40th European Workshop on Computational Geometry

Booklet of abstracts

https://eurocg2024.math.uoi.gr
March 13-15, 2024

Ioannina, Greece

Preface

The 40th European Workshop on Computational Geometry (EuroCG 2024) was held on March 13-15, 2024 in Ioannina, Greece. EuroCG is an annual, informal workshop whose goal is to provide a forum for scientists to meet, present their work, interact, and establish collaborations, in order to promote research in the field of Computational Geometry, within Europe and beyond.

Concerning the scientific program, we received 78 submissions, which underwent a limited refereeing process by the program committee in order to ensure some minimal standards and to check for plausibility. We selected 67 submissions for presentation at the workshop. For the workshop, we had 91 registered participants. EuroCG does not have formally published proceedings; therefore, we expect most of the results outlined here to be also submitted to peer-reviewed conferences and/or journals. This book of abstracts, available through the EuroCG 2024 web site, should be regarded as a collection of preprints. In addition to the 67 contributed talks, this book contains abstracts of the invited lectures. The invited speakers were Walter Didimo (University of Perugia, Italy), Ioannis Emiris (Athena Research Center and National & Kapodistrian University of Athens, Greece), and Xavier Goaoc (Université de Lorraine, Nancy, France).

Many thanks to all authors and to the members of the program committee and all external reviewers for their insightful comments. We also thank the organizing committee members: Konstantina Kyriakoudi, Maria Eleni Pavlidi, Giorgos Velissaris and Konstantina Tzouvara. Finally, we are very grateful for the generous support of our gold sponsors: the Department of Mathematics of the University of Ioannina and yWorks GmbH. We would like to thank also various contributors for their valuable contributions: Katogi Averoff, Klidarithmos publisher, Zagori Water and Green Beverages Group. Last but not least, we would like to thank the University of Ioannina for offering the lecture halls of Karolos Papoulias Conference Center.

The conference gave out a best student presentation award. The prize was voted by the EuroCG 2024 attendees to recognize the effort of young researchers to present their work clearly and elegantly. The winners were Arjen Simons, for the presentation of paper "Hausdorff morphs with fewer components", and Miriam Goetze, for the presentation of paper "Recognition Complexity of Subgraphs of 2- and 3-Connected Planar Cubic Graphs". Congratulations to Arjen and Miriam!

During the business meeting, Martin Balko presented the 2025 edition of EuroCG, which will take place near Prague, Czech Republic. A single bid was presented for 2026 by Alexandra Weinberger and, as a consequence, EuroCG 2026 will take place in Hagen, Germany.

Looking forward to seeing you next year all in Prague!

March 2024,
Michael A. Bekos and Charis Papadopoulos

Program Committee

Patrizio Angelini	John Cabot University
Gill Barequet	Technion - Israel Institute of Technology
Michael A. Bekos (co-chair)	University of Ioannina
Kevin Buchin	Technical University Dortmund
Markus Chimani	Osnabrück University
Sabine Cornelsen	University of Konstanz
Giordano Da Lozzo	Roma Tre University
Ioannis Z. Emiris	Athena Research Center & National and Kapodistrian University of Athens
Loukas Georgiadis	University of Ioannina
Joachim Gudmundsson	The University of Sydney
Siddharth Gupta	University of Warwick
Balázs Keszegh	Alfréd Rényi Institute of Mathematics
Anna Lubiw	University of Waterloo
Eunjin Oh	Pohang University of Science and Technology
Tim Ophelders	Utrecht University and TU Eindhoven
Leonidas Palios	University of Ioannina
Charis Papadopoulos (co-chair)	University of Ioannina
Maria Saumell	Czech Technical University in Prague
Lena Schlipf	Universität Tübingen
Christiane Schmidt	Linköping University
Bettina Speckmann	Eindhoven University of Technology
Joachim Spoerhase	University of Sheffield
Frank Staals	Utrecht University
Sabine Storandt	University of Konstanz
Antonios Symvonis	National Technical University of Athens
Alessandra Tappini	University of Perugia
Torsten Ueckerdt	Karlsruhe Institute of Technology
Ivor van der Hoog	Technical University of Denmark
Yelena Yuditsky	Université libre de Bruxelles

Organizing Committee

Michael A. Bekos (co-chair)	University of Ioannina
Konstantina Kyriakoudi	University of Ioannina
Charis Papadopoulos (co-chair)	University of Ioannina
Maria Eleni Pavlidi	University of Ioannina
Konstantina Tzouvara	University of Ioannina
George Velissaris	University of Ioannina

Table of Contents

Invited talks

Contributed papers

Orthogonal Graph Drawings and the Bend Minimization Problem

Walter Didimo[1]

1 University of Perugia, Italy
walter.didimo@unipg.it

Abstract

Orthogonal drawings have a rich history and rank among the most extensively studied subjects in graph drawing. They have wide-ranging applications in several real-world domains, including circuit design, computer network schematization, database design, and software engineering. In an orthogonal drawing of a graph, each edge is a sequence of horizontal and vertical segments, with a bend being the point where a horizontal segment and a vertical segment intersect. Since bends can adversely impact the readability of the graph layout, a central problem is how to compute orthogonal drawings with the minimum number of bends.

Following a brief overview on the subject, this talk focuses on a line of research that leverages the notion of "spirality", a measure of how much a component of an orthogonal drawing is rolled-up. A general strategy based on this notion was introduced over 25 years ago. This strategy has been recently revisited and refined to answer long-standing open questions, also in the presence of specific layout constraints.

40th European Workshop on Computational Geometry, Ioannina, Greece, March 13–15, 2024.
This is an extended abstract of a presentation given at EuroCG'24. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

Algebraic and combinatorial bounds on the embedding number of distance graphs

Ioannis Z. Emiris[1]

1 **Athena Research Center, Greece, and National & Kapodistrian University of Athens, Greece**
emiris@athenarc.gr

Abstract

Rigidity theory studies mechanisms with a prescribed number of degrees of freedom. Besides its fundamental nature, the theory is particularly useful in modeling robotic and architectural devices, swarms of drones or the structure of biomolecules. The main tool is distance graphs, which are weighted graphs whose edges are labeled by the prescribed distance between the corresponding nodes. Laman graphs in the plane are the most well-studied, but distance graphs can be defined and find applications in any dimension. We focus on estimating the number of Euclidean embeddings of generically minimally rigid graphs, namely graphs with a minimal number of edge constraints which are rigid for sufficiently generic edge lengths. The field is today very active, and we outline recent progress. More specifically, we exploit algebraic and combinatorial techniques to obtain nontrivial upper bounds on the maximal number of embeddings given the number of nodes. Our results also yield example graphs that provide lower bounds on the maximal number of embeddings. Conversely, we consider ways of deriving relevant root counts for polynomial systems expressed by means of distance graphs.

40th European Workshop on Computational Geometry, Ioannina, Greece, March 13–15, 2024.
This is an extended abstract of a presentation given at EuroCG'24. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

Intersection patterns of geometric set systems

Xavier Goaoc[1]

1 Université de Lorraine, Nancy, France
xavier.goaoc@loria.fr

Abstract

In this talk, I will discuss some of the ways in which the geometry or topology of a family of sets influence their intersection patterns, and illustrate how the resulting structures are harnessed by some algorithms from computational geometry and computational topology.

40th European Workshop on Computational Geometry, Ioannina, Greece, March 13–15, 2024.
This is an extended abstract of a presentation given at EuroCG'24. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

Nondango is NP-Complete

Suthee Ruangwises[1]

1 **Department of Informatics, The University of Electro-Communications, Tokyo, Japan**
ruangwises@uec.ac.jp

Abstract

Nondango is a pencil puzzle consisting of a rectangular grid partitioned into regions, with some cells containing a white circle. The player has to color some circles black such that every region contains exactly one black circle, and there are no three consecutive circles (horizontally, vertically, or diagonally) having the same color. In this paper, we prove that deciding solvability of a given Nondango puzzle is NP-complete.

Related Version arXiv:2310.11447

1 Introduction

Nondango is a pencil puzzle published by Nikoli. The puzzle consists of a rectangular grid partitioned into polyominoes called *regions*, with some cells containing a white circle. The player has to color some circles black to satisfy the following constraints [20].

1. Every region contains exactly one black circle.
2. There are no three consecutive circles (horizontally, vertically, or diagonally) having the same color (see Figure 1).

In this paper, we show that it is NP-complete to decide whether a given Nondango puzzle has a solution.

▶ **Theorem 1.1.** *Deciding solvability of a given Nondango instance is NP-complete.*

As the problem clearly belongs to NP, the nontrivial part is to prove the NP-hardness. We do so by constructing a reduction from the 1-in-3-SAT+ problem (deciding whether there is a Boolean assignment such that every clause has exactly one literal that evaluates to true, in a setting where each clause contains exactly three positive literals), which is known to be NP-complete [27].

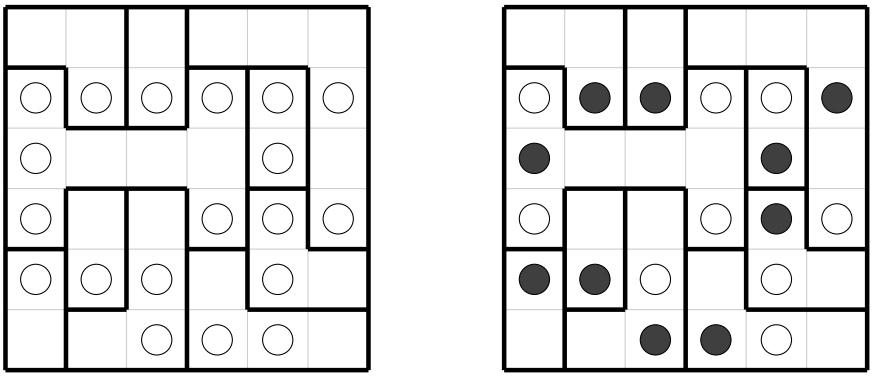


Figure 1 An example of a 6×6 Nondango puzzle (left) and its solution (right)

40th European Workshop on Computational Geometry, Ioannina, Greece, March 13–15, 2024.
This is an extended abstract of a presentation given at EuroCG'24. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

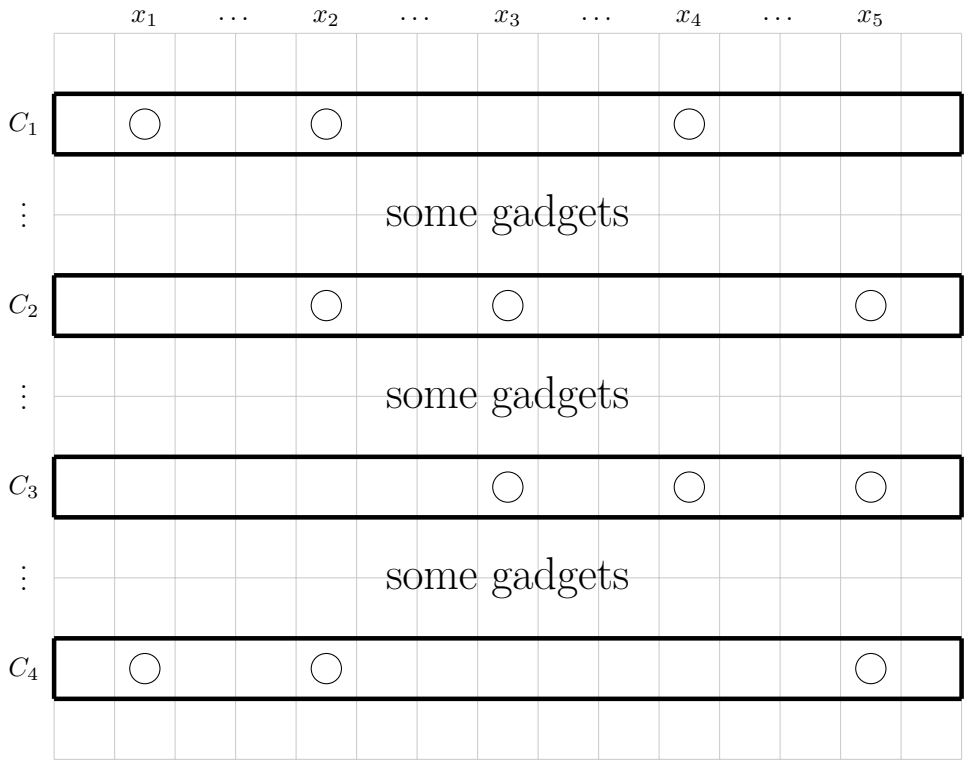


Figure 2 Basic structure of a Nondango instance transformed from a formula consisting of clauses $C_1 = x_1 \vee x_2 \vee x_4$, $C_2 = x_2 \vee x_3 \vee x_5$, $C_3 = x_3 \vee x_4 \vee x_5$, and $C_4 = x_1 \vee x_2 \vee x_5$

1.1 Related Work

Many pencil puzzles have been proved to be NP-complete, including Dosun-Fuwari [15], Fillmat [30], Five Cells [18], Goishi Hiroi [5], Hashiwokakero [4], Herugolf [14], Heyawake [11], Juosan [16], Kakuro [31], Kurodoko [21], Kurotto [16], LITS [6], Makaro [14], Moon-on-Sun [19], Nagareru [19], Nonogram [29], Norinori [6], Numberlink [1], Nurikabe [10], Nurimeizu [19], Nurimisaki [17], Pencils [23], Ripple Effect [28], Roma [9], Sashigane [17], Shakashaka [8], Shikaku [28], Slitherlink [31], Sto-Stone [3], Sudoku [31], Suguru [24], Sumplete [25], Tatamibari [2], Tilepaint [31], Toichika [26], Usowan [13], Yin-Yang [7], and Yosenabe [12].

2 Idea of the Proof

Given a 1-in-3-SAT+ formula, we will transform it into a Nondango puzzle. In the puzzle grid, each variable and each clause is represented by a column (called a *variable column*) and a row (called a *clause row*), respectively. In each clause row, a rectangular region of height 1 consists of the whole row. Inside the region, we place three circles (called *variable circles*) at columns corresponding to the three variables appearing in that clause (see Figure 2).

We interpret a black (resp. white) circle in a Nondango solution as a true (resp. false) literal. The constraint that exactly one literal in each clause is true is equivalent to that exactly one circle in that region is black. However, a more challenging task is to force every circle in each variable column to have the same color (which is equivalent to that each variable must have the same truth value in every clause it appears). We will show how to construct gadgets to enforce this constraint in the next section.

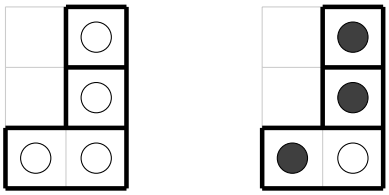


Figure 3 A gadget for creating a forced white circle (left) and its only solution (right)

3 Reduction

We use the following gadgets to construct components of the Nondango puzzle with certain properties.

3.1 Enforcing a Black Circle

Creating a circle that must be black in the solution is trivial; in a region with exactly one circle, that circle must be black in the solution. We call such circle a *forced black circle*.

3.2 Enforcing a White Circle

We can create a circle that must be white in the solution by using a gadget represented in Figure 3. As the bottom-right circle is placed next to two vertically consecutive forced black circles, it must be white in the solution (otherwise there will be three consecutive black circles in the solution). Analogously, we call such circle a *forced white circle*.

3.3 Enforcing Two Circles with Different Colors

For any region with exactly two circles, the colors of these two circles in the solution are always different. However, if we want to force two circles *in different regions* to have different colors in the solution, we can do so by using a gadget represented in Figure 4. The idea is that there are four consecutive circles arranged diagonally, with a circle at one end being forced white and at the other end being forced black. As a result, the two middle circles must have different colors in the solution (otherwise there will be three consecutive circles with the same color in the solution).

3.4 Connecting Two Consecutive Clause Rows with Common Variables

For two consecutive clause rows (e.g. clause rows for C_2 and C_3) which the corresponding clauses share a variable, we use a gadget represented in Figure 5 (see also Figure 6 for its solutions) to connect the two variable circles corresponding to that common variable. This gadget forces these two variable circles to have the same color in the solution, thus ensuring that the variable has the same truth value in both clauses. The idea behind this gadget is to use multiple copies of the gadget in Section 3.3.

3.5 Skipping a Clause Row

As the gadget in the Section 3.4 can only connect consecutive clause rows, we also provide a method to skip a clause row that does not contain the given variable. For example, if x_1

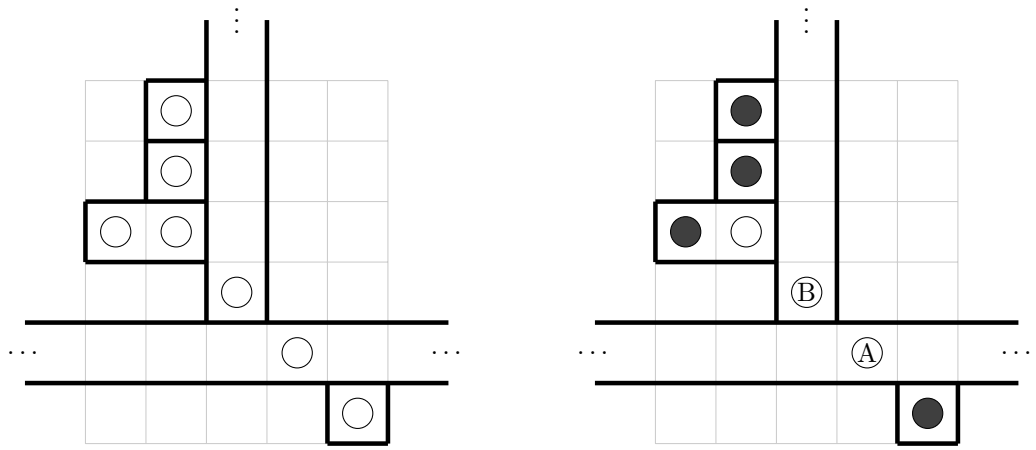


Figure 4 A gadget for enforcing two circles with different colors (left) and its only two solutions (right), where $\{A, B\} = \{\text{black}, \text{white}\}$

appears in C_2 and C_4, but not in C_3, we need to connect the clause rows of C_2 and C_4 while skipping the one of C_3[1].

We can do so by using a gadget represented in Figure 7 (see also Figure 8 for its solutions). The idea behind this gadget is that we put a forced white circle as a variable circle in the clause row we want to skip, so that the color constraint for other circles in that row will not be affected.

3.6 Filling Empty Area

We can simply make each connected empty area into one region, with one circle placed inside it, not touching any boundary. That circle is forced to be black without affecting other regions.

Recall that we interpret a black (resp. white) circle in a Nondango solution as a true (resp. false) literal. We can see that the Nondango puzzle we construct has a solution if and only if the original 1-in-3-SAT+ problem is satisfiable. As the reduction is clearly parsimonious, we can conclude that deciding solvability of a given Nondango puzzle is NP-complete.

[1] In fact, this gadget is unnecessary if we instead construct a reduction from the planar positive rectilinear 1-in-3-SAT problem, which is also NP-complete [22], where the clause rows can be arranged such that we do not need to skip a clause row. However, we include this gadget for the sake of completeness.

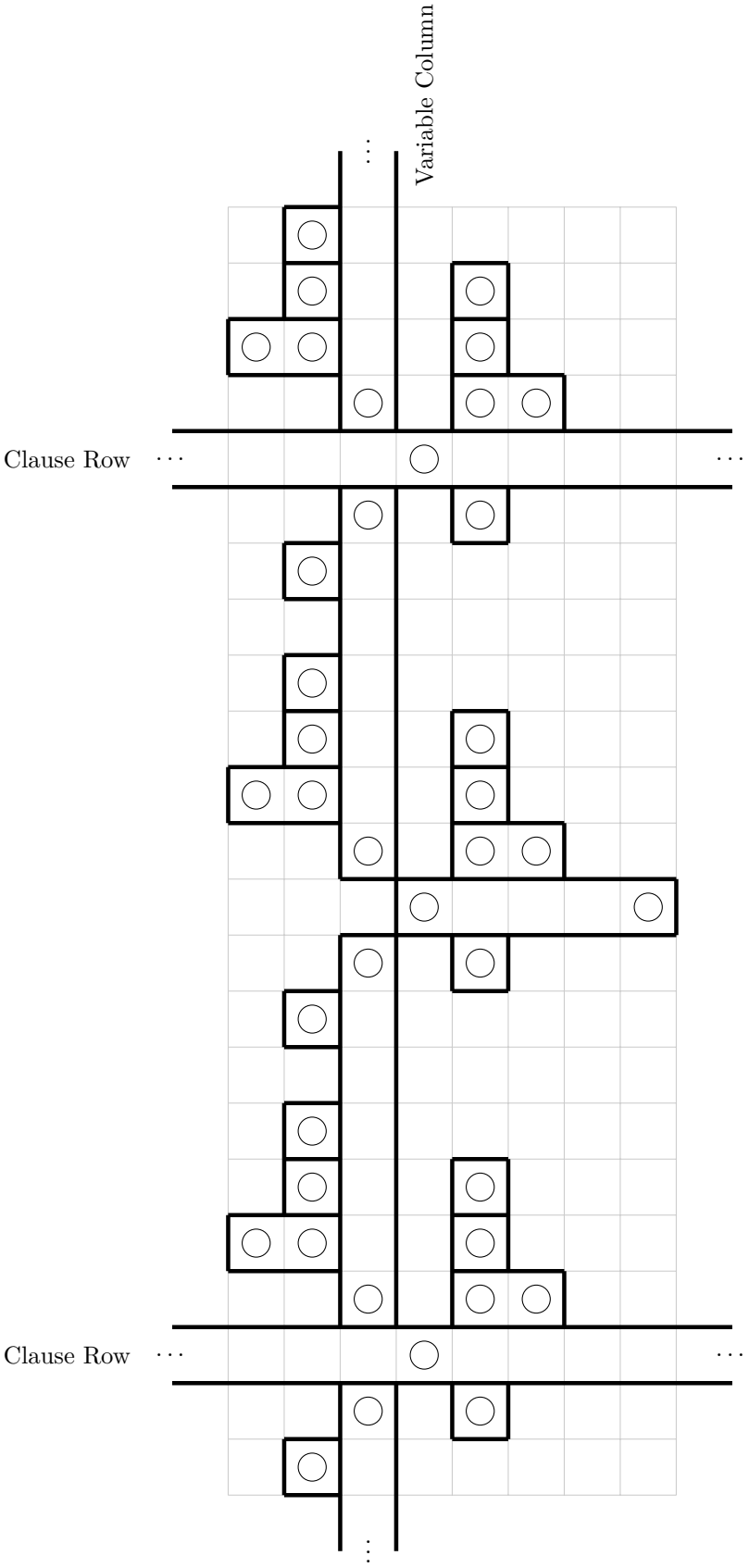


Figure 5 A gadget connecting two consecutive clause rows, forcing the two variable circles to have the same color

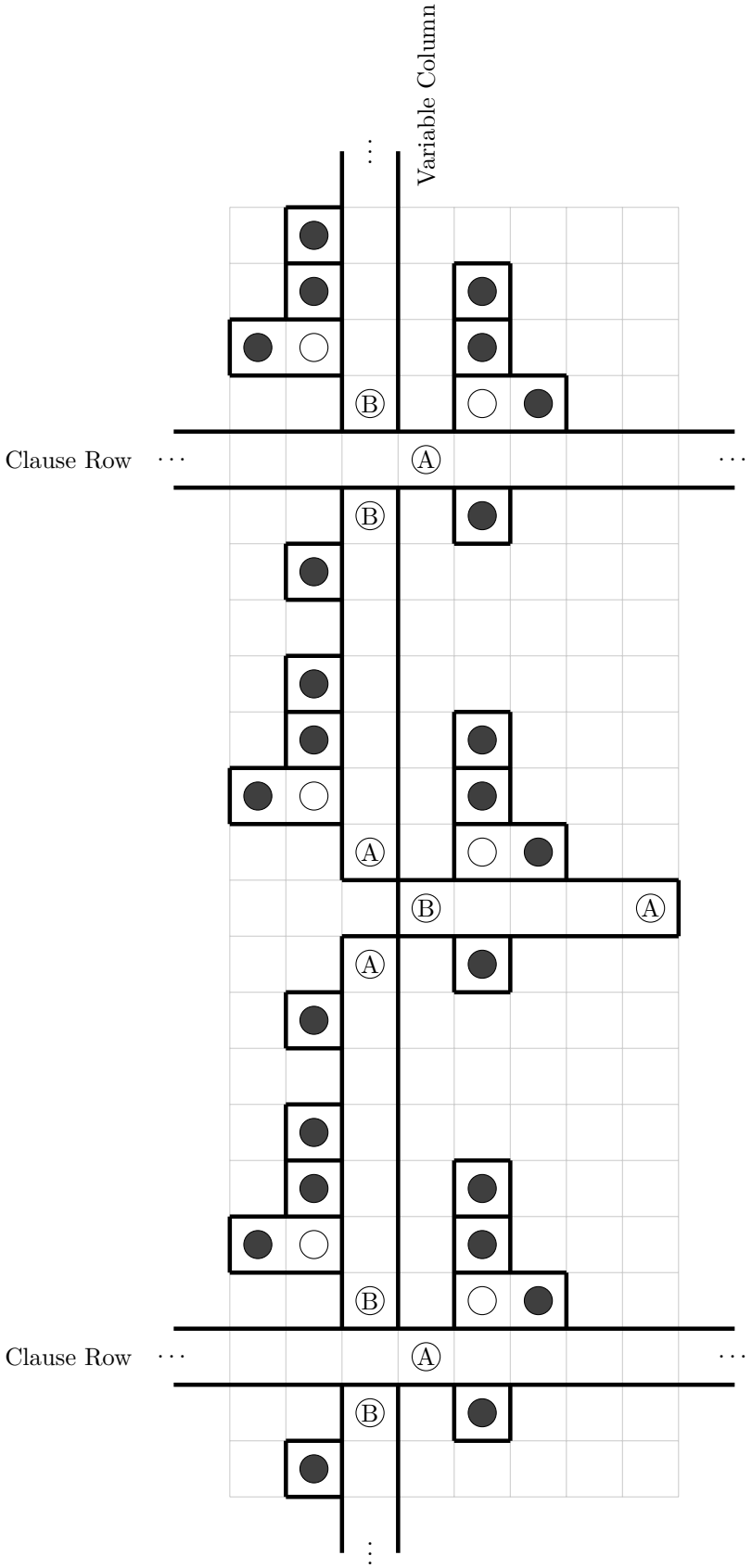


Figure 6 The only two solutions of the puzzle in Figure 5, where $\{A, B\} = \{\text{black}, \text{white}\}$

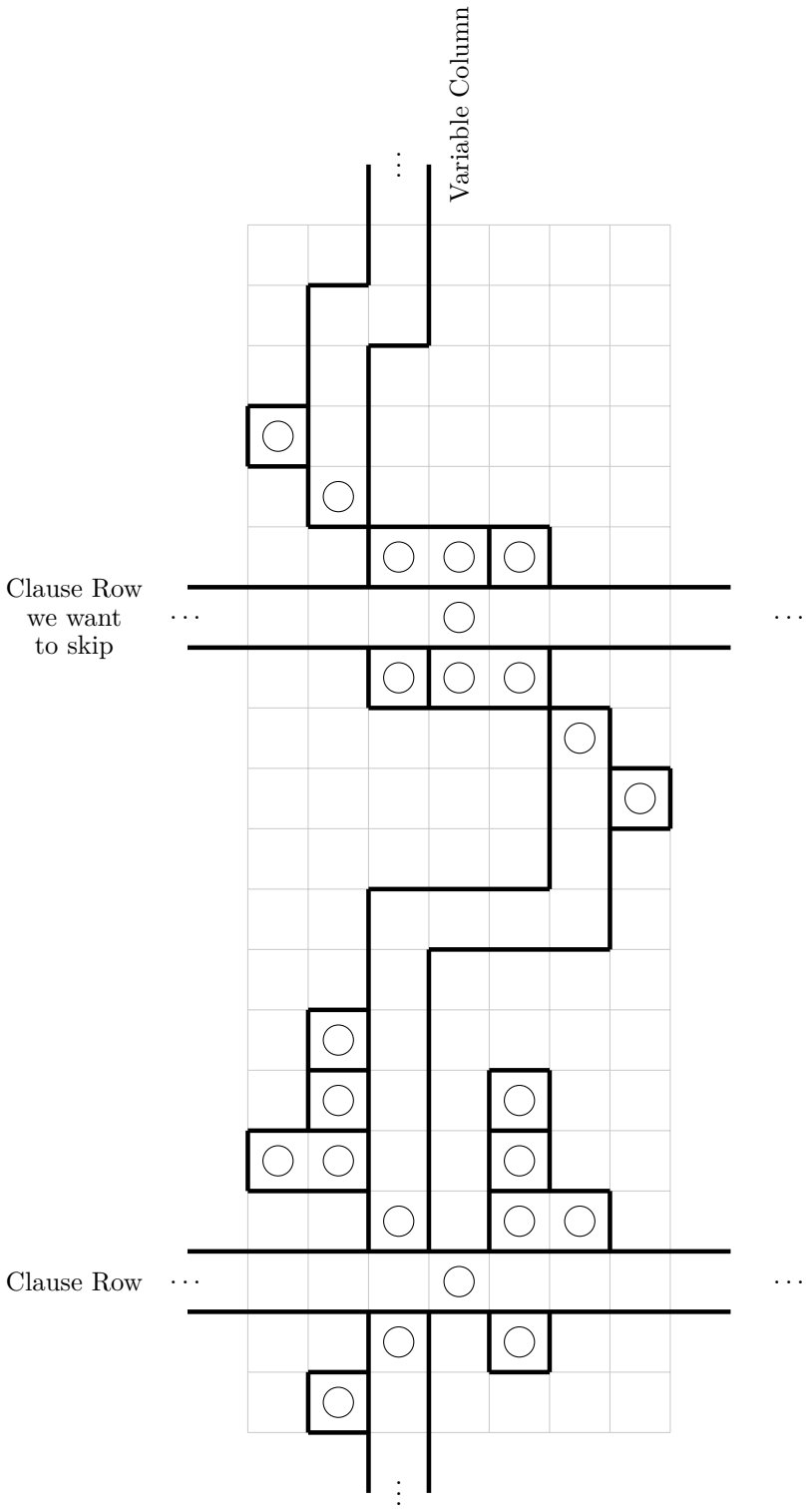


Figure 7 A gadget to skip a clause row, starting from the lower clause row and skipping the upper clause row to connect to some clause row beyond it

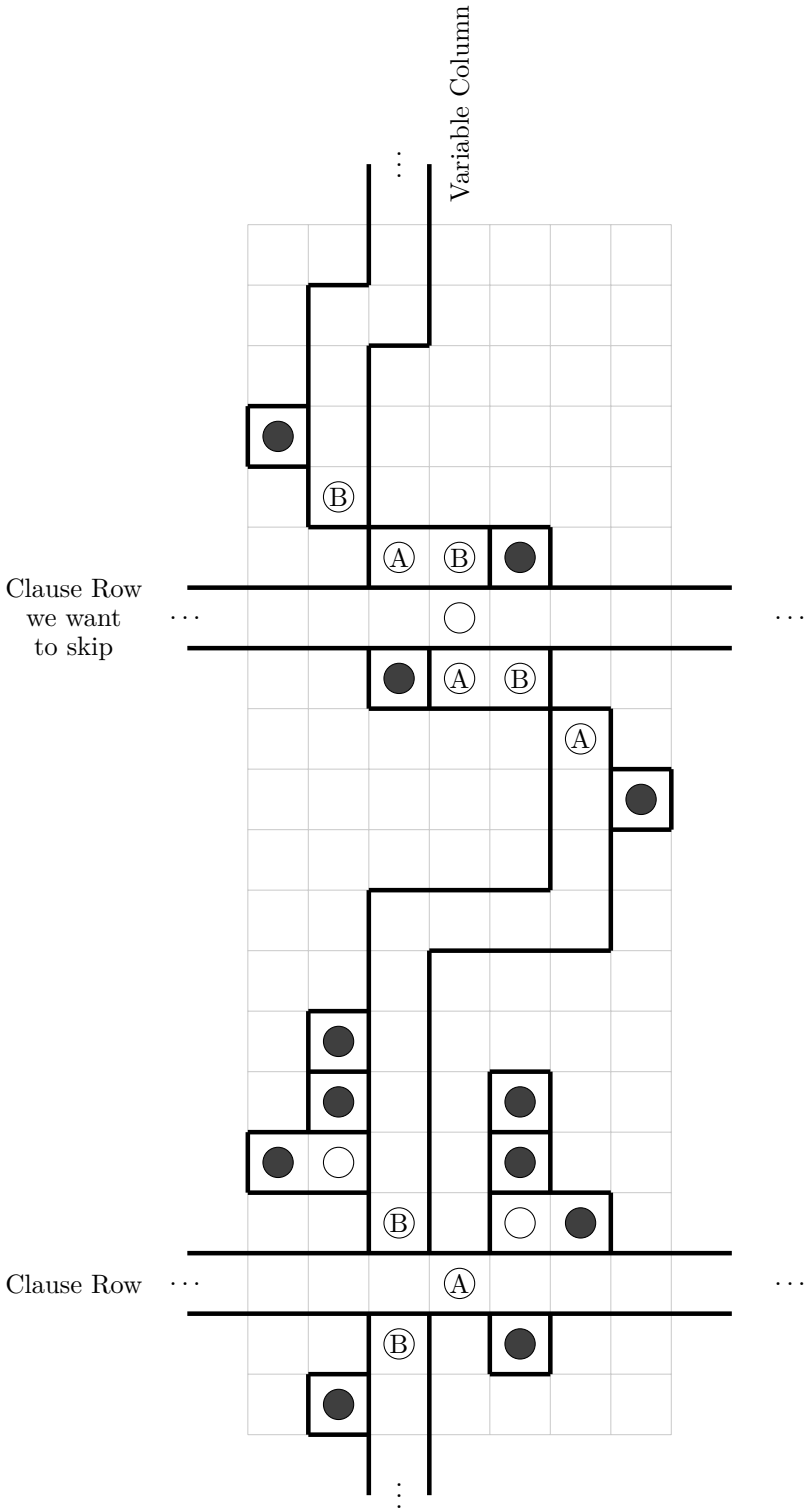


Figure 8 The only two solutions of the puzzle in Figure 7, where $\{A, B\} = \{\text{black}, \text{white}\}$

References

1. A. Adcock, E.D. Demaine, M.L. Demaine, M.P. O'Brien, F. Reidl, F.S. Villaamil and B.D. Sullivan. Zig-Zag Numberlink is NP-Complete. *Journal of Information Processing*, 23(3): 239–245 (2015).
2. A. Adler, J. Bosboom, E.D. Demaine, M.L. Demaine, Q.C. Liu and J. Lynch. Tatamibari Is NP-Complete. In *Proceedings of the 10th International Conference on Fun with Algorithms (FUN)*, pp. 1:1–1:24 (2020).
3. A. Allen and A. Williams. Sto-Stone is NP-Complete. In *Proceedings of the 30th Canadian Conference on Computational Geometry (CCCG)*, pp. 28–34 (2018).
4. D. Andersson. Hashiwokakero is NP-complete. *Information Processing Letters*, 109(9): 1145–1146 (2009).
5. D. Andersson. HIROIMONO Is NP-Complete. In *Proceedings of the 4th International Conference on Fun with Algorithms (FUN)*, pp. 30–39 (2007).
6. M. Biro and C. Schmidt. Computational complexity and bounds for Norinori and LITS. In *Proceedings of the 33rd European Workshop on Computational Geometry (EuroCG)*, pp. 29–32 (2017).
7. E.D. Demaine, J. Lynch, M. Rudoy and Y. Uno. Yin-Yang Puzzles are NP-complete. In *Proceedings of the 33rd Canadian Conference on Computational Geometry (CCCG)*, pp. 97–106 (2021).
8. E.D. Demaine, Y. Okamoto, R. Uehara and Y. Uno. Computational Complexity and an Integer Programming Model of Shakashaka. *IEICE Trans. Fundamentals*, 97.A(6): 1213–1219 (2014).
9. K. Goergen, H. Fernau, E. Oest and P. Wolf. All Paths Lead to Rome. In *Proceedings of the 24th Japan Conference on Discrete and Computational Geometry, Graphs, and Games (JCDCG^3)*, pp. 38–39 (2022).
10. M. Holzer, A. Klein and M. Kutrib. On The NP-Completeness of The Nurikabe Pencil Puzzle and Variants Thereof. In *Proceedings of the 3rd International Conference on Fun with Algorithms (FUN)*, pp. 77–89 (2004).
11. M. Holzer and O. Ruepp. The Troubles of Interior Design–A Complexity Analysis of the Game Heyawake. In *Proceedings of the 4th International Conference on Fun with Algorithms (FUN)*, pp. 198–212 (2007).
12. C. Iwamoto. Yosenabe is NP-complete. *Journal of Information Processing*, 22(1): 40–43 (2014).
13. C. Iwamoto and M. Haruishi. Computational Complexity of Usowan Puzzles. *IEICE Trans. Fundamentals*, 101.A(9): 1537–1540 (2018).
14. C. Iwamoto, M. Haruishi and T. Ibusuki. Herugolf and Makaro are NP-complete. In *Proceedings of the 9th International Conference on Fun with Algorithms (FUN)*, pp. 24:1–24:11 (2018).
15. C. Iwamoto and T. Ibusuki. Dosun-Fuwari is NP-complete. *Journal of Information Processing*, 26: 358–361 (2018).
16. C. Iwamoto and T. Ibusuki. Polynomial-Time Reductions from 3SAT to Kurotto and Juosan Puzzles. *IEICE Trans. Inf. & Syst.*, 103.D(3): 500–505 (2020).
17. C. Iwamoto and T. Ide. Computational Complexity of Nurimisaki and Sashigane. *IEICE Trans. Fundamentals*, 103.A(10): 1183–1192 (2020).
18. C. Iwamoto and T. Ide. Five Cells and Tilepaint are NP-Complete. *IEICE Trans. Inf. & Syst.*, 105.D(3): 508–516 (2022).
19. C. Iwamoto and T. Ide. Moon-or-Sun, Nagareru, and Nurimeizu are NP-Complete. *IEICE Trans. Fundamentals*, 105.A(9): 1187–1194 (2022).
20. A. Janko and O. Janko. Nondango. `https://www.janko.at/Raetsel/Nondango/index.htm`

21 J. Kölker. Kurodoko is NP-Complete. *Journal of Information Processing*, 20(3): 694–706 (2012).

22 W. Mulzer and G. Rote. Minimum-weight triangulation is NP-hard. *Journal of the ACM*, 55(2): 11:1–11:29 (2008).

23 D. Packer, S. White and A. Williams. A Paper on Pencils: A Pencil and Paper Puzzle - Pencils is NP-Complete. In *Proceedings of the 30th Canadian Conference on Computational Geometry (CCCG)*, pp. 35–41 (2018).

24 L. Robert, D. Miyahara, P. Lafourcade, L. Libralesso and T. Mizuki. Physical zero-knowledge proof and NP-completeness proof of Suguru puzzle. *Information and Computation*, 285(B): 104858 (2022).

25 S. Ruangwises. Sumplete is Hard, Even with Two Different Numbers. arXiv:2309.07161 (2023). `https://arxiv.org/abs/2309.07161`

26 S. Ruangwises. Toichika is NP-Complete. In *Proceedings of the 25th Indonesia-Japan Conference on Discrete and Computational Geometry, Graphs, and Games (IJCDCG^3)*, p. 96 (2023).

27 T.J. Schaefer. The complexity of satisfiability problems. In *Proceedings of the 10th Annual ACM Symposium on Theory of Computing (STOC)*, pp. 216–226 (1978).

28 Y. Takenaga, S. Aoyagi, S. Iwata and T. Kasai. Shikaku and Ripple Effect are NP-Complete. *Congressus Numerantium*, 216: 119–127 (2013).

29 N. Ueda and T. Nagao. NP-completeness Results for NONOGRAM via Parsimonious Reductions. Technical Report TR96-0008, Department of Computer Science, Tokyo Institute of Technology (1996).

30 A. Uejima and H. Suzuki. Fillmat is NP-Complete and ASP-Complete. *Journal of Information Processing*, 23(3): 310–316 (2015).

31 T. Yato and T. Seta. Complexity and Completeness of Finding Another Solution and Its Application to Puzzles. *IEICE Trans. Fundamentals*, 86.A(5): 1052–1060 (2003).

An Interleaving Distance for Ordered Merge Trees

Thijs Beurskens[1], Tim Ophelders[1, 2], Bettina Speckmann[1], and Kevin Verbeek[1]

1 **Department of Mathematics and Computer Science, TU Eindhoven, The Netherlands**
{t.p.j.beurskens|k.a.b.verbeek|b.speckmann}@tue.nl

2 **Department of Information and Computing Science, Utrecht University, The Netherlands**
t.a.e.ophelders@uu.nl

Abstract

Merge trees are a common topological descriptor for data with a hierarchical component. The interleaving distance, in turn, is a common distance measure for comparing merge trees. In this abstract, we introduce a form of ordered merge trees and extend the interleaving distance to a measure that preserves orders. Exploiting the additional structure of ordered merge trees, we then describe an $\mathcal{O}(n^2)$ time algorithm that computes a 2-approximation of this new distance with an additive term G that captures the maximum height differences of leaves of the input merge trees.

Related Version A full version of the paper is available at arxiv.org/abs/2312.11113.

1 Introduction

Merge trees are a common topological descriptor for data with a hierarchical component, such as terrains and scalar fields. However, standard merge trees focus solely on the hierarchy and do not represent other salient geometric features of the data. Specifically, our work is motivated by the study of braided rivers (see Figure 1). A braided river is a multi-channel river system, known to evolve rapidly [13, 17]. There exist methods to generate a *river network* from a snapshot of the terrain [7, 14, 16]. We model a river network as a hierarchy of *braids*. We use a merge tree to represent this hierarchy: each leaf represents a single channel in the network, and each internal vertex represents two braids merging (see Figure 1).

It is our goal to analyse the evolution of the channel network over time. The standard way to compare two merge trees is the interleaving distance [19]. However, the interleaving distance has two main drawbacks. Firstly, the standard interleaving distance is unable to capture any intrinsic order, e.g. from bank to bank in braided rivers, that might be present in

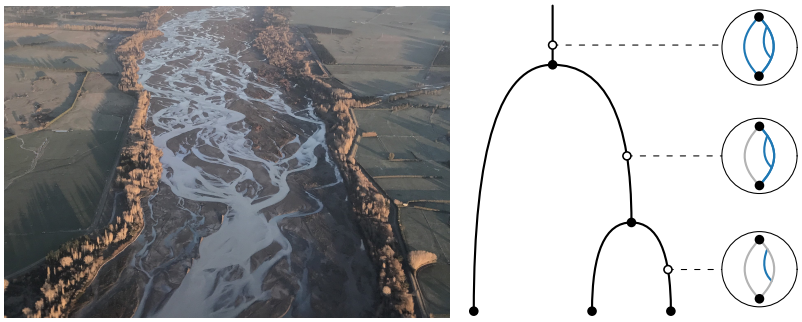


Figure 1 (Left) the Waimakariri River in New-Zealand is a braided river. Photo was taken by Greg O'Beirne [21]. (Right) representing a river network by a merge tree.

40th European Workshop on Computational Geometry, Ioannina, Greece, March 13–15, 2024.
This is an extended abstract of a presentation given at EuroCG'24. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

the data. Secondly, there is no known efficient algorithm to compute even an approximation of the interleaving distance.[1] To tackle both issues, we introduce the *monotone interleaving distance*: an order-preserving distance measure on ordered merge trees.

Contributions. We show that the monotone interleaving distance can be defined in terms of a single map between the input merge trees, two maps, or a labelling. Moreover, we give an algorithm that computes an approximation of this distance in $\mathcal{O}(n^2)$ time. Finding an efficient algorithm to compute the monotone interleaving distance exactly, or to prove NP-hardness of computing the distance, remains an open problem. We first review the relevant background in Section 2. In Section 3 we introduce a form of ordered merge trees and define monotone interleavings, monotone δ-good maps and monotone labellings. We give constructions to prove that all of these lead to the same distance. Finally, in Section 4 we describe an efficient algorithm to approximate the monotone interleaving distance. All omitted proofs can be found in the full version on arXiv.

Related work. The interleaving distance was first introduced as a measure for persistence modules [8]. It has since been well-studied from a categorical point of view [3, 4, 5, 6, 9, 10, 11, 15, 22], and has been transferred to numerous topological descriptors [2, 18, 20]. Morozov et al. [19] defined the interleaving distance for merge trees. Agarwal et al. [1] established a relation between the interleaving distance and the Gromov-Hausdorff distance.[1] The interleaving distance on merge trees was redefined by first Touli and Wang [23], and later Gasparovich et al. [12]. Touli and Wang also gave an FPT-algorithm to compute the interleaving distance. Recently, the result by Gasparovich et al. has been used to design algorithms for computing geometry aware labellings [24, 25].

2 Preliminaries

A *merge tree* is a pair (T, f), where T is a rooted tree and $f\colon T \to \mathbb{R} \cup \{\infty\}$ is a continuous height function that is increasing towards the root, with $f(v) = \infty$ if and only if v is the root. Here, f is defined not only on the vertices of T, but also on points of T interior to the edges. Specifically, f is linearly interpolated along the edges. For a point $x \in T$, we denote by T_x the subtree of T rooted at x. Furthermore, for a given value $\delta \geq 0$, we denote by x^δ the unique ancestor of x with $f(x^\delta) = f(x) + \delta \in T$.

Now consider two merge trees (T, f) and (T', f') and fix a value $\delta \geq 0$. Intuitively, a *δ-interleaving* describes a mapping α from T to T' that sends points exactly δ upwards, and a similar map β from T' to T, such that both compositions of α and β send any point to its unique ancestor 2δ higher. Figure 2 shows an example of a δ-interleaving.

▶ **Definition 1** (Morozov et al. [19])**.** Given two merge trees (T, f) and (T', f'), a pair of maps $\alpha\colon T \to T'$ and $\beta\colon T' \to T$ is called a *δ-interleaving* if for all $x \in T$ and $y \in T'$:

(C1) $f'(\alpha(x)) = f(x) + \delta$,
(C2) $\beta(\alpha(x)) = x^{2\delta}$,
(C3) $f(\beta(y)) = f'(y) + \delta$, and
(C4) $\alpha(\beta(y)) = y^{2\delta}$.

The *interleaving distance* d_{I} is defined as the smallest δ such that there exists a δ-interleaving.

[1] Agarwal et al. [1] actually prove that approximating the Gromov-Hausdorff distance with a factor better than 3 is NP-hard. As many have observed, this proof also applies to the interleaving distance.

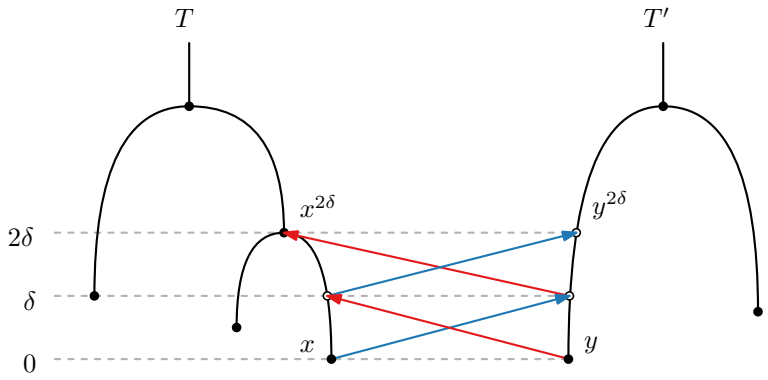


Figure 2 Two merge trees and part of a δ-interleaving. Mapping a point x from T to T' through α (in blue), and mapping it back to T via β (in red) gives the unique ancestor $x^{2\delta}$ of x.

The maps α and β are both *δ-shift maps*, i.e. continuous maps that send points exactly δ higher. Touli and Wang [23] give an alternative definition of the interleaving distance in terms of a single δ-shift map with additional requirements. They call this map a *δ-good* map. Gasparovich et al. [12] show that the interleaving distance can also be defined in terms of *labelled merge trees*: merge trees equipped with *label-maps*. Formally, let $n \geq 0$. We denote $[n] \coloneqq \{1, \ldots, n\}$. A map $\pi\colon [n] \to T$ is called a *label map* if each leaf in T is assigned at least one label. Note that π is not restricted to vertices, and may map different labels to the same point. The *induced matrix* $M = M(T, f, \pi)$ of a labelled merge tree is defined by $M_{i,j} = f(\mathrm{lca}(\pi(i), \pi(j)))$, where $\mathrm{lca}(\cdot, \cdot)$ is the *lowest common ancestor* of two points. See Figure 3 for an example of a labelled merge tree and its induced matrix.

For a matrix M, the ℓ^∞-norm is defined as $\|M\|_\infty = \max_{i,j} |M_{i,j}|$. For two *unlabelled merge trees* (T, f) and (T', f'), we refer to a pair of equally-sized label maps (π, π') as a *δ-labelling* if $\|M(T, f, \pi) - M(T', f', \pi')\|_\infty = \delta$. The δ-good interleaving distance $d_{\mathrm{I}}^{\mathrm{G}}$ and the label interleaving distance $d_{\mathrm{I}}^{\mathrm{L}}$ are defined as the smallest δ such that there exists a δ-good map or a δ-labelling. It has been shown that $d_{\mathrm{I}} = d_{\mathrm{I}}^{\mathrm{G}} = d_{\mathrm{I}}^{\mathrm{L}}$ [12, 23].

3 An Order-Preserving Interleaving Distance

We consider a new class of merge trees, which we call *ordered merge trees*. For a point $x \in T$ with $f(x) \leq h$, we denote by $x|^h$ the unique ancestor of x at height h. An ordered merge tree $(T, f, (\leq_h)_{h \geq 0})$ is a merge tree (T, f) equipped with a set of total orders on the level sets of T, such that these orders are *consistent* (see Figure 4). Formally, for two heights $h_1 \leq h_2$

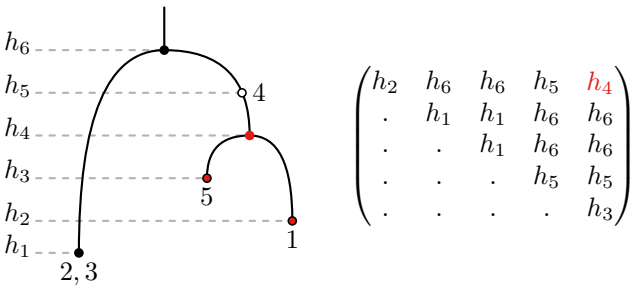


Figure 3 Example of a labelled merge tree and its induced matrix.

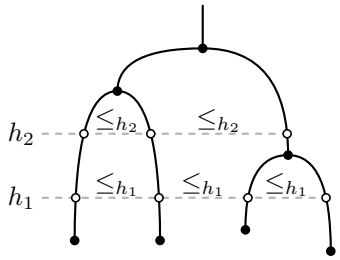


Figure 4 Two layers of an ordered merge tree.

and two points x_1, x_2 in $f^{-1}(h_1)$, we require that $x_1 \leq_{h_1} x_2$ implies $x_1|^{h_2} \leq_{h_2} x_2|^{h_2}$.

An ordered merge tree induces a binary relation $\sqsubseteq$ on the complete tree T: for $x_1, x_2 \in T$, we define $x_1 \sqsubseteq x_2$ if $x_1|^h \leq_h x_2|^h$, where $h = \max(f(x_1), f(x_2))$. This *induced relation* is not antisymmetric, and thus not a total order. If we restrict $\sqsubseteq$ to the set of leaves of T, denoted $\sqsubseteq_L$, we do obtain a total order. We refer to $\sqsubseteq_L$ as the *induced leaf-order* of T.

Monotone interleaving distance. We now define an order-preserving distance measure for ordered merge trees. Specifically, we say a δ-shift map $\alpha\colon T \to T'$ is *monotone* if for all height values $h \geq 0$ and for any two points $x_1, x_2 \in f^{-1}(h)$ it holds that $x_1 \leq_h x_2$ implies $\alpha(x_1) \leq'_{h+\delta} \alpha(x_2)$. A *monotone δ-interleaving* is a δ-interleaving (α, β) such that the maps α and β are both monotone (see Figure 5). A *monotone δ-good map*, in turn, is a δ-good map α that is also monotone. Lastly, a δ-labelling (π, π') of size n is *monotone* if for all $\ell_1, \ell_2 \in [n]$ it holds that $\pi(\ell_1) \sqsubset \pi(\ell_2)$ implies $\pi'(\ell_1) \sqsubseteq' \pi'(\ell_2)$. The *monotone interleaving distance* d_{MI} is defined as the smallest δ such that there exists a monotone δ-interleaving. Similarly, we can define the monotone δ-good, and the monotone label interleaving distances, denoted $d^{\mathrm{G}}_{\mathrm{MI}}$ and $d^{\mathrm{L}}_{\mathrm{MI}}$. Our main result is the following.

▶ **Theorem 2.** *The distances d_{MI}, $d^{\mathrm{G}}_{\mathrm{MI}}$ and $d^{\mathrm{L}}_{\mathrm{MI}}$ are equal.*

To prove Theorem 2, we describe constructions between monotone δ-interleavings, monotone δ-good maps, and monotone δ-labellings. The first construction, from a monotone δ-interleaving to a monotone δ-good map, follows directly from the regular setting (by Touli and Wang [23]). The construction of a monotone δ-labelling from a monotone δ-good map follows from a refinement of the construction by Gasparovich et al. [12].[2] We use y^F to denote the lowest ancestor of $y \in T'$ in the image of α. Moreover, for two distinct points $x_1, x_2 \in T$, we say x_1 is *smaller* than x_2 if $x_1 \sqsubseteq x_2$. The existing construction is as follows.

[2] We remark that they use a slightly different (but equivalent) definition for a δ-good map.

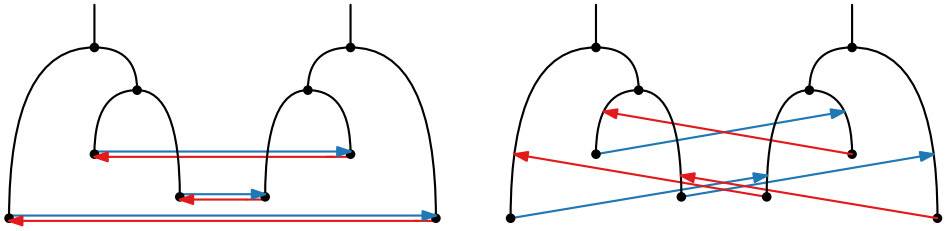


Figure 5 Parts of an optimal regular (left) and an optimal monotone (right) interleaving.

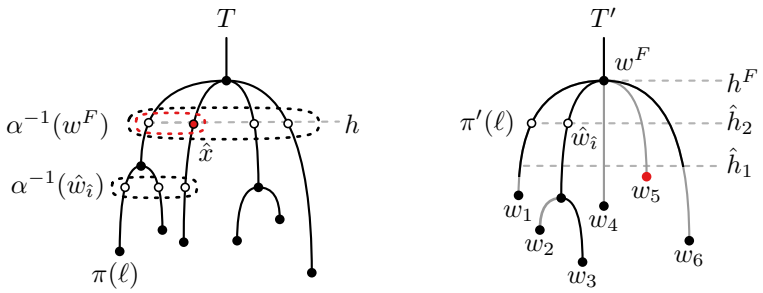


Figure 6 The refined step (S2). The grey parts of T' do not lie in the image of α. We add the pair $(\hat{x}, w_5)$ to Π. In this example, $w = w_5$, $S = \{1, 2, 3, 6\}$, $i = 5$, $S_i = \{1, 2, 3\}$ and $\hat{\imath} = 3$.

(S1) For every leaf $u \in L(T)$, add $(u, \alpha(u))$ to an initially empty set Π.
(S2) For every leaf $w \in L(T')$, take an arbitrary point $x \in \alpha^{-1}(w^F)$. Add (x, w) to Π.
(S3) Consider an arbitrary ordering $\Pi = \{(x_\ell, y_\ell) \mid \ell \in [n]\}$, and set $\pi(\ell) = x_\ell$, $\pi'(\ell) = y_\ell$.

We refine (S2), by choosing a specific $x \in \alpha^{-1}(w^F)$. Intuitively, we first identify all points $\bar{x}$ that lead to a violation of the monotonicity property if we choose x smaller than $\bar{x}$. Such a violation occurs if $\bar{x}$ is an ancestor of a labelled point whose corresponding labelled point in T' is smaller than w. We then take x to be the largest point among the points $\bar{x}$.

(S2) For every leaf $w \in L(T')$, sort the set of leaves in T'_{w^F} by induced leaf-order, denoted $W = \{w_1, \ldots, w_m\}$. Define $S \subseteq [m]$ such that for $k \in S$, w_k^F is a strict descendant of w^F. Fix $i \in [m]$ such that $w_i = w$, and define $S_i = \{k \in S \mid k < i\}$. Now consider the set $X = \alpha^{-1}(w^F)$
- If S_i is empty, take x to be the smallest point in X and add (x, w) to Π.
- If S_i is not empty, consider the largest index $\hat{\imath} \in S_i$. Define Y as the set of strict descendants of w^F that were labelled in (S1). Consider the following height values:

$$\hat{h}_1 := \max\{f'(w_k^F) \mid k \in S\}, \quad \hat{h}_2 := \max\{f'(y) \mid y \in Y\}, \quad \hat{h} = \max(\hat{h}_1, \hat{h}_2) \tag{1}$$

Consider the unique ancestor $\hat{w}_{\hat{\imath}}$ of $w_{\hat{\imath}}$ at height $\hat{h}$. Note that $\hat{w}_{\hat{\imath}}$ is a strict descendant of the point w^F and that it lies in the image of α. Let $X_{\hat{\imath}} \subset X$ be the set of ancestors of points in $\alpha^{-1}(\hat{w}_{\hat{\imath}})$. Take $\hat{x}$ to be the largest point in $X_{\hat{\imath}}$ and add $(\hat{x}, w)$ to Π.

See Figure 6 for an illustration. We can show that the resulting δ-labelling is monotone.

Lastly, we construct a monotone δ-interleaving from a monotone δ-labelling. To do so, we extend an existing construction of a δ-good map α from a δ-labelling by Gasparovich et al. [12]. Specifically, for all $x \in T$, they consider an arbitrary label ℓ from the subtree T_x and set $y_\ell = \pi'(\ell)|^{f(x)+\delta}$. Gasparovich et al. show that the point y_ℓ is well-defined, and argue that the resulting map α is a δ-good map. We can construct a δ-interleaving (α, β) by using this construction twice: first to build a map $\alpha\colon T \to T'$ and next to build a map $\beta\colon T' \to T$.

Monotone leaf-label interleaving distance. We now turn to a restriction of the (monotone) label interleaving distance. Specifically, if we restrict a label map to map *only* to the leaves of T, we obtain a *leaf-label map*. A *δ-leaf-labelling* is a pair of leaf-label maps that is also a δ-labelling. The *leaf-label interleaving distance* $d_{\mathrm{I}}^{\mathrm{LL}}$, in turn, is defined as the smallest δ for which there exists a δ-leaf labelling. We can show that this distance is an approximation of the interleaving distance, in both the regular and monotone setting. We define the *leaf-gap* G of two trees T and T' as the maximum height difference of any pair of leaves in T and T'.

▶ **Theorem 3.** *The monotone leaf-label interleaving distance between* $(T, f, (\leq_h))$ *and* $(T', f', (\leq'_h))$ *is bounded by* $2\delta + G$*, where* $\delta = d_{\mathrm{MI}}(T, T')$ *and* G *is the leaf-gap of* T *and* T'*.*

4 Approximating the Monotone Interleaving Distance

In this section we describe an algorithm to compute the monotone leaf-label interleaving distance between two ordered merge trees $(T, f, (\leq_h))$ and $(T', f', (\leq'_h))$. We denote the leaves of T and T', sorted by leaf-order, by $L(T) = \{u_1, \ldots, u_m\}$ and $L(T') = \{w_1, \ldots, w_{m'}\}$, respectively. We can show that we can permute the labels of a monotone labelling (ω, ω') on T and T', such that the resulting label maps *respect* the induced leaf-orders of their trees. That is, for any two labels $i, j \in [n]$ with $i < j$, we have $\omega(i) \sqsubseteq_L \omega(j)$ and $\omega'(i) \sqsubseteq'_L \omega'(j)$. Assume (ω, ω') is such a monotone leaf-labelling. Let $M = M(T, f, \omega)$ and $M' = M(T', f', \omega')$ be the corresponding induced matrices and set $\mathcal{M} = \mathcal{M}(T, T') \coloneqq |M - M'|$. We refer to entries $\mathcal{M}_{i,i}$ as the *diagonal* of $\mathcal{M}$, and to entries $\mathcal{M}_{i,j}$ with $j - i = 1$ as the *upper-diagonal* of $\mathcal{M}$.

▶ **Lemma 4.** *The maximum of* $\mathcal{M}$ *lies on the diagonal or upper diagonal of* $\mathcal{M}$*.*

We now describe a dynamic program to compute the monotone leaf-label interleaving distance between T and T'. We denote by $T[i]$ the subtree of T consisting of only the first i leaves. For $i \in |L(T)|$ and $j \in |L(T')|$, we maintain a value $\Delta[i, j]$ that stores the monotone leaf-label interleaving distance between $T[i]$ and $T'[j]$. Consider an optimal monotone leaf-labelling (ω, ω') for $T[i]$ and $T'[j]$ with a minimum number of k labels, such that ω and ω' respect the leaf-orders of T and T', respectively. From Lemma 4 we know that to compute $d^{\mathrm{LL}}_{\mathrm{MI}}(T[i], T'[j])$, it suffices to compute the diagonal and upper-diagonal entries of $\mathcal{M} = \mathcal{M}(T[i], T'[j])$. As $\omega(k) = u_i$ and $\omega'(k) = w_j$, we have $\mathcal{M}_{k,k} = |f(u_i) - f'(w_j)| =: \varepsilon$. To compute the other relevant elements of $\mathcal{M}$, we consider the three options for label $k - 1$:

$$\begin{array}{lll} (1) \quad \omega(k-1) = u_i, & (2) \quad \omega(k-1) = u_{i-1}, & (3) \quad \omega(k-1) = u_{i-1}, \\ \quad\ \omega'(k-1) = w_{j-1}, & \quad\ \omega'(k-1) = w_j, & \quad\ \omega'(k-1) = w_{j-1}. \end{array}$$

First assume case (1) applies. Then we know that $\mathcal{M}_{k-1,k} = |f(u_i) - f'(\mathrm{lca}(w_{j-1}, w_j))|$. Furthermore, $\Delta[i, j-1]$ captures the remaining relevant entries of $\mathcal{M}$. We set $\delta_1 = \max(\Delta[i, j-1], \mathcal{M}_{k-1,k})$. Similarly, we can set δ_2 and δ_3 for cases (2) and (3) respectively. Finally, at each iteration, we set $\Delta[i, j] = \max(\varepsilon, \min(\delta_1, \delta_2, \delta_3))$. We can show that $\Delta[i, j] = d^{\mathrm{LL}}_{\mathrm{MI}}(T[i], T'[j])$. The algorithm returns $\Delta[|L(T)|, |L(T')|]$.

We can use our algorithm to compute a monotone interleaving as follows. First, we compute the lowest common ancestors of all consecutive pairs of leaves in T or T'. This allows us to construct Δ in $\mathcal{O}(n^2)$ time, where $n = |L(T)| + |L(T')|$. Recovering an optimal leaf-labelling from the dynamic program can be done in a standard way. Next, we can construct two partial maps $\alpha_L \colon L(T) \to T'$ and $\beta_L \colon L(T') \to T$ using the construction from Section 3 in $\mathcal{O}(n)$ time. Lastly, one can recover a complete interleaving (α, β) from α_L and β_L using continuity and δ-shift map properties. Our final result follows from Theorem 3:

▶ **Theorem 5.** *Given two ordered merge trees* T *and* T'*, there exists an* $\mathcal{O}(n^2)$ *algorithm that computes a monotone* δ*-interleaving between* T *and* T'*, where* $\delta \leq 2d_{\mathrm{MI}}(T, T') + G$*.*

5 Acknowledgments

Thijs Beurskens and Tim Ophelders are supported by the Dutch Research Council (NWO) under project numbers OCENW.M20.089 and VI.Veni.212.260, respectively.

References

1 P.K. Agarwal, K. Fox, A. Nath, A. Sidiropoulos, and Y. Wang. Computing the Gromov-Hausdorff distance for metric trees. *ACM Transactions on Algorithms*, 14(2):1–20, 2018. doi:10.1145/3185466.

2 U. Bauer, X. Ge, and Y. Wang. Measuring distance between Reeb graphs. In *Proc. 30th Symposium on Computational Geometry (SoCG)*, pages 464–473, 2014. doi:10.1145/2582112.2582169.

3 H.B. Bjerkevik, M. B. Botnan, and M. Kerber. Computing the interleaving distance is np-hard. *Foundations of Computational Mathematics*, 20:1237–1271, 2020. doi:10.1007/s10208-019-09442-y.

4 H.B. Bjerkevik and M.B. Botnan. Computational Complexity of the Interleaving Distance. In *Proc. 34th Symposium on Computational Geometry (SoCG)*, pages 13:1–13:15, 2018. doi:10.4230/LIPIcs.SoCG.2018.13.

5 P. Bubenik, V. De Silva, and J.A. Scott. Metrics for generalized persistence modules. *Foundations of Computational Mathematics*, 15(6):1501–1531, 2015. doi:10.1007/s10208-014-9229-5.

6 P. Bubenik and J.A. Scott. Categorification of persistent homology. *Discrete & Computational Geometry*, 51(3):600–627, 2014. doi:10.1007/s00454-014-9573-x.

7 A.J. Chadwick, E. Steel, R.A. Williams-Schaetzel, P. Passalacqua, and C. Paola. Channel migration in experimental river networks mapped by particle image velocimetry. *Journal of Geophysical Research: Earth Surface*, 127(1):e2021JF006300, 2022. doi:10.1029/2021JF006300.

8 F. Chazal, D. Cohen-Steiner, M. Glisse, L.J. Guibas, and S.Y. Oudot. Proximity of persistence modules and their diagrams. In *Proc. 25th Symposium on Computational Geometry (SoCG)*, pages 237–246, 2009. doi:10.1145/1542362.1542407.

9 J. Curry, H. Hang, W. Mio, T. Needham, and O.B. Okutan. Decorated merge trees for persistent topology. *Journal of Applied and Computational Topology*, 6(3):371–428, 2022. doi:10.1007/s41468-022-00089-3.

10 V. De Silva, E. Munch, and A. Patel. Categorified Reeb graphs. *Discrete & Computational Geometry*, 55(4):854–906, 2016. doi:10.1007/s00454-016-9763-9.

11 V. De Silva, E. Munch, and A. Stefanou. Theory of interleavings on a category with a flow. *Theory and Applications of Categories*, 33(1):583–607, 2018. URL: http://www.tac.mta.ca/tac/volumes/33/21/33-21a.pdf.

12 E. Gasparovich, E. Munch, S. Oudot, K. Turner, B. Wang, and Y. Wang. Intrinsic interleaving distance for merge trees. arXiv:1908.00063.

13 A.D. Howard, M.E. Keetch, and C.L. Vincent. Topological and geometrical properties of braided streams. *Water Resources Research*, 6(6):1674–1688, 1970. doi:10.1029/WR006i006p01674.

14 M.G. Kleinhans, M. van Kreveld, T. Ophelders, W. Sonke, B. Speckmann, and K. Verbeek. Computing representative networks for braided rivers. *Journal of Computational Geometry*, 10(1):423–443, 2019. doi:10.20382/jocg.v10i1a14.

15 M. Lesnick. The theory of the interleaving distance on multidimensional persistence modules. *Foundations of Computational Mathematics*, 15(3):613–650, 2015. doi:10.1007/s10208-015-9255-y.

16 Y. Liu, P.A. Carling, Y. Wang, E. Jiang, and P.M. Atkinson. An automatic graph-based method for characterizing multichannel networks. *Computers & Geoscience*, 166:105180, 2022. doi:10.1016/j.cageo.2022.105180.

17 W.A. Marra, M.G. Kleinhans, and E.A. Addink. Network concepts to describe channel importance and change in multichannel systems: test results for the jamuna river, bangladesh. *Earth Surface Processes and Landforms*, 39(6):766–778, 2014. doi:10.1002/esp.3482.

18 F. Mémoli and L. Zhou. Persistent homotopy groups of metric spaces. arXiv:1912.12399.

19 D. Morozov, K. Beketayev, and G. Weber. Interleaving distance between merge trees. Manuscript, 2013.

20 E. Munch and A. Stefanou. The ℓ^∞-Cophenetic metric for phylogenetic trees as an interleaving distance. *Research in Data Science*, 17:109–127, 2019. doi:10.1007/978-3-030-11566-1_5.

21 G. O'Beirne. Waimakariri River (New Zealand), 2017. Image is slightly cropped; accessed on 30/11/2023. Photo is licensed under CC BY 4.0 (https://creativecommons.org/licenses/by/4.0/). URL: https://commons.wikimedia.org/w/index.php?curid=61302377.

22 A. Patel. Generalized persistence diagrams. *Journal of Applied and Computational Topology*, 1(3-4):397–419, 2018. doi:10.1007/s41468-018-0012-6.

23 E.F. Touli and Y. Wang. FPT-algorithms for computing the Gromov-Hausdorff and interleaving distances between trees. *Journal of Computational Geometry*, 13:89–124, 2022. doi:10.20382/jocg.v13i1a4.

24 L. Yan, T. B. Masood, F. Rasheed, I. Hotz, and B. Wang. Geometry aware merge tree comparisons for time-varying data with interleaving distances. *IEEE Transactions on Visualization and Computer Graphics*, 29(8):3489–3506, 2022. doi:10.1109/TVCG.2022.3163349.

25 L. Yan, Y. Wang, E. Munch, E. Gasparovich, and B. Wang. A structural average of labeled merge trees for uncertainty visualisation. *IEEE Transactions on Visualization and Computer Graphics*, 26(1), 2020. doi:10.1109/TVCG.2019.2934242.

Greedy Monochromatic Island Partitions

Steven van den Broek[1], Wouter Meulemans*[1], and Bettina Speckmann[1]

1 **TU Eindhoven, the Netherlands**
[s.w.v.d.broek|w.meulemans|b.speckmann]@tue.nl

Abstract

Constructing partitions of colored points is a well-studied problem in discrete and computational geometry. We study the problem of creating a minimum-cardinality partition into monochromatic islands. Our input is a set S of n points in the plane where each point has one of $k \geq 2$ colors. A set of points is monochromatic if it contains points of only one color. An island I is a subset of S such that $\mathcal{CH}(I) \cap S = I$, where $\mathcal{CH}(I)$ denotes the convex hull of I. We identify an island with its convex hull; therefore, a partition into islands has the additional requirement that the convex hulls of the islands are pairwise-disjoint. We present three greedy algorithms for constructing island partitions and analyze their approximation ratios.

Related Version arXiv:2402.13340

1 Introduction

Constructing partitions of colored points is a well-studied problem in discrete [8, 12] and computational geometry [1, 4, 5, 16]. The colors of the points can be present in the constraints and the optimization criterion in different ways. For example, one may require the partition to be *balanced*—see the survey by Kano and Urrutia [12] for many such instances—or *monochromatic* [1, 4, 5, 8]. Alternatively, one may want to minimize or maximize the *diversity* [16] or *discrepancy* [3, 7] of the partition. Furthermore, one can use different geometries to partition the points, such as triangles [1], disks [5], or lines [4].

We study the problem of creating a minimum-cardinality partition into *monochromatic islands* [2]. Our input is a set S of n points in the plane where each point has one of $k \geq 2$ colors. A set of points is *monochromatic* if it contains points of only one color. An *island* I is a subset of S such that $\mathcal{CH}(I) \cap S = I$, where $\mathcal{CH}(I)$ denotes the convex hull of I. We identify an island with its convex hull; therefore, a partition into islands has the additional requirement that the convex hulls of the islands are pairwise-disjoint.

Related work. Bautista-Santiago et al. [2] study islands and describe an algorithm that can find a monochromatic island of maximum cardinality in $O(n^3)$ time, improving upon an earlier $O(n^3 \log n)$ algorithm [10]. Dumitrescu and Pach [8] consider monochromatic island partitions and prove how many islands are sufficient and sometimes necessary for different types of input. Bereg et al. [3] use island partitions to define a notion of *coarseness* that captures how blended a set of red and blue points are. Agarwal and Suri [1] study the following problem: given red and blue points, cover the blue points with the minimum number of pairwise-disjoint monochromatic triangles. They prove that this problem is NP-hard and describe approximation algorithms. Their NP-hardness reduction can be used to prove that covering and partitioning points of only one color into the minimum number of

* W. Meulemans is partially supported by the Dutch Research Council (NWO) under project number VI.Vidi.223.137.

40th European Workshop on Computational Geometry, Ioannina, Greece, March 13–15, 2024.
This is an extended abstract of a presentation given at EuroCG'24. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

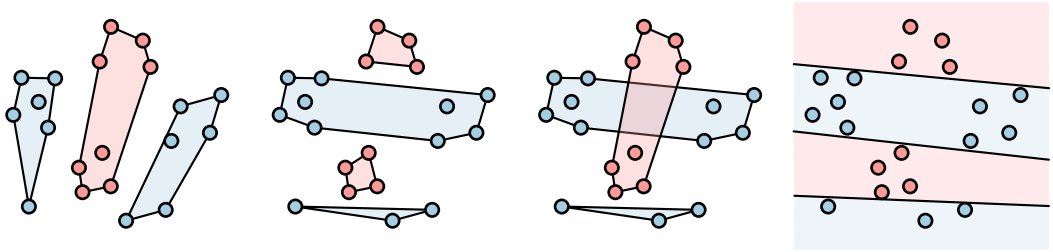


Figure 1 Left: optimal island partition; middle-left: disjoint-greedy island partition; middle-right: overlap-greedy island cover; right: line-greedy separating lines.

monochromatic islands—the points of the other colors serving only as obstacles—is NP-hard, as observed by Bautista-Santiago et al. [2]. We suspect that the problem we study is NP-hard as well, which motivates us to focus on approximation algorithms.

Overview. In the remainder, we consider only monochromatic islands. We denote by $\mathrm{Opt}_{\mathrm{P}}$ the minimum cardinality of an island partition of S. In the following sections, we use three greedy algorithms—*disjoint-greedy*, *overlap-greedy*, and *line-greedy*—to construct island partitions. Disjoint-greedy creates an island partition by iteratively picking the island that covers most uncovered points and does not intersect any island chosen before. We sketch a proof that shows that disjoint-greedy has an approximation ratio of $\Omega(n/\log^2 n)$. The overlap-greedy algorithm greedily constructs an $O(\log n)$-approximation of the minimum-cardinality island cover. We prove that any algorithm that transforms an island cover returned by overlap-greedy into an island partition has approximation ratio $\Omega(\sqrt{n})$, and describe one such algorithm that has approximation ratio $O(\mathrm{Opt}_{\mathrm{P}}^2 \log^2 n)$. Lastly, we investigate the relation between constructing a minimum-cardinality island partition and finding the minimum number of lines that separate the points into monochromatic regions. In particular, we show that greedily choosing the line that separates most pairs of points of different color induces an $O(\mathrm{Opt}_{\mathrm{P}} \log^2 n)$-approximation to the minimum-cardinality island partition. Figure 1 illustrates the greedy algorithms. The full paper contains all technical details.

2 Disjoint-Greedy

We sketch our lower bound construction that shows that disjoint-greedy has an approximation ratio of $\Omega(n/\log^2 n)$. Consider a family of problem instances that have the form of two opposing complete binary trees of height ℓ (Figure 2). Sets of points are placed close together at the nodes of these trees. The idea is that by placing sufficiently many points at the nodes, and by placing obstacle points appropriately, the disjoint-greedy algorithm iteratively picks points of two opposing nodes such that the problem instance is split into two symmetric nearly independent parts that have nearly the same structure as the original instance. This results in disjoint-greedy returning a partition into $\Omega(2^\ell)$ islands (Figure 3). However, there exists a partition such that each layer in the tree consists of a constant number of islands, resulting in $O(\ell)$ islands in total (Figure 4). In our construction, the number of red points at a node at height $i \in \{0, \ldots, \ell-1\}$ is 2^{i+6} and the number of blue points at a node is constant. Hence, the problem instance contains $\Theta(\ell \cdot 2^\ell)$ points in total and the approximation ratio of disjoint-greedy is $\Omega(2^\ell/\ell) = \Omega(n/\log^2 n)$.

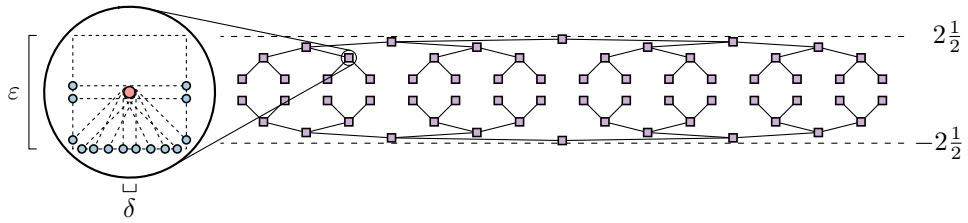


Figure 2 The problem instance for $\ell = 5$. The lines in the figure are not part of the problem instance, but illustrate its structure. The purple squares represent red and blue points lying close together inside a square. The red disk inside the square represents many red points placed together inside a disk. The centers of the purple squares lie within the strip bounded by the two dashed lines.

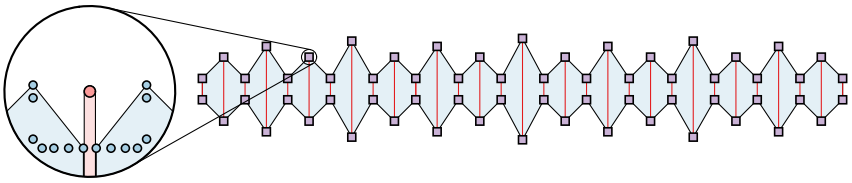


Figure 3 The solution returned by disjoint-greedy.

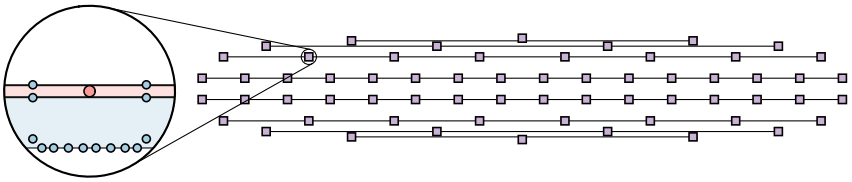


Figure 4 An alternative solution, serving as an upper bound for the optimal solution.

3 Overlap-Greedy

A greedy algorithm that iteratively picks the island that covers most uncovered points results in an $O(\log n)$ approximation to the minimum-cardinality island *cover*. This follows immediately from viewing the problem as a set cover problem, where islands form the sets. We refer to this greedy algorithm as *overlap-greedy*. Below, we explore how to transform the island cover returned by overlap-greedy into an island partition. We assume that the greedy algorithm breaks ties by choosing an island that covers the fewest previously covered points.

We first define the relation between island covers and partitions based on them. Intuitively, islands that intersect can be transformed into a set of pairwise-disjoint islands by splitting them. In the transformation, each island has a corresponding family of islands into which it is split. The union of such a family should be a subset of the original island—a subset, not equal, because a point that was originally covered by multiple islands should be part of exactly one island after the transformation. This motivates the following definition.

▶ **Definition 1** (Compatible). Families $\mathcal{I}'_1, \ldots, \mathcal{I}'_m$ are *compatible* with islands $I_1, \ldots, I_m$ if:

- Families $\mathcal{I}'_1, \ldots, \mathcal{I}'_m$ cover the same points as $I_1, \ldots, I_m$: $\bigcup_k \bigcup \mathcal{I}'_k = \bigcup_i I_i$;
- For every i, we have $\bigcup \mathcal{I}'_i \subseteq I_i$;
- Islands $\bigcup_k \mathcal{I}'_k$ are pairwise-disjoint.

Islands $I'_1, \ldots, I'_{m'}$ are compatible with islands $I_1, \ldots, I_m$ if there exists a partition of $\{I'_1, \ldots, I'_{m'}\}$ into families that are compatible with $I_1, \ldots, I_m$.

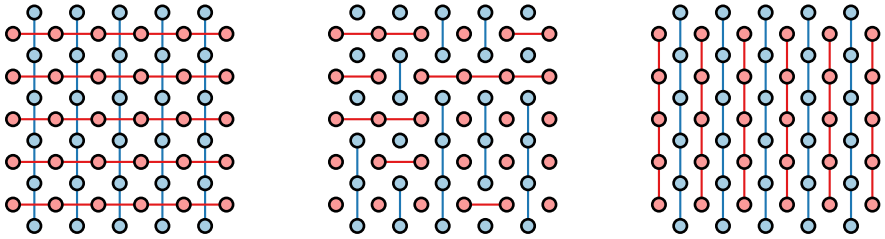


Figure 5 A lower bound on the cardinality of solutions compatible with islands returned by overlap-greedy. Left: overlap-greedy cover; middle: a partition compatible with the overlap-greedy cover; right: an optimal island partition.

Thus, we arrive at the following problem: given m islands $I_1, \ldots, I_m$ obtained by overlap-greedy, find compatible families $\mathcal{I}'_1, \ldots, \mathcal{I}'_m$ with $|\bigcup_k \mathcal{I}'_k|$ minimum. Solving this problem optimally is non-trivial. A natural approach to tackle the problem is to create an arrangement of the islands $I_1, \ldots, I_m$ and extract compatible families from that arrangement. However, two minor issues arise: the faces in the arrangement may not be convex, and the quality of the solution is not immediately clear as the number of faces in the arrangement may be arbitrarily greater than m. To resolve these issues, we build an arrangement iteratively. Before describing this process, we give a lower bound on the approximation ratio of algorithms that return islands compatible with those returned by overlap-greedy.

▶ **Lemma 2.** *Any algorithm that returns islands compatible with those returned by overlap-greedy has approximation ratio* $\Omega(\max\{Opt_P, \sqrt{n}\})$.

Proof. Let $k \in \mathbb{N}_{\geq 1}$. Consider a problem instance that consists of k evenly spaced vertical blue lines each formed by $k+1$ evenly spaced blue points, and symmetrically k evenly spaced horizontal red lines (Figure 5). The cover returned by overlap-greedy has cardinality $2k$ and is induced by exactly those lines that were just described (Figure 5, left). Any partition that is compatible with the overlap-greedy cover has cardinality at least $2k + k^2$ (Figure 5, middle). Indeed, each intersection between two lines in the overlap-greedy cover forces an additional island in a compatible partition. An optimal island partition has cardinality $\mathrm{Opt_P} = 2k + 1$ and consists of either all horizontal or all vertical islands (Figure 5, right). The number of points $n = O(k^2)$. Thus, the approximation ratio of an algorithm that produces solutions compatible with that of overlap-greedy is at least $\frac{2k+k^2}{2k+1} = \Omega(k) = \Omega(\mathrm{Opt_P}) = \Omega(\sqrt{n}) = \Omega(\max\{\mathrm{Opt_P}, \sqrt{n}\})$. ◀

3.1 Upper Bound

As mentioned earlier, our algorithm for creating a compatible island partition from an island cover works in an iterative manner. Throughout the iterations, we keep track of a restricted planar subdivision, which we call an *island arrangement*, to bound the cardinality of the island partition constructed by the algorithm. To simplify the arguments, we assume all islands in the island cover have cardinality at least three and that no three points are collinear. Then, a compatible island partition can be created from the faces of the island arrangement. See Figure 6 for an overview of the transformation from island cover to island partition.

We now define the notion of an *island arrangement*. In the following, vertices, edges, and faces of a planar subdivision are collectively referred to as *features*.

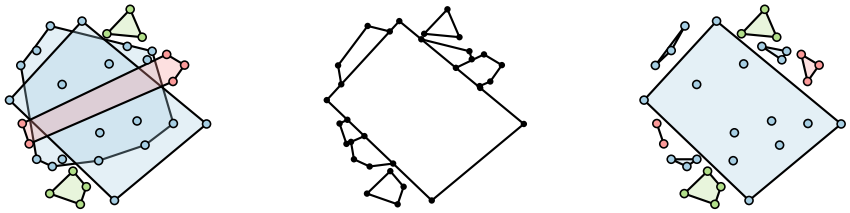


Figure 6 Left: an island cover $I_1, \ldots I_m$ returned by overlap-greedy; middle: an island arrangement of $I_1, \ldots, I_m$; right: an island partition compatible with $I_1, \ldots, I_m$ induced by the arrangement.

▶ **Definition 3** (Island arrangement). An island arrangement of islands $I_1, \ldots, I_i$ is a planar subdivision with the following additional requirements:

- Bounded faces are convex;
- Every bounded feature is a subset of $\mathcal{CH}(I_j)$ for some $1 \leq j \leq i$;
- For every $1 \leq j \leq i$, $\mathcal{CH}(I_j)$ is covered by bounded features.

Let $I_1, \ldots, I_m$ be the islands chosen by overlap-greedy for some set of points S. Let $U_i = I_i \setminus \bigcup_{j<i} I_j$ be the set of uncovered points island I_i covers. Because islands are chosen greedily by overlap-greedy, they satisfy $|U_i| \geq |U_{i+1}|$ for $i \in \{1, \ldots, m-1\}$ and for all i island I_i is such that $|U_i|$ is maximum. By using these properties, the following lemma can be proven to hold for islands $I_1, \ldots, I_m$.

▶ **Lemma 4.** *Let δ denote the boundary operator on sets. For distinct $1 \leq i < j \leq m$, the number of intersections between $\delta(\mathcal{CH}(I_i))$ and $\delta(\mathcal{CH}(I_j))$ is at most $2Opt_P$.*

Using this lemma we can prove that an island arrangement of $I_1, \ldots, I_{i-1}$ with $1 \leq i \leq m$ can be modified into an island arrangement of $I_1, \ldots, I_i$ such that the increase in the number of faces is bounded in terms of i and $\mathrm{Opt_P}$. We call this modification an *augmentation* of the arrangement. The following lemma makes one face for the new island I_i and modifies any existing features to make room. We refer to this as a *bold augmentation* of the arrangement.

▶ **Lemma 5** (Bold augmentation). *Given an island arrangement A of $I_1, \ldots, I_{i-1}$ with f faces, there exists an island arrangement A' of $I_1, \ldots, I_i$ with at most $f + 2Opt_P \cdot (i-1)$ faces such that there is exactly one face whose closure equals $\mathcal{CH}(I_i)$.*

By repeatedly applying Lemma 5, an island cover returned by overlap-greedy can be transformed into a compatible island partition. Let *bold overlap-greedy* be the algorithm that first runs overlap-greedy to obtain islands $I_1, \ldots, I_m$, then repeatedly applies the bold augmentation step to create an island arrangement A of $I_1, \ldots, I_m$, and finally extracts an island partition from the faces of A. The following bound holds on its approximation ratio.

▶ **Corollary 6.** *Bold overlap-greedy has approximation ratio $O(Opt_P^2 \log^2 n)$.*

4 Line-Greedy

In this section, we explore the relation between our problem and that of separating colors with the minimum number of lines. In particular, we show that greedily chosen separating lines induce an $O(\mathrm{Opt_P} \log^2 n)$-approximation to the minimum-cardinality island partition.

A set of lines L *separates* a set of colored points S if each face in the arrangement $\mathcal{A}(L)$ is monochromatic. The problem of finding the minimum-cardinality set of such separating lines is W[1]-hard with the parameter being the solution size [4]. Furthermore, the problem

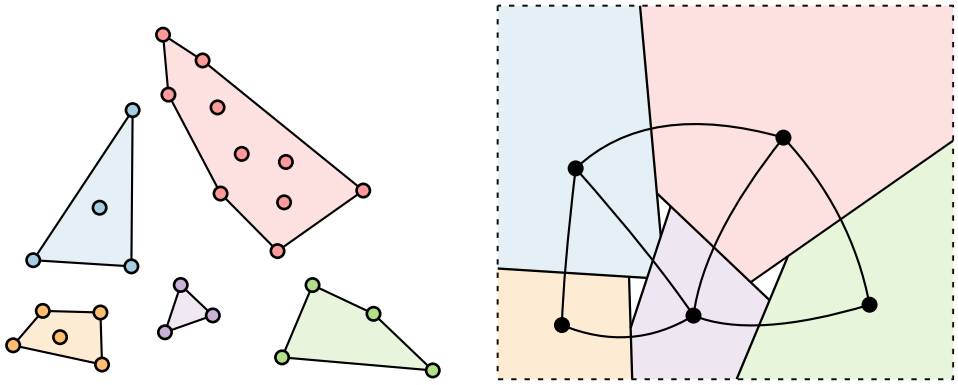


Figure 7 Left: islands; right: expanded islands and their contact graph.

is NP-hard [14] and APX-hard [6], even when allowing only axis-parallel lines. The problem can be viewed as a set cover problem where lines are used to cover line segments between pairs of points of different color. Thus, the corresponding greedy algorithm, which we refer to as *line-greedy*, yields a $O(\log n)$-approximation [11, 13]. Line-greedy can be implemented to run in $O(k\mathrm{Opt}_{\mathrm{L}} n^2 \log n)$ time [11], where $\mathrm{Opt}_{\mathrm{L}}$ is the optimal number of lines and k is the number of colors of the input set.

If L separates S, then the faces of the arrangement $\mathcal{A}(L)$ induce a partition of S into $O(|L|^2)$ islands. Conversely, an island partition $\mathcal{P}$ of S, with $|\mathcal{P}| \geq 3$, induces a set of $O(|\mathcal{P}|)$ lines that separates S. This can be shown using a construction by Edelsbrunner, Robison, and Shen [9]. We sketch their construction, adapted slightly for our use; see their paper for details and proofs. Circumscribe a rectangle around all the polygons—the convex hulls of the islands in $\mathcal{P}$. Grow the polygons, by moving their sides, until they are maximal. Extend each shared polygon side to obtain a set of lines. This set of lines separates the input points. Furthermore, each line corresponds to an edge of the contact graph of the expanded polygons (Figure 7). Because the contact graph is planar, there are at most $3|\mathcal{P}| - 6$ lines, yielding the desired result. While the exact running time of the construction is unclear, it is clearly polynomial. Pocchiola and Vegter [15] provide an alternative construction that makes use of a pseudo-triangulation of the polygons. Their algorithm runs in $O(n + |\mathcal{P}| \log n)$ time.

Thus, an optimal island partition induces an $O(\mathrm{Opt}_{\mathrm{L}})$-approximation to the optimal set of separating lines. Conversely, an optimal set of separating lines induces an $O(\mathrm{Opt}_{\mathrm{P}})$-approximation to the optimal island partition. There is an analogous relation between approximation algorithms of the two problems. In particular, we have the following result.

▶ **Lemma 7.** *Line-greedy induces an $O(Opt_P \log^2 n)$-approximation to the minimum-cardinality island partition.*

For a lower bound instance, place points in a square grid of $k = 2^\ell$ rows and columns and color them alternatingly as in a checkerboard. In addition, place points on the corners of thin axis-parallel rectangles on the sides of the grid to encourage the line-greedy algorithm to use axis-parallel lines (Figure 8). We suspect that for any $\ell \geq 1$ line-greedy returns horizontal and vertical separating lines that separate the rows and columns of the grid as shown on the left in Figure 8. However, a formal proof eludes us. If this were true, then the island partition induced by the line-greedy solution would have cardinality $\Omega(k^2)$. Because an island partition of cardinality $O(k)$ exists (Figure 8, right), this would result in an $\Omega(\sqrt{n}) = \Omega(\mathrm{Opt}_{\mathrm{P}})$ lower bound on the approximation that is attained by an island partition induced by line-greedy.

Figure 8 The figure shows an idea of a lower bound on the approximation that is attained by an island partition induced by line-greedy.

References

1 Pankaj K. Agarwal and Subhash Suri. Surface approximation and geometric partitions. *SIAM Journal on Computing*, 27(4):1016–1035, 1998. doi:10.1137/S0097539794269801.

2 Crevel Bautista-Santiago, José Miguel Díaz-Báñez, Dolores Lara, Pablo Pérez-Lantero, Jorge Urrutia, and Inmaculada Ventura. Computing optimal islands. *Operations Research Letters*, 39(4):246–251, 2011. doi:10.1016/j.orl.2011.04.008.

3 Sergey Bereg, José Miguel Díaz-Báñez, Dolores Lara, Pablo Pérez-Lantero, Carlos Seara, and Jorge Urrutia. On the coarseness of bicolored point sets. *Computational Geometry*, 46(1):65–77, 2013. doi:10.1016/J.COMGEO.2012.04.003.

4 Edouard Bonnet, Panos Giannopoulos, and Michael Lampis. On the parameterized complexity of red-blue points separation. *Journal of Computational Geometry*, 10(1):181–206, 2019. doi:10.20382/jocg.v10i1a7.

5 Sergio Cabello, José Miguel Díaz-Báñez, and Pablo Pérez-Lantero. Covering a bichromatic point set with two disjoint monochromatic disks. *Computational Geometry*, 46(3):203–212, 2013. doi:10.1016/j.comgeo.2012.06.002.

6 Gruia Călinescu, Adrian Dumitrescu, Howard J. Karloff, and Peng-Jun Wan. Separating points by axis-parallel lines. *International Journal of Computational Geometry & Applications*, 15(6):575–590, 2005. doi:10.1142/S0218195905001865.

7 José Miguel Díaz-Báñez, Ruy Fabila Monroy, Pablo Pérez-Lantero, and Inmaculada Ventura. New results on the coarseness of bicolored point sets. *Information Processing Letters*, 123:1–7, 2017. doi:10.1016/J.IPL.2017.02.007.

8 Adrian Dumitrescu and János Pach. Partitioning colored point sets into monochromatic parts. *International Journal of Computational Geometry & Applications*, 12(5):401–412, 2002. doi:10.1142/S0218195902000943.

9 Herbert Edelsbrunner, Arch D. Robison, and Xiaojun Shen. Covering convex sets with non-overlapping polygons. *Discrete Mathematics*, 81(2):153–164, 1990. doi:10.1016/0012-365X(90)90147-A.

10 Paul Fischer. Sequential and parallel algorithms for finding a maximum convex polygon. *Computational Geometry*, 7:187–200, 1997. doi:10.1016/0925-7721(95)00035-6.

11 Hossein Jowhari and Mohsen Rezapour. Monochromatic partitioning of colored points by lines. *Information Processing Letters*, 182:106402, 2023. doi:10.1016/j.ipl.2023.106402.

12 Mikio Kano and Jorge Urrutia. Discrete geometry on colored point sets in the plane - A survey. *Graphs and Combinatorics*, 37(1):1–53, 2021. doi:10.1007/S00373-020-02210-8.

13 Bing Lu, Hongwei Du, Xiaohua Jia, Yinfeng Xu, and Binhai Zhu. On a minimum linear classification problem. *Journal of Global Optimization*, 35(1):103–109, 2006.

14 Nimrod Megiddo. On the complexity of polyhedral separability. *Discrete & Computational Geometry*, 3:325–337, 1988. doi:10.1007/BF02187916.

15 Michel Pocchiola and Gert Vegter. On polygonal covers. *Contemporary Mathematics*, 223:257–268, 1999.

16 Marc van Kreveld, Bettina Speckmann, and Jérôme Urhausen. Diverse partitions of colored points. In *17th Algorithms and Data Structures Symposium (WADS)*, volume 12808 of *Lecture Notes in Computer Science*, pages 641–654, 2021. doi:10.1007/978-3-030-83508-8_46.

Hausdorff morphs with fewer components

Arjen Simons[*1], Marc van Kreveld[2], Wouter Meulemans[*1], and Tim Ophelders[†1,2]

1 **TU Eindhoven**
{a.simons1|w.meulemans}@tue.nl
2 **Utrecht University**
{m.j.vankreveld|t.a.e.ophelders}@uu.nl

Abstract

We introduce two geometry-based morphing techniques that build upon the existing Voronoi and mixed morphs. These morphs are Hausdorff morphs, meaning they linearly interpolate the Hausdorff distance between the two polygons. Our new morphs are also Hausdorff morphs, and additionally reduce the number of components in intermediate shapes. In an experimental analysis we record data on the area, perimeter and total angular change throughout the morph, and the number of holes and components. Our new morphs perform better in these aspects than their original counterparts and the new component-reduced mixed morph also appears to introduce fewer visual artifacts in the intermediate polygons.

1 Introduction

Shape morphing, also called shape interpolation, is the process of gradually transforming a source shape to a target shape over time. Good morphs produce intermediate shapes that preserve the input shapes' appearances.

We focus on abstract morphing between 2D shapes. Abstract morphs do not concern themselves with (semantic) reasons to transform certain parts of the source shape to the target shape. This type of morphing can be used to morph between complex shapes that do not have any clear correspondence.

To capture a morph is gradual, we may use distance measures between shapes: ideally, the distance to the source shape should linearly increase, while the distance to the target shape should linearly decrease. A morph is a *Hausdorff morph*, when it satisfies this property using the Hausdorff metric [5]. The *dilation morph* [5] is such a Hausdorff morph, though the resulting intermediate shapes tend to lack characteristic features of either input shapes; see Figure 1. The *Voronoi morph* [3] is also a Hausdorff morph; compared to the dilation morph, it greatly reduces the area of intermediate shapes and also retains more characteristic features. This morph does however add superfluous components and noise: extra details that are not present in the input shapes; see Figure 1. The above two morphs combine into a *mixed morph* [3], which reduces, but does not eliminate these issues. We introduce the *Component-Reduced Voronoi morph* (CRV morph): a Hausdorff morph based on improving the Voronoi morph by reducing the number of extra components in intermediate shapes. We also describe the *Mixed Component-Reduced Voronoi morph* (MCRV morph), by combining our CRV morph with the dilation morph, resulting in intermediate shapes with considerably less noise and fewer superfluous components.

[*] A. Simons and W. Meulemans are (partially) supported by the Dutch Research Council (NWO) under project number VI.Vidi.223.137.
[†] T. Ophelders is (partially) supported by the Dutch Research Council (NWO) under project no. VI.Veni.212.260.

40th European Workshop on Computational Geometry, Ioannina, Greece, March 13–15, 2024.
This is an extended abstract of a presentation given at EuroCG'24. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

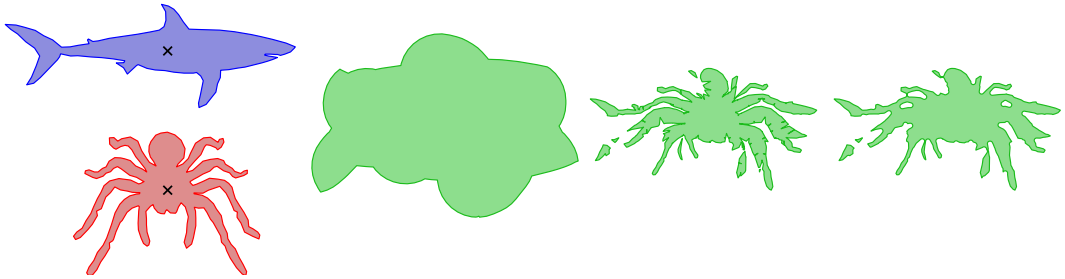


Figure 1 From left to right: two input shapes, with the origin of the plane indicated with a cross; intermediate shapes of the dilation [5], Voronoi [3] and mixed [3] morphs at $\alpha = 1/2$.

1.1 Preliminaries

Hausdorff distance. For two non-empty sets A and B, the Hausdorff distance is defined as

$$d_H(A, B) := \max\left(\sup_{a\in A}\inf_{b\in B} d(a, b),\ \sup_{b\in B}\inf_{a\in A} d(b, a)\right)$$

where d denotes the Euclidean distance. $d_H(A, B)$ is a bottleneck metric that can be described as the largest distance from all points in A and B to their closest point on the other shape.

Morphs. We regard morphs as a function from the interval $[0, 1]$. The parameter α operating on this interval can be viewed as time. The function operates on two shapes A and B, and outputs another shape C_α, where $C_0 = A$ and $C_1 = B$. The output shape is referred to as an *intermediate shape*.

Assume we are given two shapes A and B in the plane that is scaled so that $d_H(A, B) = 1$. A morph that outputs C_α is called a *Hausdorff morph* if it satisfies the *Hausdorff property*: $d_H(A, C_\alpha) = \alpha$ and $d_H(B, C_\alpha) = (1 - \alpha)$, for all $\alpha \in [0, 1]$. For shapes that do not have a Hausdorff distance of 1, we can easily scale the plane uniformly to achieve a Hausdorff distance of 1; as the considered morphs are scale-invariant, this does not affect results.

We define a *component* of a shape S, as a disjoint non-empty subset, such that two components of S are always positively separated. S is equal to the union of all its components.

Dilation morph. The *dilation morph* [5] is defined as

$$S_\alpha(A, B) := (A \oplus D_\alpha) \cap (B \oplus D_{1-\alpha}),$$

where $\oplus$ denotes the Minkowski sum defined as $\{a + b \mid a \in A, b \in B\}$, and D_α is a disc of radius α. This operation is also called *dilation*. This morph produces the maximal shape to support the Hausdorff property. Therefore, any Hausdorff morph is a subset of S_α.

Voronoi morph. The *Voronoi morph* [3] moves points in the source shape to their closest point on the target shape by a fraction of α and moves every point in the target shape to their closest point in the source shape by a fraction of $1 - \alpha$. The union of these two sets results in the intermediate shape at a given α. Formally, it is defined as

$$T_\alpha(A, B) := \{a + \alpha(c(a, B) - a) \mid a \in A\} \cup \{b + (1 - \alpha)(c(b, A) - b) \mid b \in B\},$$

where $c(p, X)$ denotes the point on a shape X that is closest to a point p. If a point is equidistant to multiple points in the other shape, all options are included. To compute

this morph, the Voronoi diagram $V(A)$ of the vertices, open edges the polygonal region of A, partitions B into regions; the Voronoi cells of $V(A)$. *Slices* of B are defined as disjoint non-empty subsets of B, such that two slices in one region of the partitioning by $V(A)$ are always positively separated. Note that two slices in adjacent regions may not be positively separated. Symmetrically, A is partitioned into slices by $V(B)$.

Mixed morph. The *mixed morph* [3] is defined as

$$M_{\alpha,\varphi}(A,B) := ((T_\alpha(A,B) \oplus D_\varphi) \ominus D_\varphi) \cap S_\alpha,$$

where $\ominus$ denotes the Minkowski difference, defined as $A \ominus B := (A^c \oplus B)^c$, where A^c is the complement of A. It first *dilates* T_α by taking the Minkowski sum with a small disc. After that, it *erodes* the shape by taking the Minkowski difference with a disc of the same radius, causing small gaps and holes to close while keeping the rest of the shape intact. To make sure it is also a Hausdorff morph, the intersection with S_α is taken.

2 Component-Reduced Voronoi Morph

We introduce the *Component-Reduced Voronoi morph* (CRV morph), which benefits from the advantages of the Voronoi morph, while reducing the number of components. This morph identifies which slices converge to extra components in the Voronoi morph, and tries to move them along with *neighboring* slices. Two slices of a shape S are neighboring if they are not positively separated. Every slice in the Voronoi morph has a *target*. This target is the site of the Voronoi cell by which the slice is partitioned. This site can be a vertex, edge or polygonal region of the other shape. Every slice is scaled and translated towards its target during the morph. In the CRV morph, the target of a slice can be set to the target of a neighboring slice. In that case, the slice is *redirected*. When two neighboring slices have the same target due to redirection, they will always be part of the same component during the morph. We call the targets of redirected slices *alternative targets*. Slices cannot always move along with a neighboring slice; the alternative target has to be valid. An alternative target is valid when the Hausdorff distance between the slice and target is smaller than $d_H(A,B)$, and it satisfies one of the following conditions: (1) it is the primary target of a neighboring slice that is part of a larger component, or (2) it is the alternative target of a neighboring slice that is redirected along with a larger component.

If a slice has a valid alternative target, it will be redirected, making the alternative target the new target of the slice. When a slice has multiple valid alternative targets, it will be redirected to the alternative target to which the directed Hausdorff distance is smallest. Assuming the final targets are given, we formally define our new morph U_α as:

$$U_\alpha(A,B) := \{a + \alpha(c(a,t_s) - a) \mid a \in A, t_s \in B\} \cup \{b + (1-\alpha)(c(b,t_s) - b) \mid b \in B, t_s \in A\},$$

where $c(p,t_s)$ is the closest point on the final target t_s of the slice in which point p is located. If a slice has no valid alternative targets, its target remains the closest point on B. Note that if no slices are redirected, $U_\alpha(A,B) = T_\alpha(A,B)$.

Alternative targets are valid only if their Hausdorff distance is smaller than that of the two input shapes. This means that redirected targets never determine the Hausdorff distance between an intermediate shape and the input shapes. Therefore, U_α is a Hausdorff morph:

▶ **Theorem 1.** *Let A and B be two compact sets in the plane with $d_H(A,B) = 1$. Then for any $0 \le \alpha \le 1$, we have $d_H(A,U_\alpha) = \alpha$ and $d_H(B,U_\alpha) = 1-\alpha$.*

2.1 Mixed Component-Reduced Voronoi morph

We also define the *Mixed Component-Reduced Voronoi morph* (MCRV morph) $U^m_{\alpha,\varphi}$, to reduce noise on the boundary of intermediate shapes in the CRV morph as

$$U^m_{\alpha,\varphi}(A,B) := ((U_\alpha(A,B) \oplus D_\varphi) \ominus D_\varphi) \cap S_\alpha,$$

where $\ominus$ denotes the Minkowski difference, defined as $A \ominus B := (A^c \oplus B)^c$, where A^c is the complement of A. D_φ denotes a disc of radius φ. This means that $U^m_{\alpha,0} = U_\alpha$. To make sure $U^m_{\alpha,\varphi}$ is a Hausdorff morph, the intersection with S_α is taken.

2.2 Algorithm

The algorithm to compute the CRV morph uses the algorithm to compute the Voronoi morph as the first step. To compute U_α we assume A and B are (sets of) solid polygons that may contain holes. The basic algorithm on these input sets works as follows:

1. Compute T_α at $\alpha = 1/2$ as described in de Kogel et al. [3]. This results in a set of slices, with each slice transformed and scaled halfway to their target.
2. Determine which slices belong to which components in the halfway Voronoi morph for both input shapes separately.
3. For each input shape construct a graph G where each slice is a vertex and two neighboring slices are connected by an edge. For each slice, determine the closest neighboring component in G, using any graph searching algorithm.
4. For both shapes A and B, sort all slices separately based on two ascending sort keys. The primary sort key is the area of the component they belong to and the secondary key is the shortest-path distance from the slice itself in G to the closest slice in G that is part of an adjacent component in the initial shape.
5. For each slice s, in order of the previously sorted slices, determine which neighbors in the original shape are valid alternative targets based on the described criteria. The valid alternative target that is closest, in terms of the directed Hausdorff distance from slice s to the target, will be set as the new target. If slice s has no valid alternative targets, the primary target remains. If slice s is redirected, it is marked a follower of its neighboring slice l that s is redirected along with. If l is redirected, the target of slice s is set to the new target of l, if the Hausdorff distance allows for it. If the Hausdorff distance does not allow s to also be redirected again, the target of s slice is reset to its primary target.
6. Each slice in A is scaled and translated to its target in B. If the target is an interior component of the other shape, the slice will be stationary throughout the morph. If the target is a vertex, the slice will be uniformly scaled towards that vertex by a fraction of α. If the target is an edge, the slice will scale perpendicular to the supporting line of that edge by a factor of α. Slices in B are scaled towards their targets in A in the same manner, except that they are scaled by a factor of $1 - \alpha$.
7. Combine slices of A and B into one multipolygon.

We sort components from small to large in order to ensure that larger components can move along with smaller components if that smaller component is redirected to the target of an even larger component. Within each component, slices are sorted based on the distance to a neighboring component in terms of slices, to allow slices with no directly neighboring components to move along with neighboring slices that can be redirected.

$U^m_{\alpha,\varphi}$ can simply be computed when U_α is constructed by dilating and eroding U_α with a disc of radius φ, and intersecting the result with S_α.

The Voronoi morph can construct any intermediate shape in $O(n^2 \log n)$ time [3]. Our morph additionally relies on a sorting algorithm and a shortest path algorithm on a graph with n edges: both can be found in standard books [2]. The all-pairs shortest path in graph can be computed in $O(V^3)$. We have at most $O(n^2)$ slices, resulting from the intersections between a polygon and the Voronoi diagram of the other polygon. Therefore, an intermediate shape can be constructed in $O(n^6)$ time. In practice such a worst case scenario is very unlikely to occur. For the shapes used in our experiments (see Section 3), computing one intermediate shape took approximately five seconds on commodity hardware.

3 Experiments

We compare the Voronoi, CRV, mixed and MCRV morphs experimentally on two data sets: animal outlines from [1] and country outlines from [4]. The animal dataset contains a collection of nine outlines of animals, averaging 143 vertices, that all comprise one component. The country dataset contains 13 country outlines, averaging 1548 vertices and 12 components. We compute the four morphs for all animal pairs and all country pairs from these sets. The input shapes are scaled to have the same area and translated to have a common centroid.

For all experiments we record the perimeter, area, total angular change (sum of all enclosed boundary loops' angular changes), and the number of components and holes. We record this for α values starting at zero and increasing in steps of $1/8$. For the perimeter and area we record the ratio between the value and a linear interpolation. The number of components and holes are discrete and directly recorded at every value of α except for zero and one.

4 Results

Figures 2 and 6 show visual examples of the four morphing techniques.

The CRV morph can effectively reduce the number of extra components created in the Voronoi morph. Table 1 shows the average number of components created in the animal data set during the CRV morph to be more than three times less than that in the Voronoi morph. In Figure 2 we see that the CRV morph removes all extra components created in the Voronoi shark-spider morph. The mixed morph can also reduce the number of components, but as indicated by label 2 in Figure 2, this type of component reduction can still result in superfluous details that are not present in the CRV morph.

In terms of area, perimeter and total angular change, Figures 3, 4 and 5 indicate that the CRV and MCRV morphs perform slightly better or similar to their Voronoi and mixed morphs counterparts. In terms of holes the CRV morph performs worse than the Voronoi morph, but most of these holes are resolved in the MCRV morph which only has slightly more holes than the Mixed morph. The number of components and holes are recorded as the average over all α values except 0 and 1.

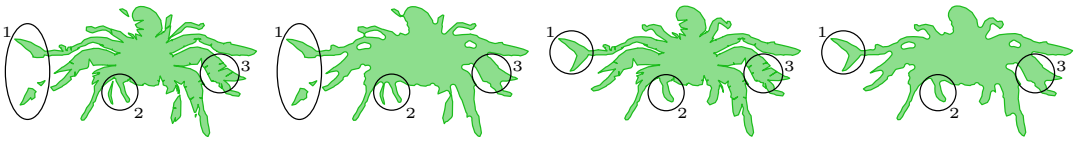


Figure 2 Intermediate shapes at $\alpha = 1/2$ when morphing between the outlines of a shark and spider shown in Figure 1. The images show the Voronoi, mixed, CRV and MCRV morphs from left to right. The labeled ellipses are highlighted parts of the shape in which details differ.

	Voronoi		Mixed		CRV		Mixed CRV	
Category	Mean	SD	Mean	SD	Mean	SD	Mean	SD
Components	11.262	4.686	4.210	2.480	3.710	2.263	3.016	2.024
Holes	0.282	0.589	0.337	0.663	1.690	1.787	0.508	0.770

Table 1 Component and hole count distributions for each morphing method for all tested values of α except 0 and 1. Only the animal data set is included, as these shapes only have one component.

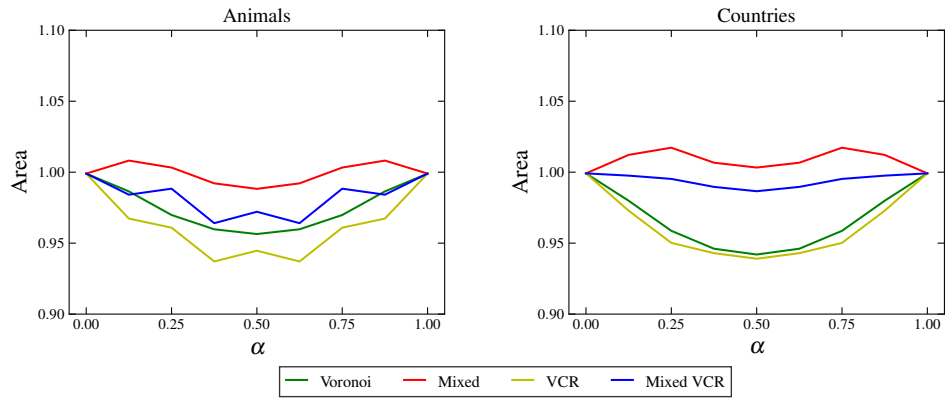


Figure 3 Normalized average area for the animals and countries data set experiments.

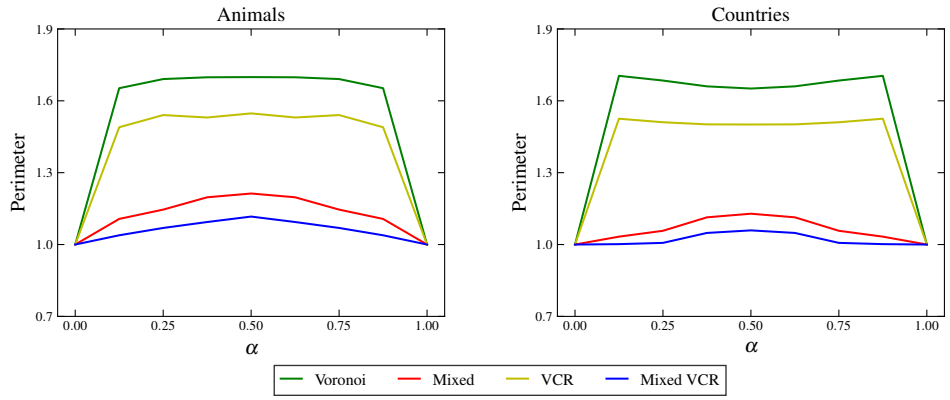


Figure 4 Normalized average perimeter for the animals and countries data set experiments.

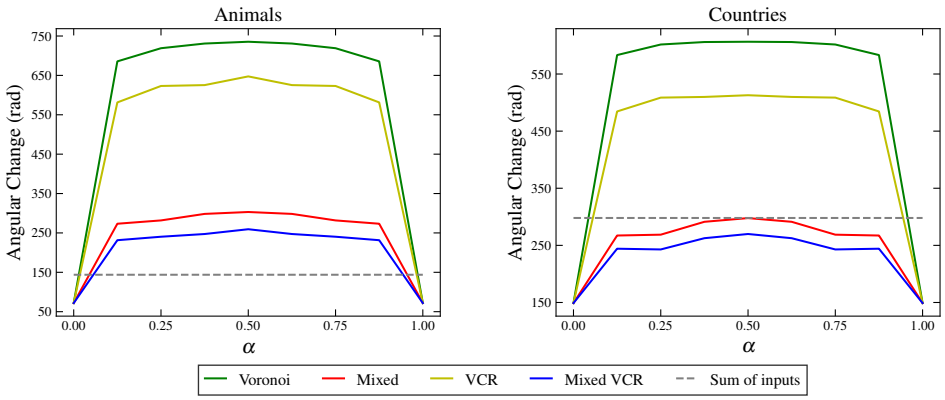


Figure 5 Average total angular change for the animals and countries data set experiments.

Figure 6 Intermediate shapes for $\alpha \in \{0, \frac{1}{4}, \frac{1}{2}, \frac{3}{4}, 1\}$ when morphing between outlines of Germany and Italy. The columns show the Voronoi, mixed, CRV and MCRV morphs from left to right.

References

1 Quirijn W. Bouts, Irina Kostitsyna, Marc van Kreveld, Wouter Meulemans, Willem Sonke, and Kevin Verbeek. Mapping Polygons to the Grid with Small Hausdorff and Fréchet Distance. In *Proc. 24th Annual European Symposium on Algorithms (ESA 2016)*, volume 57 of *LIPIcs*, pages 22:1–22:16, 2016. doi:10.4230/LIPIcs.ESA.2016.22.

2 Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. The MIT Press, 3rd edition, 2009.

3 Lex de Kogel, Marc van Kreveld, and Jordi L. Vermeulen. Abstract Morphing Using the Hausdorff Distance and Voronoi Diagrams. In *Proc. 30th Annual European Symposium on Algorithms (ESA 2022)*, volume 244 of *LIPIcs*, pages 74:1–74:16, 2022. doi:10.4230/LIPIcs.ESA.2022.74.

4 Bjørn Sandvik. Thematic Mapping World Borders Dataset, 2009. URL: https://67-20-120-230.unifiedlayer.com/downloads/world_borders.php.

5 Marc van Kreveld, Tillmann Miltzow, Tim Ophelders, Willem Sonke, and Jordi L. Vermeulen. Between shapes, using the Hausdorff distance. *Computational Geometry*, 100:101817, 2022.

On Totally-Concave Polyominoes

Gill Barequet[1], Noga Keren[1], Johann Peters[2], and Adi Rivkin[1]

1 **Dept. of Computer Science, Techion—Israel Inst. of Technology, Haifa, Israel**
`barequet@cs.technion.ac.il`, [noga.keren,adi.rivkin]@campus.technion.ac.il
2 **Dept. of Mathematics, Univ. of Waterloo, ON, Canada**
`j8peters@uwaterloo.ca`

Abstract

A polyomino is an edge-connected set of cells on the square lattice. Every row or column of a *totally-concave* (TC) polyomino consists of more than one sequence of consecutive cells of the polyomino. We show that the minimum area (number of cells) of a TC polyomino is 21 cells. We also suggest, implement, and run an efficient algorithm for counting TC polyominoes. Finally, we prove that the associated sequence $(\kappa(n))$ has a finite growth constant λ_κ, and prove the lower bound $\lambda_\kappa > 2.4474$.

1 Introduction

A *polyomino* of area n is a connected set of n cells on the square lattice $\mathbb{Z}^2$, where connectivity is through edges. Two (fixed) polyominoes are considered equivalent if one can be transformed into the other by a translation.

Counting polyominoes is a long-standing problem in discrete geometry, originating in statistical physics in the context of percolation processes [8] and popularized in Golomb's pioneering book [9] and by M. Gardner's columns in *Scientific American*; The sequence $A(n)$, which lists the number of fixed polyominoes, is currently known up to $n = 70$ [1].

The growth constant of polyominoes has also attracted much attention in the literature. Klarner [13] showed that the limit (a.k.a. *Klarner's constant*) $\lambda := \lim_{n\to\infty} \sqrt[n]{A(n)}$ exists. The convergence of $A(n+1)/A(n)$ to λ, as $n \to \infty$, was proved only three decades later by Madras [14]. The best-known lower [4] and upper [5] bounds on λ are 4.0025 and 4.5252, respectively. By applying numerical methods to the known values of $A(n)$, it is widely believed that $\lambda \approx 4.06$, and the currently best estimate of λ is 4.0625696 ± 0.0000005 [11]. (Based on the new counts of $A(n)$ till $n = 70$, a better estimate is 4.06256912(2) [12].)

In a *totally-concave* (TC) polyomino, each row and column consists of at least *two* maximal continuous sequences of cells, as is shown in Figure 1. It is hinted in Ref. [7, §14, p. 369, problem 14.5.4] that the minimum possible area of a TC polyomino is 21. Let $\kappa(n)$ be the number of TC polyominoes of size (area) n. An algorithm for computing $\kappa(n)$, for a given n, is also sought as an open problem [Ibid., problem 14.5.5]. Among other results, we settle the minimality conjecture and suggest an efficient algorithm.

In this paper, we investigate a few problems related to TC polyominoes. We prove that the minimum possible area of such a polyomino is indeed 21; suggest an efficient algorithm for counting TC polyominoes, and report the values of $\kappa(n)$ till $n = 35$; show that the seqeunce $(\kappa(n))$ has a growth constant λ_κ; and finally, prove that $\lambda_\kappa > 2.4474$.

40th European Workshop on Computational Geometry, Ioannina, Greece, March 13–15, 2024.
This is an extended abstract of a presentation given at EuroCG'24. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

Symbolic Eatable With marked edges

(a) Area 21 (b) Area 24

Figure 1 TC polyominoes of various areas and flavors. The symbolic representation in (b) distinguishes between *hidden* edges (green), *inside* edges (blue), and *outside* edges (red).

2 Minimum Area

▶ **Theorem 2.1.** *The minimum area of a TC polyomino is 21.*

The proof of this theorem follows a necessity-sufficiency format. Necessity is shown by deducing upper and lower bounds on the area of TC polyominoes in $m \times \ell$ bounding boxes; These bounds contradict each other for areas less than 21. Sufficiency is evident by example.

Proof. A lower bound on the area of a TC polyomino within an $m \times \ell$ bounding box is achieved by partitioning the edges of such a polyomino into *hidden*, *outside*, and *inside* edges, as shown in Figure 1(b). The top (resp., right/bottom/left) edge of a cell c is *hidden* if there is a cell of the polyomino immediately above (resp., to the right of/below/to the left of) c. An edge is *outside* if it is not facing any other edge. An *inside* edge is an edge facing another edge, but not immediately, that is, with a gap of at least one cell. Consider a TC polyomino. Denote by n its area, and by h, o, and i the number of hidden, outside, and inside edges, respectively, of the polyomino. For example, by these definitions, the "U-pentomino" () has $i = 2$, $o = 10$, and $h = 8$. For the area-24 TC-polyomino depicted in Figure 1(b), we have $i = 24$, $o = 24$, and $h = 46$. By duplicity of inside and outside edges in rows and columns, we have that $o = 2m + 2\ell$ and $i \geq 2m + 2\ell$. We also have that $h \geq 2n - 2$ since the polyomino is connected and, hence, it must include at least $n-1$ cell adjacencies. Since $h + o + i = 4n$, we have that $n \geq 2m + 2\ell - 1$.

For an upper bound on n, we may assume without loss of generality that $m \leq \ell$. Then, a TC polyomino within an $m \times \ell$ bounding box must be missing at least one cell from each of the ℓ columns, none of which is in the top or bottom row (for guaranteeing concavity of the columns), as well as at least two further cells, one in the top and one in the bottom row (for guaranteeing concavity of these rows). Therefore, $n \leq m\ell - \ell - 2$.

Altogether, we have that $2m + 2\ell - 1 \leq n \leq m\ell - \ell - 2$, with $m \leq \ell$. A simple case analysis shows that the smallest n satisfying these constraints is 21, with $m = 5$ and $\ell = 6$.

Hence, $n \geq 21$ is a necessary condition for a TC polyomino. On the other hand, the existence of a TC polyomino of area 21 is evident by Fig. 1(a). This completes the proof. ◀

This result was verified by our TC-polyomino counting programs (see Section 3). Figure 2 shows representatives of the 152 TC polyominoes of area 21. (None of these polyominoes have any symmetries, hence, each of the 19 drawn polyominoes has eight distinct orientations.)

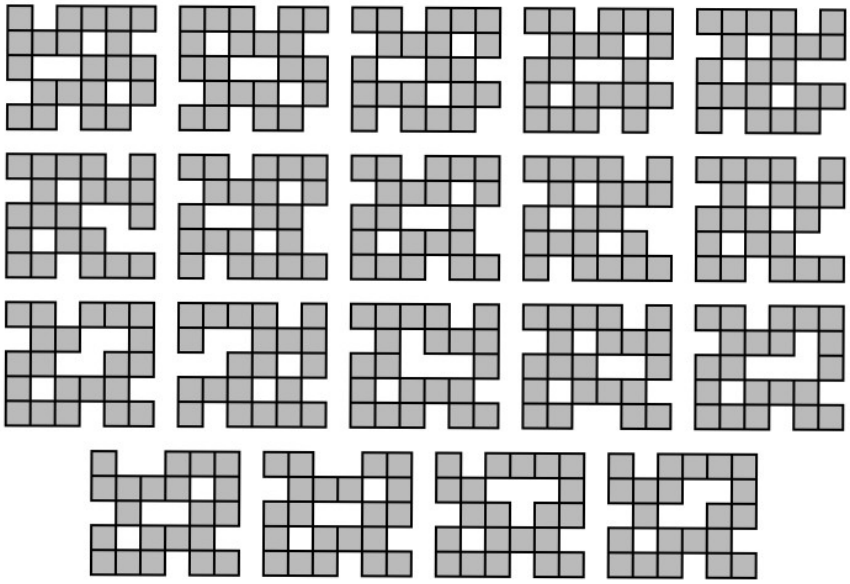


Figure 2 The 19 TC polyominoes of area 21, up to rotation and mirroring.

3 An Efficient Counting Algorithm

3.1 Algorithm

We first implemented a prototype backtracking algorithm for counting TC polyominoes. The program recursively concatenated concave columns to a growing polyomino. A branch of this procedure was abandoned when the area of the polyomino grew too large or if it was no longer possible for it to become connected with the addition of further columns. (This happened when a component of the polyomino became permanently detached.)

We then designed a much more efficient algorithm, based on Jensen's algorithm for counting all polyominoes [10, 11]. In a nutshell, Jensen's algorithm counts polyominoes within horizontal bounding strips of height h, where $1 \leq h \leq \lceil n/2 \rceil$. The algorithm considers column by column from left to right, and cell by cell from top to bottom within each column. At each cell, the algorithm considers either to have it occupied (belonging to the polyomino) or empty (not belonging). At all stages, the algorithm does not keep in memory all polyominoes but all possible *right boundaries* of polyominoes, that is, all combinations of the last h cells considered. The algorithm maintains a database whose entries have keys that are the different *signatures*, where a signature consists of a boundary plus all possible connections between cells on the boundary by cells found to the left of it. In other words, the keys reflect all possible splits of boundary cells into connected components, where the connections are to the left of the boundary. In addition, a signature also includes two bits that indicate whether or not the polyominoes associated with that entry touch the top and/or bottom of the strip. The contents of each entry in the database is statistics of all partially-built polyominoes ("partially" means that polyominoes may still consist of more than one connected component), that is, the *counts* of all polyominoes parameterized by area, having that specific signature. When the currently considered cell is chosen to be occupied, the counts of polyominoes are updated by adding the numbers of fully-built polyominoes, that is, polyominoes that consist of exactly one connected component and touch the top and bottom of the strip.

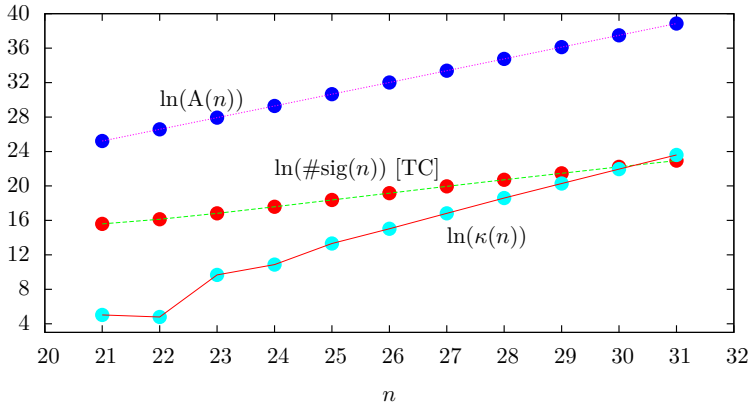


Figure 3 Plots of the number of signatures (while counting TC polyominoes), all poyominoes, and TC polyominoes.

For counting TC polyominoes, we also need to ensure that each column and each row consists of more than one consecutive sequence of cells. This is simple to achieve for columns: At the end of processing a column, we discard from the database all entries that correspond to columns that contain less than two sequences of occupied cells. For rows, we enhance the signatures by splitting each one into at most 4^h subsignatures: For each row, we keep a number as follows: '0' indicates that the first sequence of occupied cells has not been met yet; '1' means that we are in the middle of the first sequence; '2' states that we are between the first and second sequences; and '3' signifies that we have already entered the second sequence. (Once we reach '3,' we do not need to update this indicator any more.) Then, we count only polyominoes with signatures whose line indicators are all '3.' Note that the indicators of the top and bottom rows make the two bits described above redundant.

Jensen's algorithm is efficient in the sense that it's the only known algorithm whose running time, $\tilde{O}(1.732^n)$ [3], is smaller than the total number of polyominoes, $\tilde{\Theta}(\lambda^n)$. (Recall that $\lambda \approx 4.063$.) Our modification splits every signatures into at most $4^{n/2} = 2^n$ subsignatures (in practice, into much less than that), thus, the running time of the modified algorithm is $\tilde{O}(3.464^n)$, which is still much smaller than the total number of polyominoees. Figure 3 plots in a semi-logarithmic scale the number of distinct signatures encountered by the algorithm while computing $\kappa(n)$) (in red circles), together with the number of TC polyominoes (cyan) and the total number of polyominoes (blue), all as functions of n, for $21 \leq n \leq 31$.

3.2 Results

Our prototype program, implemented in Python, computed in 90 hours (elapsed time) $\kappa(n)$ up to $n = 26$ on a PC with a 64-bit system operating an i5-9400F Intel Core CPU at 2.90GHz with 12GB of RAM.

The modified version of Jensen's algorithm was implemented in `C++` and run on a 12th generation Intel(R) i9-12900KF with 128GiB of RAM. Using about 41 hours of CPU, the program computed $\kappa(n)$ up to $n = 35$, obtaining the values reported in Table 1 and agreeing with all values computed by the prototype program.

Table 1 Counts of TC polyominoes.

n	$\kappa(n)$	n	$\kappa(n)$	n	$\kappa(n)$	n	$\kappa(n)$
1–20	0	24	52,306	28	119,309,768	32	88,476,873,440
21	152	25	606,636	29	641,447,812	33	435,921,253,072
22	120	26	3,376,528	30	3,403,173,276	34	2,113,011,155,472
23	15,820	27	20,204,672	31	17,634,751,456	35	10,065,872,407,536

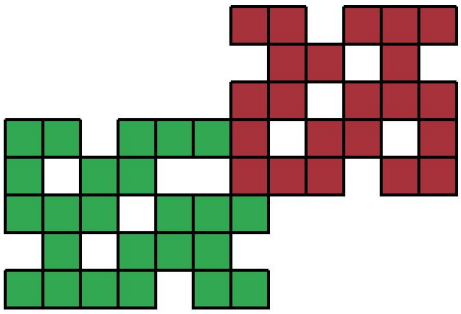


Figure 4 The concatenation of two TC polyominoes is always a TC polyomino.

4 Growth Constant

4.1 Existence

▶ **Definition 4.1.** (lexicographic order) For cells c_1, c_2, we say that $c_1 \prec c_2$ if c_1 lies in a column which is to the left of the column of c_2, *or* if c_1 lies below c_2 in the same column.

▶ **Definition 4.2.** (concatenation) Let P_1, P_2 be two polyominoes, and let c_1 (resp., c_2) be the biggest (resp., smallest) cell of P_1 (resp., P_2). The *concatenation* of P_1 and P_2 is the placement of P_2 relative to P_1, such that c_2 is found immediately on top of c_1.

▶ **Theorem 4.3.** *The limit* $\lambda_\kappa := \lim_{n\to\infty} \sqrt[n]{\kappa(n)}$ *exists and is finite.*

Proof. We follow closely the proof of existence and finiteness of Klarner's constant λ [13]. First, the sequence $\kappa(n)$ is supermultiplicative, that is, $\kappa(n)\kappa(m) \le \kappa(n+m)$ for all $m, n \in \mathbb{N}$. This is justified by a simple concatenation argument. Indeed, all TC polyominoes of area n can be concatenated with all TC polyominoes of area m (see, *e.g.*, Figure 4), yielding distinct TC polyominoes of area $n+m$. Second, there exists a constant $\mu > 0$ for which $\kappa(n) \le \mu^n$ for all $n \in \mathbb{N}$. For example, $\mu = \lambda$, the growth constant of all polyominoes. (This follows immediately from the fact that $\kappa(n) \le A(n) \le \lambda^n$.) By a lemma of Fekete (Klarner cites Ref. [15, p. 852] for similar results), the claim follows. ◀

Remark In fact, it makes more sense (see Section 4.2) to explore $((4\kappa(n))^{1/n})$ instead of $((\kappa(n))^{1/n})$. Figure 5 shows plots of the known values of $(4\kappa(n))^{1/n}$ and $\kappa(n)/\kappa(n-1)$. Surprisingly, the ratio sequence seems empirically to be monotone *decreasing* (except some low-order fluctuations), a property rarely found in other families of polyominoes.

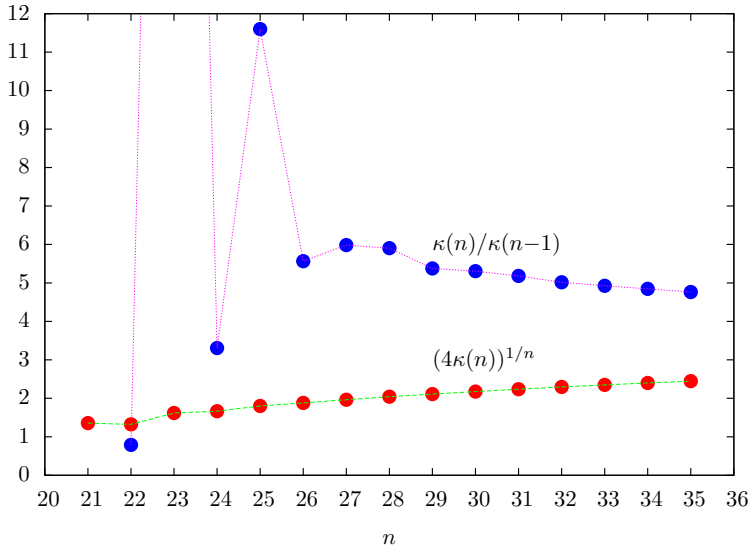


Figure 5 Plots of known values of $(4\kappa(n))^{1/n}$ and $\kappa(n)/\kappa(n-1)$.

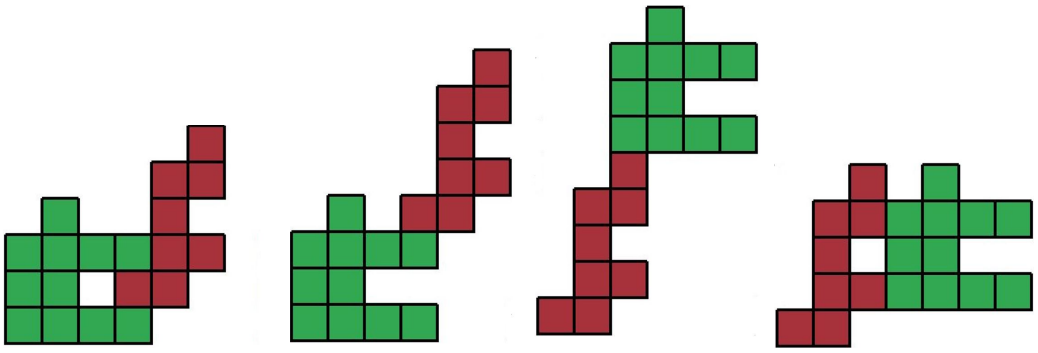


Figure 6 A few compositions of a sample pair of polyominoes.

4.2 A Lower Bound on λ_κ

We now present a computer-assisted proof of a lower bound on λ_κ.

▶ **Definition 4.4.** (composition) A *composition* of two polyominoes is a relative placement of the two polyominoes, such that they touch (edge to edge), possibly in multiple places, but do not overlap.

Figure 6 shows a few compositions of a pair of polyominoes P, Q. Note that some compositions have the property that all cells of P are smaller than all cells of Q (or vice versa), and some compositions do not. It is easy to observe that a composition of two TC polyominoes is not always a TC polyomino.

▶ **Lemma 4.5.** *(A simplified version of Theorem 1(a) in Ref. [2, p. 3]) Assume that the limit* $\mu := \lim_{n\to\infty} \sqrt[n]{Z(n)}$ *exists for a sequence* $(Z(n))$*. Let* $c_1 \neq 0, c_2$ *be some constants. Then, if* $c_1 n^{c_2} Z^2(n) \leq Z(2n)\ \forall n \in \mathbb{N}$*, then* $\sqrt[n]{c_1 (2n)^{c_2} Z(n)} \leq \mu\ \forall n \in \mathbb{N}$.

▶ **Theorem 4.6.** $\lambda_\kappa > 2.4474$.

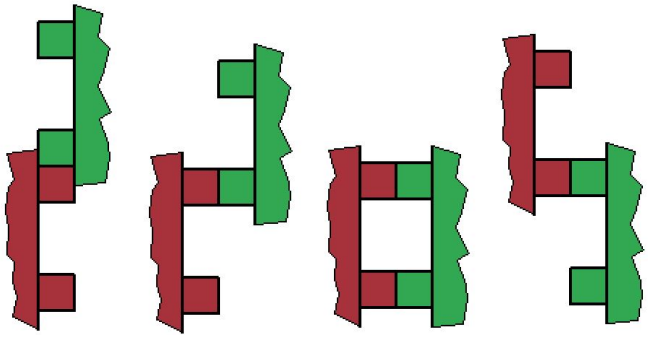


Figure 7 The at least four order-preserving compositions of a pair of TC polyominoes.

Proof. We use a composition argument, using the property that the extreme (rightmost and leftmost) columns of any TC polyomino have at least two cells. This property allows at least four lexicographic compositions of any pair of TC polyominoes P, Q that yield TC polyominoes, that is, compositions in which all cells of P are lexicographically smaller than all cells of Q. It can easily be verified that the minimum number of such compositions is obtained when both the rightmost column of P and the leftmost column of Q contain exactly two cells, with the same vertical gap between them. For such pairs of TC polyominoes, we have the four lexicographic compositions shown in Figure 7.

Consequently, we have that $4(\kappa(n))^2 \leq \kappa(2n)$. Then, Lemma 4.5 implies that any term of the form $(4\kappa(n))^{1/n}$ is a lower bound on λ_κ. Checking the known values of $\kappa(n)$, we see that $n = 35$ provides the best lower bound $\lambda_\kappa \geq (4\kappa(35))^{1/35} > 2.4474$. ◀

References

1 G. Barequet and G. Ben-Shachar, Counting polyominoes, revisited, *SIAM Symp. on Algorithm Engineering and Experiments*, Alexandria, VA, January 2024 (to appear).
2 G. Barequet, G. Ben-Shachar, and M.C. Osegueda, Concatenation arguments and their applications to polyominoes and polycubes, *Computational Geometry: Theory and Applications*, 98 (2021), 12 pp.
3 G. Barequet and M. Moffie, On the complexity of Jensen's algorithm for counting fixed polyominoes, *J. of Discrete Algorithms*, 5 (2007), 348–355.
4 G. Barequet, G. Rote, and M. Shalah, $\lambda > 4$: An improved lower bound on the growth constant of polyominoes, *Comm. of the ACM*, 59 (2016), 88–95.
5 G. Barequet and M. Shalah, Improved upper bounds on the growth constants of polyominoes and polycubes, *Algorithmica*, 84 (2022), 3559–3586.
6 G. Barequet, M. Shalah, and Y. Zheng, An improved lower bound on the growth constant of polyiamonds, *J. of Combinatorial Optimization*, 37 (2019), 424–438.
7 G. Barequet, S.W. Solomon, and D.A. Klarner, Polyominoes, *Handbook of Discrete and Computational Geometry*, 3rd ed. (E. Goodman, J. O'Rourke, and C.D. Tóth, eds.), 359–380. Chapman and Hall/CRC Press, 2017.
8 S.R. Broadbent and J.M. Hammersley, Percolation processes: I. Crystals and mazes, *Proc. Cambridge Philosophical Society*, 53 (1957), 629–641.
9 S.W. Golomb, *Polyominoes*, Princeton University Press, Princeton, NJ, 1965 (2nd ed., 1994).
10 I. Jensen, Enumerations of lattice animals and trees, *J. of Statistical Physics*, 102 (2001), 865–881.
11 I. Jensen, Counting polyominoes: A parallel implementation for cluster computing, *Proc. Int. Conf. on Computational Science*, part III, Melbourne, Australia and St. Petersburg, Russia, *Lecture Notes in Computer Science*, 2659, Springer, 203–212, June 2003.
12 I. Jensen, private communication.
13 D.A. Klarner, Cell growth problems, *Canadian J. of Mathematics*, 19 (1967), 851–863.
14 N. Madras, A pattern theorem for lattice clusters, *Annals of Combinatorics*, 3 (1999), 357–384.
15 G. Pólya and G. Szegő, Aufgaben und Lehrsätze aus der Analysis Bd. Funktionentheorie. Nullstellen. Polynome. Determinanten. Zahlentheorie (Tasks and theorems from analysis: Vol. on Function theory, zeros, polynomials, determinants, number theory), vol. 2, Springer, 1925.

Approximating Simplet Frequency Distribution for Simplicial Complexes*

Hamid Beigy[1], Mohammad Mahini[2], Salman Qadami[3], and Morteza Saghafian[4]

1 **Sharif University of Technology**
beigy@sharif.edu
2 **Sharif University of Technology**
m_mahini@ce.sharif.edu
3 **Amirkabir University of Technology**
salmanqadami@gmail.com
4 **Institute of Science and Technology Austria**
morteza.saghafian@ist.ac.at

Abstract

Simplets, constituting elementary units within simplicial complexes (SCs), serve as foundational elements for the structural analysis of SCs. Previous efforts have focused on the exact count or approximation of simplet count rather than their frequencies, with the latter being more practical in large-scale SCs. This paper enables simplet frequency analysis of SCs by introducing the Simplet Frequency Distribution (SFD) vector. In addition, we present a bound on the sample complexity required for accurately approximating the SFD vector by any uniform sampling-based algorithm. We also present a simple algorithm for this purpose and justify the theoretical bounds with experiments on some random simplicial complexes.

1 Introduction

In a range of disciplines, including biology, geology, and social science, the application of simplicial complexes is frequently employed to extract essential structural insights. *Simplicial Complexes (SCs)* are defined as networks of higher-order that possess the property of downward closure, which makes them suitable for representing higher-order relationships within network-like structures and their geometrical aspects [6, 11, 13]. In particular, SCs are used to study the geometric and combinatorial structure of protein interaction networks [12], epidemic spreading [17], co-authorship relations [26], analyze email communications [15], and investigate the functional and structural organization of the brain [18].

Analyzing network behavior using small network building blocks, commonly known as *motifs*, is common in numerous fields, including biological [1] and social networks [25]. Graphs are great examples where researchers use small building blocks called *graphlets* to understand how networks behave based on local structures [23]. Graphlet analysis has many applications in biological networks [10, 30], and social networks [2, 3]. By considering *simplets* as fundamental elements within simplicial complexes, analogous to graphlets in the context of SCs, we can examine the specific patterns formed by the simplices associated with different

* Work by the second and fourth authors is partially supported by the European Research Council (ERC), grant no. 788183, by the Wittgenstein Prize, Austrian Science Fund (FWF), grant no. Z 342-N31, and by the DFG Collaborative Research Center TRR 109, Austrian Science Fund (FWF), grant no. I 02979-N35.

40th European Workshop on Computational Geometry, Ioannina, Greece, March 13–15, 2024.
This is an extended abstract of a presentation given at EuroCG'24. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

sets of nodes [22]. This approach offers a straightforward way of analyzing complex networks' structural characteristics and their constituent parts.

Approximating Graphlet Count and Distribution. Numerous investigations have delved into the precise enumeration of graphlet types or approximating their frequencies. Several studies, like the ESU and RAGE algorithms, count the precise number of graphlets [29, 20]. Meanwhile, various algorithms like GRAFT, CC have employed sampling techniques to estimate the frequency of graphlets [5, 7, 8, 9]. For instance, Bressan in [7] introduced a random walk based method that preprocesses k-vertex graphlets, and gives a random graphlet in the time complexity of $k^{O(k)} \cdot \log \Delta$, where Δ is the maximum degree in the given graph.

Approximating Simplet Count and Distribution. Preti et al. introduced the concept of simplets, and the FRESCO algorithm that indirectly estimates the quantity of each simplet by utilizing a proxy metric referred to as *support* [21, 22]. B-Exact precisely enumerates up to 4-node configurations through combinatorial techniques [4]. Importantly, each simplet can correspond to zero, one, or more than one configuration. Kim et al. presented SC3, a sampling-based algorithm for approximating simplet count, that utilizes color coding techniques [14]. Thus far, a limited number of dedicated algorithms designed for counting simplets either precisely or approximately. The concept of simplets is relatively novel, and numerous opportunities remain untapped for their application in various contexts.

Contribution. Calculating the exact quantity of each graphlet type or simplet type is frequently prohibitively expensive, and for numerous practical purposes, obtaining an estimated count of various graphlet types and simplet types or approximating their frequency distribution is usually adequate. This paper studies the concept of the *Simplet Frequency Distribution (SFD)* for the first time (to the best of the authors' knowledge), which can be more practical in analyze of large-scale SCs. Alongside this new concept, we present an algorithm to approximating the SFD vector based on uniform sampling of simplets.

More importantly, we present an upper-bound on the sample complexity (number of samples needed) of any approximation algorithm based on a uniform sampling method. By doing this, we aim to enhance our comprehension and analysis of simplicial complexes, mapping them to vector spaces and using this vector for machine learning applications such as classification. In overview, we present the following contributions.

- Defining the concept of the Simplet Frequency Distribution (SFD) vector
- Studying an upper bound on the number of samples we need for every sampling based algorithm for approximating the SFD vector
- Proposing an algorithm for approximating the SFD vector by uniform sampling of simplets

2 Preliminaries

Within this section, we lay out the foundational concepts employed in this paper.

Simplex. A n-simplex is the convex hull of $n+1$ distinct points in n-dimensional space. A face of an n-simplex σ is the convex hull of any non-empty subset S of its vertices.

Simplicial Complex. A simplicial complex $\mathcal{K}$ is a set of simplices that is closed under taking faces, and the non-empty intersection of any two simplices $\sigma, \tau \in \mathcal{K}$ is a face of both σ, τ.

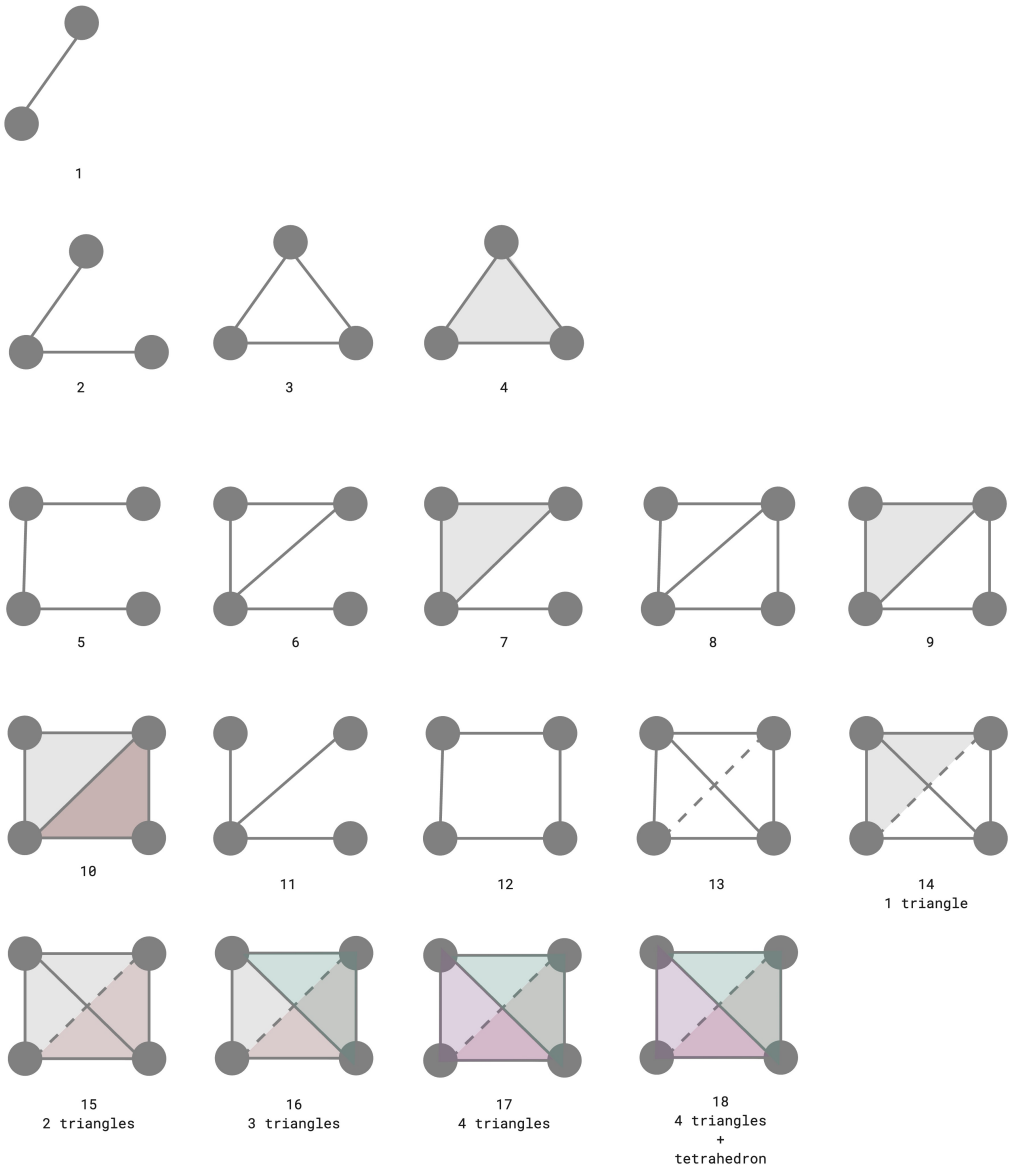


Figure 1 The set of all 18 simplet types with at least two and at most four vertices.

Simplicial complexes provide a combinatorial and topological framework for studying the structure of spaces through simplices, capturing both geometric and connectivity information.

Simplet. *Simplets* are small induced connected sub-complexes of a massive complex that appear at any frequency. A complex H is an induced sub-complex of $\mathcal{K}$ if and only if, for any simplex S in $\mathcal{K}$ whose vertices are a subset of $V(H)$, S should also be in H. So, every simplet can be identified by its vertices, typically regarded as being at least two. A *simplet set* is a set of simplets of a simplicial complex. *Simplet types* are isomorphic classes of simplets. We denote $\mathcal{S}_{\mathcal{K}}(i)$ as a set of all simplets of type i in $\mathcal{K}$, where $1 \leq i \leq N_m$, and N_m is the number of simplet types with at most m vertices. Also, we denote $\mathcal{S}^m_{\mathcal{K}}$ as the set of all simplets in $\mathcal{K}$ with at most m vertices. We assume that m is a constant small number.

Simplet Frequency Distribution. The SFD vector of complex $\mathcal{K}$ characterizes the relative frequencies of various simplets in $\mathcal{K}$. By definition, $|\mathcal{S}_{\mathcal{K}}(i)|$ is the number of simplets of type i in $\mathcal{K}$, where $i \in \{1, \ldots, N_m\}$. The frequency, denoted by $\phi_{\mathcal{K}}(i)$, is obtained by dividing $|\mathcal{S}_{\mathcal{K}}(i)|$ by $\sum_{j=1}^{N_m} |\mathcal{S}_{\mathcal{K}}(j)|$. The vector $(\phi_{\mathcal{K}}(1), \ldots, \phi_{\mathcal{K}}(N_m))$ is called the SFD vector of the $\mathcal{K}$. In Figure 2, we show an SFD vector for two sample SCs.

3 Approximating the SFD Vector

In this section, we focus on showing that if we have a method for sampling simplets uniformly from an SC, we can have an (ϵ, δ)-approximation of the SFD vector. After that, we study an algorithm for simple uniform sampling that is better than a trivial brute-force sampling method. Consider a collection of independent samples $X^k = X_1, \ldots, X_k$ drawn from a distribution ϕ over a domain D. Here, $\phi(A)$ signifies the probability of selecting an element from the set $A \subseteq D$. The empirical estimation of $\phi(A)$ based on the samples X^k is:

$$\hat{\phi}^X(A) = \frac{1}{k} \sum_{j=1}^{k} 1_A(X_j),$$

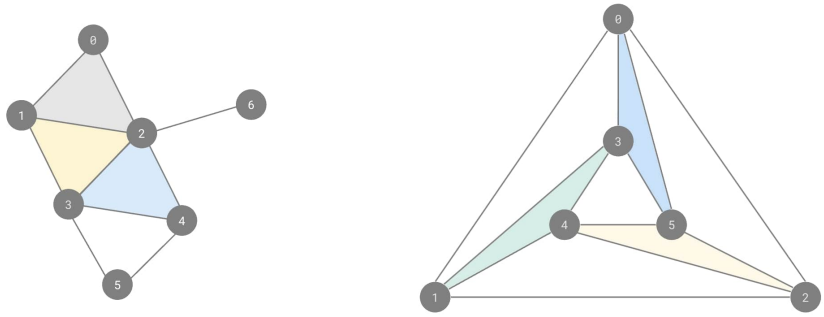
In this equation, $1_A(X_j)$ is an indicator function that equals 1 when X_j belongs to A and equals 0 otherwise. Additionally, let $\mathcal{R}$ be a family of subsets of D.

(ϵ, δ)-approximation. For any given $\epsilon, \delta \in (0, 1)$, we say $X \subseteq D$ is an (ϵ, δ)-*approximation* of $(\mathcal{R}, \phi)$, if with a probability of at least $(1 - \delta)$, it satisfies $sup_{A \in \mathcal{R}} |\phi(A) - \hat{\phi}^X(A)| \leq \epsilon$.

3.1 Sample Complexity of Approximating the SFD Vector

We utilize the concept of Vapnik-Chervonenkis dimension (VC dimension), introduced in [27]. In short, for a domain D and a collection $\mathcal{R}$ of subsets of D, the VC dimension $VC(D, \mathcal{R})$, represents the maximum size of a set $X \subseteq D$ that can be shattered by $\mathcal{R}$, which means $\{r \cap X | \forall r \in R\} = 2^{|X|}$. We use VC dimension to determine the sample complexity for approximating the SFD vector through simplet sampling models. Theorem 3.1 establishes the VC dimension of the collection of simplet sets.

▶ **Theorem 3.1** (VC Dimension of Simplets). *Let $\mathcal{R} = \{\mathcal{S}_i \mid 1 \leq i \leq N_m\}$ be a family of all simplet sets where N_m is the number of simplet types with at most m vertices, and $D = \mathcal{S}^m_{\mathcal{K}}$. Then, we have $VC(D, \mathcal{R}) = 1$.*



Simplet Type	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
$\lvert\mathcal{S}_{\mathcal{K}_1}\rvert$	**10**	**12**	**1**	**3**	**7**	**1**	**6**	0	**1**	**2**	**3**	0	0	0	0	0	0	0
$\lvert\phi_{\mathcal{K}_1}\rvert$	**0.22**	**0.26**	**0.02**	**0.07**	**0.15**	**0.02**	**0.13**	0	**0.02**	**0.04**	**0.07**	0	0	0	0	0	0	0
$\lvert\mathcal{S}_{\mathcal{K}_2}\rvert$	**12**	**12**	**5**	**3**	0	0	0	**3**	**9**	0	0	**3**	0	0	0	0	0	0
$\lvert\phi_{\mathcal{K}_2}\rvert$	**0.26**	**0.26**	**0.11**	**0.06**	0	0	0	**0.06**	**0.19**	0	0	**0.06**	0	0	0	0	0	0

Figure 2 The SFD vector and the number of at most 4-vertices simplets for two sample SCs. The simplet types in the table refer to the types in Figure 1.

Proof. We show that a set X with $|X| > 1$ can not be shattered with $(D, \mathcal{R})$. Let X be a set of simplets shattered with $(D, \mathcal{R})$, and assume that $|X| > 1$. Let s_1 and s_2 be two distinct elements of X. There are two possibilities. If elements s_1 and s_2 belong to the same simplet type, then, set $\{s_1\}$ can not be shattered because there is no set $\mathcal{S}_i$, including only s_1. Otherwise, elements s_1 and s_2 belong to different simplet types, but then $\{s_1, s_2\}$ can not be shattered because no set $\mathcal{S}_i$ contains both. Clearly every singleton set can be shattered by one of the $\mathcal{S}_i$s, hence $VC(D, \mathcal{R}) = 1$. ◀

The subsequent theorem from [24] illustrates the relationship between the upper bound on the sample complexity of sampling-based (ϵ, δ)-approximations and VC dimension.

▶ **Theorem 3.2.** *Let D be a domain and $\mathcal{R}$ be a family of subsets of D, with $VC(D, \mathcal{R}) \leq d$ and ϕ be a distribution on D. For every $\epsilon, \delta \in (0, 1)$, every set X of independent samples drawn from D using ϕ that satisfies*

$$|X| \geq \frac{c}{\epsilon^2}\left(d + \ln\frac{1}{\delta}\right),$$

is an (ϵ, δ)-approximation of $(\mathcal{R}, \phi)$ for some positive constant c.

Combining Theorem 3.1 and Theorem 3.2 we conclude our main result.

▶ **Proposition 3.3.** *Let X be a set of at least $\frac{c}{\epsilon^2}(1 + \ln\frac{1}{\delta})$ simplets sampled uniformly from simplicial complex $\mathcal{K}$. Then, X obtains an (ϵ, δ)-approximation on the SFD vector of $\mathcal{K}$.*

Proposition 3.3 shows that we can approximate the SFD vector using sampling-based algorithms, and the sample complexity of these approximations are independent of the simplicial complex size. This property suggests the usage of approximation algorithms for various simplicial complex sizes with the same sample complexity.

3.2 Simplet Uniform Sampling Algorithm

In this section, we propose a uniform sampling algorithm for simplets in a connected simplicial complex $\mathcal{K}$ that is better than a trivial brute-force method. The algorithm we present is a Monte-Carlo Markov-Chain algorithm [28], that samples sufficiently many simplets uniformly at random. We assume that $\mathcal{K}$ is connected with at least three vertices, and $m \geq 3$.

For the sampling part, we perform a random walk on a directed graph $\mathcal{P}^m_{\mathcal{K}}$ whose vertex set (states) is a set of all simplets in complex $\mathcal{K}$ with at most m vertices. Out-neighbors of every state s can be created by adding one vertex to s, removing one vertex from s, or replacing one vertex in s with another vertex out of s.

The transition probability matrix T for the random walk is such that every cell $T(i,j)$ defines the transition probability from state i to j. If i and j are not neighbors, we set $T(i,j) = 0$. Otherwise, we set $T(i,j) = min(\frac{1}{d(i)}, \frac{1}{d(j)})$ where $d(i)$ specifies the number of out-neighbors of state i. Also, for every i, if the sum of transitions from i is not equal to 1, we allocate the remaining probability to a self-loop for i. Observe that since $\mathcal{K}$ is finite, $\mathcal{P}^m_{\mathcal{K}}$ is finite and since $\mathcal{K}$ is connected, the random walk is irreducible. Indeed, since $\mathcal{K}$ is connected, there is a vertex u in $\mathcal{K}$ that is connected to at least two other vertices v, w. So the three simplets on $\{u,v,w\}$, on $\{u,v\}$, and on $\{u,w\}$ form a triangle in $\mathcal{P}^m_{\mathcal{K}}$ with positive probabilities on the edges, this means the random walk is aperiodic. Also T is symmetric, meaning $T = T^T$. This ensures that the random walk on $\mathcal{P}^m_{\mathcal{K}}$ converges to the uniform stationary distribution $(\frac{1}{|\mathcal{S}^m_{\mathcal{K}}|}, \ldots, \frac{1}{|\mathcal{S}^m_{\mathcal{K}}|})$. So, using this random walk on $\mathcal{P}^m_{\mathcal{K}}$, we can select a simplet from the input complex $\mathcal{K}$ with uniform distribution.

3.3 The SFD Vector Approximation Algorithm

Now, we propose the (ϵ, δ)-approximation algorithm on the SFD vector of $\mathcal{K}$. For input $\epsilon, \delta \in (0,1)$ and simplicial complex $\mathcal{K}$, first the algorithm calculates the number ℓ of samples needed, according to Proposition 3.3. After that it executes ℓ times the sampling algorithm, presented above, to find the set X of ℓ simplets that are chosen uniformly at random. Based on X, it computes $\hat{\phi}^X_{\mathcal{K}}(i)$, that is a (ϵ, δ)-approximation for $\phi_{\mathcal{K}}(i)$, for $1 \leq i \leq N_m$. The vector $(\hat{\phi}^X_{\mathcal{K}}(1), \ldots, \hat{\phi}^X_{\mathcal{K}}(N_m))$ is therefore a (ϵ, δ)-approximation for the SFD vector of $\mathcal{K}$.

Time Complexity of the SFD vector Approximation Algorithm The time complexity of the (ϵ, δ)-approximation algorithm, consists of two components: the number of samples and the time complexity for sample identification. Having established that $O(\frac{1}{\epsilon^2} \cdot (1 + \ln \frac{1}{\delta}))$ samples are sufficient for (ϵ, δ)-approximation, our focus shifts to analyzing the time complexity of the MCMC sampling algorithm. The mixing time t^G_{mix} in a random walk on graph G is the number of steps needed to be close to its stationary state with high probability. Lemma 3.4 limits the maximum degree of $\mathcal{P}^m_{\mathcal{K}}$, and then Lemma 3.5 shows an upper bound on $t^{\mathcal{P}^m_{\mathcal{K}}}_{mix}$ in terms of the number of vertices n, the maximum degree Δ in $\mathcal{K}$, and the diameter $diam(\mathcal{K})$, which is the length of maximum shortest path between any pair of vertices in $\mathcal{K}$.

▶ **Lemma 3.4.** *The maximum degree of $\mathcal{P}^m_{\mathcal{K}}$ satisfies $\Delta(\mathcal{P}^m_{\mathcal{K}}) \in O(m^2 \cdot \Delta)$.*

Proof. We can create neighbors of every state in $\mathcal{P}^m_{\mathcal{K}}$ by adding a new vertex, removing a vertex, or replacing two vertices. The number of neighbors by adding a new vertex is at most $m \cdot \Delta$, by removing a vertex is at most m, and by replacing two vertices is at most $m \cdot (m-1) \cdot \Delta$. Therefore, the maximum degree of every state in $\mathcal{P}^m_{\mathcal{K}}$ is in $O(m^2 \cdot \Delta)$. ◀

▶ **Lemma 3.5.** *The mixing time of the markov chain on $\mathcal{P}^m_{\mathcal{K}}$ is in $O(\log(n) \cdot \Delta \cdot diam(\mathcal{K})^2)$.*

Proof. Theorems (12.4) and (13.26) in [16] imply that the mixing time t_{mix}^{G} of a random walk on graph G with n vertices is in $O(\log(n)\cdot\Delta(G)\cdot diam(G)^2)$. For $\mathcal{P}_{\mathcal{K}}^{m}$ we can reach from every state i to every other state j with $diam(\mathcal{K})+m$ steps as follows: Assume $v\in V(i), u\in V(j)$ and assume a shortest path from v to u in $\mathcal{K}$. Starting from i, in every step, we replace one vertex from the current state with the unused closest vertex to v in the shortest path from v to u, until we reach u. After that we replace vertices that are not in j with vertices in j, starting from neighbors of u. We make sure that after each step the simplet remains connected. So, $diam(\mathcal{P}_{\mathcal{K}}^{m}) = diam(\mathcal{K})+m$ and therefore in the markov chain $\mathcal{P}_{\mathcal{K}}^{m}$ we have

$$t_{mix}^{\mathcal{P}_{\mathcal{K}}^{m}} \in O(log(n(\mathcal{P}_{\mathcal{K}}^{m}))\cdot\Delta(\mathcal{P}_{\mathcal{K}}^{m})\cdot diam(\mathcal{P}_{\mathcal{K}}^{m})^2) \in O(log(n)\cdot\Delta\cdot diam(\mathcal{K})^2).$$

◀

▶ **Corollary 3.6** (Time Complexity of (ϵ,δ)-approximation of SFD vector)**.** *Let $\mathcal{K}$ be a simplicial complex with the number of vertices n, maximum degree Δ and diameter $diam(\mathcal{K})$. The time complexity of (ϵ,δ)-approximation of SFD vector of $\mathcal{K}$ is $O(\frac{1}{\epsilon^2}\cdot(1+\ln\frac{1}{\delta})\cdot\log(n)\cdot\Delta\cdot diam(\mathcal{K})^2)$.*

In practice, for a large sparse simplicial complex $\mathcal{K}$, since Δ and $diam(\mathcal{K})$ are bounded, the above bound is sublinear in the size of $\mathcal{K}$ (i.e. the number of vertices or $\mathcal{K}$).

Implementation and Experiments. We implement an algorithm for counting the exact number of simplets of different types and another algorithm for approximating the frequencies based on uniform simplet sampling, with their source code accessible on GitHub [19]. This experimental outcome demonstrates that the confidence in the (ϵ,δ)-approximation is unrelated to the size of the input complex.

4 Conclusion

This paper introduced the Simplet Frequency Distribution (SFD) vector and a method for approximating it with simplet sampling algorithms. Also, we studied the sample complexity of approximating the SFD vector for SCs and showed that the obtained bounds are independent of SC size. We also showed that we can approximate the SFD vector with a specific error and confidence, and the time complexity depends only on the time complexity of sampling algorithm for finding a sample, that is sublinear in the algorithm we presented. It would be beneficial to have such algorithms with time complexity that is independent of the SC size.

Combining these approaches with filtrations of simplicial complexes and exploring them within alpha complexes would be interesting. Additionally, defining the vertex-specific SFD vectors for each vertex in the complex could offer valuable insights into their potential to convey more information about the global structure of the complex.

References

1 Uri Alon. Network motifs: theory and experimental approaches. *Nature Reviews Genetics*, 8(6):450–461, 2007.

2 James R Ashford, Liam D Turner, Roger M Whitaker, Alun Preece, and Diane Felmlee. Understanding the characteristics of covid-19 misinformation communities through graphlet analysis. *Online Social Networks and Media*, 27:100178, 2022.

3 Andrew Baas, Frances Hung, Hao Sha, Mohammad Al Hasan, and George Mohler. Predicting virality on networks using local graphlet frequency distribution. In *2018 IEEE International Conference on Big Data (Big Data)*, pages 2475–2482. IEEE, 2018.

4 Austin R Benson, Rediet Abebe, Michael T Schaub, Ali Jadbabaie, and Jon Kleinberg. Simplicial closure and higher-order link prediction. *Proceedings of the National Academy of Sciences*, 115(48):E11221–E11230, 2018.

5 Mansurul A Bhuiyan, Mahmudur Rahman, Mahmuda Rahman, and Mohammad Al Hasan. Guise: Uniform sampling of graphlets for large graph analysis. In *2012 IEEE 12th International Conference on Data Mining*, pages 91–100. IEEE, 2012.

6 Ginestra Bianconi. *Higher-order networks*. Cambridge University Press, 2021.

7 Marco Bressan. Efficient and near-optimal algorithms for sampling small connected subgraphs. *ACM Transactions on Algorithms*, 19(3):1–40, 2023.

8 Marco Bressan, Flavio Chierichetti, Ravi Kumar, Stefano Leucci, and Alessandro Panconesi. Counting graphlets: Space vs time. In *Proceedings of ACM International Conference on Web Search and Data Mining*, pages 557–566, 2017.

9 Marco Bressan, Flavio Chierichetti, Ravi Kumar, Stefano Leucci, and Alessandro Panconesi. Motif counting beyond five nodes. *ACM Transactions on Knowledge Discovery from Data*, 12(4):1–25, 2018.

10 Alexander Douglas et al. *Exploring how graphlet analysis can be used to identify highly-connected cancer driving genes*. PhD thesis, University of Northern British Columbia, 2022.

11 Herbert Edelsbrunner. *A short course in computational geometry and topology*. Springer, 2014.

12 Ernesto Estrada and Grant J Ross. Centralities in simplicial complexes. applications to protein interaction networks. *Journal of theoretical biology*, 438:46–60, 2018.

13 Jakob Jonsson. *Simplicial complexes of graphs*, volume 3. Springer, 2008.

14 Hyunju Kim, Jihoon Ko, Fanchen Bu, and Kijung Shin. Characterization of simplicial complexes by counting simplets beyond four nodes. In *Proceedings of the ACM Web Conference 2023*, pages 317–327, 2023.

15 Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. Graph evolution: Densification and shrinking diameters. *ACM transactions on Knowledge Discovery from Data (TKDD)*, 1(1):2–es, 2007.

16 David A Levin and Yuval Peres. *Markov chains and mixing times*, volume 107. American Mathematical Soc., 2017.

17 Zhaoqing Li, Zhenghong Deng, Zhen Han, Karin Alfaro-Bittner, Baruch Barzel, and Stefano Boccaletti. Contagion in simplicial complexes. *Chaos, Solitons & Fractals*, 152:111307, 2021.

18 Louis-David Lord, Paul Expert, Henrique M Fernandes, Giovanni Petri, Tim J Van Hartevelt, Francesco Vaccarino, Gustavo Deco, Federico Turkheimer, and Morten L Kringelbach. Insights into brain architectures from the homological scaffolds of functional connectivity networks. *Frontiers in systems neuroscience*, 10:85, 2016.

19 Mohammad Mahini and Salman Qadami. Network Signature. URL: `https://github.com/mmahini/graphlet-analysis`.

20 Dror Marcus and Yuval Shavitt. Rage–a rapid graphlet enumerator for large networks. *Computer Networks*, 56(2):810–819, 2012.

21 Giulia Preti, Gianmarco De Francisci Morales, and Francesco Bonchi. Strud: Truss decomposition of simplicial complexes. In *Proceedings of the Web Conference 2021*, pages 3408–3418, 2021.

22 Giulia Preti, Gianmarco De Francisci Morales, and Francesco Bonchi. Fresco: Mining frequent patterns in simplicial complexes. In *Proceedings of the ACM Web Conference 2022*, pages 1444–1454, 2022.

23 Pedro Ribeiro, Pedro Paredes, Miguel EP Silva, David Aparicio, and Fernando Silva. A survey on subgraph counting: concepts, algorithms, and applications to network motifs and graphlets. *ACM Computing Surveys (CSUR)*, 54(2):1–36, 2021.

24 Matteo Riondato and Evgenios M Kornaropoulos. Fast approximation of betweenness centrality through sampling. *Data Mining and Knowledge Discovery*, 30(2):438–475, 2016.

25 Rahmtin Rotabi, Krishna Kamath, Jon Kleinberg, and Aneesh Sharma. Detecting strong ties using network motifs. In *Proceedings of the 26th International Conference on World Wide Web Companion*, pages 983–992, 2017.

26 Arnab Sinha, Zhihong Shen, Yang Song, Hao Ma, Darrin Eide, Bo-June Hsu, and Kuansan Wang. An overview of microsoft academic service (mas) and applications. In *Proceedings of the 24th international conference on world wide web*, pages 243–246, 2015.

27 Vladimir N Vapnik and A Ya Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. In *Measures of complexity: festschrift for alexey chervonenkis*, pages 11–30. Springer, 2015.

28 Duncan J Watts and Steven H Strogatz. Collective dynamics of 'small-world'networks. *nature*, 393(6684):440–442, 1998.

29 Sebastian Wernicke. Efficient detection of network motifs. *IEEE/ACM transactions on computational biology and bioinformatics*, 3(4):347–359, 2006.

30 Sam Freddy Ludwien Windels. *Graphlet-adjacencies provide complementary views on the functional organisation of the cell and cancer mechanisms*. PhD thesis, UCL (University College London), 2022.

Coloring problems on arrangements of pseudolines*

Sandro Roch

Technische Universität Berlin
roch@math.tu-berlin.de

Abstract

Arrangements of pseudolines are a widely studied generalization of line arrangements. They are defined as a finite family of infinite curves in the Euclidean plane, any two of which intersect at exactly one point. One can state various related coloring problems depending on the number n of pseudolines. In this article, we show that n colors are sufficient for coloring the crossings avoiding twice the same color on the boundary of any cell, or, alternatively, avoiding twice the same color along any pseudoline. We also study the problem of coloring the pseudolines avoiding monochromatic crossings.

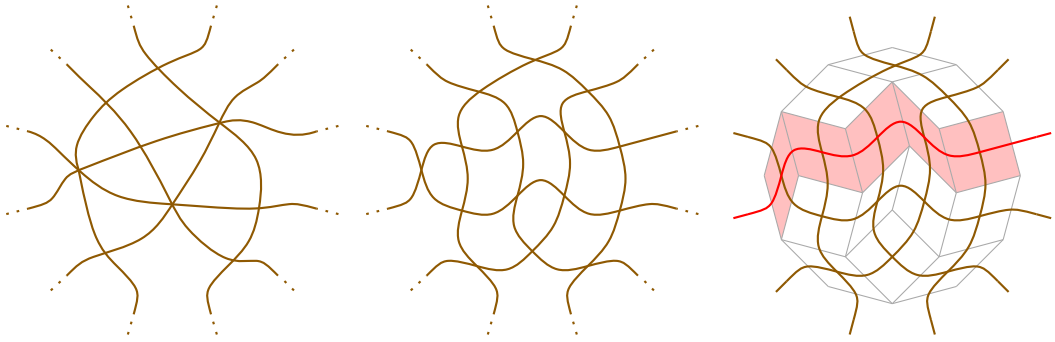
Related Version arXiv:2402.12564

1 Introduction

An *arrangement of pseudolines* or *pseudoline arrangement* is a finite family of simple continuous curves $f_1, \cdots, f_n : \mathbb{R} \to \mathbb{R}^2$ in the Euclidean plane with

$$\lim_{t\to\infty} \|f_i(t)\| = \lim_{t\to-\infty} \|f_i(t)\| = \infty,$$

and the property that each pair f_i, f_j, $i \neq j$ crosses in exactly one point. A pseudoline arrangement is *simple*, if at most two pseudolines cross in a single point, see Figure 1a and Figure 1b for examples of a non-simple and a simple arrangement of 6 pseudolines.



(a) (b) (c)

Figure 1 A non-simple (a) and a simple (b) arrangement together with a corresponding tiling (c).

Pseudoline arrangements are widely studied objects. They were first described in 1926 by Levi [18] and were further studied by Ringel [19] and Grünbaum [13]. Every line arrangement is also a pseudoline arrangement. On the other hand, there exist arrangements of at least $n \geq 8$ pseudolines that cannot be „strechted“, i.e. they are not isomorphic to any line arrangement, see [19] and [12]. But pseudoline arrangements are not only a generalization of line arrangements: Isomorphism classes of simple arrangements are in correspondence with a

* S. Roch was funded by the DFG-Research Training Group 'Facets of Complexity' (DFG-GRK 2434). Special thanks to Rimma Hämäläinen for the stimulating discussions on this topic.

40th European Workshop on Computational Geometry, Ioannina, Greece, March 13–15, 2024.
This is an extended abstract of a presentation given at EuroCG'24. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

rich variety of other objects, such as rhombic tilings of 2-dimensional zonotopes (indicated in Figure 1c), classes of reduced words of permutations and oriented matroids of rank 3. For a general introduction to pseudoline arrangements we refer to [8], [10] and [2, ch. 6].

1.1 Related work

In 2006, Felsner, Hurtado, Noy and Streinu [9] studied the *arrangement graph* $G_{\mathcal{A}}$ of a pseudoline arrangement $\mathcal{A}$, which consists of the crossings in $\mathcal{A}$ as vertices and its edges are formed by the arcs between them. They give a short argument that $G_{\mathcal{A}}$ can be colored using three colors if $\mathcal{A}$ is simple. As $G_{\mathcal{A}}$ is planar, it is clearly 4-colorable, including for non-simple arrangements. In [5] one can find an infinite family of line arrangements that require 4 colors.

In 2013, Bose et al. [3] introduced further coloring problems on line arrangements. An arrangement decomposes the Euclidean plane into *cells*: The example in Figure 1a consists of 7 *bounded cells* and 12 *unbounded cells*. One of the most remarkable results in [3] states that coloring the lines of a simple arrangement of n lines avoiding cells whose bounding lines have all the same color requires at most $\mathcal{O}(\sqrt{n})$ colors. This was improved to $\mathcal{O}(\sqrt{n/\log n})$ by Ackerman, Pach, Pinchasi, Radoičić and Tóth [1], extending it also to non-simple line arrangements. Finding line arrangements that require many colors in such a coloring seems to be a difficult task; in [3] they provide a construction that requires $\Omega\left(\log / n \log\log n\right)$ colors.

1.2 Results

In [3] and [1], the language of hypergraph coloring serves as a common formalization of the different coloring concepts and allows for the use of results from this field. If $\mathcal{H} = (V, \mathcal{E})$ is a hypergraph, a *vertex coloring* of $\mathcal{H}$ is a coloring of the vertices avoiding *monochromatic edges*, i.e. hyperedges whose contained vertices are assigned all the same color, while an *edge coloring* of $\mathcal{H}$ is a coloring of the hyperedges with no vertex being incident to two edges of the same color. The *(vertex) chromatic number* $\chi(\mathcal{H})$ is the minimal number of colors of a vertex coloring, while the *edge chromatic number* $\chi'(\mathcal{H})$ is the minimal number of colors of an edge coloring. Note that vertex coloring is not equivalent to edge coloring of the hypergraph dual.

Our results can all be stated in terms of two hypergraphs: The vertices of $\mathcal{H}_{\text{cell-vertex}}(\mathcal{A})$ are the (bounded and unbounded) cells of $\mathcal{A}$, and each crossing c defines a hyperedge consisting of the cells that contain c on their boundary. At the same time, the hypergraph $\mathcal{H}_{\text{line-vertex}}(\mathcal{A})$ is defined on the set of n pseudolines as vertices and each crossing in $\mathcal{A}$ defines a hyperedge consisting of the pseudolines involved in c. Section 2 is devoted to problems in which the crossings are colored. We show $\chi'(\mathcal{H}_{\text{cell-vertex}}) \leq n$ for every pseudoline arrangement:

▶ **Theorem 1.1.** *Let $\mathcal{A}$ be an arrangement of n pseudolines. The crossings of $\mathcal{A}$ can be colored using n colors so that no color appears twice on the boundary of any cell.*

The abovementioned results in [3] and [1] are bounds on the chromatic number of a hypergraph $\mathcal{H}_{\text{line-cell}}$ restricted to the case of line arrangements. However, none of the coloring problems that are discussed in [3] relates lines with crossings. This is done in the following two theorems, the first one of which shows $\chi'(\mathcal{H}_{\text{line-vertex}}) \leq n$:

▶ **Theorem 1.2.** *Let $\mathcal{A}$ be an arrangement of n pseudolines. The crossings of $\mathcal{A}$ can be colored using n colors so that no color appears twice along any pseudoline.*

Figure 2 shows an example of a coloring as guaranteed in Theorem 1.1 and in Theorem 1.2. In Section 3, we study the number of colors required to color the pseudolines avoiding monochromatic crossings. In addition to several minor results, we prove:

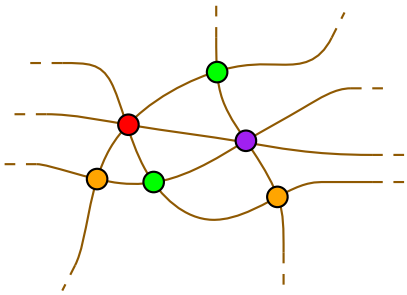


Figure 2 Coloring that fulfills the statements of both Theorem 1.1 and Theorem 1.2.

▶ **Theorem 1.3.** *Let $\mathcal{A}$ be an arrangement of n pseudolines. The pseudolines of $\mathcal{A}$ can be colored using $\mathcal{O}(\sqrt{n})$ colors avoiding monochromatic crossings of degree at least 4.*

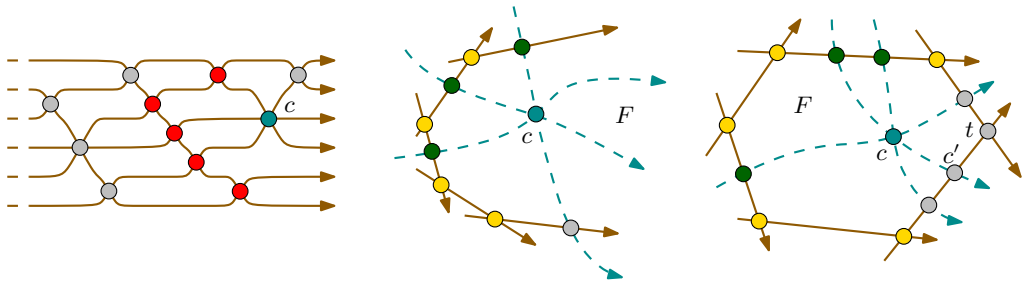
Here, the *degree* of a crossing is the number of pseudolines that intersect in said crossing.

2 Coloring crossings

In this section we sketch the proofs of Theorem 1.1 and Theorem 1.2. For complete proofs we refer to the full version.

2.1 Avoiding twice the same color on the boundary of any cell

A pseudoline arrangement can always be drawn in a way in which the pseudolines are x-monotone curves and no two crossings lie on a vertical line, see Figure 3a. This is also known as a *wiring diagram*, see [10]. We call this a *monotone drawing* and aim for coloring the crossings greedily from left to right. For any crossing c, a *conflict ancestor* is a crossing c' that lies left of c and both c and c' are on the boundary of a common cell. Figure 3a shows an example: The red crossings are conflict ancestors of c.



(a) **(b)** **(c)**

Figure 3 (a): Example for conflict ancestors; (b), (c): Case distinction for bounding their number.

Fix some crossing c in $\mathcal{A}$. Consider the arrangement $\mathcal{A}'$ obtained from $\mathcal{A}$ by dropping all pseudolines that contain c. In $\mathcal{A}'$ there is a cell F whose area contains the point c. All conflict ancestors of c lie on the boundary of F. By distinguishing the cases in which F is an unbounded or a bounded cell one can obtain that each crossing has at most $n-1$ conflict ancestors, see Figure 3b and Figure 3c and the proof in the full version. Theorem 1.1 now follows: Color the crossings from left to right. Using n colors, we can always avoid the colors

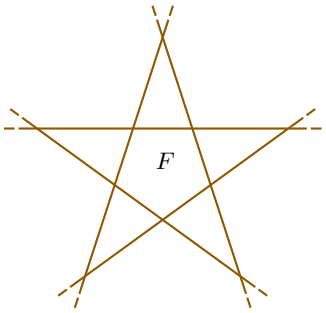


Figure 4 Construction that shows that Theorem 1.1 is tight.

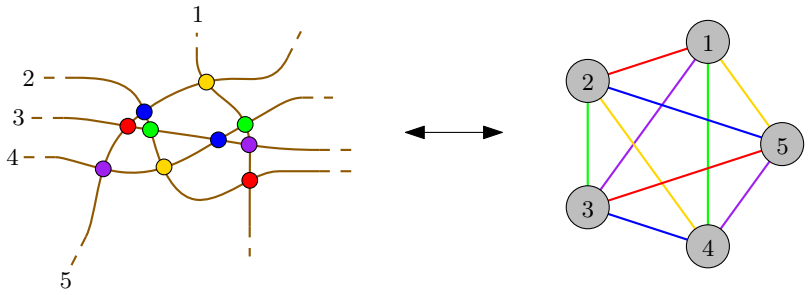


Figure 5 Coloring simple arrangements is equivalent to edge-coloring of K_n.

that were already assigned to conflict ancestors. It is easy to see that Theorem 1.1 is tight: There are arbitrarily large arrangements as in Figure 4 in which a cell F is incident to all pseudolines.

2.2 Avoiding twice the same color along any pseudoline

In view of Theorem 1.2, we now focus on coloring the crossings of a pseudoline arrangement $\mathcal{A}$ avoiding twice the same color along any pseudoline. If $\mathcal{A}$ is simple, this is equivalent to edge-coloring of the complete graph K_n (see Figure 5) and n colors are sufficient. We deduce the general case from the following recent breakthrough result by Kang et al. [15], which solved a longstanding conjecture by Erdős, Faber and Lovász [7].

▶ **Theorem 2.1** (D. Y. Kang, T. Kelly, D. Kühn, A. Methuku & D. Osthus, 2021)**.** *For every simple hypergraph $\mathcal{H}$ with n vertices, $\chi'(\mathcal{H}) \leq n$.*

Here, a hypergraph $\mathcal{H} = (V, \mathcal{E})$ is *simple* if all hyperedges have cardinality at least 2 and for all $E_1, E_2 \in \mathcal{E}$, $E_1 \neq E_2$ it holds that $|E_1 \cap E_2| \leq 1$.

Proof of Theorem 1.2. The statement is equivalent to the existence of an edge-coloring of $\mathcal{H}_{\text{line-vertex}}$ using n colors. $\mathcal{H}_{\text{line-vertex}}$ is simple, because there can be at most one pseudoline passing through any pair of crossings, otherwise this would mean a pair of pseudolines crossing twice. Then the statement follows from Theorem 2.1. ◀

It would be nice to have a bound on the required number of colors that also takes into account how far the arrangement is away from being a simple arrangement. For this purpose, we introduce $\text{mx}(\mathcal{A})$, which is defined as the maximal number of crossings along any pseudoline in $\mathcal{A}$. For simple arrangements we have $\text{mx}(\mathcal{A}) = n - 1$. Excluding *trivial arrangements* where all pseudolines cross in a single point ($\text{mx}(\mathcal{A}) = 1$), it was shown

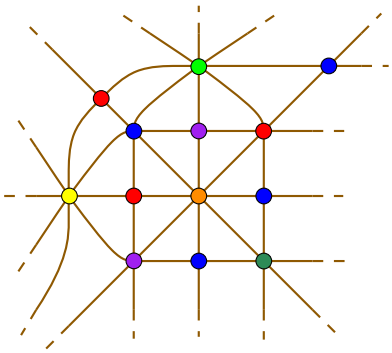


Figure 6 Minimal coloring using 7 colors of an arrangement with mx = 4.

recently in [6] that the number of pseudolines is linearly upper bounded by $\mathrm{mx}(\mathcal{A})$, in particular $n \leq 845 \cdot \mathrm{mx}(\mathcal{A})$ for large values of n. Therefore, $\mathrm{mx}(\mathcal{A})$ can be interpreted as a measure of the size of an arrangement alternatively to the number of pseudolines n.

▶ Conjecture 1. There is a constant c so that the crossings of every pseudoline arrangement $\mathcal{A}$ can be colored using $\mathrm{mx}(\mathcal{A}) + c$ colors so that no color appears twice along any pseudoline.

Figure 6 shows an arrangement $\mathcal{A}$ with $\mathrm{mx}(\mathcal{A}) = 4$ that requires 7 colors. We were unable to find any arrangement where the gap between these numbers is larger than 3. The following proposition is a consequence of a result about hypergraph coloring by Kahn [14, 16]. It shows that Conjecture 1 holds at least asymptotically and under a certain restriction.

▶ Proposition 1. For every $k, \varepsilon > 0$, there exists an $\mathrm{mx}_0 \in \mathbb{N}$ so that the following holds: If a pseudoline arrangement $\mathcal{A}$ only contains crossings of degree at most k and fulfills $\mathrm{mx}(\mathcal{A}) \geq \mathrm{mx}_0$, then its crossings can be colored using $(1 + \varepsilon) \cdot \mathrm{mx}(\mathcal{A})$ colors so that no color appears twice along any pseudoline.

3 Coloring pseudolines

Again, complete proofs for this section can be found in the full version of this article. A *pseudoline coloring* of an arrangement $\mathcal{A}$ is defined as a coloring of the pseudolines in $\mathcal{A}$ such that there are no monochromatic crossings, i.e. crossings of pseudolines of a single color class. We let $\chi_{pl}(\mathcal{A})$ denote the minimal number of colors in a pseudoline coloring of $\mathcal{A}$.

3.1 Pseudoline colorings and ordinary points

The study of pseudoline colorings is closely related to the study of *ordinary points*. An ordinary point is defined as a crossing of exactly two pseudolines, also known as *simple crossing*. Every non-trivial pseudoline arrangement contains at least $\lceil 6n/13 \rceil$ ordinary points [17]. Two pseudolines that cross each other in an ordinary point must be assigned different colors. Hence, for simple arrangements $\mathcal{A}$ we have $\chi_{pl}(\mathcal{A}) = n$. In the following we want to take a closer look at the relationship between $\chi_{pl}(\mathcal{A})$ and the structure of the ordinary points in a pseudoline arrangement. For this purpose, we define the *ordinary graph* $G_o(\mathcal{A})$ that has the n pseudolines of $\mathcal{A}$ as its vertices and two of them share an edge if and only if they cross each other in an ordinary point. Clearly, $\chi_{pl}(\mathcal{A}) \geq \chi(G_o(\mathcal{A}))$. Let $\sigma_k(n)$ denote the maximal number of ordinary points that an arrangement of n pseudolines $\mathcal{A}$ with $\chi_{pl}(\mathcal{A}) \leq k$ can have. Turán's theorem, applied on $G_o(\mathcal{A})$, gives us close bounds on $\sigma_k(n)$:

▶ Proposition 2. We have $\sigma_k(n) \in \Theta(n^2)$. More precisely, let $t_k(n)$ denote the Turán number, i.e. the maximum number of edges that a graph on n vertices without containing a $(k+1)$-clique can have. Then we have $t_k(n) - n \leq \sigma_k(n) \leq t_k(n)$.

We would like to know how much $\chi_{pl}(\mathcal{A})$ and $\chi(G_o(\mathcal{A}))$ can differ. We observe:

▶ Proposition 3. There are arbitrarily large arrangements with $\chi_{pl}(\mathcal{A}) = 2 \cdot \chi(G_o(\mathcal{A}))$.

It is unknown to us whether the factor of 2 in Proposition 3 can be further improved.

When it comes to the complexity of computing $\chi_{pl}(\mathcal{A})$ we have the following result:

▶ Proposition 4. Given an arrangement of pseudolines $\mathcal{A}$, it is NP-hard to compute $\chi_{pl}(\mathcal{A})$.

3.2 Avoiding monochromatic crossings of high degrees

Even though $\chi_{pl}(\mathcal{A})$ and $\chi(G_o(\mathcal{A}))$ can differ by a multiplicative factor, as stated in Proposition 3, for most arrangements, $\chi_{pl}(\mathcal{A})$ does not seem to be far away from $\chi(G_o(\mathcal{A}))$. This is why our focus lies now on a variant of pseudoline colorings: Instead of avoiding monochromatic crossings of any degree, including ordinary points, we only forbid crossings of certain degrees to be monochromatic. Lemma 3.1 can be proven using the Lovász Local Lemma.

▶ **Lemma 3.1.** *Let $l, r \in \mathbb{N}$ and let $\mathcal{A}$ be an arrangement of n pseudolines. Then, using*

$$\left(\frac{4(l+r)}{l-1}n\right)^{\frac{1}{l-1}} \in \mathcal{O}(n^{\frac{1}{l-1}})$$

colors, $\mathcal{A}$ can be colored avoiding monochromatic crossings of degree within $\{l, l+1, \cdots, l+r\}$.

Theorem 1.3 follows from Lemma 3.1 by first coloring the crossings of degree at most $\sqrt{n}$. If we only want to avoid monochromatic crossings of a single degree, then we can obtain a stronger result by applying a theorem by Frieze and Mubayi [11].

▶ Proposition 5. Let $\mathcal{A}$ be an arrangement of n pseudolines. Fix some $l \geq 3$. Then, the pseudolines in $\mathcal{A}$ can be colored using

$$c \cdot \left(\frac{\mathrm{mx}(\mathcal{A})}{\log \mathrm{mx}(\mathcal{A})}\right)^{\frac{1}{l-1}} \in \mathcal{O}\left(\left(\frac{n}{\log n}\right)^{\frac{1}{l-1}}\right)$$

colors avoiding monochromatic crossings of degree exactly l, where c only depends on l.

4 Conclusion and Future Work

We consider Theorem 1.1 as our main result. When coloring the crossings avoiding twice the same color along any pseudoline, Theorem 1.2 is a direct application of the recently proven Erdős-Faber-Lovász conjecture. However, for the specific hypergraphs induced by pseudoline arrangements, one could hope for a simple deterministic coloring procedure, like the one proposed in [4] that requires $\lceil (3/2)n - 2 \rceil$ colors.

We mentioned Conjecture 1 as an open problem. One may also ask whether for sufficiently large arrangements there always exists a coloring using n colors that satisfies the conditions of Theorem 1.1 and Theorem 1.2 simultaneously. When it comes to pseudoline colorings, we asked whether $\chi_{pl}(\mathcal{A})$ and $\chi(G_o(\mathcal{A}))$ can differ by a factor larger than 2. Finally, in view of Lemma 3.1, Theorem 1.3 and Proposition 5, we expect it to be possible to color the pseudolines of every arrangement using $\mathcal{O}(n^{\frac{1}{l-1}})$ colors avoiding monochromatic crossings of degree at least l.

References

1 Eyal Ackerman, János Pach, Rom Pinchasi, Radoš Radoičić, and Géza Tóth. A note on coloring line arrangements. *Electron. J. Combin.*, 21(2):Article Number 2.23, 2014. doi:10.37236/2660.

2 Anders Björner, Michel Las Vergnas, Bernd Sturmfels, Neil White, and Günter Ziegler. *Oriented matroids*, volume 46 of *Encycl. Math. Appl.* Cambridge University Press, 2nd edition, 1999. doi:10.1017/CBO9780511586507.

3 Prosenjit Bose, Jean Cardinal, Sébastien Collette, Ferran Hurtado, Matias Korman, Stefan Langerman, and Perouz Taslakian. Coloring and guarding arrangements. *Discrete Math. Theor. Comput. Sci.*, 15(3):139–154, 2013. URL: https://dmtcs.episciences.org/2072/pdf.

4 W. I. Chang and E. L. Lawler. Edge coloring of hypergraphs and a conjecture of Erdős, Faber, Lovász. *Combinatorica*, 8(3):293–295, 1988. doi:10.1007/BF02126801.

5 Man-Kwun Chiu, Stefan Felsner, Manfred Scheucher, Felix Schröder, Raphael Steiner, and Birgit Vogtenhuber. Coloring circle arrangements: New 4-chromatic planar graphs. *Eur. J. Comb.*, 2023. doi:10.1016/j.ejc.2023.103839.

6 Adrian Dumitrescu. The Dirac–Goodman–Pollack conjecture. *Discrete Comput. Geom.*, 2023. doi:10.1007/s00454-023-00487-z.

7 Paul Erdős. On the combinatorial problems which I would most like to see solved. *Combinatorica*, 1:25–42, 1981. doi:10.1007/BF02579174.

8 Stefan Felsner and Jacob E. Goodman. Pseudoline arrangements. In Csaba D. Tóth, Jacob E. Goodman, and Joseph O'Rourke, editors, *Handbook of discrete and computational geometry*, Discrete Math. Appl., chapter 5. CRC Press, Boca Raton, FL, 3rd revised and updated edition, 2017.

9 Stefan Felsner, Ferran Hurtado, Marc Noy, and Ileana Streinu. Hamiltonicity and colorings of arrangement graphs. *Discrete Appl. Math.*, 154(17):2470–2483, 2006. doi:10.1016/j.dam.2006.04.006.

10 Stefan Felsner and Helmut Weil. Sweeps, arrangements and signotopes. *Discrete Appl. Math.*, 109(1-2):67–94, 2001. doi:10.1016/S0166-218X(00)00232-8.

11 Alan Frieze and Dhruv Mubayi. Coloring simple hypergraphs. *J. Comb. Theory, Series B*, 103(6):767–794, 2013. doi:10.1016/j.jctb.2013.09.003.

12 Jacob E. Goodman and Richard Pollack. Proof of Grünbaum's conjecture on the stretchability of certain arrangements of pseudolines. *J. Comb. Theory, Ser. A*, 29:385–390, 1980. doi:10.1016/0097-3165(80)90038-2.

13 Branko Grünbaum. *Arrangements and spreads*, volume 10 of *Reg. Conf. Ser. Math.* Amer. Math. Soc., Providence, RI, 1972.

14 Jeff Kahn. Asymptotically good list-colorings. *J. Comb. Theory, Ser. A*, 73(1):1–59, 1996. doi:10.1006/jcta.1996.0001.

15 Dong Kang, Tom Kelly, Daniela Kühn, Abhishek Methuku, and Deryk Osthus. A proof of the Erdős-Faber-Lovász conjecture. *Ann. Math. (2)*, 198(2):537–618, 2023. doi:10.4007/annals.2023.198.2.2.

16 Dong Yeap Kang, Tom Kelly, Daniela Kühn, Abhishek Methuku, and Deryk Osthus. Graph and hypergraph colouring via nibble methods: A survey, 2021. arXiv:2106.13733.

17 Jonathan Lenchner. *Sylvester-Gallai Results and Other Contributions to Combinatorial and Computational Geometry*. PhD thesis, Polytechnic University, 2008.

18 Friedrich Levi. Die Teilung der projektiven Ebene durch Gerade oder Pseudogerade. Berichte Leipzig 78, 256-267, 1926.

19 Gerhard Ringel. Über Geraden in allgemeiner Lage. *Elemente der Mathematik*, 12:75–82, 1957. doi:10.5169/seals-19211.

Faces in Rectilinear Drawings of Complete Graphs*

Martin Balko[1], Anna Brötzner[2], Fabian Klute[3], and Josef Tkadlec[4]

1 **Department of Applied Mathematics, Faculty of Mathematics and Physics, Charles University, Czech Republic**
balko@kam.mff.cuni.cz

2 **Faculty of Technology and Society, Malmö University, Sweden**
anna.brotzner@mau.se

3 **Departament de Matemàtiques, Universitat Politècnica de Catalunya, Barcelona, Spain**
fabian.klute@upc.edu

4 **Computer Science Institute, Faculty of Mathematics and Physics, Charles University, Czech Republic**
josef.tkadlec@iuuk.mff.cuni.cz

Abstract

We study extremal problems about faces in *convex rectilinear drawings* of K_n, that is, drawings where vertices are represented by points in the plane in convex position and edges by line segments between the points representing the end-vertices. We show that if a convex rectilinear drawing of K_n does not contain a common interior point of at least three edges, then there is always a face forming a convex 5-gon while there are such drawings without any face forming a convex k-gon with $k \geq 6$.

A convex rectilinear drawing of K_n is *regular* if its vertices correspond to vertices of a regular convex n-gon. We characterize positive integers n for which regular drawings of K_n contain a face forming a convex 5-gon.

To our knowledge, this type of problems has not been considered in the literature before and so we also pose several new natural open problems.

1 Introduction

Let G be a graph with no loops nor multiple edges. In a *rectilinear drawing of* G the vertices are represented by distinct points in the plane and each edge corresponds to a line segment connecting the images of its end-vertices. We consider only drawings where no three points representing vertices lie on a common line. As usual, we identify the vertices and their images, as well as the edges and the line segments representing them.

A *crossing* in a rectilinear drawing D of G is a common interior point of at least two edges of D where they properly cross. A *heavy crossing* in D is a common interior point of at least three edges of D where they properly cross. We say that D is *generic* if there are no heavy crossings in D. That is, crossings in a generic drawing D are the points where exactly two edges of D cross.

We focus on rectilinear drawings of complete graphs K_n on n vertices. We say that a rectilinear drawing D of a graph K_n is *convex* if the points representing the vertices of

* M. Balko was supported by the grant no. 23-04949X of the Czech Science Foundation (GAČR) and by the Center for Foundations of Modern Computer Science (Charles University project UNCE/SCI/004). A. Brötzner was supported by grant 2021-03810 from the Swedish Research Council (Vetenskapsrådet). J. Tkadlec was supported by the Center for Foundations of Modern Computer Science (Charles University project UNCE/SCI/004). This article is part of a project that has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 810115).

40th European Workshop on Computational Geometry, Ioannina, Greece, March 13–15, 2024.
This is an extended abstract of a presentation given at EuroCG'24. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

K_n are in convex position. We say that a convex drawing D of K_n is *regular* if the points representing the vertices of K_n form a regular n-gon; see Figure 1 for regular drawings of K_8 and K_{12}.

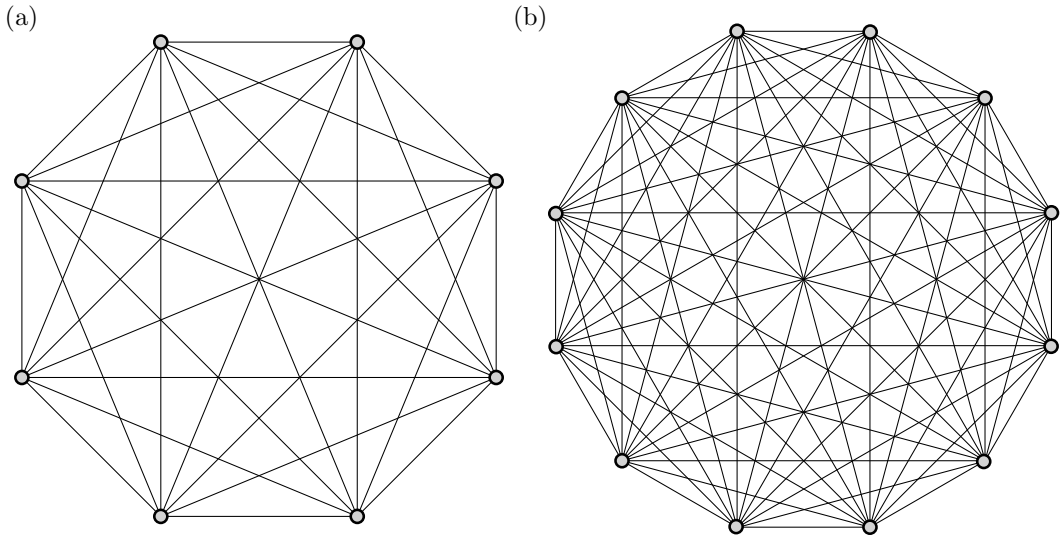


Figure 1 Regular drawings of K_8 (part (a)) and K_{12} (part (b)). Observe that none of these drawings contains a 5-face.

A *face* in a rectilinear drawing D of K_n is a non-empty connected component of $\mathbb{R}^2 \setminus D$. Note that exactly one face of D is unbounded and that every bounded face of D is a convex polygon. Thus, we can define the *size* of a bounded face F of D to be the number of vertices of the polygon that forms F. If the size of F equals k, then we call F a *k-face* of D.

In this paper, we study extremal problems about the bounded faces of a given size in convex drawings of K_n. To our knowledge, there has been no systematic study of this topic despite the fact that it offers an abundance of natural and interesting problems. For example, what is the largest face we can always find in a convex drawing of K_n for large n? What if we restrict ourselves to generic convex drawings of K_n? Or to regular drawings of K_n? In this paper, we address these questions and we pose several natural open problems.

2 Previous Work

Despite the fact that these problems are very natural and that rectilinear drawings of K_n have been studied extensively, we did not find any relevant reference in the literature. The existence of faces of a given size in regular drawings of K_n was recently considered by Shannon and Sloane [14], who computed the values from Table 1, but we are not aware of any publication. The total number of faces in a regular drawing of K_n was considered by Harborth [8] and Poonen and Rubinstein [12], but these results do not distinguish faces of different sizes and do not apply to all convex drawings of K_n. Finally, Hall [7] studied large faces in convex drawings of K_n where the vertices are points from the integer lattice.

Concerning other graph classes, Griffiths [6] calculated the number of regions enclosed by the edges of so-called regular drawings of the complete bipartite graphs $K_{n,n}$. There are also various results about the complexity of faces in the more general setting of line arrangements; for example [1, 2, 4, 5, 11]. However, we do not know any result that would imply the existence of large bounded faces in all convex drawings of sufficiently large K_n.

Closely related to our paper is the work of Poonen and Rubinstein [12] who gave a formula for the number of crossings in regular drawings of K_n and used it to count the number of faces in regular drawings of K_n. In particular, it follows from their formula that all regular drawings of K_n with odd n have $\binom{n}{4}$ crossings and thus are generic. They also showed that, apart from the center, no point is the intersection of more than 7 edges of a regular drawing of K_n for any positive integer n. We also note that these results are connected to the well-known *Blocking conjecture*; see [10, 13].

3 Our Results

First, we address the question about the maximum size of a face that we can always find in convex or regular drawings of K_n for large n. We observe that finding faces of size 3 or 4 in convex drawings of K_n is not difficult.

▶ **Proposition 3.1.** *Let n be a positive integer and D a convex drawing of K_n. Then, D contains a 3-face if and only if $n \geq 3$. Moreover, D contains a 4-face if and only if $n \geq 6$.*

To find larger faces, we restrict ourselves to generic convex drawings of K_n. In this case, we can show that a 5-face always exists if we have at least five vertices.

▶ **Theorem 3.2.** *For every positive integer n and every generic convex drawing D of K_n, the drawing D contains a 5-face if and only if $n \geq 5$.*

On the other hand, we can provide examples of generic convex drawings of K_n with arbitrarily large n that do not contain any k-face with $k \geq 6$.

▶ **Theorem 3.3.** *For every positive integer n, there is a generic convex drawing of K_n that does not contain any k-face with $k \geq 6$.*

Thus, in the case of generic convex drawings of K_n, we can settle the question about the largest face we can always find completely. A k-face with $k \in \{3, 4, 5\}$ is guaranteed in all sufficiently large drawings, while faces of sizes larger than 5 can be avoided (even simultaneously). The problem, however, becomes significantly more difficult if we allow heavy crossings.

We were not able to find a k-face with $k \geq 5$ in every sufficiently large convex drawing of K_n. In fact, finding larger faces becomes surprisingly difficult already for regular drawings of K_n. Here, however, we can at least show that a 5-face always exists in all sufficiently large regular drawings of K_n. In fact, we can even precisely characterize the values of n for which a regular drawing of K_n contains a 5-face.

▶ **Theorem 3.4.** *For a positive integer n, a regular drawing of K_n contains a 5-face if and only if $n \notin \{1, 2, 3, 4, 6, 8, 12\}$.*

The proof of Theorem 3.4 is quite involved and is based on the results obtained by Poonen and Rubinstein [12].

Finally, although we were not able to find a 5-face in all sufficiently large convex drawings of K_n, we can at least show that every convex drawing of K_7 contains at least one.

▶ **Proposition 3.5.** *Every convex drawing of K_7 contains a 5-face.*

k	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
$a(k)$	3	6	5	9	7	13	9	29	11	40	13	43	15	212	17	231	19

Table 1 The values of $a(k)$, the smallest n such that the regular drawing of K_n contains a k-face, computed by Shannon and Sloane [14].

4 Open Problems and Discussion

The study of extremal questions about faces of a given size in convex drawings of K_n offers plenty of interesting and natural problems. Here, we draw attention to some of them.

Although we were able to determine the largest size of a bounded face that appears in every sufficiently large generic convex drawing of K_n, the same question remains unsolved for general convex drawings of K_n. In particular, the following problem is open.

▶ Problem 4.1. Is there a positive integer n_0 such that for every $n \geq n_0$ every convex drawing of K_n contains a 5-face?

Since the regular drawing of K_{12} does not contain a 5-face, we have $n_0 \geq 13$, if it exists. An affirmative answer to Problem 4.1 would imply that every sufficiently large *regular* drawing of K_n contains a 5-face, a fact that was quite difficult to prove.

Considering the regular drawings of K_n, although we proved that all sufficiently large regular drawings of K_n contain a 5-face, we do not know much about larger faces. It seems plausible that we can find arbitrarily large faces in regular drawings of K_n as n grows.

▶ Problem 4.2. Is it true that for every integer $k \geq 3$ there is an integer $n(k)$ such that every regular drawing of K_n with $n \geq n(k)$ contains a k-face?

For every integer k with $3 \leq k \leq 19$, Shannon and Sloane [14] computed the value $a(k)$, which is the smallest n such that the regular drawing of K_n contains a k-face; see Table 1. Note that even if $a(k)$ exists, $n(k)$ might not. Those computations suggest that the answer to Problem 4.2 might be positive. In such a case, it would be interesting to determine the growth rate of $n(k)$ with respect to k. It follows from Proposition 3.1 and Theorem 3.4 that $n(3) = 3$, $n(4) = 6$, and $n(5) = 13$. We encourage the reader to visit website[1] to see the regular drawings for themselves.

For k odd, we trivially have $a(k) = k$ as the regular drawing of K_n with n odd contains an n-face in the center. It might be interesting to explore the size of the largest faces in such drawings if we exclude this n-face.

A more difficult version of Problem 4.2 would be to determine, for a given $k \geq 3$, all values of n such that every regular drawing of K_n contains a k-face.

Another possible direction is to count the minimum number of k-faces in a convex drawing of K_n. For example, regarding 3-faces, it is simple to show that there are always at least $n(n-3)$ by considering the area of a convex drawing around its 3-face as long as $n \geq 3$, but what is the growth rate of the minimum number of 3-faces with respect to n?

▶ Problem 4.3. What is the minimum number of 3-faces in a convex drawing of K_n? What if the drawing is generic or regular?

In the whole paper, we focused on convex drawings. The problems we considered can also be stated for all rectilinear drawings of K_n. Here, we can show that every generic rectilinear drawing of K_n with $n \geq 10$ contains a k-face with $k \geq 5$. This follows easily since, by a result

[1] fklute.com/regularkn.html

of Harborth [9], every set P of at least 10 points in the plane without three collinear contains a *5-hole*, that is, a set H of 5 points in convex position with no point of P in the interior of the convex hull of H. If we then apply this result on the vertex set of a generic rectilinear drawing of K_n and use a similar reasoning as in the proof of Theorem 3.2 on the drawing induced by the resulting 5-hole in D, then we find a bounded face of size at least 5 in D.

Finally, we considered the problem of finding a bounded face of size exactly k for a given integer k, but it also makes sense to consider more relaxed variants of the above problems where we want to find a bounded face of size at least k for a given integer k. In particular, this leads to the following potentially simpler variant of Problem 4.1.

▶ Problem 4.4. Is there a positive integer n_1 such that for every $n \geq n_1$ every convex drawing of K_n contains a bounded face of size at least 5?

We note that a simple double-counting argument based on Euler's formula yields the existence of k-faces in generic convex drawings of K_n with $k \geq 4$. If we knew that there are many 3-faces in such drawings, then the argument gives the existence of k-faces with $k \geq 5$. This also illustrates that some insight for Problem 4.3 might have consequences for our original questions.

5 Proof of Theorem 3.3

We prove that, for every positive integer n, there is a generic convex drawing of K_n that does not contain a k-face with $k \geq 6$. We apply a similar construction to the one used by Balko et al. [3].

First, we state some auxiliary definitions. For an integer $k \geq 3$, a set of k points in the plane is a *k-cup* if all its points lie on the graph of a convex function. Similarly, a set of k points is a *k-cap* if all its points lie on the graph of a concave function. Clearly, k-cups and k-caps are sets of points in convex position. A convex polygon P is *k-cap free* if no k vertices of P form a k–cap. Note that P is k-cap free if and only if it is bounded from above by at most $k-2$ segments (edges of P). Analogously, P is *k-cup free* if no k vertices of P form a k–cup. Observe that vertices of a k-face determine an a-cap and a u-cup that share the leftmost and the rightmost vertex and satisfy $a + u = k + 2$. We use $e(P)$ to denote the leftmost edge bounding P from above; see part (a) of Figure 2.

We inductively construct a certain generic convex drawing D_n of K_n with vertices represented by points $p_1, \ldots, p_n$ that form an n-cup in the plane and their x-coordinates satisfy $x(p_i) = i$; see part (b) of Figure 2. Let $V(D_n)$ denote the vertex set of D_n. We recall that we identify the vertices of K_n and the points from D_n representing them. We let $V(D_1) = \{(1,0)\}$ and $V(D_2) = \{(1,0),(2,0)\}$. Now, assume that we have already constructed the drawing D_{n-1} with $V(D_{n-1}) = \{p_1, \ldots, p_{n-1}\}$ for some integer $n \geq 3$. We choose a sufficiently large number y_n, and we let p_n be the point (n, y_n). We then set $V(D_n) = V(D_{n-1}) \cup \{p_n\}$ and we let D_n be the drawing of K_n on this vertex set. The number y_n is chosen large enough so that the following three conditions are satisfied:

1. for every $i = 1, \ldots, n-1$, every intersection point of two line segments spanned by points from $V(D_{n-1})$ lies on the left side of the line $\overline{p_i p_n}$ if and only if it lies to the left of the vertical line $x = i$ containing the point p_i,
2. if F is a 4-cap free face of D_n that is not 3-cap free, then there is no point p_i below the (relative) interior of $e(F)$,
3. no crossing of two edges of D_n lies on the vertical line containing some point p_i.

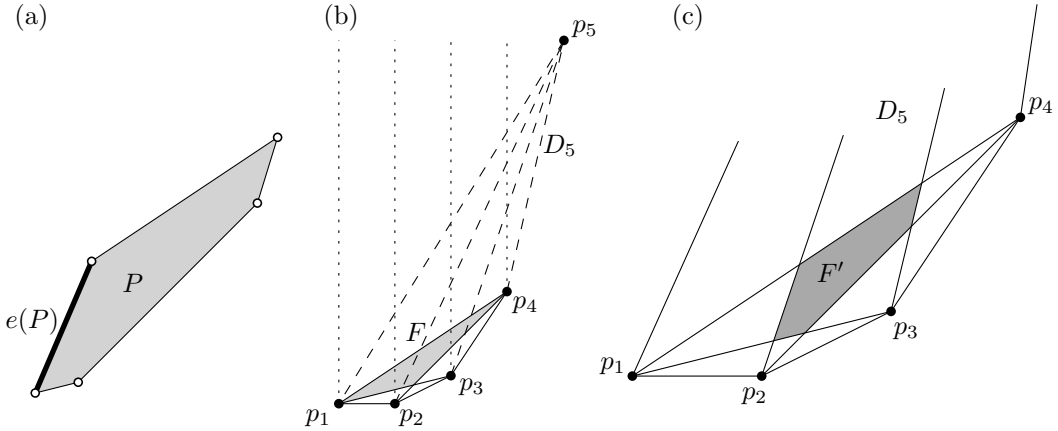


Figure 2 (a) A 4-cap free and 5-cup free polygon P that is not 3-cap free nor 4-cup free. (b) A construction of the drawing D_n for $n = 5$. If the point p_n is chosen sufficiently high above $V(D_{n-1})$, then each line segment $\overline{p_i p_n}$ with $i < n$ is very close to the vertical line containing p_i and thus all faces of D_n will be 4-cap free and 5-cup free. (c) The face F of D_{n-1} is split into new faces of D_n and contains the face F' that is 4-cap free and 5-cup free but not 3-cap free nor 4-cup free.

Choosing the point p_n is indeed possible as for a sufficiently large y-coordinate y_n of p_n we get that for each i, all the intersections of the line segments $p_i p_n$ with line segments of D_{n-1} lie very close to the vertical line $x = i$ containing the point p_i. Note that no line segment of D_n is vertical and that there are no heavy crossings in D_n. Since $p_1, \ldots, p_n$ form an n-cup, they are in convex position and D_n is a generic convex drawing of K_n.

It remains to prove that there are no k-faces with $k \geq 6$ in D. To show that, we use the following lemma.

▶ **Lemma 5.1.** *Each bounded face of D_n is a 4-cap free and 5-cup free convex polygon.*

Now, suppose for contradiction that there is a k-face F in D_n for some integer $k \geq 6$. By Lemma 5.1, the face F is a 4-cap free and 5-cup free convex polygon. On the other hand, the vertex set of F is in convex position and thus determines an a-cap and a u-cup that share the leftmost and the rightmost vertex and satisfy $a + u \geq 8$. Therefore, we either have $a \geq 4$ or $u \geq 5$, However, this contradicts the fact that F is 4-cap free and 5-cup free.

Acknowledgments. We would like to thank the organizers of the the 18th European Research Week on Geometric Graphs (GG Week 2023) where this research was initiated. We are very grateful to Alexandra Weinberger, and Daniel Perz for useful discussions and for their help with finalizing the paper.

References

1 E. M. Arkin, D. Halperin, K. Kedem, J. S. B. Mitchell, and N. Naor. Arrangements of segments that share endpoints: single face results. *Discrete Comput. Geom.*, 13(3-4):257–270, 1995.

2 B. Aronov, H. Edelsbrunner, L. J. Guibas, and M. Sharir. The number of edges of many faces in a line segment arrangement. *Combinatorica*, 12(3):261–274, 1992.

3 M. Balko, S. Chaplick, R. Ganian, S. Gupta, M. Hoffmann, P. Valtr, and A. Wolff. Bounding and computing obstacle numbers of graphs. In *30th annual European Symposium on Algorithms*, volume 244 of *LIPIcs. Leibniz Int. Proc. Inform.*, pages Art. No. 11, 13. Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern, 2022.

4 M. Balko, J. Cibulka, and P. Valtr. Drawing graphs using a small number of obstacles. *Discrete Comput. Geom.*, 59(1):143–164, 2018.

5 Z. Füredi and I. Palásti. Arrangements of lines with a large number of triangles. *Proceedings of the american mathematical society*, 92(4):561–566, 1984.

6 M. Griffiths. Counting the regions in a regular drawing of $K_{n,n}$. *J. Integer Seq.*, 13(8):Article 10.8.5, 8, 2010.

7 H. T. Hall. *Counterexamples in discrete geometry.* Phd thesis, University of California, Berkeley, 2004.

8 H. Harborth. Diagonalen im regulären n-Eck. *Elem. Math.*, 24:104–109, 1969.

9 H. Harborth. Konvexe Fünfecke in ebenen Punktmengen. *Elemente der Mathematik*, 33:116–118, 1978.

10 J. Matoušek. Blocking visibility for points in general position. *Discrete Comput. Geom.*, 42(2):219–223, 2009.

11 J. Matoušek and P. Valtr. The complexity of the lower envelope of segments with h endpoints. In *Intuitive geometry (Budapest, 1995)*, volume 6 of *Bolyai Soc. Math. Stud.*, pages 407–411. János Bolyai Math. Soc., Budapest, 1997.

12 B. Poonen and M. Rubinstein. The number of intersection points made by the diagonals of a regular polygon. *SIAM J. Discrete Math.*, 11(1):135–156, 1998.

13 A. Pór and D. R. Wood. On visibility and blockers. *J. Comput. Geom.*, 1(1):29–40, 2010.

14 S. R. Shannon and N. J. A. Sloane. Sequence A342222 in The On-Line Encyclopedia of Integer Sequences, 2021. `https://oeis.org/A342222`.

A Universal Construction for Unique Sink Orientations*

Michaela Borzechowski[1], Joseph Doolittle[2], and Simon Weber[3]

1 **Institut für Informatik, Freie Universität Berlin, Germany**
`michaela.borzechowski@fu-berlin.de`
2 **Institut für Geometrie, Technische Universität Graz, Austria**
`nerdyjoe@gmail.com`
3 **Department of Computer Science, ETH Zürich, Switzerland**
`simon.weber@inf.ethz.ch`

Abstract

Unique Sink Orientations (USOs) of cubes capture the combinatorial structure of many essential algebraic and geometric problems. It is crucial to have systematic constructions of USOs for various structural and algorithmic questions, including enumeration of USOs and algorithm analysis. While some construction methods for USOs already exist, each one of them has some significant downside. Inspired by cube tilings of space, we expand upon existing techniques to develop *generalized rewriting rules* for USOs. These rewriting rules are a new construction framework which can be applied to all USOs. Furthermore, they can generate every USO using only USOs of lower dimension.

Related Version arXiv:2211.06072

1 Introduction

A Unique Sink Orientation (USO) is an orientation of the hypercube graph, such that every non-empty face (subcube) has a unique sink. See Figure 1 for an example. USOs were first defined by Szabó and Welzl in 2001 [21]. They encode the combinatorial structure of several problems, for examples the P-matrix linear complementarity problem, linear programming, and many more [8, 11, 13, 17, 19]. USOs have also attracted attention as purely combinatorial objects, with interest in structural and algorithmic directions [2, 5, 6, 7, 9, 10, 16, 18].

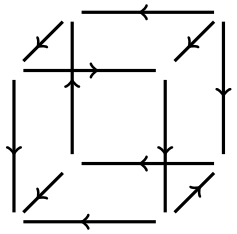


Figure 1 A Unique Sink Orientation of the 3-cube.

On the structural side, enumerating and sampling USOs are important unsolved challenges. The main issues are that USOs are hard to recognize [9] and while they are very sparse

* Michaela Borzechowski is supported by the German Research Foundation DFG within the Research Training Group GRK 2434 *Facets of Complexity*. Joseph Doolittle was supported by the Austrian Science Fund FWF, grant P 33278. Simon Weber is supported by the Swiss National Science Foundation under project no. 204320.

40th European Workshop on Computational Geometry, Ioannina, Greece, March 13–15, 2024.
This is an extended abstract of a presentation given at EuroCG'24. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

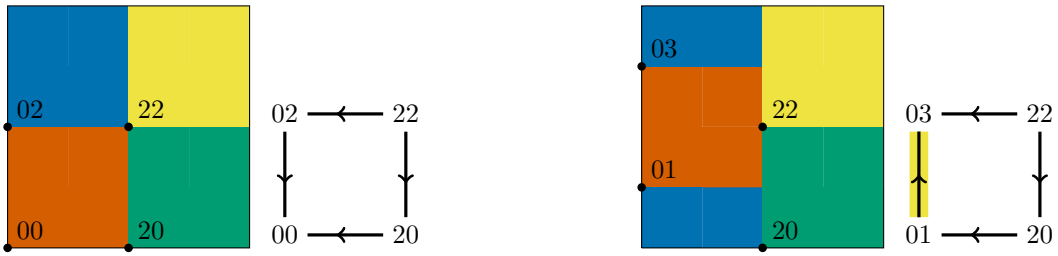


Figure 2 Two examples of $4\mathbb{Z}^2$-periodic tilings of $\mathbb{R}^2$ with their corresponding USOs.

among all cube orientations, there still exists a doubly exponential number (in terms of the cube dimension) of them [16].

A few systematic construction methods for USOs are known: the *product construction* [18], *inherited orientations*, *flipping* all edges of one dimension at once [21] and flipping equivalence classes of edges (called *phases*) that preserves the USO condition [17]. Of all these methods, only the product construction is able to increase the dimension of the USO, and only phase flips are capable of theoretically generating all USOs of a fixed dimension — however no systematic strategy for this is known and the mixing rate of the natural Markov chain based on phase flips remains unknown too. We discuss the existing construction methods more in-depth in the full version of the paper [1].

Results. Based on a remark of Schurr [17], we prove a one-to-one correspondence between so-called $4\mathbb{Z}^k$*-periodic tilings* and k-dimensional USOs. Representations of these tilings can be manipulated in the language of string rewriting, in particular this technique was used to disprove Keller's conjecture on unit cube tilings [12, 14, 15]. We generalize these construction techniques and translate them into the language of USOs. Our generalization provides a very general framework with many parameters, and every choice of parameters is a new construction which can be applied to any USO. Given both 1-dimensional USOs and a specific 2-dimensional USO (the *bow*), repeated application of constructions from our framework can be used to generate all USOs of dimension $k \geq 1$, we thus call our framework *universal*.

In the full version [1], we additionally show that we can realize all existing constructions that are applicable to all USOs as special cases of our framework. We also point out another special case of our construction as a new dimension-preserving modification, the *partial swap*.

2 Unit Cube Tilings and USOs

In a $4\mathbb{Z}^k$*-periodic tiling* [20] we tile the k-cube C of side length 4 by 2^k integer-grid aligned k-cubes (*tiles*) of side length 2, such that (i) every point of K is contained in at least one tile, and (ii) if a point is contained in multiple tiles, it lies on the boundary of all such tiles. These tiles may wrap around the boundary of C, exiting on one side and entering again on the opposite side (see Figure 2). This then defines a periodic tiling of $\mathbb{R}^k$, as infinitely repeating the tiling of C fills $\mathbb{R}^k$.

It was shown by Szabó [20] that Keller's conjecture [12] — a conjecture claiming that all cube tilings of $\mathbb{R}^k$ contain two tiles that share a facet (so-called *twins*) — can be decided by only considering these $4\mathbb{Z}^k$-periodic tilings. Note that Keller's conjecture has since been resolved and is known to hold up to dimension 7 [3] and fail for dimensions 8 and above [15].

A $4\mathbb{Z}^k$-periodic tiling can be described by a set of 2^k strings in $\{0, 1, 2, 3\}^k$, each string describing the coordinates of the bottom left corner of one tile. A set of strings describes a

valid $4\mathbb{Z}^k$-periodic tiling if and only if for every pair of strings, there is at least one coordinate in which the integer entries differ by exactly 2 [4, 14].

Schurr [17] briefly mentioned a bijection between $4\mathbb{Z}^k$-periodic tilings and USOs of the k-cube. We want to make this more explicit.

A set of strings describing a tiling also describes an orientation as follows. Each string $s \in \{0,1,2,3\}^k$ describes one vertex of the k-cube and the orientation of its incident edges:

- If $s_i = 0$ or $s_i = 1$, then s is in the lower i-facet of the cube.
- If $s_i = 2$ or $s_i = 3$, then s is in the upper i-facet of the cube.
- If $s_i = 0$ or $s_i = 2$, then the edge from s in dimension i is *downwards* oriented.
- If $s_i = 1$ or $s_i = 3$, then the edge from s in dimension i is *upwards* oriented.

In other words, the two bits of the binary encoding of s_i encode the location and the orientation of the vertex in dimension i, respectively. Equivalently, we can also retrieve a $4\mathbb{Z}^k$-periodic tiling from a k-cube and its orientation. See Figure 2 for an example and note that we always mark upwards edges by a yellow background.

It remains to show that the set of strings describes an USO if and only if the tiling it describes is valid. To see this, we use the characterization of USOs by the Szabó-Welzl condition [21]: An orientation is USO if and only if for each pair of distinct vertices v, w, there exists a dimension i in the subcube they span such that they both have the same up-map in that dimension, i.e., both have an upwards i-edge or both have a downwards i-edge. This corresponds directly to the condition that any pair of strings s, t differs by exactly 2 in the i'th coordinate:

- The difference of two strings s_i and t_i is greater than 1 if and only if the vertices s and t lie in different i-facets, and thus i is a dimension of the subcube these vertices span.
- The difference of two strings s_i and t_i is even if and only if the vertices s and t agree on the orientation of their incident edge in dimension i.

Combining these two conditions yields that s_i and t_i differ by exactly 2, as desired. Thus, the Szabó-Welzl condition is equivalent to the condition for tiling validity.

3 Rewriting Rules

The first disproof of Keller's conjecture by Lagarias and Shor [14] used string rewriting to create higher dimensional tilings from lower dimensional tilings. In this section, we generalize their technique to operations that can be applied to all USOs, so-called *generalized rewriting rules*. We first define *simple rewriting rules*, which are used to rewrite a single digit in each string of an USO. To define such a rule we need four lists, which specify what to replace each possible digit with. From Lagarias and Shor's approach we extract the conditions necessary to hold for these four lists, such that the result is again an USO.

▶ **Definition 3.1.** Let $S^{(0)}, S^{(1)}, S^{(2)}, S^{(3)} \subseteq \{0,1,2,3\}^d$ with the properties that

(i) $\left(S^{(0)} \cup S^{(2)}\right)$ defines a d-dimensional USO (a $4\mathbb{Z}^d$-periodic tiling) and $S^{(0)} \cap S^{(2)} = \emptyset$, and
(ii) $\left(S^{(1)} \cup S^{(3)}\right)$ defines a d-dimensional USO (a $4\mathbb{Z}^d$-periodic tiling) and $S^{(1)} \cap S^{(3)} = \emptyset$.

The sets $\left(S^{(0)}, S^{(1)}, S^{(2)}, S^{(3)}\right)$ define a *simple rewriting rule*. We define the function S_h to *apply* this simple rewriting rule to a k-dimensional input USO K on dimension $h \in [k]$. It maps subsets of $\{0,1,2,3\}^k$ to subsets of $\{0,1,2,3\}^{k+d-1}$. Applying the simple rewriting rule to a single vertex of the input USO is defined as follows:

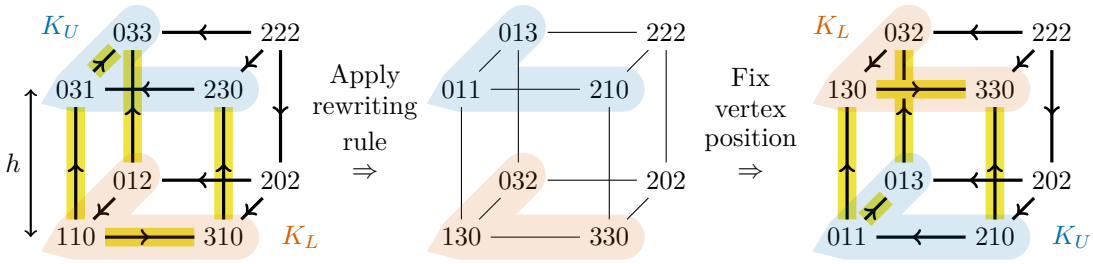
$$S_h(v) \coloneqq \left\{ v_1, \ldots, v_{h-1}, s_1, \ldots, s_d, v_{h+1}, \ldots, v_k \mid s \in S^{(v_h)} \right\}.$$

We write $S_h(K)$ (for a set $K \subseteq \{0,1,2,3\}^k$) for the union of the outputs of S_h when applied to all elements of K, i.e., $S_h(K) \coloneqq \bigcup_{v \in K} S_h(v)$.

For each string $v \in K$, $S_h(v)$ produces a set of strings which depends on the value of the entry v_h. For each element s of $S^{(v_h)}$, a string is generated by replacing the entry v_h with s. The single vertex v is thus mapped to $|S^{(v_h)}|$ vertices. Note that some of the sets $S^{(\cdot)}$ may be empty. In this case, when $|S^{(v_h)}| = 0$, no strings are produced from v.

A particularly interesting operation on USOs is the following 1-dimensional rewriting rule, which we call the *partial swap*: $(S^{(0)} = \{0\}, S^{(1)} = \{3\}, S^{(2)} = \{2\}, S^{(3)} = \{1\})$. We analyze this rewriting rule in more detail in the full version of this paper [1].

▶ **Example 3.2.** To the USO $K = \{110, 310, 012, 202, 031, 230, 033, 222\}$ we apply the partial swap in dimension $h = 2$, i.e., we rewrite the second coordinate of each vertex. In the resulting USO, the subgraphs K_L and K_U swapped places.



In the full version [1] we show the following lemma, i.e., that simple rewriting rules are correct USO constructions.

▶ **Lemma 3.3.** *Applying any rewriting rule S_h to an USO K of strings in $\{0,1,2,3\}^k$ results in a valid USO $S_h(K)$ of strings in $\{0,1,2,3\}^{k+d-1}$.*

3.1 Generalized Rewriting Rules

To arrive at their counterexamples to Keller's conjecture, Lagarias and Shor used a more general rewriting technique [14]. They do not only use the four digits $0, 1, 2, 3$ in their input tiling, but also "alternative digits" $0'$ and $1'$, which only differ from their normal counterparts for the purposes of the rewriting, but specify the same coordinate for the tiling. We generalize our construction based on this idea, by letting the input USO specify one of i labels at each vertex.

For the full generality of our rewriting framework, the sets $S^{(m)}$ are replaced by a list of i sets $S^{(m)}_{1,\dots,i}$ each, where the indices correspond to the possible labels attached to the vertices of the input USO. All the compatibility requirements are appropriately expanded.

▶ **Definition 3.4.** Let $d, i \in \mathbb{N}$, $S^{(0)}_{1,\dots,i}, S^{(1)}_{1,\dots,i}, S^{(2)}_{1,\dots,i}, S^{(3)}_{1,\dots i} \subseteq \{0,1,2,3\}^d$ where

(i) $\left(S^{(0)}_j \cup S^{(2)}_{j'}\right)$ defines a d-dimensional USO and $S^{(0)}_j \cap S^{(2)}_{j'} = \emptyset$ for all pairs $j, j' \in [i]$, and

(ii) $\left(S^{(1)}_j \cup S^{(3)}_{j'}\right)$ defines a d-dimensional USO and $S^{(1)}_j \cap S^{(3)}_{j'} = \emptyset$ for all pairs $j, j' \in [i]$.

The sets $\left(S^{(0)}_{1,\dots,i}, S^{(1)}_{1,\dots,i}, S^{(2)}_{1,\dots,i}, S^{(3)}_{1,\dots i}\right)$ define a *generalized rewriting rule*. We define the function T_h to *apply* this generalized rewriting rule to a k-dimensional input USO K on dimension $h \in [k]$. It maps subsets of $\{0,1,2,3\}^k \times [i]$ to subsets of $\{0,1,2,3\}^{k+d-1}$. Applying the generalized rewriting rule to a single vertex v labeled j of the input USO is defined as:

$$T_h(v, j) \coloneqq \left\{ v_1, \dots, v_{h-1}, t_1, \dots, t_d, v_{h+1}, \dots, v_k \mid t \in S^{(v_h)}_j \right\}.$$

We extend the function T_h from single inputs to sets similarly to Definition 3.1.

Note that we can use duplicate sets $S_j^{(m)} = S_{j'}^{(m)}$ in case we want to have fewer than i sets for some $m \in \{0,1,2,3\}$. Lemma 3.3 holds also for generalized rewriting rules, with the proof applying *mutatis mutandis* since Definition 3.4 provides the necessary disjointness and coherence conditions:

▶ **Lemma 3.5.** *Let K be an USO of strings in $\{0,1,2,3\}^k$, and $L : K \to [i]$ an additional labelling function. Then $T_h(K,L)$ is an USO of strings in $\{0,1,2,3\}^{k+d-1}$.*

Intuitively, the effect of a generalized rewriting rule can be described as follows. Given an input USO K and a rewriting rule, we replace edges of dimension h. For simplicity, we focus on a single 2-face containing two edges of this dimension h: $\{v_1, v_2\}$ and $\{w_1, w_2\}$. The rewriting rule replaces those two h-edges by the d-dimensional USOs $T_h(\{v_1, v_2\}, L)$ and $T_h(\{w_1, w_2\}, L)$. Instead of the edges $\{v_1, w_1\}$ and $\{v_2, w_2\}$, there are now 2^d new edges between $T_h(\{v_1, v_2\}, L)$ and $T_h(\{w_1, w_2\}, L)$ as can be seen in Figure 3.

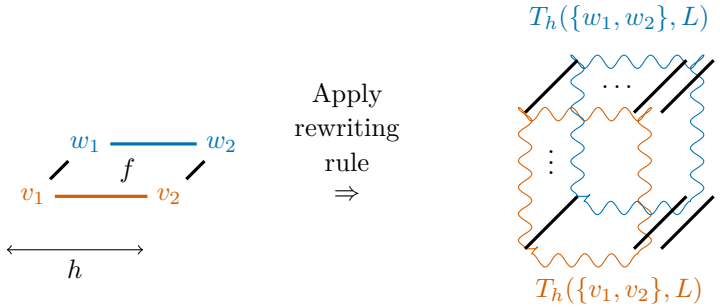


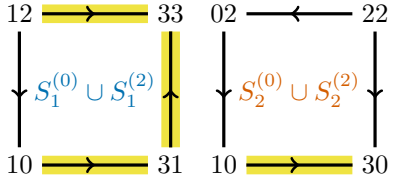
Figure 3 Sketch of the effect of the generalized rewriting rule on the 2-face f.

It holds that if $\{v_1, v_2\}$ is a downwards edge, $T_h(\{v_1, v_2\}, L) = S^{(0)}_{L(v_1)} \cup S^{(2)}_{L(v_2)}$. If $\{v_1, v_2\}$ is an upwards edge, $T_h(\{v_1, v_2\}, L) = S^{(1)}_{L(v_1)} \cup S^{(3)}_{L(v_2)}$. Analogously, the edge $\{w_1, w_2\}$ is replaced by the respective union of sets. In either case, this is guaranteed to be an USO by the conditions (i) and (ii) in Definition 3.4.

The edges between the USOs $T_h(\{v_1, v_2\}, L)$ and $T_h(\{w_1, w_2\}, L)$ copy their orientation either from the edge $\{v_1, w_1\}$ or from the edge $\{v_2, w_2\}$. Which of these edges is copied depends on whether the resulting edge is incident to a vertex in $T_h(\{v_1\}, L)$ or $T_h(\{v_2\}, L)$. This depends on how the sets $S_j^{(0)} \cup S_{j'}^{(2)}$ (and $S_j^{(1)} \cup S_{j'}^{(3)}$ respectively) are split into their parts, i.e., which vertices of the d-USOs they describe lie in which set. Note that these unions are split the same way, no matter j and j'.

▶ **Example 3.6.** The following is a generalized rewriting rule for $d = 2$.

$$S_1^{(0)} = S_1^{(1)} = \{10\}$$
$$S_1^{(2)} = S_1^{(3)} = \{12, 33, 31\}$$
$$S_2^{(0)} = S_2^{(1)} = \{10\}$$
$$S_2^{(2)} = S_2^{(3)} = \{02, 22, 30\}$$



We apply this rewriting rule to dimension $h = 1$ of the *bow* $K = \{01, 20, 03, 22\}$ with the labeling function $L(01) = 2, L(03) = 1, L(20) = 2$ and $L(22) = 1$. This means, we replace the first coordinate of each vertex. The result is $T_1(K, L) = \{101, 300, 020, 220, 103, 122, 312, 332\}$.

4 Universality of the Construction

Our construction is universal, meaning it is sufficiently general to generate all USOs, using only the 1-dimensional USOs, and the 2-dimensional "bow" as base cases.

▶ **Theorem 4.1** (Universality). *Starting with the set of both 1-dimensional USOs* $\{0,2\}$ *and* $\{1,3\}$*, one can generate every USO of dimension* $n \geq 1$ *by repeated application of generalized rewriting rules to the bow* $\{01,20,03,22\}$*, where every set* $S_j^{(m)}$ *used in a rewriting rule is a subset of some set of strings describing an USO already obtained before.*

To prove this theorem, we show the following lemma in the full version [1], which states that for any n-dimensional USO there exists a generalized rewriting rule which creates this USO by only using $(n-1)$-dimensional USOs and the bow. From Lemma 4.2, Theorem 4.1 follows as a direct consequence.

▶ **Lemma 4.2.** *Let* K *be an* n*-dimensional USO. Then there exists a generalized rewriting rule* $\left(S_{1,2}^{(0)}, S_{1,2}^{(1)}, S_{1,2}^{(2)}, S_{1,2}^{(3)}\right)$*, where each* $S_j^{(m)}$ *is a (partial)* $(n-1)$*-dimensional USO, and*

$$K = T_1(\mathit{bow} = \{01,20,03,22\}, L), \text{ for } L(01) = 2, L(03) = 1, L(20) = 2, L(22) = 1.$$

Proof (sketch). With the input labelling L, each set $S_{1,2}^{(0)}$ and $S_{1,2}^{(2)}$ is used to rewrite exactly one string of the bow. Furthermore, each of these strings has a unique digit in the second coordinate. We ignore the unused sets $S_{1,2}^{(1)}$ and $S_{1,2}^{(3)}$, and define $S_{1,2}^{(0)}$ and $S_{1,2}^{(2)}$ by simply splitting the strings of our target USO K depending on their last digit (and discarding that last digit), i.e., depending on the two n-facets of K and the edges between them:

- $S_1^{(0)}$: rewrites 03, contains vertices of the *upper* n-facet of K with an *upwards* n-edge.
- $S_2^{(0)}$: rewrites 01, contains vertices of the *lower* n-facet of K with an *upwards* n-edge.
- $S_1^{(2)}$: rewrites 22, contains vertices of the *upper* n-facet of K with an *downwards* n-edge.
- $S_2^{(2)}$: rewrites 20, contains vertices of the *lower* n-facet of K with an *downwards* n-edge.

Thus, when applied to the bow, we end up with exactly our target USO. It remains to prove that these sets form a valid generalized rewriting rule. For this, we can show that $S_1^{(0)} \cup S_1^{(2)}$ and $S_2^{(0)} \cup S_2^{(2)}$ are the two n-facets of K, while the other set combinations are the n-facets of K after applying a *partial swap*. For details we refer to the full version [1]. ◀

5 Future Work

Unfortunately, our rewriting rules exhibit a similar weakness to the phase flips of Schurr. While they are universal and each step in the universality proof is very systematic, our construction does not yet provide a suitable way to enumerate all USOs. This is in part because checking conditions (i) and (ii) of Definition 3.4 is computationally expensive. As future work, we suggest searching for more interesting special cases of (generalized) rewriting

rules, or for other more systematic ways to enumerate USOs. Our framework could also be further generalized to rewrite multiple dimensions at once, similar to the approach taken by Mackey [15] to find the 8-dimensional counterexample to Keller's conjecture.

References

1 Michaela Borzechowski, Joseph Doolittle, and Simon Weber. A universal construction for unique sink orientations, 2022. doi:10.48550/ARXIV.2211.06072.

2 Vitor Bosshard and Bernd Gärtner. Pseudo unique sink orientations, 2017. doi:10.48550/ARXIV.1704.08481.

3 Joshua Brakensiek, Marijn Heule, John Mackey, and David Narváez. The resolution of Keller's conjecture. *Journal of Automated Reasoning*, 66(3):277–300, Aug 2022. doi:10.1007/s10817-022-09623-5.

4 Keresztély Corrádi and Sándor Szabó. A combinatorial approach for Keller's conjecture. *Periodica Mathematica Hungarica*, 21(2):95–100, Jun 1990. doi:10.1007/BF01946848.

5 John Fearnley, Spencer Gordon, Ruta Mehta, and Rahul Savani. Unique end of potential line. *Journal of Computer and System Sciences*, 114:1–35, 2020. doi:10.1016/j.jcss.2020.05.007.

6 Bernd Gärtner. The random-facet simplex algorithm on combinatorial cubes. *Random Structures & Algorithms*, 20(3):353–381, 2002. doi:10.1002/rsa.10034.

7 Bernd Gärtner, Walter D. Morris jr., and Leo Rüst. Unique sink orientations of grids. *Algorithmica*, 51(2):200–235, 2008. doi:10.1007/s00453-007-9090-x.

8 Bernd Gärtner and Ingo Schurr. Linear programming and unique sink orientations. In *Proc. 17th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 749–757, 2006. doi:10.5555/1109557.1109639.

9 Bernd Gärtner and Antonis Thomas. The complexity of recognizing unique sink orientations. In Ernst W. Mayr and Nicolas Ollinger, editors, *32nd International Symposium on Theoretical Aspects of Computer Science (STACS 2015)*, volume 30 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 341–353, Dagstuhl, Germany, 2015. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.STACS.2015.341.

10 Bernd Gärtner and Antonis Thomas. The niceness of unique sink orientations. In Klaus Jansen, Claire Mathieu, José D. P. Rolim, and Chris Umans, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX-/RANDOM 2016)*, volume 60 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 30:1–30:14, Dagstuhl, Germany, 2016. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.APPROX-RANDOM.2016.30.

11 Bernd Gärtner and Emo Welzl. Explicit and implicit enforcing - randomized optimization. In *Computational Discrete Mathematics: Advanced Lectures*, pages 25–46. Springer Berlin Heidelberg, 2001. doi:10.1007/3-540-45506-X_3.

12 Ott-Heinrich Keller. Über die lückenlose erfüllung des raumes mit würfeln. *Journal für die reine und angewandte Mathematik*, 1930(163):231–248, 1930. doi:10.1515/crll.1930.163.231.

13 Lorenz Klaus. *A fresh look at the complexity of pivoting in linear complementarity*. PhD thesis, ETH Zürich, 2012. doi:10.3929/ethz-a-007604201.

14 Jeffrey C. Lagarias and Peter W. Shor. Keller's cube-tiling conjecture is false in high dimensions. *Bulletin of the American Mathematical Society*, 27(2):279–283, 1992. doi:10.1090/S0273-0979-1992-00318-X.

15 John Mackey. A cube tiling of dimension eight with no facesharing. *Discrete & Computational Geometry*, 28(2):275–279, Aug 2002. doi:10.1007/s00454-002-2801-9.

16 Jiří Matoušek. The number of unique-sink orientations of the hypercube. *Combinatorica*, 26(1):91–99, 2006. doi:10.1007/s00493-006-0007-0.

17 Ingo Schurr. *Unique Sink Orientations of Cubes*. PhD thesis, ETH Zürich, 2004. doi:10.3929/ethz-a-004844278.

18 Ingo Schurr and Tibor Szabó. Finding the sink takes some time: An almost quadratic lower bound for finding the sink of unique sink oriented cubes. *Discrete & Computational Geometry*, 31(4):627–642, 2004. doi:10.1007/s00454-003-0813-8.

19 Alan Stickney and Layne Watson. Digraph models of Bard-type algorithms for the linear complementarity problem. *Mathematics of Operations Research*, 3(4):322–333, 1978. URL: https://www.jstor.org/stable/3689630.

20 Sándor Szabó. Cube tilings as contributions of algebra to geometry. *Contributions to Algebra and Geometry*, 34(1):63–75, 1993.

21 Tibor Szabó and Emo Welzl. Unique sink orientations of cubes. In *Proceedings of the 42nd IEEE Symposium on Foundations of Computer Science*, pages 547–555, 2001. doi:10.1109/SFCS.2001.959931.

Counting Pseudoline Arrangements*

Fernando Cortés Kühnast, Stefan Felsner, and Manfred Scheucher

Institut für Mathematik, Technische Universität Berlin, Germany
lastname@math.tu-berlin.de

Abstract

Arrangements of pseudolines are classic objects in discrete and computational geometry. They have been studied with increasing intensity since their introduction almost 100 years ago. The study of the number B_n of non-isomorphic simple arrangements of n pseudolines goes back to Goodman and Pollack, Knuth, and others. It is known that B_n is in the order of $2^{\Theta(n^2)}$ and finding asymptotic bounds on $b_n = \frac{\log_2(B_n)}{n^2}$ remains a challenging task. In 2011, Felsner and Valtr showed that $0.1887 \leq b_n \leq 0.6571$ for sufficiently large n. The upper bound remains untouched but in 2020 Dumitrescu and Mandal improved the lower bound constant to 0.2083. Their approach utilizes the known values of B_n for up to $n = 12$.

We tackle the lower bound with a dynamic programming scheme. Our new bound is $b_n \geq 0.2526$ for sufficiently large n. The result is based on a delicate interplay of theoretical ideas and computer assistance.

1 Introduction

Levi [12] introduced arrangements of pseudolines as a natural generalization of line arrangements in 1926. An *arrangement of pseudolines* in the Euclidean plane $\mathbb{R}^2$ is a finite family of simple curves, called pseudolines, such that each curve approaches infinity in both directions and every pair intersects in exactly one point where the two curves cross. More generally, we call a collection of pseudolines *partial arrangement* if every pair intersects in at most one crossing-point. Pseudolines which do not intersect are said to be *parallel*. Note that, while for partial arrangements of *proper* lines the relation 'parallel' is transitive, this is no longer true in partial pseudoline arrangements.

In this article, the focus will be on *simple* arrangements, that is, no three or more pseudolines intersect in a common point (called *multicrossing*). Moreover, we consider all arrangements to be *marked*, that is, they have a unique marked unbounded cell, which is called *north-cell*. Two arrangements are *isomorphic* if one can be mapped to the other by an orientation preserving homeomorphism of the plane that also preserves the north-cell.

While it is known that the number B_n of non-isomorphic arrangements of n pseudolines grows as $2^{\Theta(n^2)}$, it remains a challenging problem to bound the multiplicative factor of the leading term of $\log_2 B_n = \Theta(n^2)$. Our focus will be on finding better estimates on the lower bound constant $c^- := \liminf_{n\to\infty} \frac{\log_2 B_n}{n^2}$. One can analogously define the upper bound constant $c^+ := \limsup_{n\to\infty} \frac{\log_2 B_n}{n^2}$ but it seems to be open whether c^+ and c^- coincide.

In the 1980's Goodman and Pollak [9] investigated pseudopoint configurations, which are dual to pseudoline arrangements, and established the lower bound $c^- \geq \frac{1}{8}$. An alternative and simpler construction for $c^- \geq \frac{1}{12}$ can be found in Matoušek's textbook [13, Chapter 6].

* The extended version of this work was accepted at SoCG'24 [11] and will be merged with Justin Dallant's manuscript 'Improved Lower Bound on the Number of Pseudoline Arrangements' [4] for the SoCG'24 proceedings. First steps towards the results presented here were made in the Bachelor's thesis of the first author [3]. S.F. was partially supported by DFG Grant FE 340/13-1. M.S. and F.C.K. were supported by DFG Grant SCHE 2214/1-1.

40th European Workshop on Computational Geometry, Ioannina, Greece, March 13–15, 2024.
This is an extended abstract of a presentation given at EuroCG'24. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

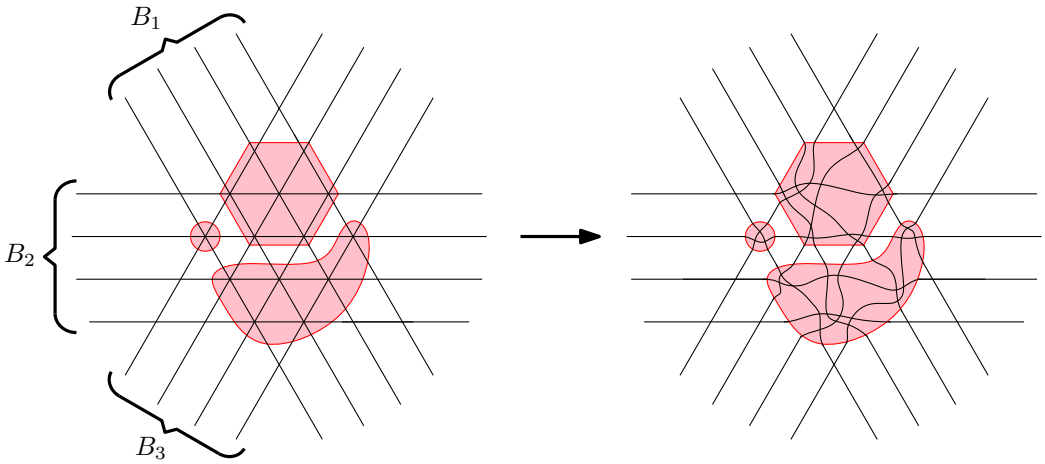


Figure 1 Left: An arrangement of 3 bundles of parallel lines and a collection of interior-disjoint patches (highlighted red) such that each multicrossing point is covered by a patch. Right: A partial pseudoline arrangement with the same parallel bundles obtained by rerouting within the patches.

Concerning the upper bound, Edelsbrunner, O'Rourke and Seidel [6] showed $c^+ < \infty$. In the 1990's Knuth [10, Section 9] improved the bounds to $c^- \geq \frac{1}{6}$ and $c^+ < 0.7925$, and he conjectured that $c^+ \leq 0.5$. The upper bound was lowered to $c^+ < 0.6974$ by Felsner [7], and in 2011, Felsner and Valtr [8] further narrowed the gap by showing $c^- > 0.1887$ and $c^+ < 0.6571$. In 2020 Dumitrescu and Mandal [5] proved the currently best lower bound $c^- > 0.2083$.

In this article, we make a substantial step on the lower bound by proving $c^- > 0.2526$.

▶ **Theorem 1.1.** *The number B_n of non-isomorphic simple arrangements of n pseudolines satisfies the inequality $B_n \geq 2^{cn^2 - O(n \log n)}$ with $c > 0.2526$.*

2 Outline

Our approach is in the spirit of several previous bounds. We consider a specific partial arrangement $\mathcal{L}$ of n lines consisting of k *bundles* $\mathcal{L}_1, \ldots, \mathcal{L}_k$ of parallel lines. We then define a class of local perturbations to $\mathcal{L}$ and consider the number of arrangements that can be obtained by these perturbations. This number is a lower bound on B_n, and it can be improved by recursively applying the same construction to each of the parallel classes $\mathcal{L}_i$.

The main difference between the approaches lies in the number of bundles k and the notion of locality. Matoušek and also Felsner and Valtr used three bundles but the locality was increased from considering just a triple intersection with its two simple resolutions to the full intersection pattern of three bundles. Dumitrescu and Mandal [5] increased the number k of bundles to up to 12 but still restricted to local resolutions of multicrossings.

Our approach combines higher values of k with an increased locality for the perturbations. As illustrated in Figure 1, we allow reroutings of the arrangement within designated regions, which we call *patches*. When rerouting the arrangement within a patch P, the order of the crossings along the pseudolines may change. The boundary information of P fully determines which pairs of pseudolines cross within P, but the order of crossings along the pseudolines is not determined in general. Outside of P, the arrangement remains unaffected, which allows us to count the number of reroutings for each patch independently. The total number of perturbations is obtained as the product of the numbers computed for the individual

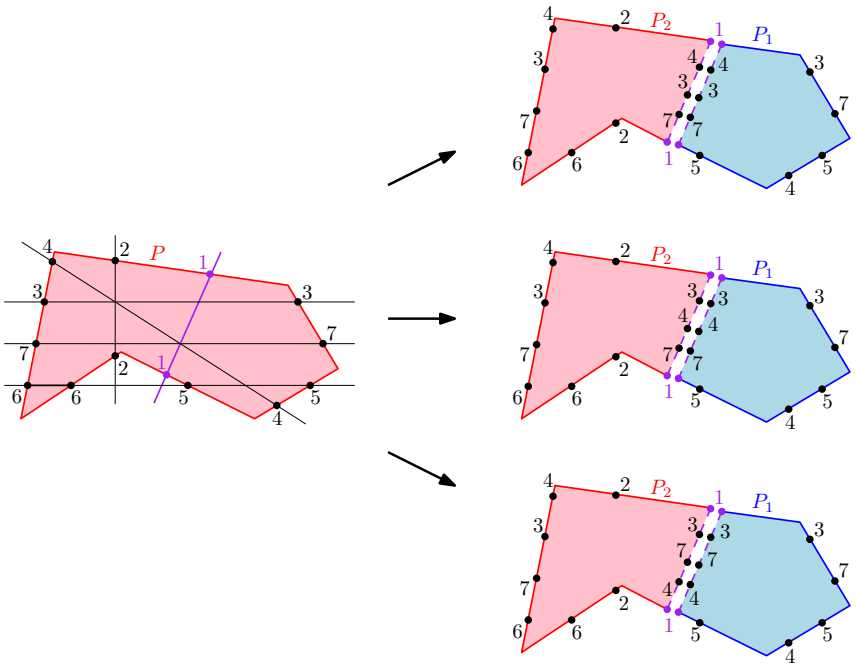


Figure 2 An illustration of how to recursively compute the number of reroutings for a patch P. When cutting along segment 1, highlighted purple, there are intersections with the segments 3, 4, and 7. As the segments 3 and 7 do no cross within P, there are only three possibilities for placing the three crossings along the segment 1, namely 4–3–7 (right top), 3–4–7 (right center) and 3–7–4 (right bottom).

patches. The number of possibilities within a patch are computed recursively via dynamic programming; Figure 2 gives an illustration. Details are given in [11].

To eventually use computer assistance, we choose patches of high regularity and reasonably small complexity. In fact, since our construction is highly regular, it is sufficient to determine the rerouting possibilities only for a small number of patch-types. Only a negligible fraction of patches along the boundaries are different. As we only want to find an asymptotic lower bound on B_n, the small number of irregular patches along the boundary of the regions will not be used in the counting.

To eventually prove Theorem 1.1, we perform the following two steps:

- In the first step (Section 3) we specify the parameters of the construction: We construct $k = 6$ bundles of $\lfloor \frac{n}{k} \rfloor$ parallel lines (see [11] for a description of the approach with $k = 4$ bundles) and cover the multicrossing points by patches. By resolving the multicrossing points within the patches, and taking the product over all patches we obtain an improved lower bound on the number $F_k(n)$ of partial arrangements with k bundles of $\lfloor \frac{n}{k} \rfloor$ parallel pseudolines.
- In the second step (Section 4), we account for crossings in bundles of pseudolines which had been parallel before. The product of the so-computed possibilities yields the improved lower bound on the number B_n of simple arrangements on n pseudolines.

3 Step 1: bundles of parallel lines, patches, and perturbations

For the start we fix an integer k and construct an arrangement $\mathcal{L}$ of k bundles of $\lfloor \frac{n}{k} \rfloor$ parallel lines as in [5]. If n is not a multiple of k, the remaining lines are discarded, or not used in the counting. We then cover all multicrossing points by a family of disjoint regions, called *patches*, and reroute the line segments within the patches so that all multicrossing points will eventually be resolved and the arrangement becomes simple.

3.1 Construction with 6 bundles

In this section we consider a partial arrangement $\mathcal{L}$ of n lines consisting of 6 bundles of $\lfloor \frac{n}{6} \rfloor$ parallel lines $\mathcal{L}_1, \ldots, \mathcal{L}_6$ following [5]. See Figure 3 for an illustration. The construction comes with four types of regions with multicrossings:

- R_i for $i \in \{3, 4, 5\}$ only contains multicrossings of order i and
- R_6 contains multicrossings of order 3 and 6.

Note that multicrossings of order 3 occur in R_3 and R_6.

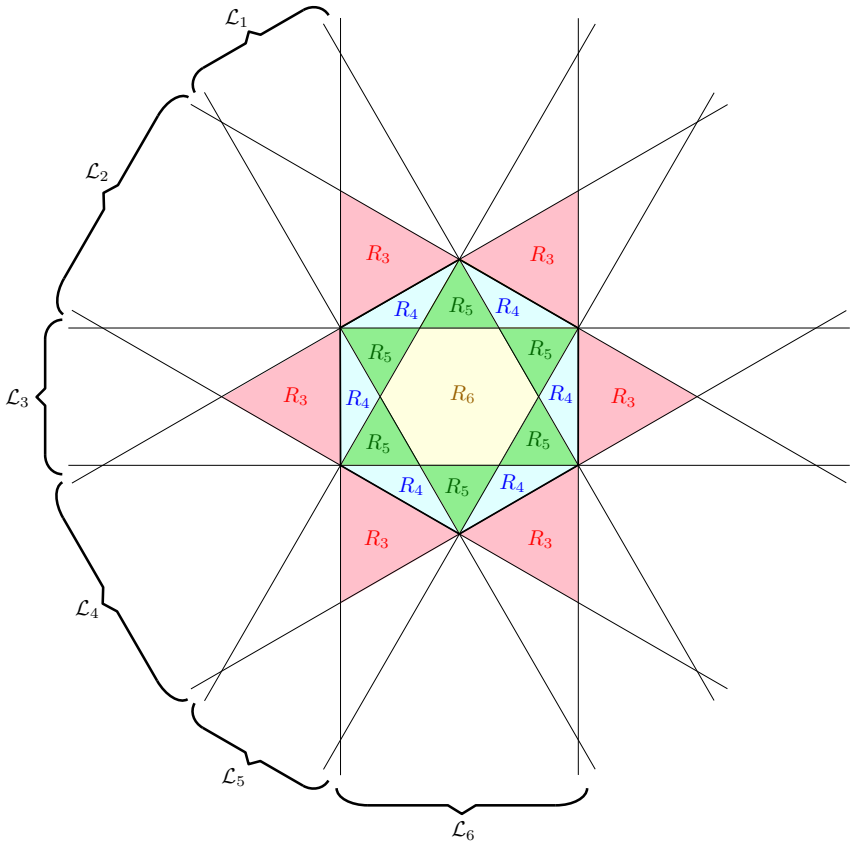


Figure 3 Construction with 6 bundles as in [5].

For each of the four regions R_i we will use a different type of patch P_i that is based on a regular tiling of the plane to ensure regularity; see Figure 4.

We have to determine the number μ_i of patches of type i. Since the number of crossings of each order is asymptotically quadratic in n and each patch contains only a constant number of crossings, the number μ_i of patches of type i is also quadratic. Again, it is important

to note that the patches along the boundary of R_i behave differently. However, since there are only linearly many of these deformed patches, they only affect lower order error terms. Hence we can omit them in the calculations.

To obtain asymptotically tight estimates on the μ_i's, we make use of the numbers $\lambda_i(n)$ of i-crossing points, which were determined by Dumitrescu and Mandal [5, Table 2]:

$$\lambda_3(n) = \tfrac{5n^2-O(n)}{144}, \qquad \lambda_4(n) = \tfrac{n^2-O(n)}{144}, \qquad \lambda_5(n) = \tfrac{n^2-O(n)}{144}, \qquad \lambda_6(n) = \tfrac{n^2-O(n)}{144}.$$

For $i = 4, 5, 6$, the number λ_i coincides with $\mu_i \cdot \#\{i\text{-fold crossings in } P_i\} + O(n)$ because only the region P_i contains i-crossings for $i = 4, 5, 6$. For $i = 3$, however, the situation is a bit more complicated because P_3 and P_6 both contains 3-crossings. More specifically, P_6 contains twice as many 3-crossings as 6-crossings. With the multiplicities given in the caption of Figure 4 we obtain:

- $\mu_3(P_3, n) = \frac{\lambda_3(n)-2\cdot\lambda_6(n)}{\#\{\text{3-crossings in } P_3\}} - O(n) = \frac{3n^2}{144\cdot 100} - O(n)$
- $\mu_4(P_4, n) = \frac{\lambda_4(n)}{\#\{\text{4-crossings in } P_4\}} - O(n) = \frac{n^2}{144\cdot 32} - O(n)$
- $\mu_5(P_5, n) = \frac{\lambda_5(n)}{\#\{\text{5-crossings in } P_5\}} - O(n) = \frac{n^2}{144\cdot 12} - O(n)$
- $\mu_6(P_6, n) = \frac{\lambda_6(n)}{\#\{\text{6-crossings in } P_6\}} - O(n) = \frac{n^2}{144\cdot 7} - O(n)$

To compute the numbers $F(P_i)$ of all possible perturbations within the patch type P_i for $i = 3, 4, 5, 6$, we ran our program and obtained:

- $F(P_3) = 19560554716747662490025595234371016704 00$
- $F(P_4) = 10233480626615962155895931163981261674$
- $F(P_5) = 32207077855497546508132740267$
- $F(P_6) = 81296061009729331372533303551 73$

We provide a computer-assisted framework [1] that can fully automatically compute $F(P)$ for a given patch P, which is given as an IPE input file [2]. See [11] for more details. The presented terms were computed within a few CPU hours on cluster nodes of TU Berlin with up to 1TB of RAM. We also provide simpler patches for which the program only needs few CPU seconds and low RAM. Those, however, give slightly worse bounds.

From $F_k(n) \geq \prod_{i=3}^{k} F(P_i)^{\mu_i(n)}$, we can now derive:

▶ **Proposition 3.1.** *$F_6(n) \geq 2^{cn^2-O(n)}$ with $c > 0.2105$.*

More specifically, by writing $c_i := \lim_{n\to\infty} \frac{\mu_i(n)}{n^2} \cdot \log_2(F(P_i))$, we can see the contributions of the patches P_3, P_4, P_5 and P_6 to the leading constant $c = c_3+c_4+c_5+c_6$ from Proposition 3.1:

$$c_3 \approx 0.0272, \qquad c_4 \approx 0.0267, \qquad c_5 \approx 0.0548, \qquad c_6 \approx 0.1019.$$

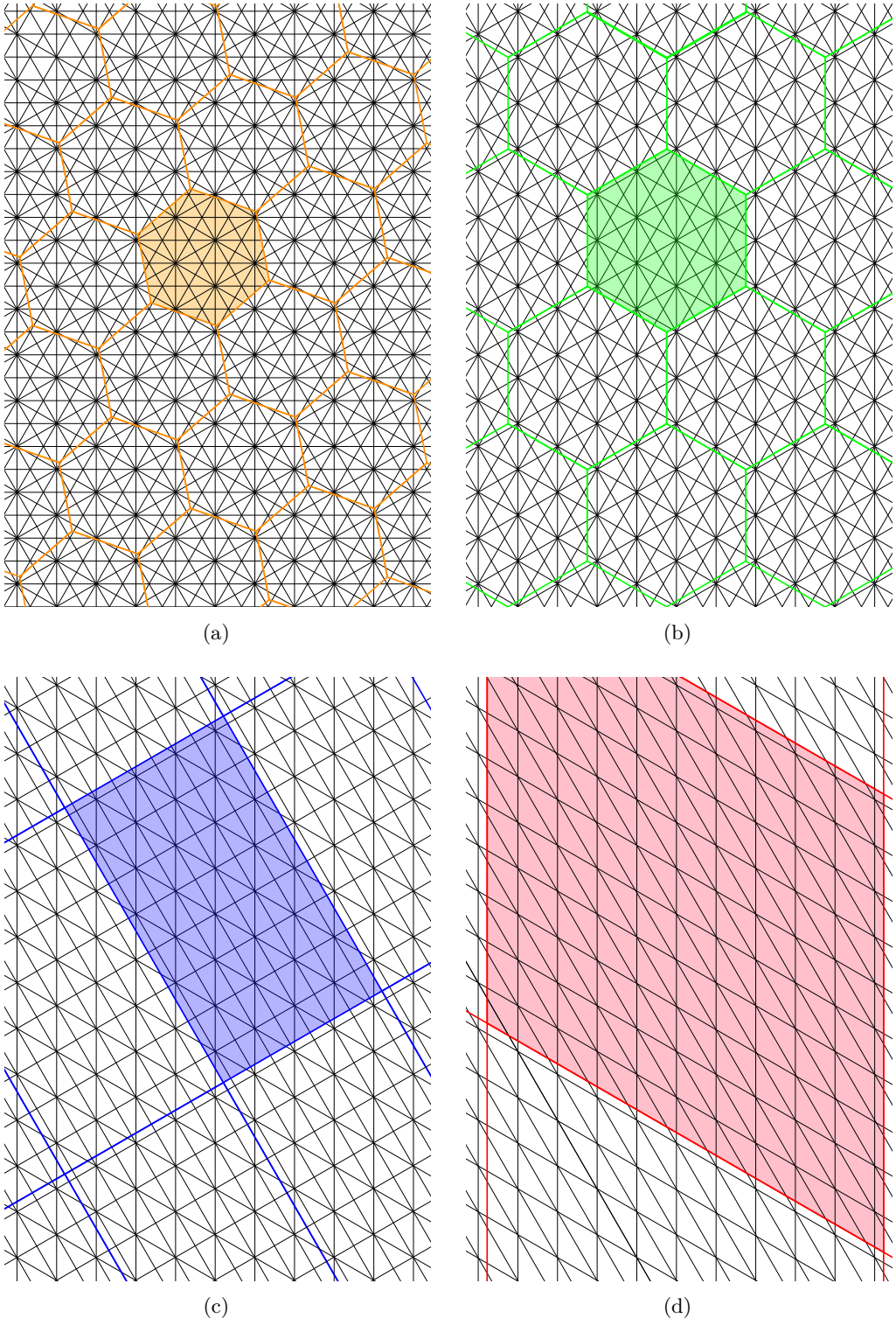


Figure 4 The four types of patches for our construction on $k = 6$ bundles:
(a) For R_6 we use a hexagonal tiling where each patch P_6 contains exactly 7 crossings of order 6 and 14 crossings of order 3.
(b) For R_5 we use a hexagonal tiling where each patch P_5 contains exactly 12 crossings of order 5.
(c) For R_4 we use a rectangular tiling where each patch P_4 contains exactly 32 crossings of order 4.
(d) For R_3 we use a rhombic tiling where each patch P_3 contains exactly 100 crossings of order 3.

4 Step 2: resolving parallel bundles

With the second and final step, we want to obtain a simple arrangement of pairwise intersecting pseudolines from a partial arrangement of k bundles of $m = \lfloor \frac{n}{k} \rfloor$ parallel pseudolines. To do so, we use a recursive scheme as in [8, 5] to make each pair of parallel pseudolines cross: For each $i = 1, \ldots, k$, we consider a disk D_i such that

(1) D_i intersects all parallel pseudolines of the bundle $\mathcal{L}_i$ and no other pseudolines, and
(2) no two disks overlap.

Within each disk D_i we can place any of the B_m arrangements of m pseudolines. This makes all the pseudolines of a bundle cross. Figure 5 gives an illustration for the case $k = 3$.

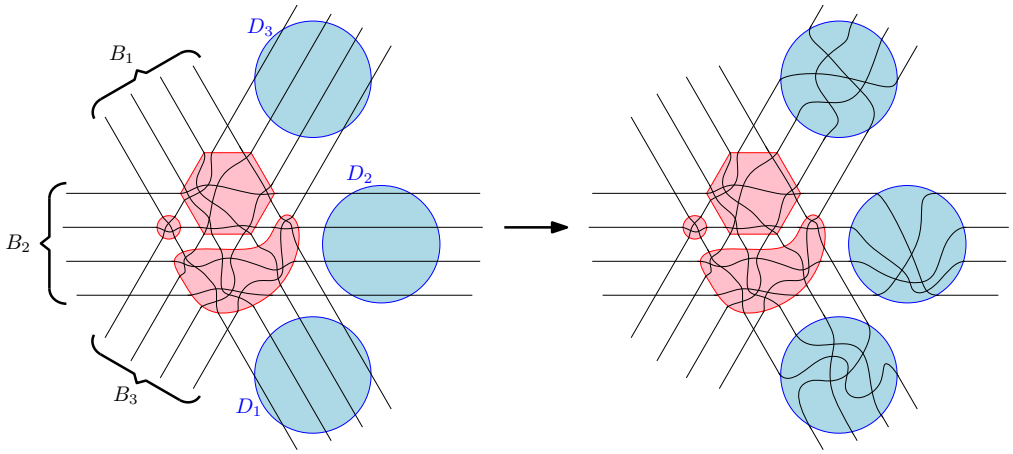


Figure 5 Left: A partial arrangement of 3 bundles of parallel pseudolines and a collection of interior-disjoint disks (highlighted blue) such that each bundle is covered by one disk.
Right: A proper pseudoline arrangement obtained by rerouting within the disks.

Since all $D_i's$ are independent and there are B_m possibilities to reroute within each D_i,

$$B_n \geq \underbrace{F_k(n)}_{\text{Step 1}} \cdot \underbrace{(B_m)^k}_{\text{Step 2}}$$

holds, where $m = \lfloor \frac{n}{k} \rfloor$. With the following lemma we can derive $c^- \geq \frac{k}{k-1} c$ where c is the constant obtained in Section 3. The construction with $k = 6$ bundles gives the lower bound $c^- > 0.2526$, and therefore completes the proof of Theorem 1.1.

▶ **Lemma 4.1.** *If* $F_k(n) \geq 2^{cn^2 - O(n)}$ *for some* $c > 0$ *then* $B_n \geq 2^{\frac{k}{k-1} cn^2 - O(n \log n)}$.

5 Discussion

We performed quite some experiments to optimize the set of parameters. To obtain the new lower bound constant $c^- > 0.2526$ presented in Theorem 1.1, we started with the $k = 6$ parallel bundles construction from [5] and covered the multicrossings with a specific selection of patches, which were inspired by regular tilings. Already the construction with $k = 4$ bundles gives $F_4(n) \geq 2^{cn^2 - O(n)}$ with $c > 0.1608$ and $c^- > 0.2144$ (see [11]), which is already an improvement to the previous best bound by Dumitrescu and Mandal [5]. While the results from [5] suggest that larger values of k give better bounds, the computations get

more and more complex. In fact, as the number k increases, the complexity of the patches increases. Since our program can only deal with patches containing about 30 to 40 segments in reasonable time, depending on the structure of crossings within it, there is a trade-off between the number of crossings within a patch and the number of bundles k in practice. This was also the reason why we use different types of patch for the four regions.

In the future we plan to investigate constructions with $k = 8$ and $k = 12$ bundles which as depicted in [5, Figures 9 and 13] come with more types of regions. It remains a challenging part to find a good tiling/patches for each of them.

Also note that as long as one fixes k, the counting approach is implicitly limited by $F_k(n)$, which is much smaller than B_n. Since $F_3(n) = 2^{cn^2+o(n^2)}$ with $c = \frac{\log_2(3)}{2} - \frac{2}{3} \approx 0.1258$ is known [8], it would be interesting to determine $\lim_{n\to\infty} \frac{\log_2 F_i(n)}{n^2}$ for $i = 4, \ldots, 12$. In particular, we wonder how far from the truth the constant in Proposition 3.1 is.

References

1 Supplemental data. https://github.com/fcorteskuehnast/counting-arrangements.

2 Otfried Cheong. The Ipe extensible drawing editor. http://ipe.otfried.org/.

3 Fernando Cortés Kühnast. On the number of arrangements of pseudolines. Bachelor's thesis, Technische Universität Berlin, Germany, 2023. https://fcorteskuehnast.github.io/files/bachelor_thesis.pdf.

4 Justin Dallant. Improved Lower Bound on the Number of Pseudoline Arrangements. arXiv:2402.13923, 2024.

5 Adrian Dumitrescu and Ritankar Mandal. New lower bounds for the number of pseudoline arrangements. *Journal of Computational Geometry*, 11:60–92, 2020. doi:10.20382/jocg.v11i1a3.

6 Herbert Edelsbrunner, Joseph O'Rourke, and Raimund Seidel. Constructing arrangements of lines and hyperplanes with applications. *SIAM Journal on Computing*, 15(2):341–363, 1986. doi:10.1137/0215024.

7 Stefan Felsner. On the Number of Arrangements of Pseudolines. *Discrete & Computational Geometry*, 18(3):257–267, 1997. doi:10.1007/PL00009318.

8 Stefan Felsner and Pavel Valtr. Coding and Counting Arrangements of Pseudolines. *Discrete & Computational Geometry*, 46(3), 2011. doi:10.1007/s00454-011-9366-4.

9 Jacob E. Goodman and Richard Pollack. Multidimensional Sorting. *SIAM Journal on Computing*, 12(3):484–507, 1983. doi:10.1137/0212032.

10 Donald E. Knuth. *Axioms and Hulls*, volume 606 of *LNCS*. Springer, 1992. doi:10/bwfnz9.

11 Fernando Cortés Kühnast, Stefan Felsner, and Manfred Scheucher. An Improved Lower Bound on the Number of Pseudoline Arrangements. arXiv:2402.13107, 2024.

12 Friedrich Levi. Die Teilung der projektiven Ebene durch Gerade oder Pseudogerade. *Berichte über die Verhandlungen der Sächsischen Akademie der Wissenschaften zu Leipzig, Mathematisch-Physische Klasse*, 78:256–267, 1926.

13 Jiří Matoušek. *Lectures on Discrete Geometry*. Springer, 2002. doi:10.1007/978-1-4613-0039-7.

Recognition of Unit Segment and Polyline Graphs is $\exists\mathbb{R}$-Complete*

Michael Hoffmann[1], Tillmann Miltzow[2], Simon Weber[1], and Lasse Wulf[3]

1 **Department of Computer Science, ETH Zürich, Switzerland**
2 **Department of Information and Computing Sciences, Utrecht University, The Netherlands**
3 **Institute of Discrete Mathematics, Graz Institute of Technology, Austria**

Abstract

Given a set of objects O in the plane, the corresponding intersection graph is defined as follows. A vertex is created for each object and an edge joins two vertices whenever the corresponding objects intersect. We study here the case of unit segments and polylines with exactly k bends. In the recognition problem, we are given a graph and want to decide whether the graph can be represented as the intersection graph of certain geometric objects. In previous work it was shown that various recognition problems are $\exists\mathbb{R}$-complete, leaving unit segments and polylines as few remaining natural cases. We show that recognition for both families of objects is $\exists\mathbb{R}$-complete.

Related Version *Full Version*: `arXiv:2401.02172`

1 Introduction

Many real-life problems can be mathematically described in the language of graphs. For instance, Cellnex Telecom owns more than 2000 cell towers in Switzerland. We want to assign each tower a frequency such that no two towers that overlap in coverage use the same frequency. This becomes a graph coloring problem. Every cell tower becomes a vertex, overlap indicates an edge and a frequency assignment corresponds to a proper coloring of the vertices, see Figure 1.

In many contexts, we have additional structure on the graph that may or may not help us to solve the underlying algorithmic problem. For instance, it might be that the graph arises as the intersection graph of unit disks in the plane (each unit disk gives a vertex, and two vertices are adjacent if their corresponding disks overlap). In that case, the coloring problem can be solved more efficiently [10], and there are better approximation algorithms for the clique problem [11]. This motivates a systematic study of geometric intersection graphs.

It is known for a host of geometric shapes that it is $\exists\mathbb{R}$-complete to recognize their intersection graphs [28, 14, 25, 27]. The class $\exists\mathbb{R}$ consists of all of those problems that are polynomial-time equivalent to deciding whether a polynomial $p \in \mathbb{Z}[X_1, \ldots, X_n]$ has a root. We will introduce $\exists\mathbb{R}$ in more detail below. $\exists\mathbb{R}$-completeness is known for the recognition problems of intersection graphs for segments, disks, unit disks, rays, grounded segments, downward rays, and a few other examples.

* This research started at the *19th Gremo Workshop on Open Problems* in Binn VS, Switzerland, in June 2022. We thank the organizers for the invitation and for providing a very pleasant and inspiring working atmosphere. Tillmann Miltzow is generously supported by the Netherlands Organisation for Scientific Research (NWO) under project no. VI.Vidi.213.150. Lasse Wulf is generously supported by the Austrian Science Fund (FWF): W1230.

40th European Workshop on Computational Geometry, Ioannina, Greece, March 13–15, 2024.
This is an extended abstract of a presentation given at EuroCG'24. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

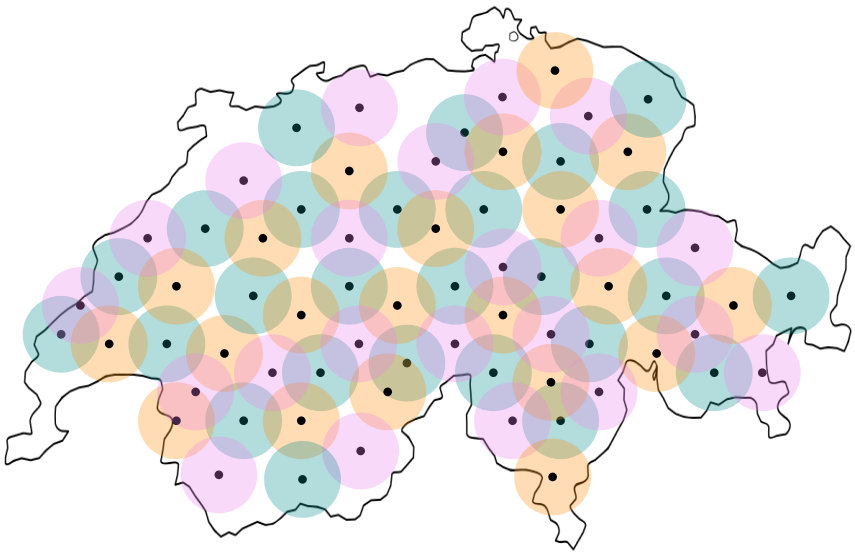


Figure 1 A fictional illustration of mobile coverage of Switzerland using cell towers.

In this work, we focus on two geometric objects; unit segments and polylines with exactly k bends. Although we consider both types of geometric objects natural and well studied, to the best of our knowledge the complexity of their recognition problem was left open.

1.1 Definition and Results

Given a finite set of geometric objects O, we denote by $G(O) = (V, E)$, the corresponding *intersection graph*. The set of vertices is the set of objects ($V = O$) and two objects are adjacent ($uv \in E$) if they intersect ($u \cap v \neq \emptyset$). We are interested in intersection graphs that come from different families of geometric objects.

Examples for a family of geometric objects are segments, disks, unit disks, unit segments, rays, and convex sets, to name a few of the most common ones. In general, given a geometric body $O \subset \mathbb{R}^2$ we denote by O^{+} the family of all translates of O. Similarly, we denote by $O^{\oplus}$ the family of all translates and rotations of O. For example, the family of all unit segments can be denoted as $u^{\oplus}$, where u is a unit segment.

Classes of geometric objects $\mathcal{O}$ naturally give rise to classes of graphs $C(\mathcal{O})$: Given a family of geometric objects $\mathcal{O}$, we denote by $C(\mathcal{O})$ the class of graphs that can be formed by taking the intersection graph of a finite subset from $\mathcal{O}$.

If we are given a graph, we can ask if this graph belongs to a geometric graph class. Formally, let C be a fixed graph class, then the recognition problem for C is defined as follows. As input, we receive a graph G and we have to decide whether $G \in C$. We denote the corresponding algorithmic problem by Recognition(C). For example the problem of recognizing unit segment graphs can be denoted by Recognition($C(u^{\oplus})$). We will use the term Unit Recognition for this problem. Furthermore, we define PolyLine Recognition as the recognition problem of intersection graphs of polylines with k bends.

We can also say that Recognition($C(\mathcal{O})$) asks about the existence of a *representation* of a given graph. A representation or *realization* of a graph G using a family of objects $\mathcal{O}$ is a function $r : V \mapsto \mathcal{O}$ such that $r(v) \cap r(w) \neq \emptyset \iff vw \in E$. For simplicity, for a set $V' \subseteq V$, we define $r(V') = \bigcup_{v \in V'} r(v)$.

Results. We show $\exists\mathbb{R}$-completeness of the recognition problems of two very natural geometric graph classes.

▶ **Theorem 1.1.** *Unit Recognition is $\exists\mathbb{R}$-complete.*

▶ **Theorem 1.2.** *PolyLine Recognition is $\exists\mathbb{R}$-complete, for any fixed $k \geq 1$.*

1.2 Discussion

To supply the appropriate context for our results, we give a comprehensive overview over important geometric graph classes and the current knowledge about the complexity of their recognition problems in Figure 2.

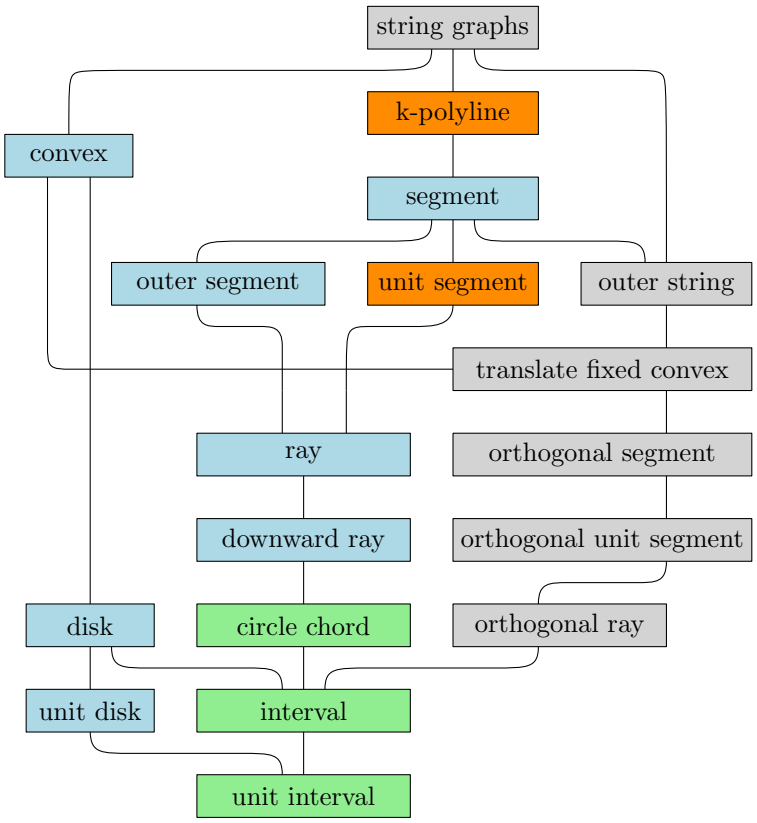


Figure 2 Each box represents a different geometric intersection graph class. Those marked in green can be recognized in polynomial time. Those in blue are known to be $\exists\mathbb{R}$-complete. The ones in gray are NP-complete, and the orange ones are the new results presented in this paper. Relevant references: [12, 14, 21, 22, 23, 24, 25, 27, 28, 30, 32, 33, 36, 43]

Refining the Hierarchy. We see our main contribution in refining the hierarchy of geometric graph classes for which recognition complexity is known. Both unit segments as well as polylines with k bends are natural objects that are well studied in the literature. However, the recognition of the corresponding graph classes was not studied previously. Polylines with an unbounded number of bends are equivalent to strings[1], while polylines

[1] It is possible to show that polylines with an unbounded number of bends are as versatile as strings with respect to the types of graphs that they can represent, since the number of intersections of any two strings can always be bounded from above [36, 37].

with 0 bends are just segments. Polylines with k bends thus naturally slot in between strings and segments, and their corresponding graph class is thus also an intermediate class between the class of segment intersection graphs and string graphs, as can be seen in Figure 2. By showing that recognition for polylines with k bends is ∃ℝ-complete for all constant k, we see that the switch from ∃ℝ-completeness (segment intersection graphs) to NP-membership (string graphs) really only happens once k is infinite. Similarly, unit segment intersection graphs slot in between segment and ray intersection graphs. Intuitively, recognition of a class intermediate to two classes that are ∃ℝ-hard to recognize should also be ∃ℝ-hard, and our Theorem 1.1 confirms this intuition in this case.

Large Coordinates. One of the consequences of ∃ℝ-completeness is that there are no short representations of solutions known. Some representable graphs may only be representable by objects with irrational coordinates, or by rational coordinates with nominator and denominator of size at least $2^{2^{n^c}}$, for some fixed $c > 0$. In other words, the numbers to describe the position might need to be doubly exponentially large [27] for some graphs. For "flexible" objects like polylines, rational solutions can always be obtained by slightly perturbing the representation. For more "sturdy" objects like unit segments this may not be possible, however it is known that for example unit disks admit rational solutions as well [28].

Unraveling the Broader Story. Given the picture of Figure 2, we wish to get a better understanding of when geometric graph recognition problems are ∃ℝ-complete and when they are contained in NP. Figure 2 indicates that ∃ℝ-hardness comes from objects that are complicated enough to avoid a complete combinatorial characterization. Such characterizations are known for example for unit interval graphs, interval graphs and circle chord graphs. On the other hand, if the geometric objects are too flexible, the recognition problem is in NP. The prime example is string graphs [36]. We want to summarize this as: recognition problems are ∃ℝ-complete if the underlying family of geometric objects is at a sweet spot of neither being too simplistic nor too flexible.

Studying the figure further we observe two different types of ∃ℝ-complete families. The first type of family encapsulates all rotations $O^{\oplus}$ of a given object O (i.e., segments, rays, unit segments etc.). The second type of family contains translates and possibly homothets of geometric objects that have some curvature themselves (i.e., disks and unit disks). However in case we fix a specific object without curvature, i.e., a polygon, and consider all translations of it then the recognition problem also lies in NP [30]. Therefore, broadly speaking, curvature or rotation seem to be properties needed for ∃ℝ-completeness and the lack of it seems to imply NP-membership. We wish to capture parts of this intuition in the following conjectures:

▶ Conjecture 1. Let O be a convex body in the plane with at least two distinct points. Then Recognition($O^{\oplus}$) is ∃ℝ-complete.

▶ Conjecture 2. Let O be a convex body in the plane. Then Recognition($O^{\dotplus}$) is ∃ℝ-complete if and only if O has curvature.

1.3 Existential Theory of the Reals

The class of the existential theory of the reals ∃ℝ (pronounced as 'ER') is a complexity class defined through its canonical problem ETR, which also stands for Existential Theory of the Reals. In this problem we are given a sentence of the form $\exists x_1, \ldots, x_n \in \mathbb{R} : \Phi(x_1, \ldots, x_n)$, where Φ is a well-formed quantifier-free formula consisting of the symbols $\{0, 1, x_1, \ldots, x_n, +, \cdot, \geq, >, \wedge, \vee, \neg\}$, and the goal is to check whether this sentence is true.

The class ∃ℝ is the family of all problems that admit a polynomial-time many-one reduction to ETR. It is known that NP ⊆ ∃ℝ ⊆ PSPACE [13]. The reason that ∃ℝ is an

important complexity class is that a number of common problems, mainly in computational geometry, have been shown to be complete for this class. Schaefer established the current name and pointed out first that several known NP-hardness reductions actually imply $\exists\mathbb{R}$-completeness [33]. Early examples are related to recognition of geometric structures: points in the plane [29, 42], geometric linkages [34, 1], segment graphs [25, 27], unit disk graphs [28, 21], ray intersection graphs [14], and point visibility graphs [14]. In general, the complexity class is more established in the graph drawing community [26, 16, 35, 18]. Yet, it is also relevant for studying polytopes [31, 17], Nash-Equilibria [6, 38, 20, 8, 9], and matrix factorization problems [15, 40, 41, 39]. Other $\exists\mathbb{R}$-complete problems are the Art Gallery Problem [3, 44], covering polygons with convex polygons [2], geometric packing [5] and training neural networks [4, 7].

2 Proof Techniques

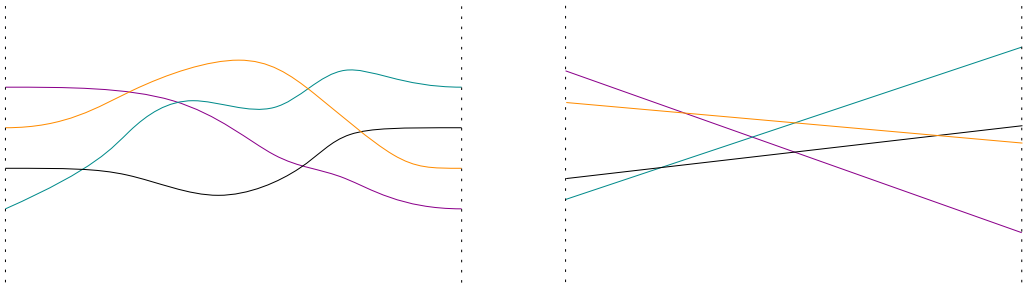


Figure 3 The pseudoline arrangement on the left is combinatorially equivalent to the (truncated) line arrangement on the right; hence, it is stretchable.

The techniques used in this paper are similar to previous work. Due to space constraints we only give some rough proof sketches, all the details can be found in the full version of the paper. $\exists\mathbb{R}$-membership can be established straightforwardly by constructing concrete formulae or invoking a characterization of $\exists\mathbb{R}$ using *real* verification algorithms, similar to the characterization of NP [19]. For $\exists\mathbb{R}$-hardness, we are in essence reducing from the SimpleStretchability problem. In this problem, we are given a simple *pseudoline arrangement* as an input, and the question is whether this arrangement is *stretchable*. A pseudoline arrangement $\mathcal{A}$ is a set of n curves that are x-monotone. Furthermore, any two curves intersect exactly once and no three curves meet in a single point. We assume that there exist two vertical lines on which each curve starts and ends. The problem is to determine whether there exists a combinatorially equivalent (truncated) line arrangement. See Figure 3 for an example.

Given the initial pseudoline arrangement $\mathcal{A}$, we construct a graph that is representable by unit segments iff $\mathcal{A}$ is stretchable. This graph is created by enhancing $\mathcal{A}$ with more curves (see Figure 4) and taking their intersection graph. Figures 5 to 7 give some intuition on the proof that if $\mathcal{A}$ is stretchable, this graph is representable by unit segments: The line arrangement certifying stretchability is first squeezed into a canonical form, then all features can be represented easily. On the other hand, if this graph is representable, the unit segments representing the vertices corresponding to $\mathcal{A}$ witness stretchability of $\mathcal{A}$. To prove this, we show that cycles can be used to enforce a certain order of intersections of objects with the cycle. For this, we can use the same proof for unit segments and polylines.

Knowing that the connectors (green in Figure 4) must intersect the cycle in the correct

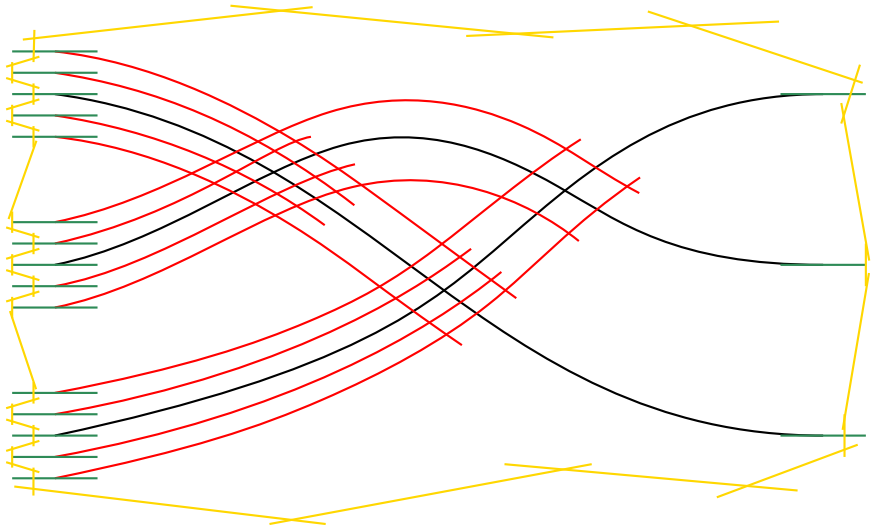


Figure 4 $\mathcal{A}$ (black), enhanced with probes (red), connectors (green) and a cycle (yellow).

order in any representation, the intersection pattern of the pseudolines and the probes (red) guarantees that unit segments representing the pseudolines must have the same combinatorial structure as $\mathcal{A}$, finishing the proof. The ideas of order-enforcing cycles and probes have already been used in different contexts [14].

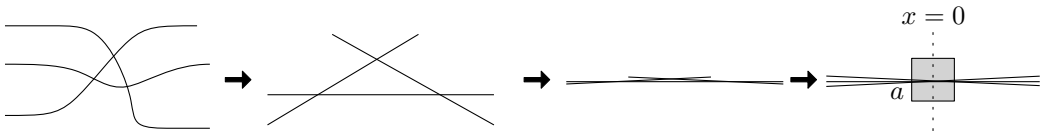


Figure 5 Stretching and then squeezing a pseudoline arrangement.

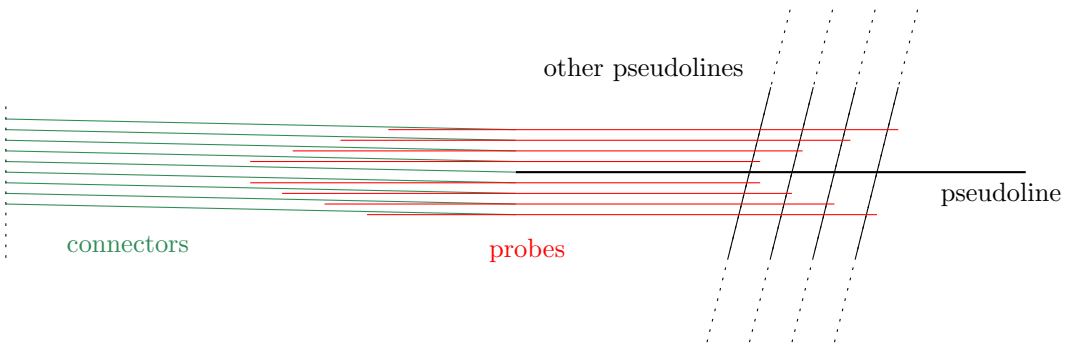


Figure 6 Representing connectors and probes by unit segments.

The ∃ℝ-hardness proof for PolyLine Recognition follows this previous proof for unit segments closely, and only adds some additional order-enforcing cycles. The enhanced pseudoline arrangement for polylines is shown in Figure 8. For each pseudoline we create a twin. Using the $2k$ additional order-enforcing cycles we enforce that in any realization of the graph by polylines, each polyline representing a pseudoline must intersect its twin within that cycle. We can then show that if two polylines intersect $2k$ times (with both polylines

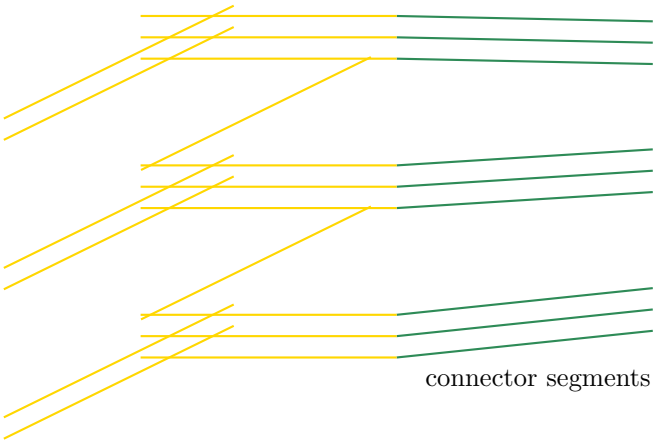


Figure 7 Attaching the connectors to the cycle using unit segments in a sawtooth pattern.

visiting these intersection points in the same order), they must in total use at least $2k - 1$ bends. This ensures that at least one of the k-polylines must spend all of its k bends to realize these intersections. Thus, this polyline is actually a straight line in the region labelled "canvas" in Figure 8. Using the same arguments as for unit segments we can then see that the arrangement formed by these straight lines is combinatorially equivalent to $\mathcal{A}$, and thus $\mathcal{A}$ is stretchable.

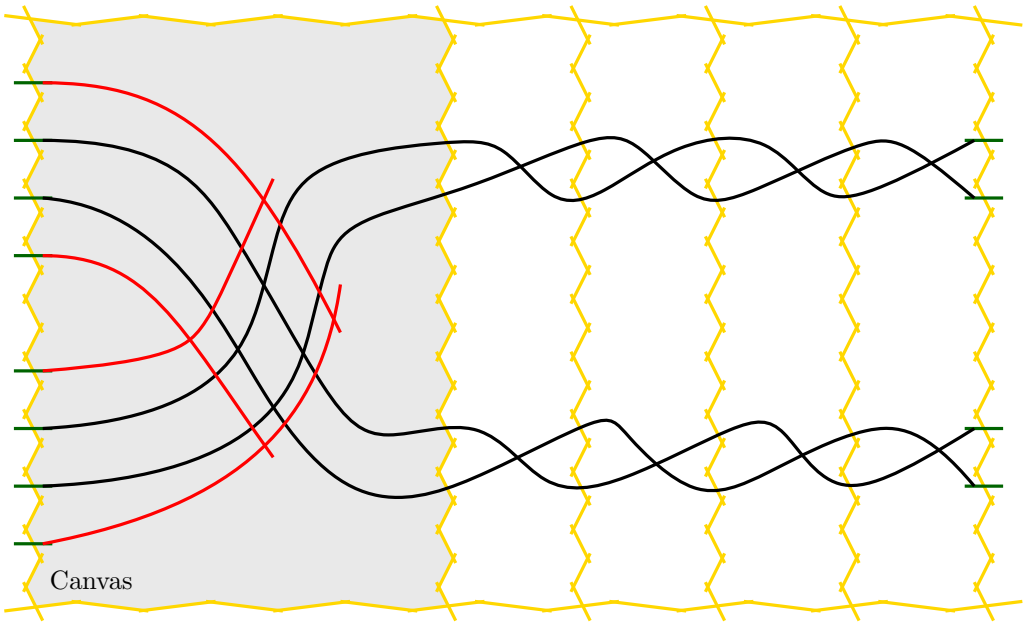


Figure 8 $\mathcal{A}$ *twinned* and enhanced with probes, connectors and $2k + 1$ cycles (yellow). Weaving the twinned pseudolines ensures that at least one of the two twins contains no bends in the canvas.

References

1 Zachary Abel, Erik Demaine, Martin Demaine, Sarah Eisenstat, Jayson Lynch, and Tao Schardl. Who needs crossings? Hardness of plane graph rigidity. In *32nd International Symposium on Computational Geometry (SoCG 2016)*, pages 3:1–3:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPIcs.SoCG.2016.3.

2 Mikkel Abrahamsen. Covering Polygons is Even Harder. In Nisheeth K. Vishnoi, editor, *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 375–386, 2022. doi:10.1109/FOCS52979.2021.00045.

3 Mikkel Abrahamsen, Anna Adamaszek, and Tillmann Miltzow. The art gallery problem is ∃ℝ-complete. *Journal of the ACM*, 69(1):4:1–4:70, 2022. doi:10.1145/3486220.

4 Mikkel Abrahamsen, Linda Kleist, and Tillmann Miltzow. Training neural networks is ∃ℝ-complete. In *NeurIPS 2021*, pages 18293–18306, 2021. URL: https://proceedings.neurips.cc/paper/2021/hash/9813b270ed0288e7c0388f0fd4ec68f5-Abstract.html.

5 Mikkel Abrahamsen, Tillmann Miltzow, and Nadja Seiferth. Framework for ∃ℝ-completeness of two-dimensional packing problems. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1014–1021, 2020. doi:10.1109/FOCS46700.2020.00098.

6 Marie Louisa Tølbøll Berthelsen and Kristoffer Arnsfelt Hansen. On the computational complexity of decision problems about multi-player Nash equilibria. *Theory of Computing Systems*, 66(3):519–545, 2022. doi:10.1007/s00224-022-10080-1.

7 Daniel Bertschinger, Christoph Hertrich, Paul Jungeblut, Tillmann Miltzow, and Simon Weber. Training fully connected neural networks is ∃ℝ-complete. *CoRR*, abs/2204.01368, 2022. To appear at NeurIPS 2024. arXiv:2204.01368.

8 Vittorio Bilò and Marios Mavronicolas. A catalog of ∃ℝ-complete decision problems about Nash equilibria in multi-player games. In *33rd Symposium on Theoretical Aspects of Computer Science (STACS 2016)*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPIcs.STACS.2016.17.

9 Vittorio Bilò and Marios Mavronicolas. ∃ℝ-complete decision problems about symmetric Nash equilibria in symmetric multi-player games. In *34th Symposium on Theoretical Aspects of Computer Science (STACS 2017)*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPIcs.STACS.2017.13.

10 Csaba Biró, Édouard Bonnet, Dániel Marx, Tillmann Miltzow, and Pawel Rzazewski. Fine-grained complexity of coloring unit disks and balls. *Journal of Computational Geometry*, 9(2):47–80, 2018. doi:10.20382/jocg.v9i2a4.

11 Marthe Bonamy, Édouard Bonnet, Nicolas Bousquet, Pierre Charbit, Panos Giannopoulos, Eun Jung Kim, Pawel Rzazewski, Florian Sikora, and Stéphan Thomassé. EPTAS and subexponential algorithm for maximum clique on disk and unit ball graphs. *Journal of the ACM*, 68(2):9:1–9:38, 2021. doi:10.1145/3433160.

12 Kellogg S. Booth and George S. Lueker. Testing for the consecutive ones property, interval graphs, and graph planarity using *PQ*-tree algorithms. *Journal of Computer and System Sciences*, 13(3):335–379, 1976. doi:10.1016/S0022-0000(76)80045-1.

13 John Canny. Some algebraic and geometric computations in PSPACE. In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, STOC '88, page 460–467, New York, NY, USA, 1988. Association for Computing Machinery. doi:10.1145/62212.62257.

14 Jean Cardinal, Stefan Felsner, Tillmann Miltzow, Casey Tompkins, and Birgit Vogtenhuber. Intersection Graphs of Rays and Grounded Segments. *Journal of Graph Algorithms and Applications*, 22(2):273–294, 2018. doi:10.7155/jgaa.00470.

15 Dmitry Chistikov, Stefan Kiefer, Ines Marusic, Mahsa Shirmohammadi, and James Worrell. On Restricted Nonnegative Matrix Factorization. In Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi, editors, *43rd International Colloquium*

on Automata, Languages, and Programming (ICALP 2016), volume 55 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 103:1–103:14, Dagstuhl, Germany, 2016. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ICALP.2016.103.

16 Michael G. Dobbins, Linda Kleist, Tillmann Miltzow, and Paweł Rzążewski. ∀∃ℝ-Completeness and Area-Universality. In Andreas Brandstädt, Ekkehard Köhler, and Klaus Meer, editors, *Graph-Theoretic Concepts in Computer Science (WG 2018)*, volume 11159 of *Lecture Notes in Computer Science*, pages 164–175. Springer, 2018. doi:10.1007/978-3-030-00256-5_14.

17 Michael Gene Dobbins, Andreas Holmsen, and Tillmann Miltzow. A universality theorem for nested polytopes. *arXiv*, 1908.02213, 2019.

18 Jeff Erickson. Optimal curve straightening is ∃ℝ-complete. *arXiv:1908.09400*, 2019.

19 Jeff Erickson, Ivor van der Hoog, and Tillmann Miltzow. Smoothing the gap between NP and ER. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1022–1033, 2020. doi:10.1109/FOCS46700.2020.00099.

20 Jugal Garg, Ruta Mehta, Vijay V. Vazirani, and Sadra Yazdanbod. ∃ℝ-completeness for decision versions of multi-player (symmetric) Nash equilibria. *ACM Transactions on Economics and Computation*, 6(1), 2018. doi:10.1145/3175494.

21 Ross Kang and Tobias Müller. Sphere and Dot Product Representations of Graphs. *Discrete & Computational Geometry*, 47(3):548–569, 2012. doi:10.1007/s00454-012-9394-8.

22 Jan Kratochvíl. String graphs. I. The number of critical nonstring graphs is infinite. *Journal of Combinatorial Theory. Series B*, 52(1):53–66, 1991. doi:10.1016/0095-8956(91)90090-7.

23 Jan Kratochvíl. String graphs. II. Recognizing string graphs is NP-hard. *Journal of Combinatorial Theory. Series B*, 52(1):67–78, 1991. doi:10.1016/0095-8956(91)90091-W.

24 Jan Kratochvíl and Jiří Matoušek. NP-hardness results for intersection graphs. *Commentationes Mathematicae Universitatis Carolinae*, 30(4):761–773, 1989.

25 Jan Kratochvíl and Jiří Matoušek. Intersection graphs of segments. *Journal of Combinatorial Theory. Series B*, 62(2):289–315, 1994. doi:10.1006/jctb.1994.1071.

26 Anna Lubiw, Tillmann Miltzow, and Debajyoti Mondal. The complexity of drawing a graph in a polygonal region. *Journal of Graph Algorithms and Applications*, 26(4):421–446, 2022. doi:10.7155/jgaa.00602.

27 Jiří Matoušek. Intersection graphs of segments and ∃ℝ. *ArXiv 1406.2636*, 2014.

28 Colin McDiarmid and Tobias Müller. Integer realizations of disk and segment graphs. *Journal of Combinatorial Theory. Series B*, 103(1):114–143, 2013. doi:10.1016/j.jctb.2012.09.004.

29 Nicolai Mnëv. The universality theorems on the classification problem of configuration varieties and convex polytopes varieties. In *Topology and geometry – Rohlin seminar*, pages 527–543, 1988.

30 Tobias Müller, Erik Jan van Leeuwen, and Jan van Leeuwen. Integer representations of convex polygon intersection graphs. *SIAM Journal on Discrete Mathematics*, 27(1):205–231, 2013. doi:10.1137/110825224.

31 Jürgen Richter-Gebert and Günter M. Ziegler. Realization spaces of 4-polytopes are universal. *Bulletin of the American Mathematical Society*, 32(4):403–412, 1995.

32 Alexandre Rok and Bartosz Walczak. Outerstring graphs are χ-bounded. *SIAM Journal on Discrete Mathematics*, 33(4):2181–2199, 2019. doi:10.1137/17M1157374.

33 Marcus Schaefer. Complexity of some geometric and topological problems. In David Eppstein and Emden R. Gansner, editors, *Graph Drawing*, pages 334–344, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg. doi:10.1007/978-3-642-11805-0_32.

34 Marcus Schaefer. *Realizability of Graphs and Linkages*, pages 461–482. Thirty Essays on Geometric Graph Theory. Springer, 2013. doi:10.1007/978-1-4614-0110-0_24.

35 Marcus Schaefer. Complexity of geometric k-planarity for fixed k. *Journal of Graph Algorithms and Applications*, 25(1):29–41, 2021. doi:10.7155/jgaa.00548.

36 Marcus Schaefer, Eric Sedgwick, and Daniel Štefankovič. Recognizing string graphs in NP. *Journal of Computer and System Sciences*, 67(2):365–380, 2003. Special Issue on STOC 2002. doi:10.1016/S0022-0000(03)00045-X.

37 Marcus Schaefer and Daniel Štefankovič. Decidability of string graphs. *Journal of Computer and System Sciences*, 68(2):319–334, 2004. Special Issue on STOC 2001. doi:10.1016/j.jcss.2003.07.002.

38 Marcus Schaefer and Daniel Štefankovič. Fixed points, Nash equilibria, and the existential theory of the reals. *Theory of Computing Systems*, 60(2):172–193, Feb 2017. doi:10.1007/s00224-015-9662-0.

39 Marcus Schaefer and Daniel Štefankovič. The complexity of tensor rank. *Theory of Computing Systems*, 62(5):1161–1174, 2018. doi:10.1007/s00224-017-9800-y.

40 Yaroslav Shitov. A universality theorem for nonnegative matrix factorizations. *ArXiv 1606.09068*, 2016.

41 Yaroslav Shitov. The complexity of positive semidefinite matrix factorization. *SIAM Journal on Optimization*, 27(3):1898–1909, 2017. doi:10.1137/16M1080616.

42 Peter W. Shor. Stretchability of Pseudolines is NP-Hard. In Peter Gritzmann and Bernd Sturmfels, editors, *Applied Geometry And Discrete Mathematics*, volume 4 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 531–554, 1991. doi:10.1090/dimacs/004/41.

43 Jeremy Spinrad. Recognition of circle graphs. *Journal of Algorithms. Cognition, Informatics and Logic*, 16(2):264–282, 1994. doi:10.1006/jagm.1994.1012.

44 Jack Stade. Complexity of the boundary-guarding art gallery problem. *CoRR*, abs/2210.12817, 2022. doi:10.48550/arXiv.2210.12817.

The Complexity of Geodesic Spanners using Steiner Points

Sarita de Berg[1], Tim Ophelders[1,2], Irene Parada[3], Frank Staals[1], and Jules Wulms[2]

1 **Department of Information and Computing Sciences, Utrecht University, the Netherlands**
S.deBerg@uu.nl, T.A.E.Ophelders@uu.nl, F.Staals@uu.nl

2 **Department of Mathematics and Computer Science, TU Eindhoven, the Netherlands**
J.H.H.M.Wulms@tue.nl

3 **Department of Mathematics, Universitat Politècnica de Catalunya, Spain**
Irene.Parada@upc.edu

Abstract

A geometric t-spanner on a set S of n point sites in a metric space P is a subgraph of the complete graph on S such that for every pair of sites p, q the distance in $\mathcal{G}$ is a most t times the distance $d(p, q)$ in P. We call a connection between two sites in the spanner a *link*. In some settings, such as when P is a simple polygon with m vertices and a link is a shortest path in P, links can consist of $\Theta(m)$ segments and thus have non-constant complexity. The total spanner complexity is a recently-introduced measure of how compact a spanner is. In this paper, we study what happens if we are allowed to introduce k Steiner points to reduce the spanner complexity. We study such Steiner spanners in simple polygons, polygonal domains, and on edge-weighted trees.

Surprisingly, we show that Steiner points have only limited utility. For a spanner that uses k Steiner points, we provide an $\Omega(nm/k)$ lower bound on the worst-case complexity of any $(3-\varepsilon)$-spanner, and an $\Omega(mn^{1/(t+1)}/k^{1/(t+1)})$ lower bound on the worst-case complexity of any $(t-\varepsilon)$-spanner, for any constant $\varepsilon \in (0, 1)$ and integer constant $t \geq 2$. These lower bounds hold in all settings.

On the positive side, for trees we show how to build a $2t$-spanner that uses k Steiner points of complexity $O(mn^{1/t}/k^{1/t} + n\log(n/k))$, for any integer $t \geq 1$. We generalize this result to forests, and then apply it to obtain a $2\sqrt{2}t$-spanner in a simple polygon or a $6t$-spanner a in polygonal domain with total complexity $O(mn^{1/t}(\log k)^{1+1/t}/k^{1/t} + n\log^2 n)$.

Related Version A full version of the paper is available at https://arxiv.org/abs/2402.12110.

1 Introduction

Consider a set S of n point *sites* in a metric space P. In applications such as (wireless) network design [3], regression analysis [10], vehicle routing [7, 14], and constructing TSP tours [5], it is desirable to have a compact network that accurately captures the distances between the sites in S. Spanners provide such a representation. Formally, a *geometric t-spanner* $\mathcal{G}$ is a subgraph of the complete graph on S, so that for every pair of sites p, q the distance $d_{\mathcal{G}}(p, q)$ in $\mathcal{G}$ is at most t times the distance $d(p, q)$ in P [12]. The quality of a spanner can be expressed in terms of the *spanning ratio* t and a term to measure how "compact" it is. Typical examples are the *size* of the spanner, that is, the number of edges of $\mathcal{G}$, its weight (the sum of the edge lengths), or its diameter [13].

When the sites represent physical locations, there are often other objects (e.g. buildings, lakes, roads, mountains) that influence the shortest path between the sites. In such settings, we need to explicitly incorporate the environment. We consider the case where

40th European Workshop on Computational Geometry, Ioannina, Greece, March 13–15, 2024.
This is an extended abstract of a presentation given at EuroCG'24. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

this environment is modeled by a polygon P with m vertices, and possibly containing holes. The distance between a pair of points $p, q \in P$ is then given by their *geodesic distance*: the length of a shortest path between p and q that is fully contained in P. This setting has been considered before. For example, Abam, Adeli, Homapou, and Asadollahpoor [1] present a geodesic $(\sqrt{10}+\varepsilon)$-spanner of size $O(n \log^2 n)$ when P is a simple polygon, and a geodesic $(5+\varepsilon)$-spanner of size $O(n\sqrt{h}\log^2 n)$ when the polygon has $h > 1$ holes. Abam, de Berg, and Seraji [2] even obtained a $(2+\varepsilon)$-spanner of size $O(n \log n)$ when P is actually a terrain. To avoid confusion between the edges of P and the edges of $\mathcal{G}$, we will from hereon use the term *links* to refer to the edges of $\mathcal{G}$.

As argued by de Berg, van Kreveld, and Staals [8], each link in a geodesic spanner may correspond to a shortest path containing $\Omega(m)$ polygon vertices. Therefore, the *spanner complexity*, defined as the total number of line segments that make up all links in the spanner, more appropriate measures how compact a geodesic spanner is. The above spanners of [1, 2] all have worst-case complexity $\Omega(mn)$, hence they present an algorithm to construct a $2\sqrt{2}t$-spanner in a simple polygon or a $6t$-spanner in a polygon with holes with complexity $O(mn^{1/t} + n\log^2 n)$, for any integer $t > 1$.[1] These complexity bounds are still relatively high. De Berg, van Kreveld, and Staals [8] also show that these results are almost tight. In particular, for sites in a simple polygon, any geodesic $(3-\varepsilon)$-spanner has worst-case complexity $\Omega(nm)$, and for any constant $\varepsilon \in (0, 1)$ and integer constant $t \geq 2$, a $(t-\varepsilon)$-spanner has worst-case complexity $\Omega(mn^{1/(t-1)} + n)$.

Problem statement. A very natural question is then if we can reduce the total complexity of a geodesic spanner by allowing *Steiner points*. That is, by adding an additional set $\mathcal{S}$ of k vertices in $\mathcal{G}$, each one corresponding to a (Steiner) point in P. For the original sites $p, q \in S$ we still require that their distance in $\mathcal{G}$ is at most t times their distance in P (i.e. $d_{\mathcal{G}}(p, q) \leq t d(p, q)$), but the graph distance from a Steiner point $p' \in \mathcal{S}$ to any other site is unrestrained. Allowing for such Steiner points has proven to be useful in reducing the weight [4, 9] and size [11] of spanners. In our setting, it allows us to create additional "junction" vertices, thereby allowing us to share high-complexity subpaths. See Figure 1 for an illustration. Indeed, if we are allowed to turn every polygon vertex into a Steiner point, Clarkson [6] shows that, for any $\varepsilon > 0$, we can obtain a $(1+\varepsilon)$-spanner of complexity $O((n+m)/\varepsilon)$. However, the number of polygon vertices m may be much larger than the number of Steiner points we can afford. Hence, we focus on the scenario in which the number of Steiner points k is (much) smaller than m.

Our contributions. Surprisingly, we show that in this setting Steiner points have only limited utility. In the full version, we show that there is a set of n sites in a simple polygon with $m = \Omega(n)$ vertices for which any $(2-\varepsilon)$-spanner (with $k < n/2$ Steiner points) has complexity $\Omega(mn^2/k^2)$. Similarly, we give an $\Omega(mn/k)$ and $\Omega(mn^{1/(1+t)}/k^{1/(1+t)})$ lower bound on the complexity of a $(3-\varepsilon)$- and a $(t-\varepsilon)$-spanner, respectively. These results dash our hopes for a near linear complexity spanner with "few" Steiner points and constant spanning ratio.

These lower bounds actually hold in a more restricted setting. Namely, when the metric space is simply an edge-weighted tree that has m vertices, and the n sites are all placed in

[1] De Berg, van Kreveld, and Staals [8] claim that the refinement by Abam, de Berg, and Seraji [2] can be applied to obtain a $(2t+\varepsilon)$-spanner of the same complexity (increased by a constant factor dependent on ε). However, some details of how this refinement influences the complexity are missing.

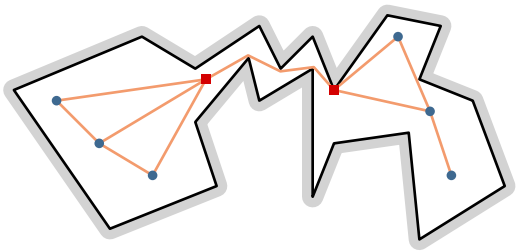


Figure 1 A spanner in a simple polygon that uses two Steiner points (red squares). By adding the two Steiner points, we no longer need multiple links that pass through the middle section of P.

leaves of the tree. In Section 2, we show that in this setting we can efficiently construct a spanner whose complexity is relatively close to optimal. In particular, our algorithm constructs a $2t$-spanner of complexity $O(mn^{1/t}/k^{1/t} + n\log(n/k))$. A slight extension of this algorithm allows us to deal with a forest as well.

This algorithm for constructing a spanner on an edge-weighted tree turns out to be the crucial ingredient for constructing low-complexity spanners for point sites in simple polygons. In particular, in Section 3, we combine some of the techniques developed by de Berg, van Kreveld, and Staals [8] and the Steiner spanner for a forest to build a $2\sqrt{2}t$-spanner of complexity $O(mn^{1/t}(\log k)^{1+1/t}/k^{1/t} + n\log^2 n)$. The main challenge here is to argue that the links used still have low complexity, even when they are now embedded in the polygon. For $k = O(1)$ our spanner thus matches the result of de Berg, van Kreveld, and Staals [8]. In the full version, we extend these results to polygonal domains, where we get a $6t$-spanner of similar complexity. Omitted proofs are contained in the full version.

2 Steiner spanners for trees

In this section, we consider spanners on an edge-weighted rooted tree T. We allow only positive weights. We denote by n the number of leaves and by m the number of vertices in T. The complexity of a link between two sites (or Steiner points) $p, q \in T$ is the number of edges in the shortest path $\pi(p, q)$, and the distance $d(p, q)$ is equal to the sum of the weights on this (unique) path. In the full version, we prove several lower bounds on the complexity of any spanner that uses k Steiner points. Among these is a general lower bound of $\Omega(mn^{1/(t+1)}/k^{1/(t+1)})$ for any $(t - \varepsilon)$-spanner. The goal in this section is to construct a spanner of complexity close to this lower bound. We denote by $T(v)$ the subtree of T rooted at vertex v. For an edge $e \in T$ with upper endpoint v_1 and lower endpoint v_2, we denote by $T(e) := T(v_2) \cup \{e\}$ the subtree of e rooted at v_1. The following lemma states that we can build a low complexity spanner for a tree (without using Steiner points).

▶ **Lemma 2.1** (de Berg et al. [8]). *For any integer $t \geq 1$, we can build a $2t$-spanner for T of size $O(n\log n)$ and complexity $O(mn^{1/t} + n\log n)$ in $O(n\log n + m)$ time.*

Spanner construction. To construct a Steiner spanner $\mathcal{G}$, we start by partitioning the sites in k sets $S_1, \ldots S_k$ by an in-order traversal of the tree. The first $\lceil n/k \rceil$ sites encountered are in S_1, the second $\lceil n/k \rceil$ in S_2, etc. After this, the sites are reassigned into k new disjoint sets $S'_1, \ldots S'_k$. For each of these sets, we consider a subtree $T'_i \subseteq T$ whose leaves are the set S'_i. There are four properties that we desire of these sets and their subtrees.

1. The size of S'_i is $O(n/k)$.
2. The trees T'_i cover T, i.e. $\bigcup_i T'_i = T$.
3. The trees T'_i are disjoint apart from Steiner points.
4. Each tree T'_i contains only $O(1)$ Steiner points.

As we prove later, these properties ensure that we can construct a spanner on each subtree T'_i to obtain a spanner for T. We obtain sets S'_i and the corresponding trees T'_i as follows.

We color the vertices and edges of the tree T using k colors $\{1, \ldots, k\}$ in two steps. In this coloring an edge or vertex is allowed to have more than one color. First, for each set S_i, we color the smallest subtree that contains all sites in S_i by color i. Note that after this step there are no uncolored vertices that have an incident descendant edge that is colored. In the next step, we color the remaining uncolored edges and vertices. These edges and their (possibly already colored) upper endpoints are colored using a bottom-up approach. We assign each uncolored edge and its upper endpoint the color with the lowest index i that is assigned also to its lower endpoint.

After coloring T, we place a Steiner point s_i at the root of tree T_i formed by all edges and vertices of color i for $i \in \{1, \ldots, k\}$. Observe that it may happen that more than one Steiner point is assigned to some vertex. Slightly abusing our notation, we denote the vertex that the Steiner point s_i is placed at by s_i as well.

For each Steiner point s_i, we define a subtree $T'_i \subseteq T$. The sites in T'_i will be the set S'_i. The tree T'_i is a subtree of $T(s_i)$. When s_i is the only Steiner point at the vertex, then $T'_i = T(s_i) \setminus \bigcup_j (T(s_j) \setminus \{s_j\})$ for s_j a descendant of s_i. In other words, we look at the tree rooted at s_i up to and including the next Steiner points, see Figure 2(a). When s_i is not the only Steiner point at the vertex, we include only subtrees $T(e)$ of s_i (up to the next Steiner points) that start with an edge e that has color i and no color $j > i$. See Figure 2(b). Whenever s_i has the lowest or highest index of the Steiner points at s_i, we also include all $T(e')$ that start with an edge e' of color $j < i$ or $j > i$, respectively.

By creating T'_i in this way, s_i is not a leaf of T'_i. We therefore adapt T'_i by adding an edge of weight zero between the vertex at s_i and a new leaf corresponding to s_i. On each subtree T'_i, we construct a $2t$-spanner using the algorithm of Lemma 2.1. These k spanners connect at the Steiner points, which we formally prove in the spanner analysis.

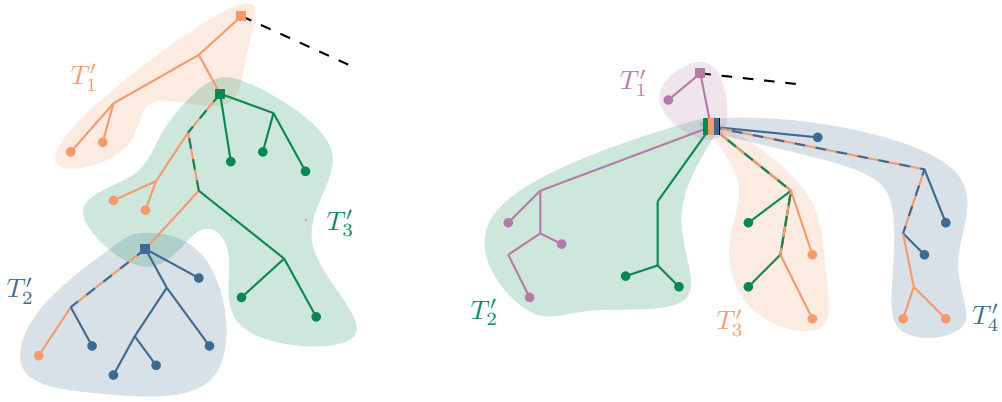
Analysis. To prove that $\mathcal{G}$ is indeed a low complexity $2t$-spanner for $\mathcal{G}$, we first show that the four properties stated before hold for S'_i and T'_i. We often apply the following lemma, that limits the number of colors an edge can be assigned by our coloring scheme.

▶ **Lemma 2.2.** *An edge can have at most two colors.*

Proof. First of all, observe that an edge can receive more than one color only in the first step of the coloring. Suppose for contradiction that there is an edge e in T that has three colors $i < j < \ell$. Let v be the lower endpoint of e. Then there must be three sites $p_i \in S_i$, $p_j \in S_j$, $p_\ell \in S_\ell$ in $T(v)$. Because these sets are defined by an in-order traversal, p_i must appear before p_j in the traversal. Similarly, p_j appears before p_ℓ. Additionally, there must be a site $p'_j \in S_j$ in $T \setminus T(v)$, otherwise the color j would not be assigned to e. The site p'_j must appear before p_i or after p_ℓ in the traversal. In the first case, p_i must be in S_j as it appears between two sites in S_j. In the second case, we find the contradiction $p_\ell \in S_j$. ◀

▶ **Lemma 2.3.** *The sets S'_i and trees T'_i adhere to the four stated properties.*

We are now ready to prove that our algorithm computes a spanner with low complexity.



(a) **(b)**

Figure 2 The tree T_i is the subtree whose edges and vertices have color i. A Steiner point (square) is placed at the root of T_i. The shaded areas show the trees T'_i. The examples show the case when the Steiner points are **(a)** at different vertices or **(b)** share a vertex.

▶ **Theorem 2.4.** *Let T be a tree with n leaves and m vertices. For any integer $t \geq 1$, we can build a $2t$-spanner $\mathcal{G}$ for T of size $O(n \log(n/k))$ and complexity $O(mn^{1/t}/k^{1/t} + n \log(n/k))$ in $O(n \log(n/k) + m + K)$ time, where K is the output size.*

Proof sketch. Let n_i and m_i denote the number of leaves and vertices in a subproblem T'_i. Properties 1 and 4 imply that $n_i = O(n/k)$, and property 3 implies that $\sum_i m_i = O(m)$. Lemma 2.1 then implies the size and complexity of $\mathcal{G}$. To bound the spanning ratio, consider the path $\pi(p, q)$ between two sites p, q. Properties 2 and 3 imply that this path exits a subtree T'_i and enters another subtree only at Steiner points. As within each subtree there is a $2t$-spanner, this is also the spanning ratio of $\mathcal{G}$. ◀

The output size K is either the size or complexity of $\mathcal{G}$, depending on whether we report the edges implicitly or explicitly. In the full version, we extend the result to a forest of trees.

3 Steiner spanners in simple polygons

We consider the problem of computing a t-spanner using k Steiner points for n point sites in a simple polygon P with m vertices. To obtain a low-complexity spanner we combine ideas from [2] and [8] with the forest spanner of Section 2.

We partition the polygon P recursively into two subpolygons P_ℓ and P_r by a vertical line segment λ such that roughly half of the sites lie in either subpolygon. For the line segment λ, we then consider the shortest path tree SPT_λ. This shortest path tree includes all sites in S and vertices of P. The segment λ is split into multiple edges at the projections of the sites. See Figure 3 for an example. The tree is rooted at the lower endpoint of λ. Observe that there are $O(m + n)$ vertices in SPT_λ.

Let $SPT_{i,j}$ denote the shortest path tree of the j-th subproblem at the i-th level of the recursion. We exclude all vertices from $SPT_{i,j}$ that have no site as a descendant. This ensures that all leaves of the tree are sites. Let $\mathcal{F} = \cup_{i,j} SPT_{i,j}$ be the forest consisting of all trees $SPT_{i,j}$. A site in S or vertex of P can occur in multiple trees $SPT_{i,j}$, but they are seen as distinct sites and vertices in the forest $\mathcal{F}$. For this forest we construct a $2t$-spanner $\mathcal{G}_\mathcal{F}$. A Steiner point in $\mathcal{G}_\mathcal{F}$ corresponds to either a vertex of P or a point on λ. Let $\mathcal{S}$ denote

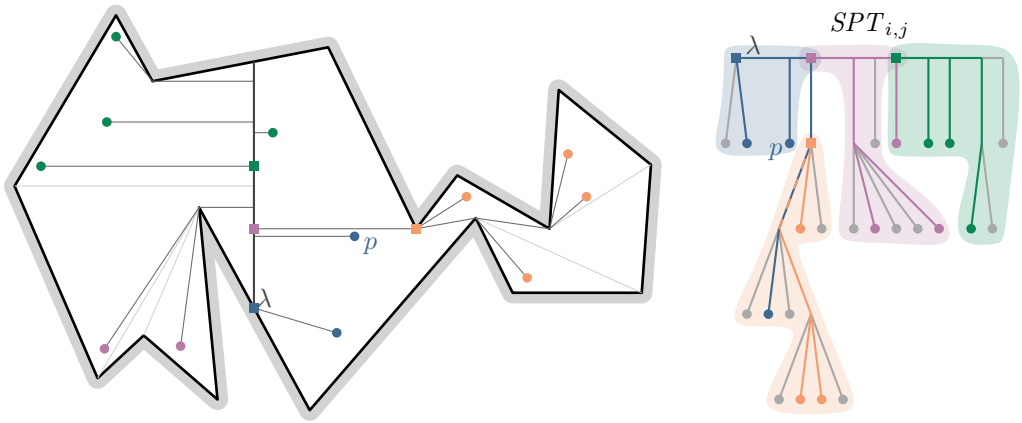


Figure 3 The shortest path tree of λ in P' and its $SPT_{i,j}$. The grey nodes and edges are not included in $SPT_{i,j}$, but can be assigned to a T'_i as indicated by the colored backgrounds. The squares show the Steiner points in $SPT_{i,j}$ and P'. The sites in P' are colored as the trees T'_i.

the set of Steiner points. To obtain a spanner $\mathcal{G}$ in the simple polygon, we add a link (p, q), $p, q \in S \cup \mathcal{S}$, to $\mathcal{G}$ whenever there is a link in $\mathcal{G}_{\mathcal{F}}$ between (a copy of) p and q.

To bound the complexity of the links in $\mathcal{G}$, we show that any path in a subtree T'_i as defined in Section 2 uses vertices of P in that subtree only. This implies that the bound on the complexity of the forest spanner also holds for the complexity of the links in the polygon. As the number of sites and vertices in $\mathcal{F}$ is increased by a factor $O(\log n)$ compared to n and m, we obtain the following theorem.

▶ **Theorem 3.1.** *Let S be a set of n point sites in a simple polygon P with m vertices, and $t \geq 1$ be any integer constant. For any $k \in \{1, \ldots, n\}$, we can build a geodesic $2\sqrt{2}t$-spanner with at most k Steiner points, size $O(n \log n \log(n/k))$, and complexity $O(mn^{1/t}(\log k)^{1+1/t}/k^{1/t} + n \log^2 n)$ in $O(n \log^2 n + m \log n + K)$ time, where K is the output size.*

References

1 Mohammad Ali Abam, Marjan Adeli, Hamid Homapour, and Pooya Zafar Asadollahpoor. Geometric spanners for points inside a polygonal domain. In *Proc. 31st International Symposium on Computational Geometry, SoCG*, volume 34 of *LIPIcs*, pages 186–197. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2015.

2 Mohammad Ali Abam, Mark de Berg, and Mohammad Javad Rezaei Seraji. Geodesic spanners for points on a polyhedral terrain. *SIAM J. Comput.*, 48(6):1796–1810, 2019.

3 Khaled M. Alzoubi, Xiang-Yang Li, Yu Wang, Peng-Jun Wan, and Ophir Frieder. Geometric spanners for wireless ad hoc networks. *IEEE Trans. Parallel Distributed Syst.*, 14(4):408–421, 2003.

4 Sujoy Bhore and Csaba D. Tóth. Light Euclidean Steiner spanners in the plane. In *Proc. 37th International Symposium on Computational Geometry, SoCG*, volume 189 of *LIPIcs*, pages 15:1–15:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

5 Glencora Borradaile and David Eppstein. Near-linear-time deterministic plane Steiner spanners for well-spaced point sets. *Comput. Geom.*, 49:8–16, 2015.

6 Kenneth L. Clarkson. Approximation algorithms for shortest path motion planning. In *Proc. 19th Annual ACM Symposium on Theory of Computing, STOC*, pages 56–65. ACM, 1987.

7 Vincent Cohen-Addad, Arnold Filtser, Philip N. Klein, and Hung Le. On light spanners, low-treewidth embeddings and efficient traversing in minor-free graphs. In *Proc. 61st IEEE Annual Symposium on Foundations of Computer Science, FOCS*, pages 589–600. IEEE, 2020.

8 Sarita de Berg, Marc J. van Kreveld, and Frank Staals. The complexity of geodesic spanners. In *Proc. 39th International Symposium on Computational Geometry, SoCG*, volume 258 of *LIPIcs*, pages 16:1–16:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023.

9 Michael Elkin and Shay Solomon. Narrow-shallow-low-light trees with and without Steiner points. *SIAM J. Discret. Math.*, 25(1):181–210, 2011.

10 Lee-Ad Gottlieb, Aryeh Kontorovich, and Robert Krauthgamer. Efficient regression in metric spaces via approximate Lipschitz extension. *IEEE Trans. Inf. Theory*, 63(8):4838–4849, 2017.

11 Hung Le and Shay Solomon. Truly optimal Euclidean spanners. In *Proc. 60th IEEE Annual Symposium on Foundations of Computer Science, FOCS*, pages 1078–1100. IEEE Computer Society, 2019.

12 Joseph S. B. Mitchell and Wolfgang Mulzer. Proximity algorithms. In *Handbook of Discrete and Computational Geometry (3rd Edition)*, chapter 32, pages 849–874. Chapman & Hall/CRC, 2017.

13 Giri Narasimhan and Michiel H. M. Smid. *Geometric Spanner Networks*. Cambridge University Press, 2007.

14 Jan Remy, Reto Spöhel, and Andreas Weißl. On Euclidean vehicle routing with allocation. *Comput. Geom.*, 43(4):357–376, 2010.

Enumerating At Most k-Out Polygons

Waseem Akram[1] and Katsuhisa Yamanaka[2]

1 **Indian Institute of Technology, Kanpur, India**
akram@iitk.ac.in
2 **Iwate University, Morioka, Japan**
yamanaka@iwate-u.ac.jp

Abstract

Let S be a set of n points in the Euclidean plane. A *simple polygon of* S is a simple polygon such that every vertex is a point of S. A simple polygon P of S is an *at most k-out polygon* if at most k points of S are outside P and the other points are either vertices of P or inside P. In this paper, we consider the problem of enumerating all the at most k-out polygons of S. We propose an algorithm that enumerates all the at most k-out polygons in $\mathcal{O}(n^3 \log n)$-delay and $\mathcal{O}(n^2)$ space.

1 Introduction

Let S be a set of n points in the Euclidean plane and general position, i.e., no three points are collinear. A *simple polygon of* S is a simple polygon such that every vertex is a point in S. In this paper, we focus on enumeration (or listing) problems of simple polygons of S.

For several classes of simple polygons of S, enumeration problems have been studied. A simple polygon of S is a *non-crossing spanning cycle* of S if every point in S is a vertex of the polygon. The non-crossing spanning cycles are appealing objects in the area of computational geometry and have been studied in the contexts of counting [6, 8, 15], random generation [1, 12, 13, 17], and enumeration [8, 15]. It was an open problem whether there exists an output-polynomial[1] time enumeration algorithm for non-crossing spanning cycles. Yamanaka *et al.* [16] proposed a new class of simple polygons, which is a relaxed version of the non-crossing spanning cycles. A *surrounding polygon* of S is a simple polygon such that every point in S is either a vertex of the polygon or inside the polygon. They also proposed an algorithm that enumerates all the surrounding polygons of S in $\mathcal{O}(n^2 \log n)$ time for each. The running time was improved to $\mathcal{O}(n^2)$ time for each [14]. Very recently, by using the enumeration algorithm of the surrounding polygons, Eppstein [5] showed that non-crossing spanning cycles of a point set can be enumerated in output-polynomial time.

Empty convex polygons are also an important class of simple polygons of S. A simple polygon of S is an *empty convex polygon* if the polygon is convex and every point is either a vertex of the polygon or outside the polygon. The empty convex polygons have been studied in the contexts of counting [3, 7, 11, 10] and enumeration [4]. Terui *et al.* [14] proposed a new class of simple polygons, which is a generalization of the empty convex polygons. A simple polygon of S is an *empty polygon* if every point in S is either a vertex of the polygon or outside the polygon. They proposed an algorithm that enumerates all the empty polygons of S in $\mathcal{O}(n^2)$ time for each.

In this paper, we propose a new class of simple polygons of S. A simple polygon P of S is an *at most k-out polygon* if there are at most k points outside P and the other points are either vertices of P or inside P. See Figure 1 for examples. The class of at most k-out

[1] An enumeration algorithm is *output-polynomial* if the algorithm enumerates all the objects in polynomial-time of the input and output size.

40th European Workshop on Computational Geometry, Ioannina, Greece, March 13–15, 2024.
This is an extended abstract of a presentation given at EuroCG'24. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

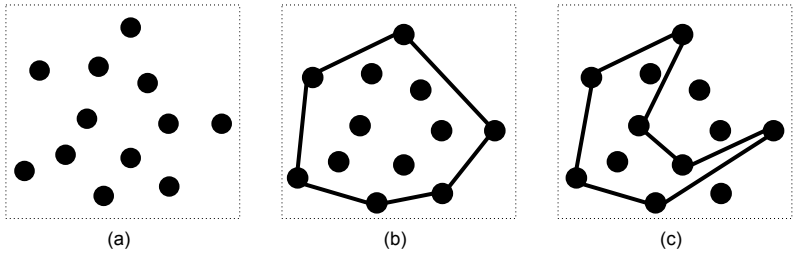


Figure 1 (a) A given point set S and (b), (c) two at most 3-out polygons of S.

polygons is a generalization of the class of surrounding polygons in the sense that the set of at most k-out polygons of S coincides with (1) the set of surrounding polygons of S when $k = 0$ and (2) the set of simple polygons of S when $k = n - 3$. We design an algorithm that enumerates all the at most k-out polygons of S in polynomial delay[2]. Our enumeration algorithm is based on the reverse-search technique by Avis and Fukuda [2].

Due to space limitations, all the proofs are omitted.

2 Preliminaries

Let S be a set of n points in the Euclidean plane. Throughout this paper, we assume that S is in general position, i.e., no three points are collinear. The *upper-left point* of $S' \subseteq S$ is the point with the minimum x-coordinate among S'. If ties exist, we choose the point with the maximum y-coordinate among them.

A sequence $P = \langle p_1, p_2, \ldots, p_t \rangle$, $(t \leq n)$, of points in S is a *simple polygon of* S if the alternating sequence $\langle p_1, (p_1, p_2), p_2, (p_2, p_3), \ldots, p_t, (p_t, p_1) \rangle$ of points and line segments forms a simple polygon. Let $P = \langle p_1, p_2, \ldots, p_t \rangle$ be a simple polygon of S. We suppose that the vertices of P appear in counterclockwise order starting from the upper-left vertex p_1 among $\{p_1, p_2, \ldots, p_t\}$. We denote by $\mathsf{in}(P) \subseteq S$ and $\mathsf{out}(P) \subseteq S$ the sets of the points inside and outside P, respectively. Note that each of $\mathsf{in}(P)$ and $\mathsf{out}(P)$ does not include any vertex on P. We denote by $p_i \prec p_j$ if $i < j$ holds, and we say that p_j is *larger than* p_i on P. $\mathsf{pred}(p_i)$ and $\mathsf{succ}(p_i)$ denote the predecessor and successor of p_i of P, respectively. Note that the successor of p_t is p_1. For two edges $(p_i, \mathsf{succ}(p_i))$ and $(p_j, \mathsf{succ}(p_j))$ of P, we say that $(p_j, \mathsf{succ}(p_j))$ is *larger than* $(p_i, \mathsf{succ}(p_i))$ if $i < j$ holds. Suppose that P has 4 or more vertices. A vertex p_i of P is *embeddable* if the triangle consisting of $\mathsf{pred}(p_i)$, p_i, and $\mathsf{succ}(p_i)$ does not intersect the interior of P and includes no point in $\mathsf{out}(P)$. An *embedment* of an embeddable vertex p_i of P is to remove two edges $(\mathsf{pred}(p_i), p_i)$ and $(p_i, \mathsf{succ}(p_i))$ and insert the edge $(\mathsf{pred}(p_i), \mathsf{succ}(p_i))$. We denote by $\mathsf{emb}(P, p_i)$ the simple polygon obtained from P by applying the embedment of p_i to P. See Figure 2 for examples. A point $p \in \mathsf{out}(P)$ is *insertable* to an edge $(p_i, \mathsf{succ}(p_i))$ of P if the triangle consisting of p, p_i, and $\mathsf{succ}(p_i)$ does not intersect the interior of P and includes no point in $\mathsf{out}(P)$. For a point $p \in \mathsf{out}(P)$ insertable to an edge $(p_i, \mathsf{succ}(p_i))$ of P, the *insertion* of p to $(p_i, \mathsf{succ}(p_i))$ is to remove $(p_i, \mathsf{succ}(p_i))$ and insert the two edges (p_i, p) and $(p, \mathsf{succ}(p_i))$. We denote by $\mathsf{ins}(P, (p_i, \mathsf{succ}(p_i)), p)$ the simple polygon obtained from P by applying the insertion of p to $(p_i, \mathsf{succ}(p_i))$ on P. See Figure 3 for examples.

[2] An enumeration algorithm is polynomial delay if the algorithm enumerates all the objects such that the delay time of any two consecutive outputs is bounded by a polynomial of the input size.

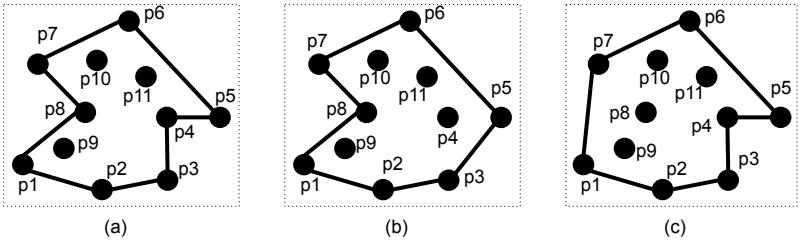


Figure 2 (a) A simple polygon of a point set $S = \{p_1, p_2, \ldots, p_{11}\}$, where p_4 and p_8 are embeddable vertices. (b) The simple polygon of S obtained from the polygon of (a) by embedding p_4. (c) The simple polygon of S obtained from the polygon of (a) by embedding p_8.

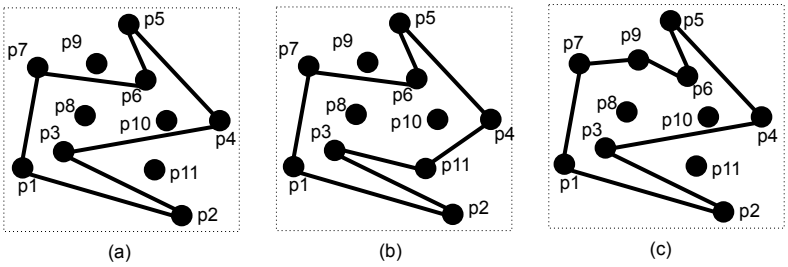


Figure 3 (a) An at most 2-out polygon of a point set with no embeddable vertices and $\mathsf{out}(P) = \{p_9, p_{11}\}$. (b) The polygon obtained from the polygon of (a) by inserting p_{11} to (p_3, p_4). (c) The polygon obtained from the polygon of (a) by inserting p_9 to (p_6, p_7).

The *convex hull*, denoted by $\mathsf{CH}(S)$, of S is the simple polygon with the smallest area that contains all the points in S. A simple polygon P of S is an *at most k-out polygon* of S if $|\mathsf{out}(P)| \leq k$ holds. Figure 1 shows examples of at most k-out polygons. We denote the set of the at most k-out polygons of S by $\mathcal{S}_{\leq k}(S)$.

3 Enumeration of at most k-out polygons

Let S be a set of n points in the Euclidean plane. In Section 3.1, we define a tree structure on $\mathcal{S}_{\leq k}(S)$, called a family tree. By traversing the family tree, we enumerate all the polygons in $\mathcal{S}_{\leq k}(S)$. In Section 3.2, we design an algorithm to traverse the family tree.

3.1 Family tree of at most k-out polygons

Let $P = \langle p_1, p_2, \ldots, p_t \rangle$ $(t \leq n)$ be a polygon in $\mathcal{S}_{\leq k}(S) \setminus \{\mathsf{CH}(S)\}$. Suppose that p_1 is the upper-left vertex of $\{p_1, p_2, \ldots, p_t\}$ and the vertices of P are arranged in the counterclockwise order. Let $p \in \mathsf{out}(P)$ be a point insertable to an edge $(p_i, \mathsf{succ}(p_i))$. Then, we define the *distance* of $(p_i, \mathsf{succ}(p_i))$ from p as the Euclidean distance between the midpoint of $(p_i, \mathsf{succ}(p_i))$ and p. The distance from p to $(p_i, \mathsf{succ}(p_i))$ is denoted by $\mathsf{dist}((p_i, \mathsf{succ}(p_i)), p)$. Note that, if p is not insertable to an edge $(p_i, \mathsf{succ}(p_i))$, the distance from p to $(p_i, \mathsf{succ}(p_i))$ is not defined. We denote the closest edge of P among the edges insertable from p by $\mathsf{cloe}(P, p)$. If ties exist, the largest edge is $\mathsf{cloe}(P, p)$.

We denote the set of the points insertable to at least one edge of P by $\mathsf{iout}(P) \subseteq \mathsf{out}(P)$. A point $p \in \mathsf{iout}(P)$ is the *closest outside point*, denoted by $\mathsf{clop}(P)$, of P if

$$\mathsf{dist}(\mathsf{cloe}(P, p), p) = \min_{q \in \mathsf{iout}(P)} \{\mathsf{dist}(\mathsf{cloe}(P, q), q)\}$$

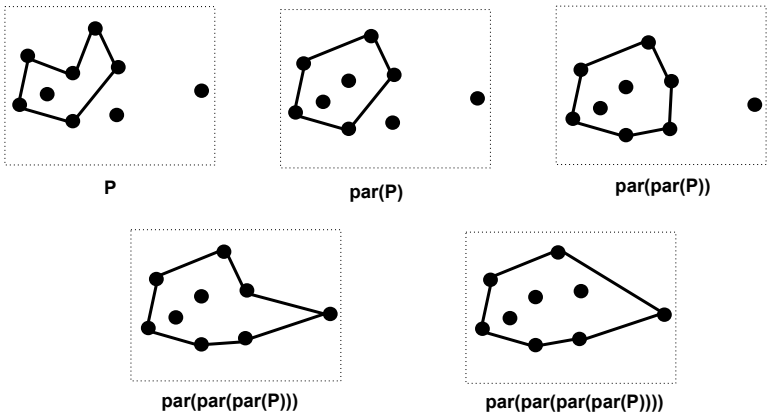


Figure 4 A parent sequence of at most 2-out polygon P.

holds. If ties exist, the point with the largest x-coordinate and y-coordinate values is chosen as the closest outside point.

▶ **Lemma 3.1.** *Let P be a polygon in $\mathcal{S}_{\leq k}(S) \setminus \{\mathsf{CH}(S)\}$. There exists either an embeddable vertex of P or an insertable point in $\mathsf{out}(S)$.*

We denote by $\mathsf{larg}(P)$ the largest embeddable vertex of P. For convenience, we define $\mathsf{larg}(P) := \emptyset$ if P has no embeddable vertex. Then, we define the *parent* of P as follows:

$$\mathsf{par}(P) := \begin{cases} \mathsf{emb}(P, \mathsf{larg}(P)) & \text{if } P \text{ has an embeddable vertex,} \\ \mathsf{ins}(P, \mathsf{cloe}(P, \mathsf{clop}(P)), \mathsf{clop}(P)) & \text{otherwise.} \end{cases}$$

▶ **Lemma 3.2.** *Let P be an at most k-out polygon in $\mathcal{S}_{\leq k}(S) \setminus \{\mathsf{CH}(S)\}$. Then, the parent $\mathsf{par}(P)$ of P is an at most k-out polygon of S, always exists and is unique.*

By repeatedly finding the parents from P, we obtain a sequence of at most k-out polygons of S. The *parent sequence* $\mathsf{PS}(P) = \langle P_1, P_2, \ldots, P_\ell \rangle$ of P is a sequence of at most k-out polygons such that the first polygon is P itself and P_i is the parent of P_{i-1} for each $i = 2, 3, \ldots, \ell$. See Figure 4 for an example. As we can see in the following lemma, the last polygon in a parent sequence is always $\mathsf{CH}(S)$.

▶ **Lemma 3.3.** *For a polygon $P \in \mathcal{S}_{\leq k}(S) \setminus \{\mathsf{CH}(S)\}$, the last polygon of $\mathsf{PS}(P)$ is $\mathsf{CH}(S)$.*

From Lemma 3.3, for any at most k-out polygon, the last polygon of its parent sequence is the convex hull of S. By merging the parent sequences for all the at most k-out polygons in $\mathcal{S}_{\leq k}(S)$, we have a tree structure rooted at $\mathsf{CH}(S)$. We call such a tree the *family tree* of $\mathcal{S}_{\leq k}(S)$. An example of the family tree is shown in Figure 5.

3.2 Enumeration algorithm of at most k-out polygons

A pair (p_i, p) of a vertex p_i of P and a point $p \in \mathsf{in}(P)$ is *digable* if the triangle consisting of p_i, p, and $\mathsf{succ}(p_i)$ lies inside P and does not contain any point of S. A *dig operation* to a digable pair (p_i, p) removes the edge $(p_i, \mathsf{succ}(p_i))$ and inserts the two edges (p_i, p) and $(p, \mathsf{succ}(p_i))$. $\mathsf{dig}(P, p_i, p)$ denotes the resulting polygon. Note that $\mathsf{dig}(P, p_i, p)$ is also a polygon in $\mathcal{S}_{\leq k}(S)$. A vertex p_i of P is *removable* if (1) $|\mathsf{out}(P)| < k$ holds, (2) the triangle consisting of $\mathsf{pred}(p_i), p_i$, and $\mathsf{succ}(p_i)$ lies inside P, and (3) the triangle does not

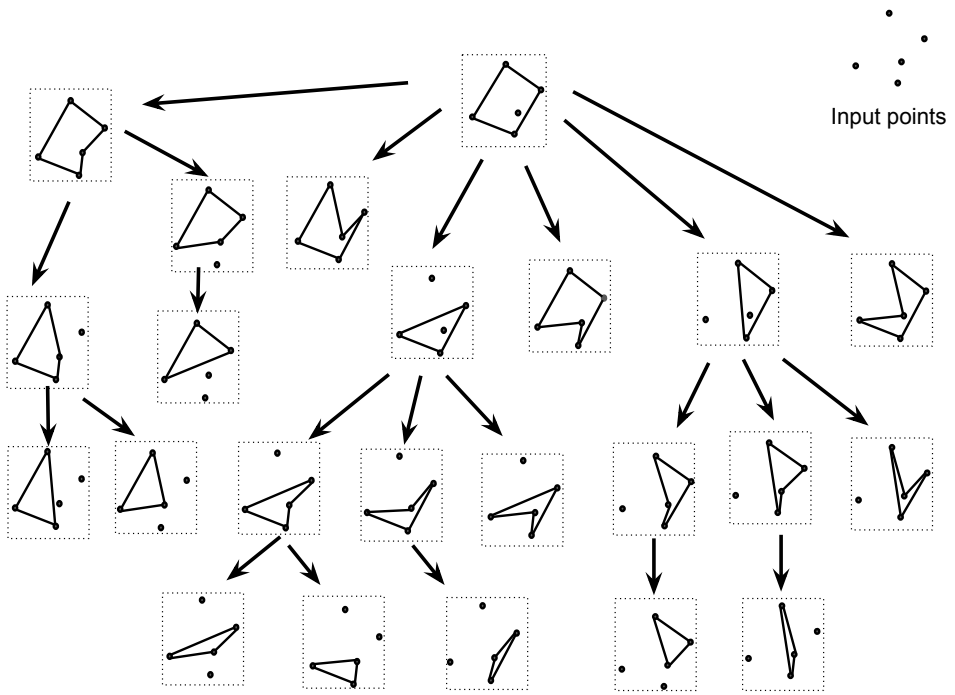


Figure 5 An example of a family tree

contain any point of S. A remove operation to a removable vertex p_i removes the two edges $(\mathsf{pred}(p_i), p_i)$ and $(p_i, \mathsf{succ}(p_i))$, and inserts an edge $(\mathsf{pred}(p_i), \mathsf{succ}(p_i))$ to P. $\mathsf{rmv}(P, p_i)$ denotes the resulting polygon. Note that $\mathsf{rmv}(P, p_i)$ is also a polygon in $\mathcal{S}_{\leq k}(S)$.

It can be observed that $\mathsf{dig}(P, p_i, p)$ and $\mathsf{rmv}(P, p_j)$ are children of P if $P = \mathsf{par}(\mathsf{dig}(P, p_i, p))$ and $P = \mathsf{par}(\mathsf{rmv}(P, p_j))$ holds, respectively. We say that a digable pair (p_i, p) and a removable vertex p_j are *active* if $\mathsf{dig}(P, p_i, p)$ and $\mathsf{rmv}(P, p_j)$ are children of P, respectively. Now, we have the following lemma.

▶ **Lemma 3.4.** *Let $P = \langle p_1, p_2, \ldots, p_t \rangle$, $(t \leq n)$, be an at most k-out polygon of a set of n points. Let (p_i, p) be a digable pair, where p_i is a vertex of P and $p \in \mathsf{in}(P)$, and let p_j be a removable vertex of P. Then,*

1. *(p_i, p) is active if $p = \mathsf{larg}(\mathsf{dig}(P, p_i, p))$ holds and*
2. *p_j is active if $\mathsf{rmv}(P, p_j)$ has no embeddable vertex, $p_j = \mathsf{clop}(\mathsf{rmv}(P, p_j))$ holds, and $(\mathsf{pred}(p_j), \mathsf{succ}(p_j)) = \mathsf{cloe}(\mathsf{rmv}(P, p_j), p_j)$.*

As stated in the following lemma, we can check whether a given pair (p_i, p) and a given vertex p_j of P are active, respectively.

▶ **Lemma 3.5.** *Let P be an at most k-out polygon of a set S of n points.*

1. *Given a pair (p_i, p), where p_i is a vertex of P and $p \in \mathsf{in}(P)$, and given $\mathsf{larg}(P)$, one can check whether (p_i, p) is active in $\mathcal{O}(\log n)$ time and*
2. *given a vertex p_i and the number of the embeddable vertices, denoted by $\#\mathsf{emb}(P)$, of P, we can check whether p_i is active in $\mathcal{O}(n^2 \log n)$ time*

with $\mathcal{O}(n^2)$-time preprocessing and $\mathcal{O}(n^2)$-additional space for triangular range queries on S and $\mathcal{O}(n \log n)$-time preprocessing and $\mathcal{O}(n)$-additional space for ray shooting queries on P.

Algorithm 1: `Enum`(S, k)

Construct the convex hull $\mathsf{CH}(S)$ of the input point set S;
Preprocess S for triangular range queries;
`Find-Children`$(\mathsf{CH}(S), \emptyset, 0)$;

Algorithm 2: `Find-Children`$(P = \langle p_1, p_2, \ldots, p_t \rangle, p_j, \#\mathsf{emb}(P))$

Output P;
Preprocess P for ray shooting queries;
if $p_j = \emptyset$ **then** $q = p_0$;
else $q = \mathsf{pred}(p_j)$;
`/* Note that` $p_j = \mathsf{larg}(P)$ `and` p_0 `is a sentinel vertex satisfying` $p_0 \prec p_i$ `for each` $i = 1, 2, \ldots, t$. `*/`
foreach *point* p_i *with* $q \prec p_i$ **do**
  foreach *point* $p \in \mathsf{in}(P)$ **do**
    if *the pair* (p_i, p) *is active* **then**
      `Find-Children`$(\mathsf{dig}(P, p_i, p), p, \#\mathsf{emb}(\mathsf{dig}(P, p_i, p)))$;
foreach *vertex* p_i *of* P **do**
  if p_i *is active* **then** `Find-Children`$(\mathsf{rmv}(P, p_i), \emptyset, \#\mathsf{emb}(\mathsf{rmv}(P, p_i)))$;

Now, we are ready to describe the pseudo-codes of our enumeration algorithm. Algorithm 1 is the main routine and Algorithm 2 is a subroutine to enumerate children.

Algorithm 1 first constructs the convex hull $\mathsf{CH}(S)$ of the input point set S. Then, it executes a preprocess to S for efficiently answering triangular range queries. Note that the preprocess for triangular range queries is executed only once in our algorithm. Algorithm 2 first outputs P and executes a preprocess to the given polygon P for efficiently answering ray shooting queries. The preprocess is done once for a recursive call. Next, the algorithm enumerates all the children of P by dig and remove operations. Note that the above two preprocesses allow us to use Lemma 3.5. Hence, we can check whether or not candidate pairs and vertices are active. We have our main theorem.

▶ **Theorem 3.6.** *Let S be a set of n points in the Euclidean plane, and let k be an integer with $0 \leq k \leq n-3$. One can enumerate all the at most k-out polygons in $\mathcal{S}_{\leq k}(S)$ in $\mathcal{O}(n^3 \log n |\mathcal{S}_{\leq k}(S)|)$ time and $\mathcal{O}(n^2)$ space.*

By applying the alternative output method [9], we can enumerate in polynomial-delay.

▶ **Corollary 3.7.** *Let S be a set of n points in the Euclidean plane, and let k be an integer with $0 \leq k \leq n-3$. One can enumerate all the at most k-out polygons in $\mathcal{S}_{\leq k}(S)$ in $\mathcal{O}(n^3 \log n)$-delay and $\mathcal{O}(n^2)$ space.*

4 Conclusions

We have designed an algorithm that enumerates all the polygons in $\mathcal{S}_{\leq k}(S)$ in $\mathcal{O}(n^3 \log n)$-delay and $\mathcal{O}(n^2)$ space, where S is the set of n points in Euclidean plane and general position. Our future work include improving the running time of the algorithm.

Acknowledgements

The authors thank anonymous referees for their helpful suggestions. The first author thanks his PhD advisor Sanjeev Saxena for his help, encouragement, comments, and discussions. This work was supported by JSPS KAKENHI Grant Number JP23K11027.

References

1 Thomas Auer and Martin Held. Heuristics for the generation of random polygons. In *Proceedings of the 8th Canadian Conference on Computational Geometry*, pages 38–43, 1996.

2 David Avis and Komei Fukuda. Reverse search for enumeration. *Discrete Applied Mathematics*, 65(1-3):21–46, 1996.

3 Sang Won Bae. Faster counting empty convex polygons in a planar point set. *Inf. Process. Lett.*, 175:106221, 2022. doi:10.1016/j.ipl.2021.106221.

4 David P. Dobkin, Herbert Edelsbrunner, and Mark H. Overmars. Searching for empty convex polygons. *Algorithmica*, 5(4):561–571, 1990. doi:10.1007/BF01840404.

5 David Eppstein. Non-crossing hamiltonian paths and cycles in output-polynomial time. In Erin W. Chambers and Joachim Gudmundsson, editors, *Proceedings of the 39th International Symposium on Computational Geometry, June 12-15, 2023, Dallas, Texas, USA*, volume 258 of *LIPIcs*, pages 29:1–29:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023. URL: https://doi.org/10.4230/LIPIcs.SoCG.2023.29, doi:10.4230/LIPICS.SOCG.2023.29.

6 Dániel Marx and Tillmann Miltzow. Peeling and nibbling the cactus: Subexponential-time algorithms for counting triangulations and related problems. In *32nd International Symposium on Computational Geometry, SoCG 2016, June 14-18, 2016, Boston, MA, USA*, pages 52:1–52:16, 2016. doi:10.4230/LIPIcs.SoCG.2016.52.

7 Joseph S. B. Mitchell, Günter Rote, Gopalakrishnan Sundaram, and Gerhard J. Woeginger. Counting convex polygons in planar point sets. *Inf. Process. Lett.*, 56(1):45–49, 1995. doi:10.1016/0020-0190(95)00130-5.

8 Yu Nakahata, Takashi Horiyama, Shin-ichi Minato, and Katsuhisa Yamanaka. Compiling crossing-free geometric graphs with connectivity constraint for fast enumeration, random sampling, and optimization. *CoRR*, abs/2001.08899, 2020. URL: https://arxiv.org/abs/2001.08899, arXiv:2001.08899.

9 Shin-ichi Nakano and Takeaki Uno. Generating colored trees. *Proceedings of the 31th Workshop on Graph-Theoretic Concepts in Computer Science, (WG 2005)*, LNCS 3787:249–260, 2005.

10 Günter Rote and Gerhard J. Woeginger. Counting convex k-gons in planar point sets. *Information Processing Letters*, 41(4):191–194, 1992. doi:10.1016/0020-0190(92)90178-X.

11 Günter Rote, Gerhard J. Woeginger, Binhai Zhu, and Zhengyan Wang. Counting k-subsets and convex k-gons in the plane. *Inf. Process. Lett.*, 38(3):149–151, 1991. doi:10.1016/0020-0190(91)90237-C.

12 Christian Sohler. Generating random star-shaped polygons. In *Proceedings of the 11th Canadian Conference on Computational Geometry*, pages 174–177, 1999.

13 Sachio Teramoto, Mitsuo Motoki, Ryuhei Uehara, and Tetsuo Asano. Heuristics for generating a simple polygonalization. IPSJ SIG Technical Report 2006-AL-106(6), Information Processing Society of Japan, May 2006.

14 Shunta Terui, Katsuhisa Yamanaka, Takashi Hirayama, Takashi Horiyama, Kazuhiro Kurita, and Takeaki Uno. Enumerating empty and surrounding polygons. *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.*, 106(9):1082–1091, 2023. URL: https://doi.org/10.1587/transfun.2022dmp0007, doi:10.1587/TRANSFUN.2022DMP0007.

15 Manuel Wettstein. Counting and enumerating crossing-free geometric graphs. *Journal of Computational Geometry*, 8(1):47–77, 2017.

16 Katsuhisa Yamanaka, David Avis, Takashi Horiyama, Yoshio Okamoto, Ryuhei Uehara, and Tanami Yamauchi. Algorithmic enumeration of surrounding polygons. *Discrete Applied Mathematics*, 303:305–313, 2021. doi:10.1016/j.dam.2020.03.034.

17 Chong Zhu, Gopalakrishnan Sundaram, Jack Snoeyink, and Joseph S. B. Mitchell. Generating random polygons with given vertices. *Computational Geometry: Theory and Applications*, 6:277–290, 1996.

LITE: A Framework for Lattice-Integrated Embedding of Topological Descriptors

Michael E. Van Huffel[1] **and Matteo Palo**[2]

1 **Department of Mathematics, ETH Zürich**
michavan@student.ethz.ch
2 **Department of Mathematics, ETH Zürich**
mapalo@student.ethz.ch

Abstract

This paper introduces LITE (Lattice Integrated Topological Embedding), a novel approach to converting persistence diagrams into finite-dimensional vectors using discrete measure-based functionals. Our primary focus in this work is on identity and frequency-based transforms but we do not restrict our framework to them. Our comparative studies reveal that LITE is competitive with, and often superior to, topological data analysis methods from the literature in common benchmark classification tasks. This work offers a new viewpoint for data scientists, challenges prevailing diagram vectorization techniques, and lays groundwork for simpler, more effective use of persistence diagrams in machine learning.

Related Version arXiv:2312.17093

1 Introduction

Topological Data Analysis (TDA) has emerged as a transformative approach in data science, providing useful insights into the underlying structure of complex datasets through the capture of their topological features. The effectiveness of machine learning algorithms, particularly in pattern recognition and feature extraction, underscores the importance of understanding data geometry. TDA offers a more sophisticated exploration of this geometric landscape, leading to numerous successful applications across various fields. Notable examples include neuroscience [2, 9], materials science [24], and environmental science [11].

Persistent homology, a core methodology in TDA, systematically keeps track of the appearing and disappearing of topological characteristics across a sequence of nested topological spaces [12, 25]. These topological features are typically represented through persistence diagrams (PDs). However, the space of these diagrams is unstructured: they vary in the number of points they contain, and operations like addition and scalar multiplication are not clearly defined. This lack of structure [4, 18], poses significant challenges in integrating PDs into machine learning workflows, where such a space is often crucial for diverse techniques including classification, neural networks, and feature selection.

Precise Problem Formulation. The unstructured nature of PDs, hinders their straightforward integration into traditional machine learning pipelines. This necessitates the development of innovative embedding techniques to effectively transform these diagrams into elements within a space suitable for machine learning workflows.

1.1 Related Work and Contribution

To address the unstructured nature of persistence diagram spaces, two main methods have been highlighted in literature: vectorization and kernel-based methods. Vectorization includes

40th European Workshop on Computational Geometry, Ioannina, Greece, March 13–15, 2024.
This is an extended abstract of a presentation given at EuroCG'24. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

Persistence Images [1] and Persistence Landscapes [3], with their multi-parameter extensions for increased robustness [5, 23], and modern techniques like ATOL, which quantizes diagram spaces, and PersLay, introducing a NN architecture for vectorization. The kernel-based approach crafts specific kernels, such as the multi-scale [19], weighted Gaussian [16], and sliced Wasserstein kernels [7], offering performance comparable to vectorization methods, despite representational and scalability challenges.

This work contributes to the computational geometry literature by introducing LITE, a new vectorization framework in TDA that conceives PDs as measures in $\mathbb{R}^2_+$, discretizes these measures on a lattice, and transforms them into finite-dimensional vectors through identity and frequency-based transforms. Our approach, distinguished by its simplicity and effectiveness, challenges the prevailing trends in TDA literature on embedding diagrams into vector spaces. We achieve results comparable to those in the TDA literature on classical graph classification benchmark tasks, and with frequency-based transforms, we even often surpass them.

1.2 Basic Definitions

In the realm of computational geometry, persistent homology is a key technique for analyzing topological features across scales. It utilizes a filtration process, forming a sequence of nested topological spaces $X_0 \subseteq X_1 \subseteq \cdots \subseteq X_n = X$, to dissect the dataset's topological structure at various levels of granularity. This analysis is typically represented using PDs. These diagrams are multisets of points in the extended half-plane $\Omega = \{(x, y) \in \mathbb{R}^2 | x \leq y\}$, including the diagonal $\partial\Omega = \{(x, x) \in \mathbb{R}^2\}$ with infinite multiplicity. Each point (x, y) in the diagram corresponds to a topological feature, with x and y indicating the *birth* and *death* of the feature, respectively. The *persistence* of a feature is quantified as $y - x$, representing its lifespan within the filtration. For our analysis, we assume that all features in our PDs exhibit finite persistence. To compare PDs, we use the p-Wasserstein distance. For diagrams D_1 and D_2, it is mathematically defined as:

$$\mathrm{W}_p\left(D_1, D_2\right) = \left(\inf_{\gamma} \sum_{x \in D_1} \|x - \gamma(x)\|_p^p\right)^{\frac{1}{p}}$$

Here, γ ranges over all bijections between D_1 and D_2, and $\|\cdot\|_p$ denotes the p-norm on $\mathbb{R}^2$.

In [8], an alternative interpretation of persistence diagrams is presented, defining them as measure expressed by $\mu = \sum_{\mathbf{x} \in D} m(\mathbf{x})\delta_{\mathbf{x}}$, where δ is the Kronecker delta, $D \subset \Omega$ is locally finite, and $m(\mathbf{x}) \in \mathbb{N}$ is the multiplicity of each $\mathbf{x}$, for all $\mathbf{x} \in D$. This results in μ being a locally finite measure supported on Ω with an integer mass on each point of its support.

Following [10], we define the *p-persistence* of a measure μ, for finite $p \geq 1$, as $\mathrm{Pers}_p(\mu) := \int_\Omega d(x, \partial\Omega)^p \, \mathrm{d}\mu(x)$. Here, the term $d(x, \partial\Omega) := \inf_{y \in \partial\Omega} d(x, y)$ signifies the distance from a point $x \in \Omega$ to its orthogonal projection onto the diagonal $\partial\Omega$. We define $\mathcal{M}^p$ as the set of all persistence measures with finite *p-persistence*. Similar to PDs, we use the p-Optimal Partial Transport distance to compare persistence measures, which we omit defining here due to space constraints. When the measures have the same mass over the space Ω, this metric coincides with the p-Wasserstein distance. For detailed information, see our extended arXiv version [15] and [13] for an introduction to the field.

2 Methodology

This section presents the LITE vectorization process, outlined in Algorithm 1. All proofs are provided in extended version on arXiv [15].

Algorithm 1 Lattice Integrated Topological Embedding (LITE)

Require: f: Transform Function with Hyperparameters, Grid $\{N, M\}$, Finesse Δ, PDs list
1: Discretize PDs on grid
2: Compute Functional on grid
Ensure: Embedding of PDs

2.1 Discretized Persistence Diagrams

The framework of our work is rooted in the computation of *discretized persistence diagrams (PDs)*, where measures are confined to allocating mass exclusively at points on a lattice measure space Γ_p, as detailed in Lemma 2.1. The discretization process consists of two main steps: a shifting step, where we transform the measure $\mu \in \mathcal{M}^p$ induced by a persistence diagram using $\tau(\mu) = (x_1, x_2 - x_1)$ for all (x_1, x_2) such that $\mu(x_1, x_2) \neq 0$, to convert birth-death coordinates to birth-persistence coordinates; and a mapping step, utilizing Proposition 2.2 to obtain a persistence measure $\nu^\star$ on Γ_p.

▶ **Definition 2.1.** Let $G_{N,M}$ be a regular grid on $\mathbb{R}^2_+$ consisting of points $\{(x_i, y_j) \mid x_i = i\Delta, y_j = j\Delta, i = 0, \ldots, N-1, j = 0, \ldots, M-1\}$ where Δ is the grid finesse. Define $\mathcal{S}$ as the σ-algebra containing all subsets of $G_{N,M}$, and $\mu : \mathcal{S} \to [0, \infty]$ as a measure such that for any $A \in \mathcal{S}$, $\mu(A) = \sum_{(x_i,y_j)\in A} m_{ij} \in \mathcal{M}^p$ where m_{ij} is an integer mass assigned to the point (x_i, y_j). The triple $\Gamma_p = (G_{N,M}, \mathcal{S}, \mu)$ constitutes a discrete measure space.

▶ **Proposition 2.2.** *Let $\Gamma_p \subset \mathcal{M}^p$ be a discrete measure space as outlined in Definition 2.1. For a persistence diagram D and a measure $\mu = \sum_{\mathbf{x}\in\tau(D)} m(\mathbf{x})\delta_{\mathbf{x}} \in \mathcal{M}^p$, consider the 1-Wasserstein distance W_1 between μ any $\nu \in \Gamma_p$. Consider the optimization problem*

$$\nu^\star = \arg\min_{\nu'\in\Gamma_p} W_1(\mu, \nu'), \quad s.t. \quad \nu'(G_{N,M}) = \mu(D)\,.$$

If we choose $\nu = \sum_{\mathbf{x}\in\tau(D)} m(\Theta(\mathbf{x}))\delta_{\Theta(\mathbf{x})}$, where $\Theta : \mathbb{R}^2_+ \to G_{N,M}$ is a mapping that assigns each point $\mathbf{x} \in \tau(D)$ to the closest point in $G_{N,M}$ minimizing $\|\mathbf{x} - \mathbf{x}'\|_1$, then it holds that $\nu = \nu^\star$ is the solution to the optimization problem.

2.2 Functionals on Persistence Measures

In this study, we define a functional $\Psi_\mu(f)$ for $\mu \in \Gamma_p$ as $\Psi_\mu(f) := \int_\Omega f(\mathbf{x}, \cdot)d\mu(\mathbf{x}) = \sum_{\mathbf{x}\in D} m(\mathbf{x})f(\mathbf{x}, \cdot)$, utilizing the discrete nature of μ. This functional maps from lattice measure space Γ_p to a function space $\mathcal{F}(\Gamma_p)$. Here we focus on three functions for frequency and time-frequency distribution, $f(\mathbf{x}, \cdot)$: the Gabor Transform and the Wavelet Transform for time-frequency distributions, along with the Fourier Transform for frequency analysis. These transforms map to the frequency domain, situating $\mathcal{F}(\Gamma_p)$ as a vector space. We additionally employ the identity transform $f(\mathbf{x}, \cdot) = \mathbf{x}$. The rationale for these transforms is detailed in our extended work on arXiv [15]. All these transforms output coefficients or magnitude-phase numbers on a lattice. We convert these into vectors by flattening the lattice

into a one dimensional array.

While it is possible to establish the stability of our vectorization method for a fixed $\Delta > 0$, assuming that for all $x \in D$ and $x' \in D'$, the condition $\|x - x'\| > \Delta$ holds, proving stability with a universal constant for general PDs is not feasible. This limitation arises due to the existence of scenarios where points from PDs can be made arbitrarily close but still are mapped to different bins in the grid.

3 Results

In this section, we concisely demonstrate how LITE preserves topological information, rivaling state-of-the-art methods in TDA. Our experiments focus on two classification tasks: graph-based and point cloud classification from dynamical systems. Experimental setups and implementation details of our methods for the Graph Classification tasks are reported in our arXiv version, [15].

3.1 Graph Classification

We evaluated our methods using established graph classification benchmarks. This included social graph datasets `IMDB-B` and `IMDB-M`, as well as chemoinformatics and bioinformatics datasets `COX2`, `DHFR`, `MUTAG`, and `PROTEINS`, all sourced from [22].

The highest accuracies achieved with our frequency transforms (LITE) as well as the accuracy for the identity transform (LITE-`IdT`) are presented in Table 1.

Dataset	SV†	P†	MP†	Perslay*	ATOL*	BBA†	LITE (Our)		LITE-`IdT` (Our)	
							Mean*	Max†	Mean*	Max†
`MUTAG`	88.3	79.2	86.1	89.8	88.3	90.4	89.8	91.7	89.2	90.7
`COX2`	78.4	76.0	79.9	80.9	79.4	81.2	80.6	82.4	79.4	80.4
`PROTEINS`	72.6	65.4	67.5	74.8	71.4	74.7	72.8	73.6	72.2	73.2
`DHFR`	78.4	70.9	81.7	80.3	82.7	80.5	81.8	83.1	81.2	82.7
`IMDB-B`	72.9	54.0	68.7	71.2	74.8	69.4	68.4	69.8	67.2	68.3
`IMDB-M`	50.3	36.3	46.9	48.8	47.8	46.7	43.7	44.4	43.1	44.3

Table 1 Comparative Analysis of Classification Accuracy with topological methods on Benchmark Graph Datasets. Note: Symbol † compare with *Max* metric, while ⋆ with *Mean* due to different experimental setup.

Aligning with [6], [21], and [14], we benchmark our frequency transforms against leading TDA methods (SV [22], P[1, 3], MP [5, 7, 16, 19], Perslay [6], ATOL [21] and BBA [14], see arXiv version [15] for more details on these methods). In Table 1, our results with the frequency transforms are at the state-of-the-art for the Biomedical benchmark datasets, consistently outperforming traditional methods like P and MP, and rivaling advanced techniques like ATOL, PersLay[1], SV, and BBA. Remarkably, the identity transform often surpasses P, MP, SV and ATOL in biomedical tasks, challenging current embedding approaches in TDA literature. Our method's effectiveness, using simple grid discretization, critiques the trend towards complex vectorizations, suggesting straightforward techniques might be more

[1] Direct comparison with Perslay for `IMDB`, `PROTEINS` is limited due to Perslay's unique preprocessing [21].

efficient. Although our methods demonstrate very good performance overall, it should be acknowledged that for the Social datasets, they are slightly below the current best methods.[2]

3.2 Dynamical systems (`Orbit5K` Dataset)

The `Orbit5K` dataset, used in TDA for classifying DNA microarray flows, features chaotic trajectories in the unit cube $[0,1]^2$ with topologies varying by a parameter $\rho > 0$ (see Figure 1). For each ρ class in the `Orbit5K` dataset, we form point clouds by iterating the following recursive equations for a sequence of 1000 points, beginning from a random initial point (x_0, y_0) in $[0,1]^2$:

$$x_{n+1} = x_n + \rho y_n(1 - y_n) \mod 1,$$
$$y_{n+1} = y_n + \rho x_{n+1}(1 - x_{n+1}) \mod 1.$$

We generated 700 training and 300 testing datasets for each $\rho \in \{2.5, 3.5, 4, 4.1, 4.3\}$ class, conducting a one-versus-one classification and using persistence diagrams for both H_1 and H_0 homologies, following the approach described in [17]. We employ vanilla random forest classifier as in the graph classification tasks and the same transforms with hyperparameter settings (see extended version on arXiv). We additionally adopt a regular square grid of 64×64 and 128×128 for all transforms in this learning task. Our highest accuracy results are in Table 2, with the timings of the various algorithms to vectorize the persistence diagrams of the dataset presented in Table 3.

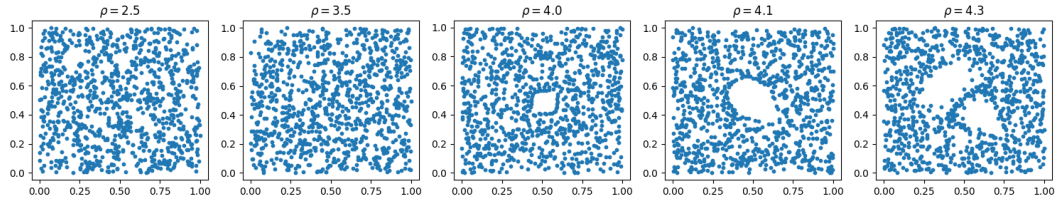


Figure 1 Representative Point Cloud Samples from the `Orbit5K` Dataset.

Aligning with [6] and [14], our comparison includes four kernel methods (PSS-K [20], PWG-K [16], SW-K [7], PF-K [17]), one neural network (Perslay from [6]), Persistence Images (PI from [1]), and a rectangle-based classification (BBA from [14]). Our frequency-based methods surpass most of the kernel methods, PI, BBA and Perslay in performance, though they fall slightly behind the NN. The identity transform outperforms certain kernel methods and is comparable to PI, but generally shows suboptimal performance. Regarding the timings of some of the vectorization methods, from Table 3, it is clear that our method is the most efficient among all the others in the table, providing a significant improvement in the computational time.

PSS-K	PWG-K	SW-K	PF-K	PI	Perslay	BBA	LITE (Our)	LITE-`IdT` (Our)
72.38	76.63	83.6	85.9	82.5	87.7	83.3	84.6	82.0

Table 2 Comparative Classification Accuracy on the `Orbit5K` Dataset.

[2] Our work replicates the biomedical dataset results from [21], but applying their code to social networks yielded a 4% lower performance, in comparison to what we reported directly from thier work in Table 1.

PSS-K	PWG-K	SW-K	PF-K	PI	LITE-FOUR	LITE-GABOR
126.8	14.07	10.08	68.56	73.90	9.15	10.12
LITE-coif1	LITE-coif2	LITE-coif3	LITE-db1	LITE-sb2	LITE-db3	LITE-IdT
9.53	10.25	10.33	9.21	9.33	9.50	8.84

Table 3 Comparative timings in seconds averaged over 5 runs required by various methods to vectorize the Orbit5K Dataset. For LITE and PI, a grid of 1×32 for the diagram of H_0 and 32×32 for the H_1 diagram has been used. For the PSS-K, PF-K, and PWG-K methods, an RBF kernel approximation has been used to speed up computations.

4 Conclusions and further work

Our study introduces a novel vectorization framework for persistence diagrams using functional-based, particularly frequency, transforms. This method is effective and often outperforms existing TDA vectorization techniques in various graph and synthetic dynamical particle classifications. Its simplicity and potential for enhancement, including the use of neural networks for the function transform $f(\mathbf{x}, \cdot)$ to improve performance and applicability, are promising directions for future research.

References

1 Henry Adams, Tegan Emerson, Michael Kirby, Rachel Neville, Chris Peterson, Patrick Shipman, Sofya Chepushtanova, Eric Hanson, Francis Motta, and Lori Ziegelmeier. Persistence images: a stable vector representation of persistent homology. *Journal of Machine Learning Research*, 18(8):1–35, 2017.

2 Keri L. Anderson, Jeffrey S. Anderson, Sourabh Palande, and Bei Wang. Topological data analysis of functional mri connectivity in time and space domains. *Connectomics in neuroImaging : second international workshop, CNI 2018, held in conjunction with MICCAI 2018, Granada, Spain, September 20, 2018 : proceedings. CNI (Workshop)*, 11083:67–77, 2018. URL: https://api.semanticscholar.org/CorpusID:52287547.

3 Peter Bubenik. Statistical topological data analysis using persistence landscapes. *Journal of Machine Learning Research*, 16(3):77–102, 2015.

4 Peter Bubenik and Alexander Wagner. Embeddings of persistence diagrams into hilbert spaces. *Journal of Applied and Computational Topology*, 4(3):353–385, 2020. URL: https://link.springer.com/article/10.1007/s41468-020-00056-w, doi:10.1007/s41468-020-00056-w.

5 Mathieu Carrière and Andrew Blumberg. Multiparameter persistence image for topological machine learning. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 22432–22444. Curran Associates, Inc., 2020. URL: https://proceedings.neurips.cc/paper_files/paper/2020/file/fdff71fcab656abfbefaabecab1a7f6d-Paper.pdf.

6 Mathieu Carriere, Frederic Chazal, Yuichi Ike, Theo Lacombe, Martin Royer, and Yuhei Umeda. Perslay: A neural network layer for persistence diagrams and new graph topological signatures. In Silvia Chiappa and Roberto Calandra, editors, *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pages 2786–2796. PMLR, 26–28 Aug 2020. URL: https://proceedings.mlr.press/v108/carriere20a.html.

7 Mathieu Carrière, Marco Cuturi, and Steve Oudot. Sliced wasserstein kernel for persistence diagrams. In *International Conference on Machine Learning*, volume 70, pages 664–673, Jul 2017.

8 Frédéric Chazal, Vin de Silva, Marc Glisse, and Steve Oudot. The structure and stability of persistence modules. *arXiv preprint arXiv:1207.3674*, Mar 2013.

9 Yuri Dabaghian, Facundo Mémoli, Loren Frank, and Gunnar Carlsson. A topological paradigm for hippocampal spatial map formation using persistent homology. *PLoS Computational Biology*, 8(8):e1002581, 2012. Epub 2012 Aug 9. doi:10.1371/journal.pcbi.1002581.

10 Vincent Divol and Théo Lacombe. Understanding the topology and the geometry of the space of persistence diagrams via optimal partial transport. *arXiv preprint arXiv:1901.03048*, 2019.

11 Irene Donato, Matteo Gori, Marco Pettini, Giovanni Petri, Sarah De Nigris, Roberto Franzosi, and Francesco Vaccarino. Persistent homology analysis of phase transitions. *Physical Review E*, 93(5):052138, 2016. URL: https://link.aps.org/doi/10.1103/PhysRevE.93.052138, doi:10.1103/PhysRevE.93.052138.

12 Herbert Edelsbrunner and John Harer. Persistent homology - a survey. *Contemporary Mathematics*, 453:257–282, 2008.

13 Herbert Edelsbrunner and Dmitriy Morozov. *Persistent Homology*. CRC Press, 3 edition, 2017. To appear.

14 Olympio Hacquard. Statistical learning on measures: an application to persistence diagrams. *arXiv preprint arXiv:2303.08456*, 2023.

15 Michael Etienne Van Huffel and Matteo Palo. Lite, 2024. arXiv:2312.17093.

16 Genki Kusano, Yasuaki Hiraoka, and Kenji Fukumizu. Persistence weighted gaussian kernel for topological data analysis. In *International Conference on Machine Learning*, volume 48, pages 2004–2013, Jun 2016.

17 Tam Le and Makoto Yamada. Persistence fisher kernel: A riemannian manifold kernel for persistence diagrams. In *Advances in Neural Information Processing Systems*, pages 10027–10038, 2018.

18 Yuriy Mileyko, Sayan Mukherjee, and John Harer. Probability measures on the space of persistence diagrams. *Inverse Problems*, 27(12):124007, nov 2011. URL: https://dx.doi.org/10.1088/0266-5611/27/12/124007, doi:10.1088/0266-5611/27/12/124007.

19 Jan Reininghaus, Stefan Huber, Ulrich Bauer, and Roland Kwitt. A stable multi-scale kernel for topological machine learning. *arXiv preprint arXiv:1412.6821*, 2014.

20 Jan Reininghaus, Stefan Huber, Ulrich Bauer, and Roland Kwitt. A stable multi-scale kernel for topological machine learning. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

21 Martin Royer, Frederic Chazal, Clément Levrard, Yuhei Umeda, and Yuichi Ike. Atol: Measure vectorization for automatic topologically-oriented learning. In Arindam Banerjee and Kenji Fukumizu, editors, *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, pages 1000–1008. PMLR, 13–15 Apr 2021. URL: https://proceedings.mlr.press/v130/royer21a.html.

22 Quoc Hoan Tran, Van Tuan Vo, and Yoshihiko Hasegawa. Scale-variant topological information for characterizing complex networks. *arXiv preprint arXiv:1811.03573*, 2018.

23 Oliver Vipond. Multiparameter persistence landscapes. *Journal of Machine Learning Research*, 21(61):1–38, 2020. URL: http://jmlr.org/papers/v21/19-054.html.

24 Kazuko Yamasaki, Avi Gozolchiani, and Shlomo Havlin. Climate Networks Based on Phase Synchronization Analysis Track El-Niño. *Progress of Theoretical Physics Supplement*, 179:178–188, January 2009. doi:10.1143/PTPS.179.178.

25 Afra Zomorodian and Gunnar Carlsson. Computing persistent homology. *Discrete and Computational Geometry*, 33(2):249–274, 2005. URL: https://www.mendeley.com/catalogue/337da92b-4e42-38b2-b353-1bfa55eb1b69/, doi:10.1007/s00454-004-1146-y.

Covering Geometric Sets with Lines*

Sándor P. Fekete[1], Chek-Manh Loi[1], and Michael Perk[1]

1 **Department of Computer Science, TU Braunschweig**
{fekete,loi,perk}@ibr.cs.tu-bs.de

Abstract

We consider the Set Cover Problem for geometric neighborhoods: Given a family $\mathcal{R} = \{R_1, \ldots, R_n\}$ of n connected regions in the plane, find as few lines as possible, such that each region is intersected by some line. Even special cases of this problem are known to be NP-complete, and a spectrum of work has focused on theoretical results such as approximation algorithms; previous practical work has been limited, and included the case in which each R_i is a single point, and the task is to decide whether a small number of lines suffice. We present exact methods for a variety of more general scenarios, including provably optimal solutions for sets with up to 2000 points, and near-optimal solutions for sets of polygons with up to 650 polygonal regions and a total of about 4000 vertices.

1 Introduction

The Set Cover Problem (SCP) is an NP-complete problem of fundamental importance, both in theory and practice. For general instances, the greedy algorithm [7] provides an $\mathcal{O}(\log n)$-approximation algorithm, which is best possible in the worst case, unless P=NP. Many variants of the SCP are geometric, e.g., using line segments, rays, convex polygons or star-shaped polygons for covering point sets, lines or other geometric objects. The geometry of an SCP may be helpful: For covering discrete point sets by lines with a limited number of directions we can get constant-factor approximation algorithms [13]. The underlying geometry can also give rise to additional difficulties: As shown by Abrahamsen et al. [1], the Art Gallery Problem (which amounts to covering a simple polygon by a minimum number of star-shaped subpolygons) is $\exists\mathbb{R}$-complete, making it unlikely that it even belongs to NP.

Our Contributions

We study *practically* useful methods for covering a set $\mathcal{R}$ of n geometric regions by a smallest number of lines, see Figure 1 for examples. In particular, we provide the following results.

- Different methods for efficiently computing a discrete set of candidate lines that limit the size of the ensuing set cover instance.
- Exact approaches for near-optimal solutions for covering points or polygons with lines.
- An experimental evaluation for a wide spectrum of benchmark instances.

Related Work

There is a large body of work on geometric Set Cover and Hitting Set problems, so we only point to a very limited selection of most closely related work; for a more extensive overview, see the relatively recent paper by Fekete et al. [13], who considered covering a finite set of points in the plane by a minimum number of lines with a limited number of different slopes.

* This work was supported by DFG project "Computational Geometry: Solving Hard Optimization Problems" (CG:SHOP), FE407/21-1.

40th European Workshop on Computational Geometry, Ioannina, Greece, March 13–15, 2024.
This is an extended abstract of a presentation given at EuroCG'24. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

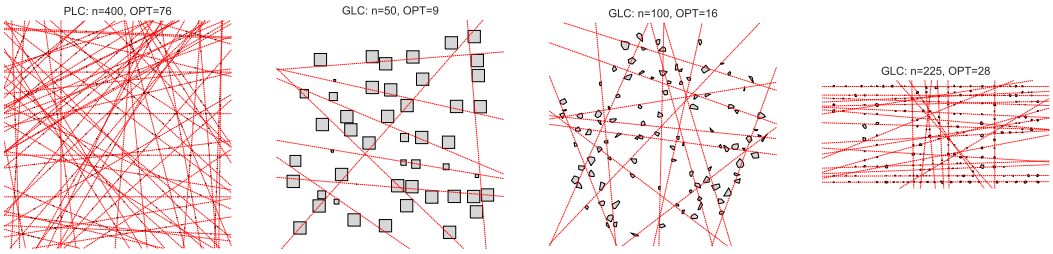


Figure 1 Small example instances that were solved optimally. From left to right: A PLC instance with 400 points and an optimal solution with 76 lines. Three GLC instances (randomly generated squares on a grid, randomly generated polygons in a circle, and the TSPLIB tsp225 [5] instance) and their optimal solutions.

The Point Line Cover (PLC) problem [22] asks for a smallest set of lines to cover a given set of points. It was shown to be NP-hard [25], APX-hard [6] and Max-SNP Hard [23]. Grantson and Levcopoulos [18] developed an $\mathcal{O}(\log OPT)$ algorithm for the PLC. Hassin and Megiddo [19] studied hitting geometric objects with the fewest lines having a small number of distinct slopes. Gaur and Bhattacharya [15] considered covering points with axis-parallel lines in d dimensions. Many other problems related to finding a small set of lines that hit a given set of objects have also been studied; see, e.g., [8, 9, 12, 16, 17, 21, 24].

When considering the coverage of geometric regions (i.e., neighborhoods) instead of discrete points, Aronov et al. [3] provide an $\mathcal{O}(\log \log OPT)$-approximation for hitting set for axis-aligned rectangles and axis-aligned boxes in 3D, based on ϵ-nets. Hitting a set of unit disks has been considered for finding a minimum number of relays for connecting a given set of relays [10]. While the simple greedy approximation algorithm is efficient and worst-case optimal, a logarithmic approximation factor is not good enough in practice. Estivill-Castro et al. [11] evaluated implementations for the PLC, but only considered cases in which the optimal solution is known to be small, i.e., $OPT \leq 7$.

2 Preliminaries

Given a point set $\mathcal{P} \subset \mathbb{R}^2$ of size n, the Point Line Cover Problem (PLC) asks for a smallest set of lines that covers all points in $\mathcal{P}$. We denote the set of all possible lines as $\mathcal{L}$. In the General Line Cover Problem (GLC), we are given a set of disjunct regions $\mathcal{R}$ and ask for a smallest set of lines that intersect all regions in at least one point. It is easy to see that a line intersects a region iff it intersects its convex hull, so we can restrict ourselves to *convex* regions; moreover, we focus on *compact, disjoint* regions with $\mathcal{O}(1)$ complexity.

3 Computing a Candidate Set $\mathcal{L}$

Computing optimal solutions can be subdivided into two steps, as follows.

1. Compute a discrete set $\mathcal{L}$ of candidate lines that contains an optimal line cover.
2. Solve the resulting Set Cover instance.

In theory, the first step is comparatively easy, while the second is NP-hard. However, the focus on practical computation implies that merely polynomial-time computation of a candidate set is insufficient, especially when aiming for small input for the ensuing SCP. A variety of possible approaches for solving those SCP instances is considered in Section 4.

3.1 Point Line Cover

For the PLC, Gajentaan and Overmars [14] showed that determining whether a set of n points in the plane has three collinear points is 3SUM-hard. Even though subquadratic algorithms for 3SUM exist [2, 4], the improvements over $\mathcal{O}(n^2)$ are often times marginal. We propose an $\mathcal{O}(n^2)$ algorithm for determining $\mathcal{L}$ for a given set of points $\mathcal{P}$. The algorithm iterates over all combinations of two points $p_i, p_j \in \mathcal{P}$ and calculates a tuple $t_{ij} = (m, b)$ with m being the gradient of the line ℓ between p_i, p_j and b being the y value of ℓ at $x = 0$. While iterating, we calculate a hash $h(t_{ij})$ to identify every line in $\mathcal{L}$. This allows us to add all collinear points to a set within a single pass over all point pairs.

3.2 General Line Cover

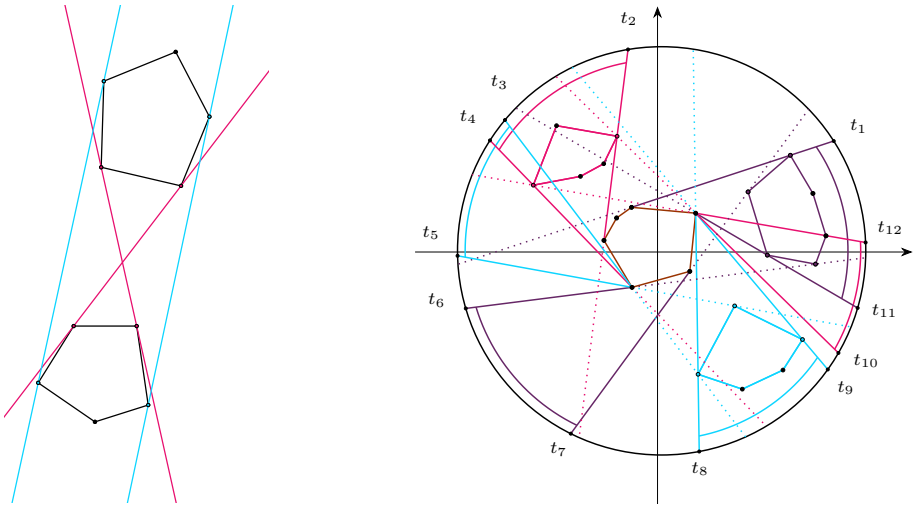


Figure 2 (Left) Four tangents from Lemma 3.1. The red tangents are interior, the blue ones exterior. In this example each polygon has four tangential points. (Right) Rotating tangent algorithm to compute $\mathcal{L}$.

Tangent Lines. For two disjoint compact, convex regions in the GLC, we can reduce the set of candidate lines to two exterior tangents using Lemma 3.1. Even neglecting the required time for computing a pair of tangents for regions with a significant number of vertices, the runtime for computing $\mathcal{L}$ remains $\mathcal{O}(n^3)$. This makes this precomputation prohibitively expensive, even before computing the SCP solution. See Section 5 for an experimental evaluation.

▶ **Lemma 3.1.** *Let r_i and r_j be two disjoint compact, convex regions in the plane.*
1. *There are four extremal lines that intersect both r_i and r_j.*
2. *It suffices to consider exterior tangents as potential lines.*

Proof. The first claim is relatively straightforward by considering degrees of freedom and events during continuous modification of a stabbing line.

For the second claim, consider a set of intersected regions $r_1, \ldots, r_s$ (in this order) and an interior tangent between r_{i_1} and r_{i_2} with $i_1 < i_2$; w.l.o.g., let t have positive slope. For any tangential position, we distinguish between regions on the left and right; w.l.o.g., let r_{i_1} be to the left and r_{i_2} to the right of t.

Now rotate t continuously in clockwise direction while maintaining tangential position relative to r_{i_1}. Then all intersections remain intact, until a tangential event with a region $r_{i_3} \neq r_{i_1}$ happens.

We distinguish:

1) If $i_3 > i_1$, then r_{i_3} is to the left, and we have an exterior tangent for r_{i_1} and r_{i_3} that stabs $r_1, \ldots, r_s$.

2) If $i_3 < i_1$, then r_{i_3} is to the right. Then we continue analogously, rotating t counterclockwise around r_{i_3} until we get an event at a region r_{i_4}, with further case distinction.

2.1) If $i_4 > i_3$, then r_{i_3} is to the right, and we have an exterior tangent for r_{i_3} and r_{i_4} that stabs $r_1, \ldots, r_s$.

2.2) If $i_4 < i_3$, then r_{i_4} is to the right. This brings us back to the same situation we had with r_{i_1} and r_{i_2}, but with $i_4 < i_1$, so we can continue in this manner, leading to a sequence $i_1, i_2, i_3, i_4, \ldots$ of event regions. If the current tangential index i_j ever increases, we have identified an exterior tangent; however, the available index set is finite, so a decrease below $i_j = 1$ guarantees an exterior tangent.

◀

Rotating Tangents. A more efficient method for computing a candidate set of tangent lines $\mathcal{L}$ is illustrated in Figure 2 (right). This *Rotating Tangent* (RT) algorithm considers a tangent line that rotates continuously around a compact convex region R_i, and exploits the fact that it intersects another disjoint compact in a contiguous circular arc of directions. This induces a circular arc graph for each of the regions; any maximal clique in this graph corresponds to a maximal subset of intersected regions [20].

Therefore, we can compute all maximal subsets containing a given region in linear time after presorting the events. Computing the tangents and initializing the sweep lines for all polygons takes $\mathcal{O}(n^2)$; presorting and computing all sets during the sweep takes $\mathcal{O}(n^2 \log n)$.

Eliminating Subsets. With either method, the resulting $\mathcal{L}$ may produce a family that contains proper subsets. In principle, these could be eliminated post-construction in worst case $\mathcal{O}(n^5)$; more efficient practical methods (e.g., using k-d trees or other decompositions) could be employed. This turned out to have limited benefit, as some of the methods for the SCP already deal with subsets in a relatively effective manner during their solution process.

4 Solving Set Cover Instances

Now we consider different approaches for finding a subset of $\mathcal{L}$ that covers all regions. Throughout this section, we adopt a standardized notation for the underlying Set Cover problems, where the objective is to cover elements in E by selecting sets from $\mathcal{S}$.

Integer Programming

An Integer Programming formulation is shown in Figure 3; in the worst case, this can result in n^2 sets with 2 elements. Alternatively, we remove all 2-sets from $\mathcal{S}$, i.e., $\mathcal{S}' = \{S_i \mid S_i \in \mathcal{S}, |S_i| > 2\}$ and introduce new binary variables y_j for $j \in E$, see the right side of Figure 3. Constraints are satisfied by choosing a set or its newly introduced variable for covering; the latter option incurs a penalty term of $\frac{1}{2}$ for covering remaining point pairs by lines.

$$\begin{aligned} \min \quad & \sum_{S_i \in \mathcal{S}} x_i & \\ \text{s.t.} \quad & \sum_{\substack{S_i \in \mathcal{S} \\ j \in S_i}} x_i \geq 1 & \forall j \in E \\ & x_i \in \{0,1\} & \forall S_i \in \mathcal{S} \end{aligned}$$

$$\begin{aligned} \min \quad & \sum_{S_i \in \mathcal{S}'} x_i + \frac{1}{2} \sum_{j \in E} y_j & \\ \text{s.t.} \quad & \sum_{\substack{S_i \in \mathcal{S}' \\ j \in S_i}} x_i + y_j \geq 1 & \forall j \in E \\ & x_i \in \{0,1\} & \forall S_i \in \mathcal{S}' \\ & y_j \in \{0,1\} & \forall j \in E \end{aligned}$$

Figure 3 Two possible Integer Programming formulations for PLC and GLC. (Left) Basic set cover IP. (Right) Formulation without sets of size 2.

$$\begin{aligned} \min \quad & \sum_{S_i \in \mathcal{S}} x_i & \\ \text{s.t.} \quad & \bigvee_{\substack{S_i \in \mathcal{S} \\ j \in S_i}} x_i & \forall j \in E \\ & x_i \in \mathbb{B} & \forall S_i \in \mathcal{S} \end{aligned}$$

$$\begin{aligned} \min \quad & 2 \cdot \sum_{S_i \in \mathcal{S}'} x_i + \sum_{j \in E} y_j & \\ \text{s.t.} \quad & \bigvee_{\substack{S_i \in \mathcal{S}' \\ j \in S_i}} x_i \vee y_j & \forall j \in E \\ & x_i \in \mathbb{B} & \forall S_i \in \mathcal{S}' \\ & y_j \in \mathbb{B} & \forall j \in E \end{aligned}$$

Figure 4 Two possible Constraint Programming formulations for PLC and GLC. (Left) Constraint programming formulation. (Right) Formulation without sets of size 2.

Constraint Programming Formulation

The IP can be directly converted into a Constraint Programming formulation, see Figure 4. For a formulation without sets of size 2, we multiply the objective function by a factor of 2 to ensure that the values remain integer.

Large Neighborhood Search

We also tested a Large Neighborhood Search (LNS): Iteratively remove a subset from the current solution until a certain number of elements are uncovered, then solve the restricted set cover problem with an exact solver for the improved IP formulation. In this process, the neighborhood size is adapted to ensure optimal solvability.

5 Experimental Results

Our implementation uses exact number types and predicates and was tested on a workstation with an AMD Ryzen 9 7900 (12×3.7 GHz) CPU and 98GB of RAM[1]. IP solvers are denoted by *IP* (Figure 3 left) and *IP-2SET* (Figure 3 right), CP-SAT solvers as *CP-SAT* (Figure 4 left) and *CP-SAT-2SET* (Figure 4 right). We used *IP-2SET* as the exact solver within the LNS algorithm, denoted by *LNS (IP-2SET)*. To account for the lack of

[1] Source code and data: `https://gitlab.ibr.cs.tu-bs.de/alg/geometric-covering`

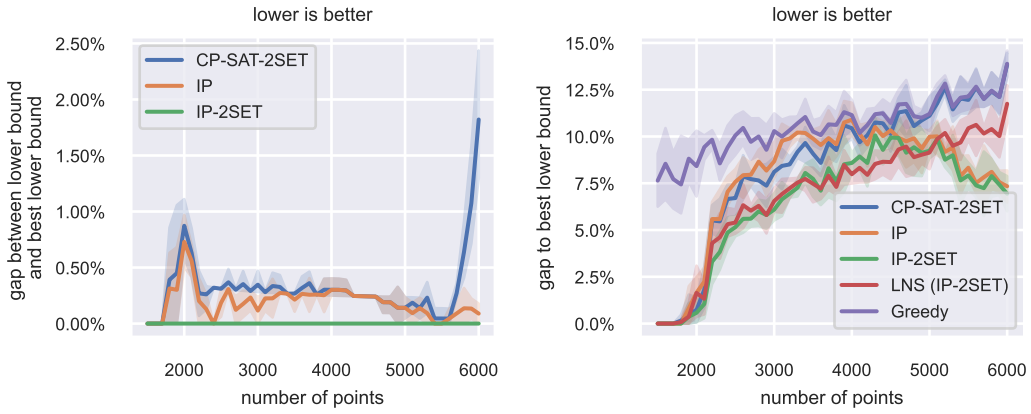


Figure 5 Solvers executed on the *plc_points* instance set with a 600s time limit. (Left) Quality of the lower bounds produced by the different solvers in comparison with the best lower bound. (Right) Upper bound quality (gaps to best lower bound) of all implemented solvers.

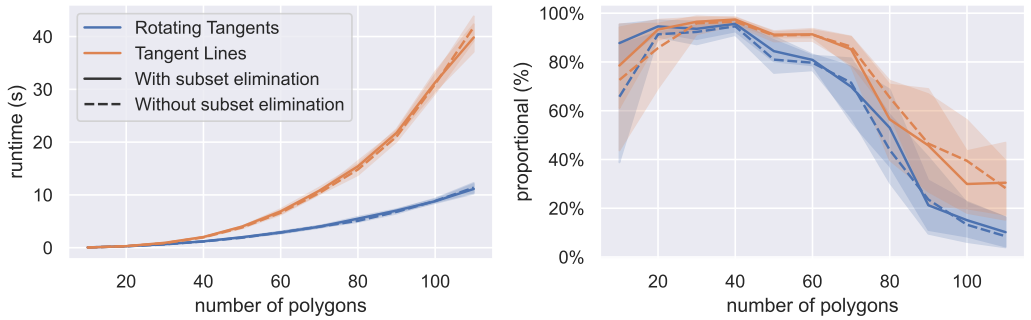


Figure 6 Solvers and line construction algorithms executed on the *glc_polygons_sm* instance set without any time limit (until OPT was found). (Left) Runtime of the RT and TL approach for computing all lines for the set cover problem. (Right) Proportion of time in the full solving process spent during line construction.

publicly available benchmarks, we generated instance sets *plc_points*, *glc_polygons_sm*, *glc_polygons*, *glc_squares* within a fixed-size canvas as follows: For the PLC, we randomly computed lines and chose points on these lines. For the GLC, we placed random point clouds (at locations randomly chosen or according to point locations in TSPLIB [5] instances) and used their convex hull while ensuring no intersections occurred. This yielded several hundred instances; see Figure 1 for examples.

5.1 Point Line Cover

Figure 5 compares the lower bounds from all approaches to the best lower bound found by any method. As the LNS solver can only produce lower bounds for small neighborhoods and *CP-SAT* exceeded the memory limits, they were excluded from this evaluation. Figure 5 shows that the *IP-2SET* can reliably find the best lower bounds of all implemented approaches, even though set cover seems to be suited for SAT-based solvers. The right side of Figure 5 shows that the initial greedy solution already provides reasonably good solutions for all tested instances. *CP-SAT-2SET* performed worse than the IP-based approaches.

Depending on the instance, either *IP-2SET* or the LNS-based approach yielded the best upper bounds, with LNS performing poorly on instances with more than 5000 points.

5.2 General Line Cover

Covering Set Computation

See Figure 6 for a comparison between the two methods for subset computation: RT is considerably faster than the TL approach, despite producing a slightly larger set of candidate lines. Moreover, subset elimination can drastically reduce the number of lines for the SCP solver. However, this has almost no effect on the ensuing SCP computation times.

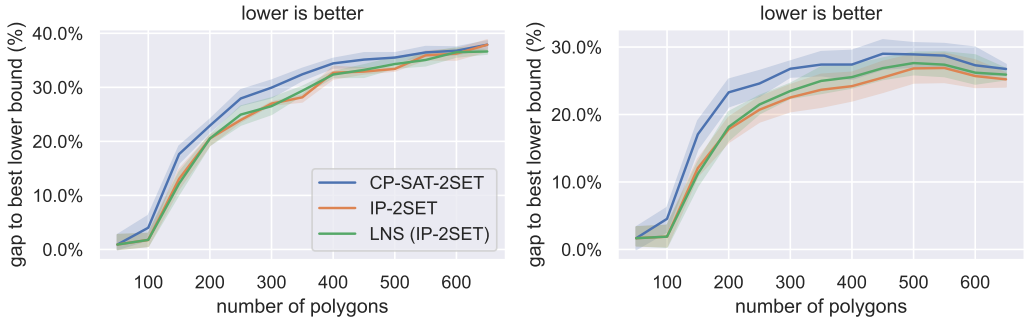


Figure 7 Solvers executed with a 600s time limit. (Left) Upper bound quality (gaps to best lower bound) for convex polygons (*glc_polygons*). (Right) Upper bound quality (gaps to best lower bound) for square instances (*glc_squares*).

Upper bounds

Figure 7 compares the performance of the best solvers from the previous section on two benchmark sets (i) convex polygons of various sizes and (ii) squares, see Figure 1 for examples and solutions. In Figure 7, we can see that *IP-2SET* again beats the other approaches, while *LNS* produces similarly good and sometimes better upper bounds than the integer program. Overall, the gap between the upper and lower bounds is slightly smaller for the unit square instances, and the performance of all approaches is worse than for the PLC.

6 Conclusion

We have shown that geometric covering problems can be practically solved to near optimality for a wide range of instances. A spectrum of further refinements remains to be studied. This includes specialized methods for congruent regions (such as squares or disks, which arise from error bounds for imprecise data), but also higher-dimensional scenarios. As Estivill-Castro et al. [11] showed, there are FPT-type practical approaches for finding PLC solutions with only few lines; it is conceivable that similar ideas can be extended to GLC instances.

References

1 M. Abrahamsen, A. Adamaszek, and T. Miltzow. The Art Gallery Problem is $\exists\mathbb{R}$-complete. *J. ACM*, 69(1):4:1–4:70, 2022.

2 B. Aronov, M. de Berg, J. Cardinal, E. Ezra, J. Iacono, and M. Sharir. Subquadratic algorithms for some 3Sum-hard geometric problems in the algebraic decision-tree model. *Comput. Geom.*, 109:101945, 2023.

3 B. Aronov, E. Ezra, and M. Sharir. Small-size ϵ-nets for axis-parallel rectangles and boxes. *SIAM J. Comput.*, 39(7):3248–3282, 2010.

4 I. Baran, E. D. Demaine, and M. Puatracscu. Subquadratic algorithms for 3SUM. *Algorithmica*, 50(4):584–596, 2008.

5 B. Bixby and G. Reinelt. TSPLIB, a Traveling Salesman Problem library. *ORSA Journal on Computing*, 3:376–384, 1991.

6 B. Brodén, M. Hammar, and B. J. Nilsson. Guarding lines and 2-link polygons is APX-hard. In *Canadian Conf. Comput. Geometry (CCCG)*, pages 45–48, 2001.

7 V. Chvátal. A greedy heuristic for the set-covering problem. *Math. Oper. Res.*, 4(3):233–235, 1979.

8 M. Dom, M. R. Fellows, and F. A. Rosamond. Parameterized complexity of stabbing rectangles and squares in the plane. In *Intern. Workshop on Algorithms and Computation (WALCOM)*, pages 298–309. 2009.

9 H. Edelsbrunner and M. Sharir. The maximum number of ways to stabbing convex non-intersecting sets in the plane is $2n - 2$. *Discrete & Computational Geometry*, 5(1):35–42, 1990.

10 A. Efrat, S. P. Fekete, J. S. B. Mitchell, V. Polishchuk, and J. Suomela. Improved approximation algorithms for relay placement. *ACM Trans. Algorithms*, 12(2):20:1–20:28, 2016.

11 V. Estivill-Castro, A. Heednacram, and F. Suraweera. Reduction rules deliver efficient FPT-algorithms for covering points with lines. *ACM J. Exp. Algorithmics*, 14, 2009.

12 G. Even, R. Levi, D. Rawitz, B. Schieber, S. M. Shahar, and M. Sviridenko. Algorithms for capacitated rectangle stabbing and lot sizing with joint set-up costs. *ACM Trans. Alg.*, 4(3):34:1–34:17, 2008.

13 S. P. Fekete, K. Huang, J. S. B. Mitchell, O. Parekh, and C. A. Phillips. Geometric hitting set for segments of few orientations. *Theory Comput. Syst.*, 62(2):268–303, 2018.

14 A. Gajentaan and M. H. Overmars. On a class of $O(n^2)$ problems in computational geometry. *Computational Geometry*, 5(3):165–185, 1995.

15 D. R. Gaur and B. Bhattacharya. Covering points by axis parallel lines. In *European Workshop on Computational Geometry (EuroCG)*, pages 42–45, 2007.

16 D. R. Gaur, T. Ibaraki, and R. Krishnamurti. Constant ratio approximation algorithms for the rectangle stabbing problem and the rectilinear partitioning problem. In *European Symposium on Algorithms (ESA)*, pages 211–219. 2000.

17 P. Giannopoulos, C. Knauer, G. Rote, and D. Werner. Fixed-parameter tractability and lower bounds for stabbing problems. *Comp. Geometry*, 46:839–860, 2013.

18 M. Grantson and C. Levcopoulos. Covering a set of points with a minimum number of lines. In T. Calamoneri, I. Finocchi, and G. F. Italiano, editors, *International Conference on Algorithms and Complexity (CIAC)*, pages 6–17, 2006.

19 R. Hassin and N. Megiddo. Approximation algorithms for hitting objects with straight lines. *Discret. Appl. Math.*, 30(1):29–42, 1991.

20 W.-L. Hsu and K.-H. Tsai. Linear time algorithms on circular-arc graphs. *Information Processing Letters*, 40(3):123–129, 1991.

21 S. Kovaleva and F. C. Spieksma. Approximation algorithms for rectangle stabbing and interval stabbing problems. *SIAM J. Discrete Math.*, 20(3):748–768, 2006.

22 S. Kratsch, G. Philip, and S. Ray. Point line cover: The easy kernel is essentially tight. *ACM Trans. Algorithms*, 12(3):40:1–40:16, 2016.

23 V. A. Kumar, S. Arya, and H. Ramesh. Hardness of set cover with intersection 1. In *International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 624–635, 2000.

24 S. Langerman and P. Morin. Covering things with things. *Discrete & Computational Geometry*, 33(4):717–729, 2005.

25 N. Megiddo and A. Tamir. On the complexity of locating linear facilities in the plane. *Operations Research Letters*, 1(5):194–197, 1982.

Covering line segments with drones: the minmax criterion*

José-Miguel Díaz-Bañez, José Manuel Higes, Alina Kasiuk, and Inmaculada Ventura

Departamento de Matemática Aplicada II, Universidad de Sevilla
{dbanez, jhiges, akasiuk, iventura}@us.es

Abstract

We are given a set of line segments (e.g., tubes in a solar plant) to be inspected by drones. The limited capacity of the batteries imposes periodical visits (tours) to a fixed base station. The objective is to assign a set of tours for each drone so that the segments are covered as quickly as possible, i.e. to minimize the maximum time spent by the drones. In this paper, we prove that this problem is NP-hard even when the segments are positioned on a line and the scenario involves two drones. An approximation algorithm is proposed with constant factor ranging form 1 to 2.

1 Introduction

As technology advances, unmanned aerial vehicles (UAVs), commonly referred to as drones, are assuming an ever-expanding role in the inspection of industrial structures. For example, the manual inspection of high-voltage power transmission lines or solar plants are both time-consuming and expensive [6]. Hence, the use of drones equipped with cameras enables efficient fault detection.

In the case of Concentrated Solar Power plants (CSP), which represent a growing technology for electricity production through renewable energies, the plant comprises an array of receiver tubes subjected to high thermal stress. Therefore, the identification of broken glass envelopes is crucial to maintain the proper functioning of the CSP plant [5]. In this context, promptly detecting a fault enables the company to take immediate action in the repair process. In this paper, we tackle the problem of minimizing the time required for a team of drones to effectively traverse a set of tubes, represented as line segments. This problem falls within the domain of arc routing problems (ARPs), which involve determining a set of tours with the minimum total cost while traversing a set of links (arcs or edges) known as required links in a graph [4]. However, in contrast to vehicles in classical ARPs, a drone has the flexibility to enter a line through any of its points, traverse a portion of that line, exit through another of its points, and subsequently travel directly to any point on another line, and so forth. Hence, employing drones for service in ARPs introduces substantial modifications to the conventional methods of modeling and solving these problems [2].

In a recent paper [1], the authors study a problem focused on minimizing the total time required for one drone to cover a set of line segments positioned along a given line. Here, the total time is defined as the sum of the lengths of the necessary tours. The drone has limited battery endurance, maintains a constant flying speed, and returns to a base station when its battery is running low. They show that this one-dimensional variant can be solved in polynomial time. They also address the problem of minimizing the number of tours needed

* This work is partially supported by grants PID2020-114154RB-I00 and TED2021-129182B-I00 funded by MCIN/AEI/10.13039/501100011033 and the European Union NextGenerationEU/PRTR.

40th European Workshop on Computational Geometry, Ioannina, Greece, March 13–15, 2024.
This is an extended abstract of a presentation given at EuroCG'24. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

to cover all the segments. In this paper, we consider the same one-dimensional covering problem, with a focus on minimizing the maximum distance (time) traveled by the drones of the team—essentially, reducing the time required for the team to cover the segments. This objective is meaningful when the company aims to expedite the task and promptly repair the broken tubes.

1.1 Problem Statement

Let $\alpha = \{a_1, a_2, \cdots, a_n\}$ be a set of disjoint line segments on a line. For $i = 1, 2, \cdots, n$, $a_i = [x_i, y_i]$, with $x_1 < y_1 < x_2 < y_2 < \cdots < x_n < y_n$. We are given a team of k identical drones that must traverse all the segments. The drones are constrained by finite battery endurance, maintain uniform velocity during flight, and execute takeoff and battery recharging protocols at a fixed point B, the designated base station, situated exterior to the line. We assume that the recharging time is negligible as the batteries are replaced instantly. Let $L > 0$ denote the maximum distance achievable by a drone when initiating and concluding its trajectory (tour) at the fixed point B. For a tour labeled as t, we refer to its length as $\ell(t)$. The length of a collection of tours, denoted as $T = \{t_1, \cdots, t_m\}$, is represented as $\ell(T)$. It is calculated as the sum of the lengths of individual tours, that is, $\sum_{i=1}^{m} \ell(t_i)$. Our objective is to determine a set of tours for each drone in such a way that we minimize the maximum length traveled by any drone (the makespan) while ensuring that all segments are covered. Formally, the *Minmax problem* for k drones can be stated as follows:

▶ **Problem 1.1.** *Minmax-k: compute a set of tours for each drone, $T_1, T_2, \cdots, T_k$, such that:*

$$\alpha \subset T_1 \cup T_2 \cup \cdots \cup T_k \text{ and,} \tag{1}$$

$$\max_{j=1,\cdots,k} \ell(T_j) \text{ is minimized.} \tag{2}$$

See Figure 1 for an example of a set of tours covering line segments with $k = 2$ drones.

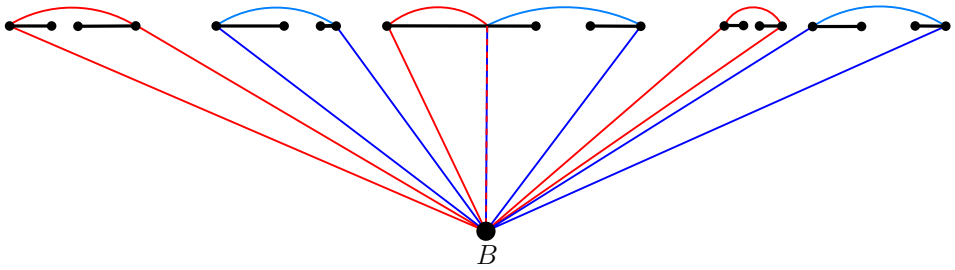


Figure 1 Covering tours for two drones. Blue and red tours correspond to different drones.

The problem of minimizing the maximal number of tours (instead of total length) performed by the k drones can be easily addressed using the approach of [1]. Indeed, let m be the minimum number of tours required by just a drone to cover all segments in the set S. Thus, $\lceil \frac{m}{k} \rceil$ is the solution, and the problem can be solved in $O(n + m)$ time with a greedy approach proposed in [1]. In the same paper, it has been demonstrated that the *Minsum problem*, that is, minimizing the total length of a set of tours performed by a single drone can also be solved in polynomial time. This can be easily extended to solve the Minsum problem for k drones by allowing one drone to perform all the tours while the others remain inactive. In this paper, we prove that transitioning from the Minsum problem to the Minmax criterion for two drones results in an NP-hard problem. Then, an approximation algorithm with factor ranging from 1 to 2 is proposed.

2 NP-Hardness

It is known that the *two way-balanced partition problem* is NP-complete [3]: Given a finite set of positive integers $S = \{s_i\}_{i=1}^{n}$, determine if there is a subset $A \subset S$ of cardinality $\lfloor n/2 \rfloor$, with $\sum_{s_i \in A} s_i = \sum_{s_j \in S \setminus A} s_j$.

We can prove that the following variant is NP-hard:

▶ **Problem 2.1.** *Minmax partition problem: Given a finite set of positive integers $S = \{s_i\}_{i=1}^{2n}$, determine a subset $A \subset S$ of cardinality n, with $M_2 = \sum_{s_i \in A} s_i \geq M_1 = \sum_{s_j \in S \setminus A} s_j$, such that M_2 is minimum for all possible subsets of S of cardinality n.*

In this section, we outline the key ideas for proving the NP-hardness of the Minmax-2 problem through a reduction from Problem 2.1. The following result will be one crucial tool in our construction:

▶ **Proposition 2.2.** *For a set of positive numbers $S = \{s_i\}_{i=1}^{2n}$ and two positive constants $K, C > 0$, define the set $S' = \{s'_i\}_{i=1}^{2n}$ with $s'_i = Ks_i + C$. Then a subset $A \subset S$ is a solution to Problem 2.1 for S if and only if the subset A', defined by $s'_i \in A'$ if $s_i \in A$, is a solution to Problem 2.1 for the set S'.*

2.1 Construction

The following construction can be done in polynomial time. Given a line, an exterior point B, and a positive constant $L > 0$, let O be the projection point of the base B on the line located as in Figure 2. Let $\{s_i\}_{i=1}^{2n}$ be a set of positive integers and assume w.l.o.g. $R = \max_i\{s_i\} = s_{2n}$. Take ϵ a small number less than $1/3$, for example $\epsilon = 10^{-8}$.

- *Step 1:* For $i = 0, \cdots, 2n$, determine a sequence of right hand points y_i, with $c_i = d(y_i, B)$, that satisfies:

$$c_0 > \max\left(\frac{L}{2} - \frac{\epsilon L}{4n}, L - y_0 - \frac{L}{3}, \frac{\sqrt{5}L}{6}\right),$$

$$y_{i+1} - y_i + c_i + c_{i+1} = L,$$

$$c_{i+1}^2 = y_{i+1}^2 + \left(\frac{L}{3}\right)^2.$$

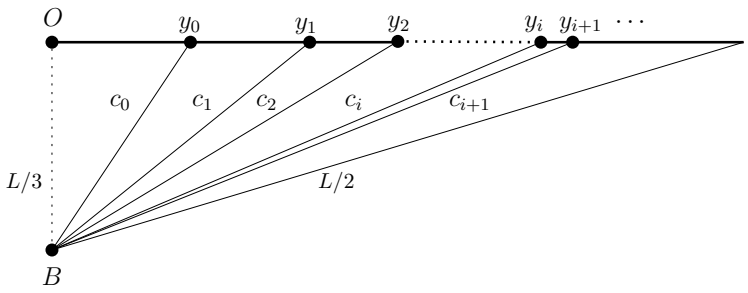


Figure 2 Diagram for Step 1. Right hand points y_i.

- *Step 2:* Determine the number sequence:

$$\{s'_i = Ks_i + 2c_{2n}\}_{i=1}^{2n} \text{ with } K = \frac{L - 2c_{2n}}{R}.$$

- *Step 3:* For $i = 1, \cdots, 2n$, determine the sequence of right hand points x_i, with $b_i = d(x_i, B)$ that satisfies:

$$c_i + (y_i - x_i) + b_i = s'_i,$$

$$b_i^2 = x_i^2 + \left(\frac{L}{3}\right)^2.$$

- *Step 4:* For $i = 1, \cdots, 2n$, draw the subsequence of right hand segments $[x_i, y_i]$ for i even, and left hand segments $[-y_i, -x_i]$ for i odd (Figure 3).

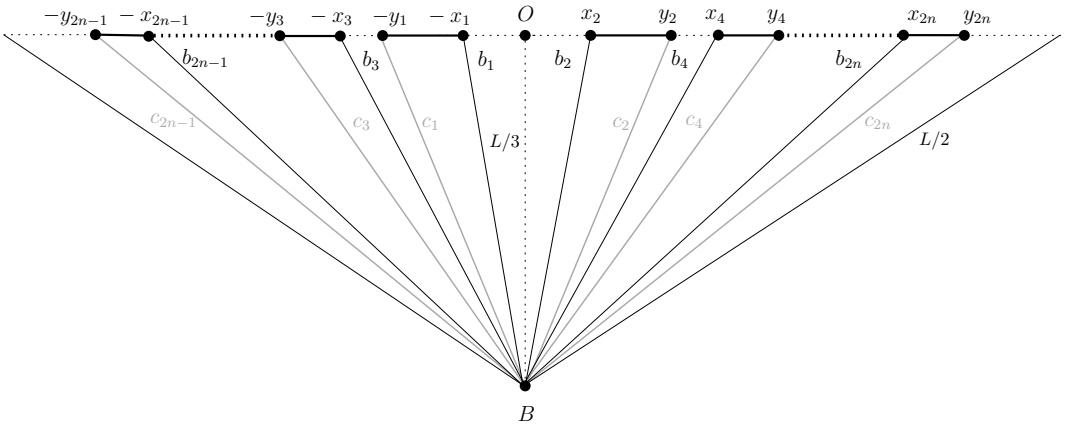


Figure 3 Diagram for step 4. Final construction.

▶ **Theorem 2.3.** *The Minmax-2 problem is NP-hard.*

Proof. We will provide a brief outline of the proof. The problem Minmax-2 is clearly in NP. Given a set of positive integers $S = \{s_i\}_{i=1}^{2n}$, perform the above construction in polynomial time. Set $K = \frac{L-2c_{2n}}{R}$ and $C = 2c_{2n}$. We can prove that the following facts are true for our construction:

- *Statement 1:* The solution of the Minsum problem for one drone in our construction is determined by a the set of tours $T = \{t_i\}_{i=1}^{2n}$ with each tour t_i covering exactly the i-th segment. Notice that the length of each tour t_i is $s'_i = Ks_i + C$ by Step 3.
- *Statement 2:* The tours of any solution $\{T_1, T_2\}$ for the Minmax-2 problem in the construction correspond to tours t_i derived from a solution to the Minsum problem for a single drone.

Now, given a solution A for Problem 2.1 for S, K and C, define the set S' and A' as in Proposition 2.2 and take a solution of the Minsum problem for one drone (by applying the polynomial time algorithm of [1] to our construction). By Statement 1, each tour t_i of this solution covers exactly the i-th segment and has length s'_i. Thus, assigning each tour t_i to T_2 if and only if $s'_i \in A'$, we will have a solution of the Minmax-2 problem by Statement 2, the minimality of A' and the fact that s'_i is the length of the tour t_i.

In the other direction, given a solution for the Minmax-2 problem in our construction, $\{T_1, T_2\}$, assign $s'_i \in A'$ if and only if $t_i \in T_2$ (assume w.o.l.g that $\ell(T_2) \geq \ell(T_1)$). By Statement 2, the length of each t_i is s'_i; therefore we obtain a solution of Problem 2.1 for S' and, by Proposition 2.2, we can consequently derive a solution for S. ◀

3 An approximation algorithm

Greedy tour distribution for two drones, G2D-algorithm: In [1], an algorithm based on dynamic programming to compute a set of tours $T = \{t_1, \cdots, t_m\}$ solving the minsum problem for one drone was proposed. Getting T as the initial step, the G2D-algorithm just distribute the tours of T into two sets, T_1 and T_2, so that, in each step, the difference between the sum of the lengths of the tours in each set is $|\ell(T_2) - \ell(T_1)| = aL$ with $0 \leq a \leq 1$. To do it, given $T = \{t_1, \cdots, t_m\}$, add for $i = 1$ the tour t_1 to T_1 and in each subsequent step $i > 1$ add the tour $t_i \in T$ to the set T_1 or T_2 with minimum total length.

▶ **Observation 3.1.** *Assume w.l.o.g. that $\ell(T_2) \geq \ell(T_1)$ in G2D-algorithm; then, for some $a \in [0, 1]$, $\ell(T_2) = \ell(T_1) + aL$.*

▶ **Theorem 3.2.** *Let $\{T_1^*, T_2^*\}$ be any solution of the Minmax-2 problem and let $\{T_1, T_2\}$ be the final distribution of the G2D-algorithm. Assume $\ell(T_1) \leq \ell(T_2)$ and $\ell(T_1^*) \leq \ell(T_2^*)$. Then:*

a) $\ell(T_1) + \dfrac{aL}{2} \leq \ell(T_2^*) \leq \ell(T_1) + aL$

b) If $a = 0$, then $\{T_1, T_2\}$ is optimal for the Minmax-2 problem.

c) If $a \in (0, 1]$, then $\ell(T_2) \leq \Delta \cdot \ell(T_2^*)$, with $\Delta = \dfrac{\Gamma + 2}{\Gamma + 1}$ and $\Gamma = \dfrac{2\ell(T_1)}{\ell(T_2) - \ell(T_1)}$.

▶ **Observation 3.3.** *By the results of [1], G2D-algorithm computes T_1 and T_2 in $O(n^2) + O(nm)$ time, where n is the number of segments and m the total number of tours.*

▶ **Observation 3.4.** *As $\ell(T_1)$ increases (for example, if $\ell(T)$ is large) then Δ tends to 1.*

▶ **Observation 3.5.** *G2D-algorithm can be generalized for $k > 2$ drones with the same time complexity; we just have to properly distribute the m tours of T into k sets $T_1, \cdots, T_k$. The approximation factor is $\Delta = \dfrac{\Gamma + k}{\Gamma + 1}$ with $\Gamma = \dfrac{k \cdot \ell(T_1)}{\ell(T_k) - \ell(T_1)}$, where T_1 is the set with minimum total length and T_k the set with maximum total length.*

4 Conclusions and future research

We have proved that covering a set of line segments on a line with drones using the minmax criterion is NP-hard, even when considering only two drones. We found a reduction from a variant of the finite partition problem. It is highly likely that we can extend this result to the Minmax-k problem for $k > 2$ drones by drawing connections to the multiway number partition problem and employing similar ideas as presented in Section 2.

Another avenue for future research is the enhancement of the G2D-algorithm. After getting $\{T_1, T_2\}$ (assume $\ell(T_2) \geq \ell(T_1)$), two tools can be employed for further improvement. With the *Cutting technique*, the idea is to break one tour of T_2 into two sub-tours and reducing the cost by allocating one sub-tour to the other drone as illustrated in Figure 4 (a).

On the other hand, the *Enlarging technique* can be applied when a tour $t_i \in T_1$ with a length strictly less than L is found to be in contact with a tour $t_j \in T_2$. Then we can simultaneously enlarge the tour t_i and reduce the length of the tour t_j (Figure 4 (b)). We plan to conduct a series of experiments to test how much the approximation factor improves with these tools.

References

1 Sergey Bereg, José-Miguel Díaz-Báñez, Alina Kasiuk, Miguel-Angel Pérez-Cutiño, and Fabio Rodríguez. Covering segments on a line with drones. In *Proc. 20th Spanish Meeting on Computational Geometry (EGC23)*, page 5, 2023.

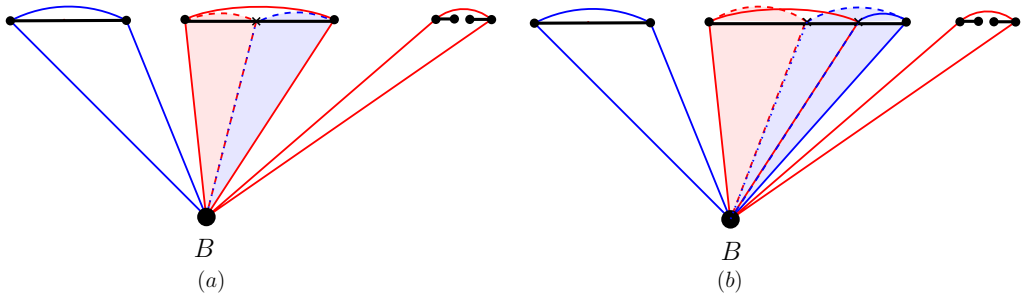


Figure 4 (a) Cutting a red tour. (b) Enlarging a blue tour and reducing a red tour.

2 James F. Campbell, Ángel Corberán, Isaac Plana, and José M. Sanchis. Drone arc routing problems. *Networks*, 72(4):543–559, 2018.

3 Michael R Garey. Computers and intractability: A guide to the theory of np-completeness, freeman. *Fundamental*, 1997.

4 Giusy Macrina, Luigi Di Puglia Pugliese, Francesca Guerriero, and Gilbert Laporte. Drone-aided routing: A literature review. *Transportation Research Part C: Emerging Technologies*, 120:102762, 2020.

5 Miguel-Angel Pérez-Cutiño, Juan Valverde, and José-Miguel Díaz-Báñez. Detecting broken receiver tubes in csp plants using intelligent sampling and dual loss. *Applied Intelligence*, pages 1–16, 2023.

6 Sékou Togola, Sountongnoma Martial Anicet Kiemde, and Ahmed Dooguy Kora. Real time and post-processing flight inspection by drone: A survey. In *Proc. 43rd International Conference on Telecommunications and Signal Processing (TSP)*, pages 399–402. IEEE, 2020.

Bipartite Dichotomous Ordinal Graphs* †

Patrizio Angelini[1], Sabine Cornelsen[2], Carolina Haase[3], Michael Hoffmann[4], Eleni Katsanou[5], Fabrizio Montecchiani[6], and Antonios Symvonis[5]

1 **John Cabot University, Rome, Italy**
pangelini@johncabot.edu
2 **Department of Computer and Information Science, University of Konstanz, Germany**
sabine.cornelsen@uni-konstanz.de
3 **Trier University, Germany**
haasec@uni-trier.de
4 **Department of Computer Science, ETH Zürich, Switzerland**
hoffmann@inf.ethz.ch
5 **School of Applied Mathematical and Physical Sciences, National Technical University of Athens, Greece**
ekatsanou@mail.ntua.gr, symvonis@math.ntua.gr
6 **Engineering Department, University of Perugia, Italy**
fabrizio.montecchiani@unipg.it

Abstract

A *dichotomous ordinal graph* consists of an undirected graph $G = (V, E_s \cup E_\ell)$ with an ordered partition of the set of edges into a set E_s of *short* edges and a set E_ℓ of *long* edges. A *geometric representation* of a dichotomous ordinal graph is a straight-line drawing Γ of G such that the short edges of G are exactly those edges that have length at most one in Γ.

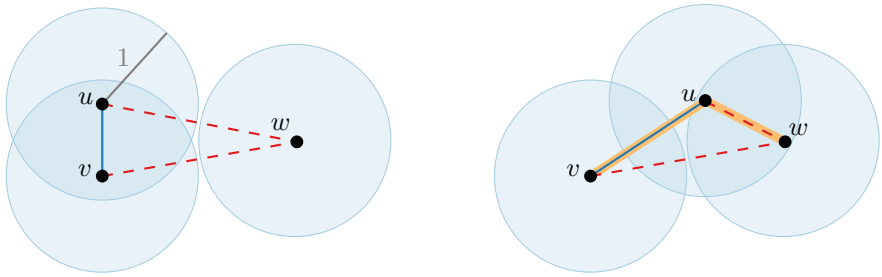
We characterize for which bipartite graphs all ordered partitions of the edge set admit a geometric representation as a dichotomous ordinal graph. On the one hand, such a representation always exists if the graph is a subgraph of $K_{3,m}$, for an arbitrary m, or a subgraph of $K_{4,6}$. On the other hand, there exist dichotomous ordinal $K_{4,7}$ and $K_{5,5}$ that do not admit a geometric representation. Moreover, any bipartite dichotomous ordinal graph admits a geometric representation if the short edges induce an outerplanar graph and any dichotomous ordinal graph admits a geometric representation if the short edges induce a subgraph of the rectangular grid.

1 Introduction

A *dichotomous ordinal graph* consists of an undirected graph $G = (V, E_s \cup E_\ell)$ with a partition of the edges into a set E_s of *short* edges and a set E_ℓ of *long* edges. A *geometric representation* of a dichotomous ordinal graph is a straight-line drawing Γ of G such that the short edges of G are exactly those edges that have length at most one in Γ. Fig. 1 shows two straight-line drawings of the same dichotomous ordinal graph. The drawing in (a) is a geometric representation of it, whereas the drawing in (b) is not.

* This work was initiated at the Annual Workshop on Graph and Network Visualization (GNV2023), Chania, Greece, June 2023.
† Research of FM partially supported by MUR of Italy, under PRIN Project n. 2022ME9Z78 - NextGRAAL— Next-generation algorithms for constrained GRAph visuALization.

40th European Workshop on Computational Geometry, Ioannina, Greece, March 13–15, 2024.
This is an extended abstract of a presentation given at EuroCG'24. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.



(a) valid drawing of $\triangle uvw$ **(b)** invalid drawing of $\triangle uvw$

Figure 1 valid (a) and invalid (b) drawing of a dichotomous ordinal triangle $\triangle uvw$; short edge uv (blue) and long edges uw and vw (red/dashed)

Related Results. It is NP-hard to decide whether a dichotomous ordinal graph admits a geometric representation, even if the underlying graph is a complete graph and the short edges induce a planar graph [1, Lemma 1] or the underlying graph is a complete bipartite graph [10, Theorem 4]. In the latter case, the problem is even known to be $\exists\mathbb{R}$-complete [10].

Angelini et al. [2] investigated for which graphs G any ordered partition of the edges admits a geometric representation as a dichotomous ordinal graph. This is the case if G is a *double-wheel* (a simple cycle and two additional vertices connected to all vertices of the cycle), *2-degenerate* (can be reduced to the empty graph by repeatedly removing vertices of degree at most two), *subcubic* (each vertex has degree at most three), or 4-colorable and the short edges induce a *caterpillar* (tree such that the removal of degree one vertices yields a path). On the negative side, they [2] proved that if G is the double-wheel plus one edge, then there exists a partition of the edge set of G into short and long edges that doesn't admit a geometric representation as a dichotomous ordinal graph.

Closely related is the notion of *ordinal embeddings*. Given a set of objects $x_1, \ldots, x_n$ in an abstract space together with a set of *ordinal constraints* of the form $dist(x_i, x_j) < dist(x_k, x_l)$, we are asked to compute a set of points $p_1, \ldots, p_n$ in the d-dimensional Euclidean space $\mathbb{R}^d$ such that, by preserving as many ordinal constraints as possible, it returns a good approximation of the displacement of $x_1, \ldots, x_n$. Ordinal embeddings were first studied in the 60's by Shepard [11, 12] and Kruskal [8, 9] in the context of psychometric data analysis. Recently, there have been applications in the field of Machine Learning [14]. The computation of ordinal embeddings is also known in the literature as *non-metric multi-dimensional scaling*. For an extensive literature review on ordinal embeddings refer to [15].

Of particular interest in relation to our work is the application of ordinal embeddings in the problem of recognizing Euclidean Multidimensional preferences [3, 5, 10] in the field of Computational Social Science. The objects are either *voters* or *alternatives*, which, together with the ordinal constraints (i.e., the voters' preferences), naturally define a bipartite graph. However, the goal is to find an embedding in $\mathbb{R}^d$ where all constraints are satisfied rather than to seek for an approximation. Efficient algorithms exist when $d = 1$ [4, 5], while for any $d \geq 2$ the problem is as hard as the existential theory of the reals [10]. The case where a voter either *likes* or *dislikes* a preference has also been studied [6, 10]. Note that, in this setting, an embedding that employs short and long edges can fully represent the likeness/dislikeness of voters to alternatives. This is precisely the problem this paper is devoted to.

Our Results. A dichotomous ordinal graph $G = (U \cup W, E_s \cup E_\ell)$ is *bipartite* if $E_s \cup E_\ell \subseteq U \times W$. We study in particular complete bipartite dichotomous ordinal $K_{n,m}$, i.e., bipartite

graphs $G = (U \cup W, E_s \cup E_\ell)$ with $|U| = n$, $|W| = m$, and $E_s \cup E_\ell = U \times W$. We show that subgraphs of dichotomous ordinal $K_{3,m}$, $m \in \mathbb{N}$ or $K_{4,6}$ always admit a geometric representation (Theorem 2.1) while there are dichotomous ordinal $K_{4,7}$ (Theorem 2.2) and $K_{5,5}$ (Theorem 2.3) that do not admit a geometric representation. Further, a bipartite dichotomous ordinal graph always admits a geometric representation if the short edges induce an outerplanar graph (Theorem 3.1) or a subgraph of the grid (Theorem 3.2). In both cases, the subgraph of short edges can even be drawn without edge crossings. However, there are bipartite dichotomous ordinal graphs that do not admit a geometric representation even though the subgraph of short edges is planar (Theorems 2.2 and 2.3).

Preliminaries. Let $G = (V, E_S \cup E_\ell)$ be a dichotomous ordinal graph and assume that there exists a set of long edges whose removal creates different connected components. We can draw these connected components far apart. Now the long edges of G between different connected components will be drawn with a length greater than one. This yields the following.

▶ Observation 1. A dichotomous ordinal graph admits a geometric representation if and only if each subgraph induced by a connected component of the short edges does.

2 Complete Bipartite Graphs — A Characterization

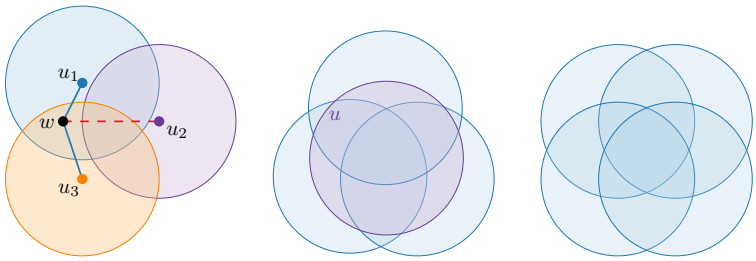
A convenient way to reason about geometric representations for bipartite graphs is in terms of arrangements of unit circles. Consider a bipartite dichotomous ordinal graph $G = (U \cup W, E)$ and suppose that the vertices of U are already drawn as points in the plane. Then, to obtain a geometric representation for G the task is to place each $w \in W$ such that for each $u \in U$ the point w lies in the unit disk centered at u if and only if the edge uw is short; see Fig. 2a.

A related question is the existence of a representation of a graph as a *unit disk graph*, where vertices are represented by unit disks, and they are connected by an edge if and only if the corresponding disks intersect. The main difference compared to dichotomous ordinal graphs lies in the different types of edges. In a unit disk representation, there are only two types: edge and non-edge, and all of them have to be faithfully represented. In a geometric realization of dichotomous ordinal graphs, there are three types of edges: long, short, and non-edges, and we have no constraints concerning the last type.

Let $U = \{u_1, \ldots, u_n\}$, let C_i denote the unit circle centered at u_i, and let D_i denote the corresponding unit disk. Let $\mathcal{C}$ denote the arrangement of $C_1, \ldots, C_n$. With every vertex $w \in W$ we associate a subset $V(w) \subseteq U$ such that $u \in V(w)$ if and only if the edge uw is short. We refer to $V(w)$ as a *singleton*, a *pair*, or a *triple* if $V(w)$ contains one, two, or three vertices, respectively. A subset $X \subseteq U$ is *realized* by a drawing of U if there is a cell r in $\mathcal{C}$ such that $r \subseteq D_i$ if and only if $u_i \in X$. Then there exists a geometric realization for G if and only if there exists a drawing/placement of U such that $V(w)$ is realized for all $w \in W$.

▶ **Theorem 2.1.** *Every dichotomous ordinal $K_{3,m}$, for $m \in \mathbb{N}$, and every dichotomous ordinal $K_{4,m}$, for $m \leq 6$, admits a geometric representation.*

Proof. For $K_{3,m}$ we can draw $U = \{u_1, u_2, u_3\}$ so that all eight subsets of U are realized; see Fig. 2a. For $|U| \geq 4$ such a universal placement is not possible because an arrangement of n circles has at most $n(n-1)+2$ cells [13]. So an arrangement of four circles has at most 14 cells, whereas a four-element set has 16 subsets. However, for $|U| = 4$ and $|W| \leq 6$ we can always obtain a geometric representation as follows. Let $V(W) \subset 2^U$ denote the set of subsets of U that are associated to some vertex of W.



(a) $K_{3,m}$ **(b)** $K_{4,m}$ all pairs **(c)** $K_{4,m}$ all triples

Figure 2 Regions for the other side.

If there are at least three pairs in $V(W)$, then, given that $|V(W)| \leq |W| \leq 6$, the number of triples plus the number of singletons in $V(W)$ together is at most three. Thus, as $|U| = 4$, there exists a vertex $u \in U$ such that $\{u\} \notin V(W)$ and $U \setminus \{u\} \notin V(W)$. So we can use the drawing depicted in Fig. 2b, where we assign u to the central circle. As all subsets of U other than $\{u\}$ and $U \setminus \{u\}$ are realized, this is a valid geometric representation of G.

Otherwise, there are at most two pairs in $V(W)$. We use the drawing depicted in Fig. 2c, where we assign the vertices of U to the circles so that both pairs in $V(W)$ appear consecutively in the circular order of circles. (This works regardless of whether or not these pairs share a vertex.) As all subsets of U other than the two pairs that correspond to opposite circles in the drawing are realized, this is a valid geometric representation of G. ◀

▶ **Theorem 2.2.** *There is a dichotomous ordinal $K_{4,7}$ that does not admit a geometric representation.*

Proof Sketch. Let $U = \{u_1, u_2, u_3, u_4\}$ and $W = \{w_1, \ldots, w_7\}$ denote the vertex partition. For each w_i, we can specify an associated set $U_i \subseteq U$ (such that exactly the edges between w_i and U_i are short; see Fig. 3a). We choose all four subsets of size three and the three subsets of size two that contain u_4, and distribute them among the vertices of W arbitrarily. In any geometric representation, each set U_i corresponds to a cell in the induced arrangement $\mathcal{C}$ of unit circles. Two more cells are required implicitly: The outer cell, which corresponds to $\emptyset \subset U$, and a cell that corresponds to the whole set U and is required by Helly's Theorem [7] because disks are convex and we specified all triples to be among the sets U_i. Using these properties of $\mathcal{C}$ we can show that it cannot be realized using unit circles. ◀

▶ **Theorem 2.3.** *There is a dichotomous ordinal $K_{5,5}$ that does not admit a geometric representation.*

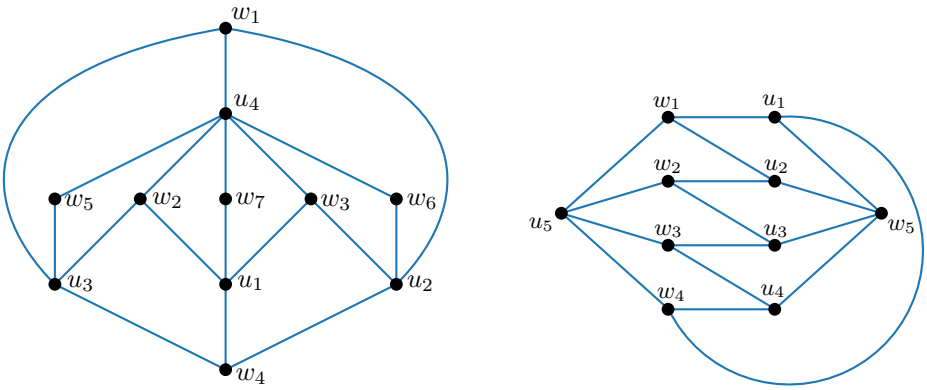
Proof Sketch. Let $U = \{u_1, \ldots, u_5\}$ and $W = \{w_1, \ldots, w_5\}$ denote the vertex partition. To each $w_i \in W$, we associate a set $U_i \subseteq U$ of "short neighbors" (see Fig. 3b):

$$U_i = \{u_i, u_{i\oplus 1}, u_5\}, \text{for } 1 \leq i \leq 4, \text{ and } U_5 = U \setminus \{u_5\},$$

where $i \oplus 1 = (i \bmod 4) + 1$. In any geometric representation, each set U_i corresponds to a cell in the induced arrangement $\mathcal{C}$ of unit circles. Using the existence of these cells we can analyze $\mathcal{C}$ geometrically and show that it cannot be realized using unit circles. ◀

3 Short Outerplanar Graphs and Short Subgraphs of the Grid

We show that every bipartite dichotomous ordinal graph admits a geometric representation if the subgraph G_s induced by the short edges is outerplanar or a subgraph of the grid. In



(a) short edges of $K_{4,7}$ **(b)** short edges of $K_{5,5}$

Figure 3 A dichotomous ordinal $K_{4,7}$ and $K_{5,5}$, respectively, that does not admit a geometric representation. The drawn edges are the short edges. Edges between vertices labeled u on one hand and w on the other hand, that are not drawn, are long.

the first case, we construct a planar drawing of G_s in which the BFS-layers are drawn on horizontal lines. See Fig. 4b. In the second case, we suitably perturb the grid. See Fig. 5.

▶ **Theorem 3.1.** *A bipartite dichotomous ordinal graph admits a geometric representation if the subgraph induced by the short edges is outerplanar.*

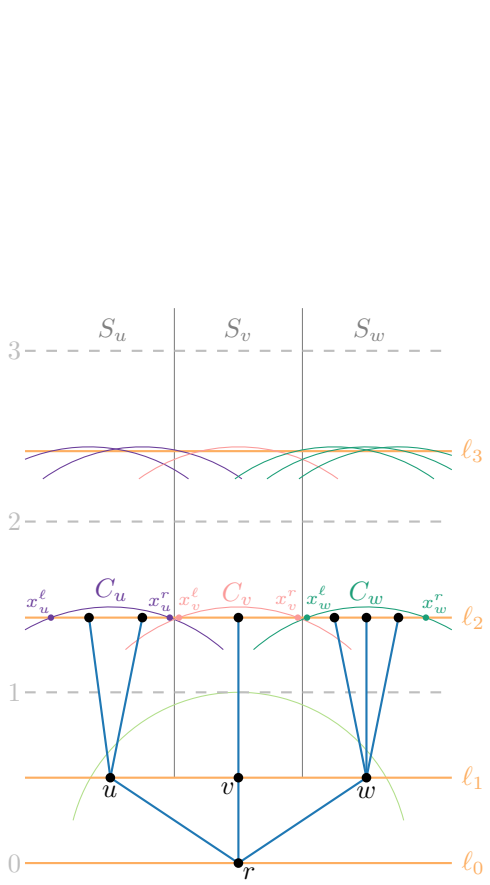
Proof Sketch. Let $G = (V, E_s \cup E_\ell)$ be a bipartite dichotomous ordinal graph such that $G_s = (V, E_s)$ is outerplanar. By Observation 1, we may assume that G_s is connected. We root G_s at an arbitrary vertex r. Let V_k, $k = 0, \dots$ be the BFS layers of G_s rooted at r, i.e., $V_0 = \{r\}$, V_1 is the set of neighbors of r, and V_{k+1}, $k \geq 1$ is the set of neighbors of the vertices in V_k that are not already in V_{k-1}. We say that w is a *child* of v and v is a *parent* of w if vw is an edge of G_s, $v \in V_k$ and $w \in V_{k+1}$ for some k. By outerplanarity, each vertex has at most two parents. We construct a planar drawing of G_s with the following properties.

- The root r is drawn with y-coordinate $y_0 = 0$. All vertices in layer V_k, $k > 0$ are on a horizontal line ℓ_k with y-coordinate y_k strictly between $k-1$ and k.
- The distance between a vertex and its children is at most 1 while the distance between two vertices on consecutive layers is greater than 1 if they are not adjacent in G_s.
- For each vertex v there is a vertical strip S_v such that (a) v is in S_v, (b) S_w is contained in the union of the strips of w's parents, (c) S_u and S_v are internally disjoint if u and v are on the same layer.

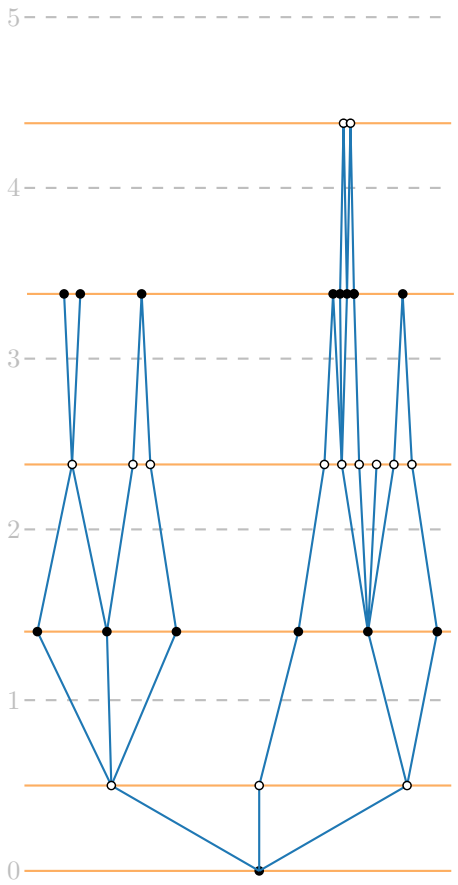
Special care has to be taken if a vertex $w \in V_{k+1}$ has two parents u and v, i.e., if w closes an internal face. In that case, we want to draw w on line ℓ_{k+1}, on the common boundary of S_u and S_v, and with distance exactly one to both u and v. ◀

▶ **Theorem 3.2.** *A dichotomous ordinal graph $G = (V, E_s \cup E_\ell)$ admits a geometric representation if the set of short edges induces a subgraph of the grid.*

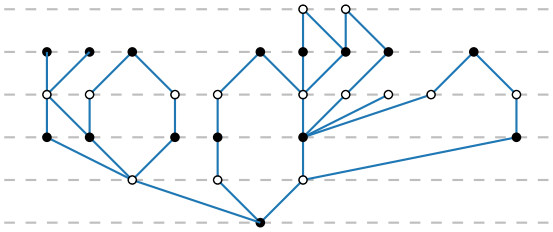
Proof Sketch. Extend $G_s = (V, E_s)$ by the remaining grid edges and require the new edges to be long. Use the construction in Fig. 5 to place the vertices. Then the short edges are shorter than $n^2 + 1/2$, while the long edges have length at least $n^2 + 1$. Scale the drawing. ◀



(a) construction for a tree



(b) geometric realization when G_s is the graph in (c)



(c) bipartite outerplanar graph

Figure 4 How to construct a geometric realization of a bipartite dichotomous ordinal graph if the short edges induce an outerplanar graph.

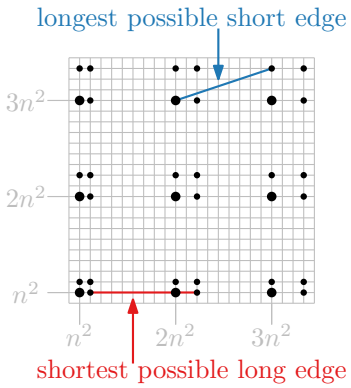


Figure 5 For each grid point (i, j), $1 \leq i \leq n$, $1 \leq j \leq n$ there are four possible points. If $i > 1$, the x-coordinate is in^2 if the edge between $(i-1, j)$ and (i, j) is short and $in^2 + i$ otherwise. If $j > 1$, the y-coordinate is jn^2 if the edge between $(i, j-1)$ and (i, j) is short and $jn^2 + j$ otherwise.

4 Conclusion

We leave open the questions whether bipartite dichotomous ordinal graphs always admit a geometric realization in any of the following cases: (i) the underlying graph is planar; (ii) the underlying graph is 3-degenerate; or (iii) the graph induced by the short edges is a 2-tree. Questions (i) and (ii) are open even for non-bipartite dichotomous ordinal graphs.

References

1 Md. Jawaherul Alam, Stephen G. Kobourov, Sergey Pupyrev, and Jackson Toeniskoetter. Weak unit disk and interval representation of graphs. In Ernst W. Mayr, editor, *Graph-Theoretic Concepts in Computer Science (WG'15)*, volume 9224 of *LNCS*, pages 237–251. Springer, 2015. doi:10.1007/978-3-662-53174-7_17.

2 Patrizio Angelini, Michael A. Bekos, Martin Gronemann, and Antonios Symvonis. Geometric representations of dichotomous ordinal data. In Ignasi Sau and Dimitrios M. Thilikos, editors, *Graph-Theoretic Concepts in Computer Science*, pages 205–217. Springer, 2019. doi:10.1007/978-3-030-30786-8_16.

3 Joseph F. Bennett and William L. Hays. Multidimensional unfolding: Determining the dimensionality of ranked preference data. *Psychometrika*, 25(1):27–43, 1960.

4 Jiehua Chen, Kirk Pruhs, and Gerhard J. Woeginger. The one-dimensional Euclidean domain: finitely many obstructions are not enough. *Social Choice and Welfare*, 48(2):409–432, 2017.

5 Jean-Paul Doignon and Jean-Claude Falmagne. A polynomial time algorithm for unidimensional unfolding representations. *J. Algorithms*, 16(2):218–233, 1994.

6 Edith Elkind and Martin Lackner. Structure in dichotomous preferences. In Qiang Yang and Michael Wooldridge, editors, *IJCAI*, pages 2019–2025. AAAI Press, 2015.

7 Eduard Helly. Über Mengen konvexer Körper mit gemeinschaftlichen Punkten. *Jahresbericht der Deutschen Mathematiker-Vereinigung*, 32:175–176, 1932. URL: http://eudml.org/doc/145659.

8 J. Kruskal. Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika*, 29(1):1–27, 1964.

9 J. Kruskal. Nonmetric multidimensional scaling: A numerical method. *Psychometrika*, 29(2):115–129, 1964.

10 Dominik Peters. Recognising multidimensional Euclidean preferences. In Singh and Markovitch, editors, *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17)*, pages 642–648, 2017. doi:10.1609/AAAI.V31I1.10616.

11 Roger N. Shepard. The analysis of proximities: Multidimensional scaling with an unknown distance function. I. *Psychometrika*, 27(2):125–140, 1962.

12 Roger N. Shepard. The analysis of proximities: Multidimensional scaling with an unknown distance function. II. *Psychometrika*, 27(3):219–246, 1962.

13 Jacob Steiner. Einige Gesetze über die Theilung der Ebene und des Raumes. *J. für die reine und angewandte Mathematik*, 1:349–364, 1826. doi:10.1515/crll.1826.1.349.

14 Yoshikazu Terada and Ulrike von Luxburg. Local ordinal embedding. In *ICML*, volume 32 of *JMLR Workshop and Conference Proceedings*, pages 847–855. JMLR.org, 2014.

15 Leena Chennuru Vankadara, Michael Lohaus, Siavash Haghiri, Faiz Ul Wahab, and Ulrike von Luxburg. Insights into ordinal embedding algorithms: A systematic evaluation. *J. Mach. Learn. Res.*, 24:191:1–191:83, 2023. URL: http://jmlr.org/papers/v24/21-1170.html.

Bounds on the Edge-length Ratio of 2-outerplanar Graphs

Emilio Di Giacomo[1], Walter Didimo[1], Giuseppe Liotta[1], Henk Meijer[2], Fabrizio Montecchiani[1], and Stephen Wismath[3]

1 **Università degli Studi di Perugia, Perugia, Italy**
{walter.didimo,emilio.digiacomo,giuseppe.liotta,fabrizio.montecchiani}@unipg.it
2 **University College Roosevelt, The Netherlands**
hendrikus.meijer@gmail.com
3 **University of Lethbridge, Canada**
wismath@uleth.ca

Abstract

The edge-length ratio of a planar straight-line drawing Γ of a graph G is the largest ratio between the lengths of every pair of edges of Γ. If the ratio is measured by considering only pairs of edges that are incident to a common vertex, we talk about local edge-length ratio. The (local) edge-length ratio of a planar graph is the infimum over all (local) edge-length ratios of its planar straight-line drawings. It is known that the edge length ratio of outerplanar graphs is upper bounded by a constant, while there exist graph families with non-constant outerplanarity that have non-constant lower bounds to their edge-length ratios. In this paper we prove an $\Omega(\sqrt{n})$ lower bound on the local edge-length ratio (and hence on the edge-length ratio) of the n-vertex 2-outerplanar graphs. We also prove a constant upper bound to the edge length ratio of Halin graphs.

1 Introduction

Let Γ be a planar straight-line drawing of a planar graph $G = (V, E)$. For any edge $e \in E$, let $|e|_\Gamma$ be the length of the segment representing e in Γ. The *edge-length ratio of* Γ, denoted as $\rho(\Gamma)$, is the maximum ratio between the lengths of every two edges in Γ; the *local edge-length ratio* $\rho_\ell(\Gamma)$ *of* Γ is the maximum ratio between the lengths of two adjacent edges. Formally,

$$\rho(\Gamma) = \max_{(u,v),(z,w)\in E} \frac{|(u,v)|_\Gamma}{|(z,w)|_\Gamma}, \qquad \rho_\ell(\Gamma) = \max_{(u,v),(v,w)\in E} \frac{|(u,v)|_\Gamma}{|(v,w)|_\Gamma}.$$

The *edge-length ratio* $\rho(G)$ *of* G is the infimum of $\rho(\Gamma)$ over the set $\mathcal{D}(G)$ of all planar straight-line drawings Γ of G, i.e., $\rho(G) = \inf_{\Gamma\in\mathcal{D}(G)} \rho(\Gamma)$. Analogously, the *local edge-length ratio* $\rho_\ell(G)$ *of* G is defined as $\rho_\ell(G) = \inf_{\Gamma\in\mathcal{D}(G)} \rho_\ell(\Gamma)$.

We remark that since the publication of the first book on graph drawing [7], minimizing the maximum edge length provided that the shortest edge has length one (i.e. minimizing the edge-length ratio) is among the most relevant optimization goals, because of its impact on the readability of the computed visualization. Eades and Wormald [9] prove that deciding whether a biconnected planar graph has edge-length ratio one is NP-hard, and Cabello et al. [5] extend this result to triconnected instances. Borrazzo and Frati [4] prove that the edge-length ratio of n-vertex planar 3-trees is $\Omega(n)$. As for n-vertex planar 2-trees, Blažej et al. [3] prove an $\Omega(\log n)$ lower bound. Notably, both the lower bound by Borrazzo and Frati and the lower bound by Blažej et al. use graph families whose outerplanarity grows as a function of n. In contrast, Lazard et al. [10] show that graphs with outerplanarity one (i.e. outerplanar graphs) have a constant upper bound to their edge-length ratio.

40th European Workshop on Computational Geometry, Ioannina, Greece, March 13–15, 2024.
This is an extended abstract of a presentation given at EuroCG'24. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

A natural question that stems from the previous literature is whether outerplanarity one is a hard cutoff for achieving constant edge-length ratio. We answer this question in the affirmative, proving that graphs with outerplanarity two, i.e. the *2-outerplanar graphs*, have unbounded edge-length ratio. Nonetheless we prove a constant upper bound on the edge-length ratio of a well-studied family of 2-outerplanar graphs. Our results are as follows.

- We describe a family of n-vertex 2-outerplanar graphs whose *local* edge-length ratio is in $\Omega(\sqrt{n})$ which implies a lower bound also for the edge-length ratio of these graphs. It is worth noticing that while graph families with $O(1)$ local edge-length ratios are known [3], no family with $\omega(1)$ local edge-length ratio was previously known.
- We show that Halin graphs have edge-length ratio at most 3. We remark that Halin graphs are well-known subjects of study in the graph drawing literature; see e.g. [2, 6, 8].

Our approach for the lower bound builds upon ideas of Borrazzo and Frati [4]. Our upper bound is proved by translating the problem of computing drawings with bounded edge-length ratio to a topological question about (a variant of) level planarity with limited edge span. As a byproduct, the proof regarding the edge-length ratio upper bound of Halin graphs fixes an imprecision in the literature about the span of their weakly leveled planar drawings. For reasons of space some proofs are omitted or sketched.

2 Lower Bound

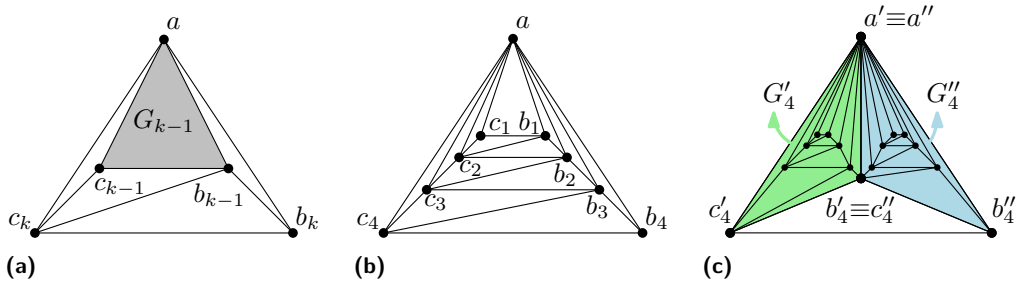


Figure 1 (a) Definition of the graph G_k. (b) Graph G_4 (c) Example of a graph G of Theorem 2.3.

We define a family of 2-outerplanar graphs G_k, for every $k \geq 1$, such that G_k has $n = 2k + 1$ vertices. The graph G_1 is a 3-cycle C_1. Assume that G_{k-1} has been defined and that its outer face is a 3-cycle C_{k-1} whose vertices are denoted as a, b_{k-1}, and c_{k-1}; then G_k is obtained by adding two vertices b_k and c_k, and the edges (a, b_k), (a, c_k), (b_k, c_k), (b_k, b_{k-1}), (c_k, c_{k-1}), and (c_k, b_{k-1}), embedded as shown in Fig. 1a. Note that G_k is 2-outerplanar and has $2k + 1$ vertices (see Fig. 1b for an example with $k = 4$). Let Γ be an embedding-preserving planar straight-line drawing of G_k. For $i = 1, 2, \ldots, k$, we denote by Δ_i the triangle that represents C_i in Γ and by $p(\Delta_i)$ its perimeter. We assume that the shortest edge over all triangles Δ_i has length 1; if not, we scale the drawing so to achieve this condition. The next lemma is a consequence of results by Borrazzo and Frati [4, pp.140-142].

▶ **Lemma 2.1.** *Let Γ be an embedding-preserving planar straight-line drawing of G_k, for $k \geq 2$; then $p(\Delta_i) > p(\Delta_{i-1}) + \gamma$, with $\gamma = 0.3$.*

We first prove a lower bound on $\rho_\ell(G_k)$ that holds if we consider only drawings that preserve the planar embedding of G_k. We then remove this restriction.

▶ **Lemma 2.2** (⋆). *Let Γ be an embedding-preserving planar straight-line drawing of G_k, for $k \geq 2$; then $\rho_\ell(\Gamma) \geq \sqrt{\frac{3k}{40}}$.*

Sketch. If $k \leq \frac{40}{3}$ the the statement is trivially true, since $\rho_\ell(\Gamma) \geq 1$. Thus, we can assume that $k > \frac{40}{3}$. By using induction, Lemma 2.1, and the fact that $p(\Delta_1) > \gamma$, we can prove that $p(\Delta_i) > \gamma \cdot i$, for every $i = 1, 2, \dots k$. Let L denote the length of the longest edge e_1 of the triangle Δ_k incident to a. We have that $L \geq \frac{p(\Delta_k)}{4} > \frac{\gamma \cdot k}{4} = \frac{3k}{40}$. Let e_2 be the shortest edge of Γ. Edge e_2 is incident to vertex a or to a neighbor v of a. If e_2 is incident to a, then $\rho_\ell(\Gamma) \geq \frac{|e_1|_\Gamma}{|e_2|_\Gamma} \geq \frac{3k}{40}$. Otherwise, edge $e_3 = (v, a)$ has vertex a in common with e_1 and vertex v in common with e_2. By definition we have $|e_1|_\Gamma \leq \rho_\ell(\Gamma)|e_3|_\Gamma \leq \rho_\ell(\Gamma)^2|e_2|_\Gamma = \rho_\ell(\Gamma)^2$ and $\rho_\ell(\Gamma) \geq \sqrt{|e_1|_\Gamma} \geq \sqrt{\frac{3k}{40}}$. ◀

To prove the next theorem we construct a 2-outerplanar graph with $n = 4k$ vertices such that, in every embedding, it contains a copy of G_k embedded as in Lemma 2.2 (see Fig. 1c).

▶ **Theorem 2.3** (⋆). *For every integer $k \geq 2$, there exists a 2-outerplanar graph G with $n = 4k$ vertices such that $\rho_\ell(G) \geq \sqrt{\frac{3n}{160}}$.*

3 Edge-length Ratio of Halin Graphs

A *k-span weakly level planar drawing* (*k-SWLP drawing*) Γ is a straight-line planar drawing whose vertices lie on a set of horizontal equispaced lines, called *levels*, and whose edges intersect at most $k + 1$ levels. Notice that, in a k-SWLP drawing edges between vertices that are consecutive in the same level are allowed. We assume that the levels are numbered from top to bottom and that the distance between consecutive levels is 1. An edge that intersects $k + 1$ levels has *span* k. A graph is *k-SWPL* if it has a k-SWLP drawing.

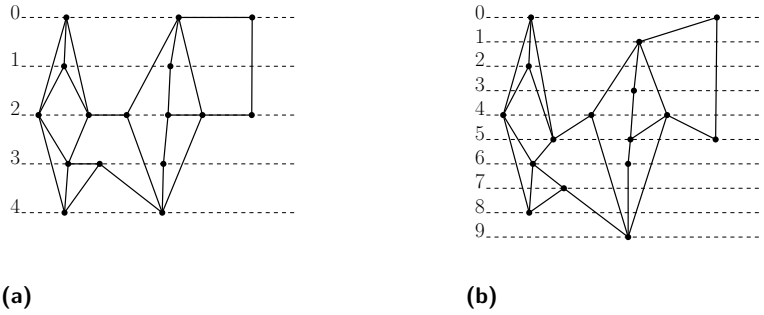


Figure 2 (a) A 2-SWLP drawing Γ; (b) The 5-SWLP drawing Γ' obtained from Γ.

▶ **Lemma 3.1** (⋆). *If G is a k-SWLP graph for some $k \geq 1$, then $\rho(G) \leq 2k + 1$.*

Proof. Let Γ be a k-SWLP drawing of G. We first transform Γ into a $(2k + 1)$-SWLP drawing such that every edge has span at least one, i.e., no edge has both end-vertices on the same level. To this aim we split each level i into two levels, numbered $2i$ and $2i + 1$, and assign the vertices of level i alternating between $2i$ and $2i + 1$. Let Γ' be the resulting drawing (see Fig. 2 for an example). For an arbitrarily chosen value $\varepsilon > 0$, we squeeze horizontally the drawing Γ' so that its width is ε. After this squeezing, for every edge e we have $1 \leq |e|_{\Gamma'} \leq 2k + 1 + \varepsilon$. It follows that $\rho(\Gamma) \leq 2k + 1 + \varepsilon$ and $\rho(G) \leq 2k + 1$. ◀

In the remainder we exploit Lemma 3.1 to prove a constant upper bound on the edge-length ratio of *Halin graphs*. A *Halin graph* (see Figs. 3a and 4b) is a 3-connected embedded planar graph G such that, by removing the edges along the boundary C of its outerface, one gets a tree T whose internal vertices have degree at least 3 and whose leaves are incident to the outerface of G. We call T the *characteristic tree* of G and we call C the *adjoint cycle* of G. A Halin graph G is *trivial* if it is a wheel graph, i.e., if T has only one non-leaf vertex.

Bannister et al. [1, Thm. 17] state that all Halin graphs are 1-SWLP which, together with Lemma 3.1, would imply an upper bound of 3 to their edge-length ratio. Unfortunately, K_4 is a Halin graph that is not 1-SWLP and the proof technique of [1] fails even for instances of Halin graphs different from K_4. Namely, let T be the characteristic tree of the Halin graph of Fig. 3a. According to the proof of Theorem 17 of [1] a leveling of T is computed as follows: Choose a leaf of T as the root and assign it to level 0; at Step i, assign to level $i+1$ the previously-unassigned nodes that are either children of nodes at level i or that belong to a path from one such children to its leftmost or rightmost leaf descendant in T. However, for any possible choice of the root of T, one obtains the leveling of Fig. 3b that has an edge between two non-consecutive vertices on a same level.

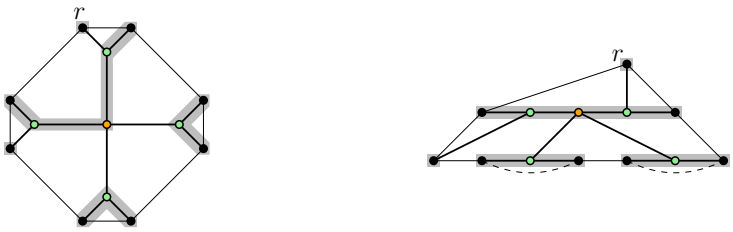


Figure 3 (a) A Halin graph; the vertices are grouped according to a leveling obtained with the technique in [1] choosing r as the root; (b) the corresponding level drawing.

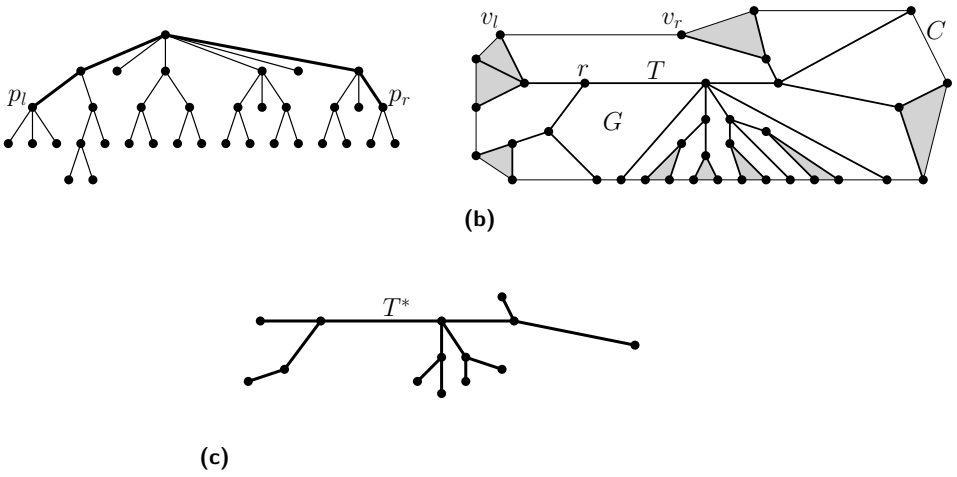


Figure 4 (a) The external path of a tree T; (b) A Halin graph G; the thick edges form the characteristic tree T, while the thin edges from the adjoint cycle C; the tufts of T are highlighted with gray areas. (c) The pruned tree T^* of T.

We now prove that all Halin graphs except K_4 are 1-SWLP (Lemmas 3.3 and 3.4), which by Lemma 3.1 implies an upper bound of 3 to the edge-length ratio of these graphs. Let T be an ordered rooted tree. The *external path* of T is defined as follows. If T is a single vertex r, then the external path of T coincides with r; otherwise it is the path connecting the parent p_l of the leftmost leaf of T and the parent p_r of the rightmost leaf of T (see Fig. 4a).

Let G be a non-trivial Halin graph. A *tuft* of the characteristic tree T of G is a maximal set of at least two leaves having the same parent, and such that this parent is adjacent to exactly one other internal vertex of T (in Fig. 4b the tufts are highlighted with gray areas). The *pruned tree* T^* of T is obtained by removing all leaves from T (see Fig. 4c).

▶ **Lemma 3.2** (⋆). *Let G be a Halin graph distinct from K_4 and let T be the characteristic tree of G. The number of tufts of T is equal to the number of leaves of the pruned tree of T.*

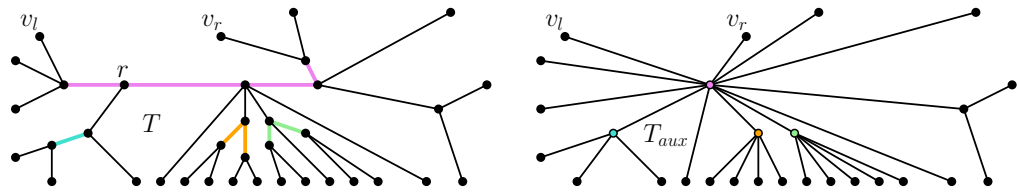


Figure 5 (a) A decomposition in characteristic paths of the characteristic tree of the Halin graph of Fig. 4b; (b) The corresponding auxiliary tree T_{aux}.

▶ **Lemma 3.3** (⋆). *Let G be a non-trivial Halin graph and let T be the characteristic tree of G rooted at any non-leaf vertex. Let v_l be the leftmost leaf and v_r be the rightmost leaf of T. If both v_l and v_r belong to a tuft, then $G \setminus (v_l, v_r)$ has a 1-SWLP drawing Γ such that v_l is the first vertex of the topmost level and v_r is the last vertex of the same level.*

Sketch. We simplify the characteristic tree T of G by collapsing into single vertices a set of suitably defined paths called *characteristic paths* and illustrated in Fig. 5. The external path of T is a characteristic path. For each vertex v of a characteristic path π and for each child w of v that is not in π, let T' be the tree rooted at w. The external path of T' is a characteristic path of T. Denote by T_{aux} the tree obtained by collapsing the characteristic paths into vertices; for a vertex v of T_{aux} that corresponds to a path π of T, we say that π is the *pertinent path* of v. We compute first a 1-SWLP drawing Γ_{aux} of T_{aux} (see Fig. 6a). The level of each vertex is equal to its depth in T_{aux} and the order of the vertices in each level is given by the left-to-right order of T_{aux}. We now replace each vertex of T_{aux} by its pertinent path, thus obtaining a 1-SWLP drawing Γ_T of T (see Figs. 6b and 6c). It is easy to see that all the edges of the adjoint cycle that are distinct from (v_l, v_r) can be added to the drawing without crossings and with no span increase. We finally move v_l and v_r to the topmost level. Let $v_l = v_1, v_2 \ldots, v_k = v_r$ be the leaves of T in the order they appear counterclockwise along the adjoint cycle C of G. Since v_l belongs to a tuft, v_2 is a sibling of v_l and they are both drawn on level 1. Also, their parent is the first vertex on level 0. Thus, v_l can be moved to the left of the leftmost vertex of level 0 without crossings and with no span increase. By a symmetric argument, v_r can be moved to the right of the righmost vertex of level 0 (see Fig. 6c). ◀

Lemma 3.3 allows us to compute a drawing of a Halin graph except for one edge. In the next lemma we explain how to cope with this issue.

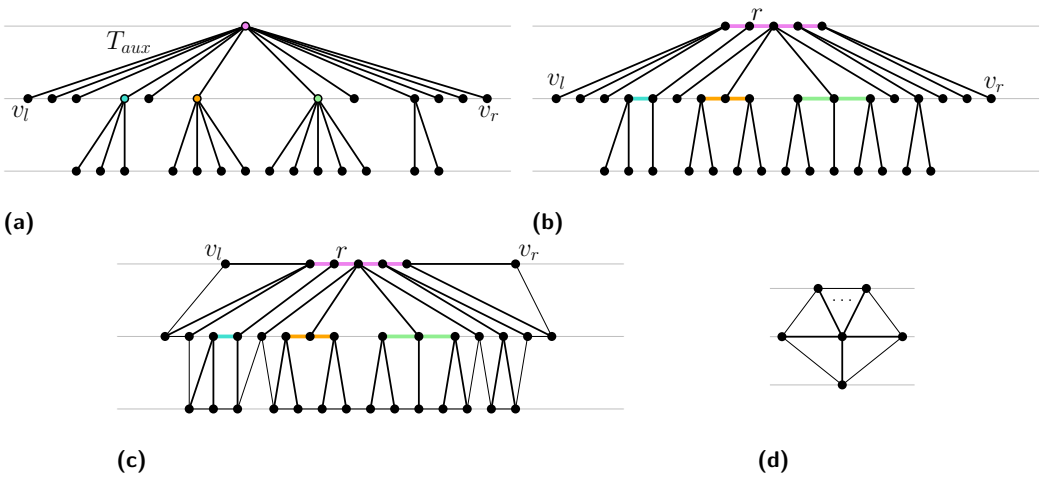


Figure 6 (a) A 1-SWLP drawing Γ_{aux} of T_{aux}; (b) A 1-SWLP drawing of T obtained from Γ_{aux} by replacing each vertex with its pertinent path; (c) A 1-SWLP drawing of $G \setminus (v_l, v_r)$ with the properties of Lemma 3.3; (d) A 1-SWLP drawing of a trivial Halin graph.

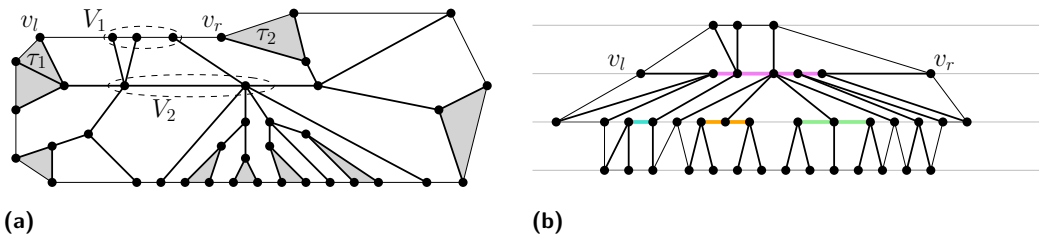


Figure 7 Illustration of Lemma 3.4, Case 1: (a) A Halin graph G; (b) A 1-SWLP drawing of G.

▶ **Lemma 3.4** ($\star$). *Every Halin graph G distinct from K_4 has a 1-SWLP drawing.*

Proof. Let T be the characteristic tree of G and C be its adjoint cycle. If G is trivial, a 1-SWLP drawing of G is computed as in Fig. 6d. Otherwise, T has at least one edge and, by Lemma 3.2, at least two tufts. A leaf not belonging to any tuft is a *single leaf*.

Case 1: T has at least one single leaf. We remove a maximal set V_1 of consecutive single leaves along C (see Fig. 7a). By Lemma 3.3 we compute a drawing of the resulting graph such that the leaf v_l preceding V_1 walking clockwise along C and the leaf v_r following V_1 walking clockwise along C, are the first and the last vertex, respectively, on the topmost level. To construct a 1-SWLP drawing of G, we put the single leaves of V_1 on a new level above the topmost in the order they appear along C. See Fig. 7b.

Case 2: T has no single leaves. If T^* has at most 1 internal vertex, then T^* is a single edge or it is a star with at least three edges; a 1-SWLP drawing of G can be constructed as in Figs. 8a and 8b. Otherwise, T^* has one edge e^* whose end-vertices are both non-leaves. Further, e^* is shared by two faces each having an edge belonging to C. Removing these two edges and e^* (possibly smoothing the end-vertices of e^* if they have degree two after the removal) we get two subgraphs G_a and G_b of G (Fig. 8c) for which we compute two 1-SWLP drawings according to Lemma 3.3 (Figs. 8d and 8e). We then combine the two drawings by

mirroring vertically and horizontally one of them. This allows us to add the three removed edges without crossings and without span increase (Fig. 8f). ◀

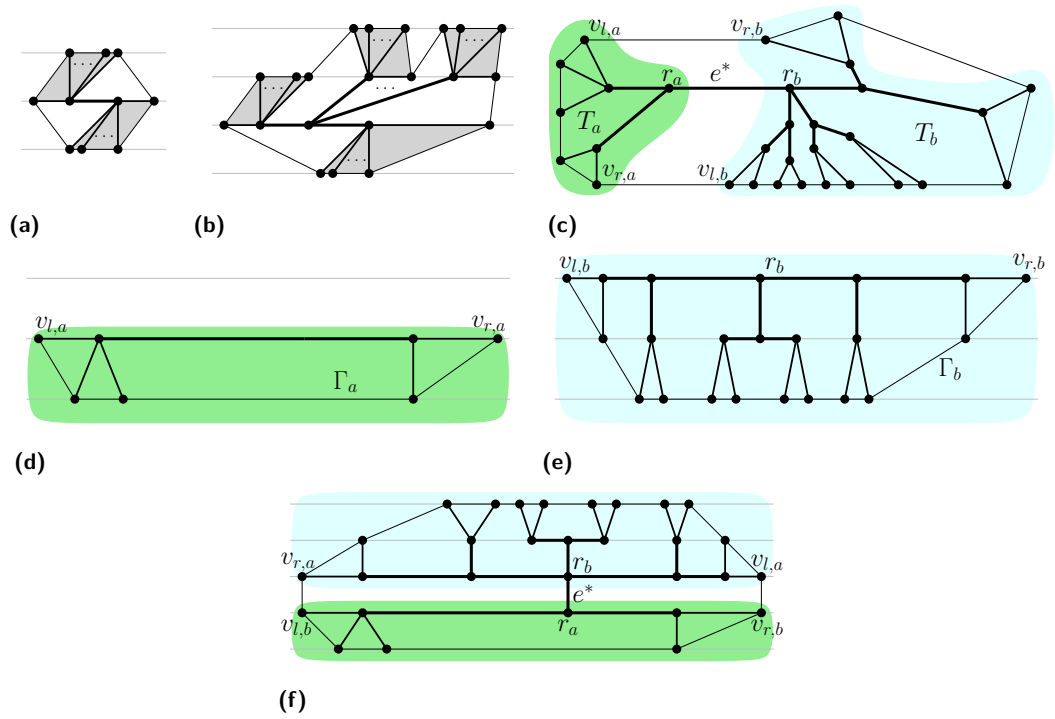


Figure 8 Illustration of Lemma 3.4, (a)-(b) Case 2a; (c)-(f) Case 2b.

▶ **Theorem 3.5.** *If G is a Halin graph, then $\rho(G) \leq 3$.*

4 Open Problems

(i) Is the bound of Theorem 2.3 asymptotically tight? (ii) Study other sub-families of 2-outerplanar graphs that have constant (local) edge-length ratio.

References

1 Bannister, M.J., Devanny, W.E., Dujmović, V., Eppstein, D., Wood, D.R.: Track layouts, layered path decompositions, and leveled planarity. Algorithmica **81**(4), 1561–1583 (2019). 10.1007/s00453-018-0487-5

2 Bekos, M.A., Kaufmann, M., Raftopoulou, C.N.: AVDTC of generalized 3-halin graphs. In: IISA 2016. pp. 1–6. IEEE (2016). 10.1109/IISA.2016.7785421

3 Blažej, V., Fiala, J., Liotta, G.: On edge-length ratios of partial 2-trees. Int. J. Comput. Geom. Appl. **31**(2-3), 141–162 (2021). 10.1142/S0218195921500072

4 Borrazzo, M., Frati, F.: On the planar edge-length ratio of planar graphs. J. Comput. Geom. **11**(1), 137–155 (2020). 10.20382/jocg.v11i1a6

5 Cabello, S., Demaine, E.D., Rote, G.: Planar embeddings of graphs with specified edge lengths. J. Graph Algorithms Appl. **11**(1), 259–276 (2007). 10.7155/jgaa.00145

6 Chaplick, S., Da Lozzo, G., Di Giacomo, E., Liotta, G., Montecchiani, F.: Planar drawings with few slopes of Halin graphs and nested pseudotrees. In: WADS 2021. Lecture Notes in Computer Science, vol. 12808, pp. 271–285. Springer (2021). 10.1007/978-3-030-83508-8_20

7 Di Battista, G., Eades, P., Tamassia, R., Tollis, I.G.: Graph Drawing: Algorithms for the Visualization of Graphs. Prentice-Hall (1999)

8 Di Giacomo, E., Liotta, G., Meijer, H.: Computing straight-line 3D grid drawings of graphs in linear volume. Comput. Geom. **32**(1), 26–58 (2005). 10.1016/j.comgeo.2004.11.003

9 Eades, P., Wormald, N.C.: Fixed edge-length graph drawing is NP-hard. Discret. Appl. Math. **28**(2), 111–134 (1990). 10.1016/0166-218X(90)90110-X

10 Lazard, S., Lenhart, W.J., Liotta, G.: On the edge-length ratio of outerplanar graphs. TCS **770**, 88–94 (2019). 10.1016/j.tcs.2018.10.002

Simplified and Improved Bounds on the VC-Dimension for Elastic Distance Measures*

Frederik Brüning[1] **and Anne Driemel**[1]

1 **Department of Computer Science, University of Bonn, Germany**

Abstract

We study the Vapnik–Chervonenkis dimension (VC-dimension) of range spaces, where the ground set consists of either polygonal curves in $\mathbb{R}^d$ or polygonal regions in the plane that may contain holes and the ranges are balls defined by an elastic distance measure, such as the Hausdorff distance, the Fréchet distance and the dynamic time warping distance (DTW). We show for the Fréchet distance of polygonal curves and the Hausdorff distance of polygonal curves and planar polygonal regions that the VC-dimension is upper-bounded by $O(dk\log(km))$, where k is the complexity of the center of a ball, m is the complexity of the polygonal curve or region in the ground set, and d is the ambient dimension. For $d \geq 4$ this bound is tight in each of the parameters d, k and m separately. For DTW of polygonal curves, our analysis directly yields an upper-bound of $O(\min(dk^2\log(m), dkm\log(k)))$.

Related Version arXiv:2308.05998

1 Introduction

The Vapnik–Chervonenkis dimension (VC-dimension) [15] is a measure of complexity for range spaces. Knowing the VC-dimension of a range space can be used to determine sample bounds for various computational tasks. These include sample bounds on the test error of a classification model in statistical learning theory [14] or sample bounds for an ε-net [11] or an (η, ε)-approximation [10] in computational geometry. Sample bounds based on the VC-dimension have been successfully applied in the context of kernel density estimation [12], neural networks [2, 13], coresets [5, 8, 9], clustering [1, 3] and other data analysis tasks.

We study range spaces, where the ground set consists of polygonal curves or polygonal regions and the ranges consist of balls defined by the Hausdorff distance. Previous to our work, Driemel, Nusser, Phillips and Psarros [7] derived almost tight bounds on the VC-dimension in the setting of polygonal curves. At the heart of their approach lies the definition of a set of boolean functions (predicates) which can be used to determine if a query curve is contained in a ball of given radius around a center curve. Their proof of the VC-dimension bound uses the worst-case number of operations needed to determine the truth values of each predicate.

In this paper, we extend the known set of predicates to be able to decide the Hausdorff distance between polygonal regions with holes in the plane. We give an improved analysis for the VC-dimension that considers each predicate as a combination of sign values of polynomials. This approach does not use the computational complexity of the distance evaluation, but instead uses the underlying structure of the range space defined by a system of polynomials directly. Our techniques extend to other elastic distance measures, such as the Fréchet distance and the dynamic time warping distance (DTW).

* This work was funded by 390685813 (Germany's Excellence Strategy – EXC-2047/1: Hausdorff Center for Mathematics); 416767905; and the Deutsche Forschungsgemeinschaft (DFG, German ResearchFoundation) – 459420781 (FOR AlgoForGe)

40th European Workshop on Computational Geometry, Ioannina, Greece, March 13–15, 2024.
This is an extended abstract of a presentation given at EuroCG'24. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

1.1 Preliminaries

Let X be a set. We call a set $\mathcal{R}$ where any $r \in \mathcal{R}$ is of the form $r \subseteq X$ a ***range space*** with ***ground set*** X. We say a subset $A \subseteq X$ is ***shattered*** by $\mathcal{R}$ if for any $A' \subseteq A$ there exists an $r \in \mathcal{R}$ such that $A' = r \cap A$. The ***VC-dimension*** of $\mathcal{R}$ (denoted by $VCdim(\mathcal{R})$) is the maximal size of a set $A \subseteq X$ that is shattered by $\mathcal{R}$. We define the ball with radius Δ and center c under the distance measure d_ρ on a set X as $b_\rho(c, \Delta) = \{x \in X \mid d_\rho(x, c) \leq \Delta\}$. We study range spaces with ground set $(\mathbb{R}^d)^m$ of the form

$$\mathcal{R}_{\rho,k} = \{b_\rho(c, \Delta) \mid \Delta \in \mathbb{R}_+, c \in (\mathbb{R}^d)^k\}.$$

Let $\mathcal{R}$ be a range space with ground set X, and F be a class of real-valued functions defined on $\mathbb{R}^d \times X$. For $a \in \mathbb{R}$ let $\boldsymbol{sgn(a)} = 1$ if $a \geq 0$ and $\boldsymbol{sgn(a)} = 0$ if $a < 0$. We say that $\mathcal{R}$ is a ***t-combination*** of $sgn(F)$ if there is a boolean function $g : \{0,1\}^t \to \{0,1\}$ and functions $f_1, \ldots, f_t \in F$ such that for all $r \in \mathcal{R}$ there is a parameter vector $y \in \mathbb{R}^d$ such that $r = \{x \in X \mid g(sgn(f_1(y, x)), \ldots, sgn(f_t(y, x))) = 1\}$.

Central to our approach is the following well-known theorem for bounding the VC-dimension of such range spaces. The theorem can be proven by investigating the complexity of arrangements of zero sets of polynomials (see full version [4]).

▶ **Theorem 1.1** ([2], Theorem 8.3). *Let F be a class of functions mapping from $\mathbb{R}^d \times X$ to $\mathbb{R}$ so that, for all $x \in X$ and $f \in F$ the function $y \to f(y, x)$ is a polynomial on $\mathbb{R}^d$ of degree no more than l. Suppose that $\mathcal{R}$ is a t-combination of $sgn(F)$. Then we have*

$$VCdim(\mathcal{R}) \leq 2d \log_2(12tl).$$

Let $\|\cdot\|$ denote the standard Euclidean norm. Let $X, Y \subseteq \mathbb{R}^d$ for some $d \in \mathbb{N}$. The ***directed Hausdorff distance*** from X to Y is defined as $d_{\overrightarrow{H}}(X, Y) = \sup_{x \in X} \inf_{y \in Y} \|x - y\|$ and the ***Hausdorff distance*** between X and Y is defined as $d_H(X, Y) = \max\{d_{\overrightarrow{H}}(X, Y), d_{\overrightarrow{H}}(Y, X)\}$. If a set X consists of a single point $p \in \mathbb{R}^d$, we may write p instead of $\{p\}$ to simplify the notation, e.g. $d_H(p, Y)$ instead of $d_H(\{p\}, Y)$. Let $d, m \in \mathbb{N}$. A sequence of vertices $p_1, \ldots, p_m \in \mathbb{R}^d$ defines a ***polygonal curve*** P by connecting consecutive vertices to create the edges $\overline{p_1, p_2}, \ldots, \overline{p_{m-1}, p_m}$. We may think of P as an element of $\mathbb{X}_m^d := (\mathbb{R}^d)^m$ and write $P \in \mathbb{X}_m^d$. We may also think of P as a continuous function $P : [0, 1] \to \mathbb{R}^d$ by fixing m values $0 = t_1 < \ldots < t_m = 1$, and defining $P(t) = \lambda p_{i+1} + (1 - \lambda)p_i$ where $\lambda = \frac{t - t_i}{t_{i+1} - t_i}$ for $t_i \leq t \leq t_{i+1}$. We call P a ***closed curve*** if $p_1 = p_m$ and we call P ***self-intersecting*** if there exist $s \in [0, 1]$, $t \in (0, 1)$ with $s \neq t$ such that $P(s) = P(t)$. In the case that P is a closed curve in $\mathbb{R}^2$ which is not self-intersecting, we call the union of P with its interior a ***simple polygonal region*** S (without holes). We denote with ∂S the boundary of S, which is P. Given a simple polygonal region S_0 and a set of pairwise disjoint simple polygonal regions $S_1, \ldots, S_h$ in the interior of S_0, we also consider the set $S = S_0 - (S_1 \cup \cdots \cup S_h)$ a polygonal region and we call $S_1, \ldots, S_h$ the ***holes*** of S.

Let $s, t \in \mathbb{R}^d$. We denote with $\ell(\overline{st})$ the line supporting $\overline{st}$. We define the stadium centered at $\overline{st}$ with radius $\Delta \in \mathbb{R}_+$ as $D_\Delta(\overline{st}) = \{x \in \mathbb{R}^d \mid \exists p \in \overline{st}, \|p - x\| \leq \Delta\}$. Let $e_1, e_2 \in \mathbb{X}_2^d$ be two edges. We define the double stadium of the edges e_1 and e_2 with radius Δ as $D_{\Delta,2}(e_1, e_2) = D_\Delta(e_1) \cap D_\Delta(e_2)$.

Let X be a set of subsets (called sites) of $\mathbb{R}^2$. The ***Voronoi region*** $reg(A)$ consists of all points $p \in \mathbb{R}^2$ for which A is the closest among all sites in X, i.e. $reg(A) = \{p \in \mathbb{R}^2 \mid d_{\overrightarrow{H}}(p, A) < d_{\overrightarrow{H}}(p, U) \text{ for all } U \in X \setminus \{A\}\}$. The ***Voronoi diagram*** is $vd(X) = \mathbb{R}^2 \setminus \cup_{A \in X} reg(A)$. We call the set $bisec(A, B) = \{p \in \mathbb{R}^2 \mid d_{\overrightarrow{H}}(p, A) = d_{\overrightarrow{H}}(p, B)\}$ the ***bisector*** of A and B. The ***Voronoi edge*** of A, B is defined as $ve(A, B) = vd(X) \cap bisec(A, B)$ and the ***Voronoi vertices*** of A, B, C are defined as $vv(A, B, C) = vd(X) \cap bisec(A, B) \cap bisec(B, C)$.

		new	ref.	old
discrete polygonal curves	DTW	$O(dk^2 \log(m))$	Thm. 4, [4]	-
		$O(dkm \log(k))$	Thm. 4, [4]	
	Hausdorff	$O(dk \log(km))$	Thm. 2, [4]	$O(dk \log(dkm))$ [7]
	Fréchet	$O(dk \log(km))^{(*)}$	Thm. 3, [4]	
continuous polygonal curves	**Hausdorff**	$O(dk \log(km))$	**Thm. 3.5**	$O(d^2k^2 \log(dkm))$ [7]
	Fréchet	$O(dk \log(km))^{(*)}$	Thm. 27, [4]	
	weak Fréchet	$O(dk \log(km))^{(*)}$	Thm. 27, [4]	$O(d^2k \log(dkm))$ [7]
polygons $\mathbb{R}^2$	**Hausdorff**	$O(k \log(km))$	**Thm. 3.6**	-

Table 1 Overview of VC-dimension bounds. Results marked with $^{(*)}$ were independently obtained by Cheng and Huang [6].

2 Results

For the ***Hausdorff distance*** of ***polygonal regions*** (with holes) in the plane, we show that the VC-dimension of $\mathcal{R}_{d_H,k}$ is bounded by $O(k \log(km))$. For the ***Fréchet distance*** and the ***Hausdorff distance*** of ***polygonal curves***, in the discrete and the continuous case, we show that for the VC-dimension of $\mathcal{R}_{\rho,k}$ our techniques imply the same bound of $O(dk \log(km))$. Parallel and independent to our work, Cheng and Huang [6] obtained the same result for the Fréchet distance using very similar techniques. The bounds improve upon the upper bounds of [7] in all of the considered cases. An overview of our results with references to theorems and comparison to [7] and the independent results from [6] is given in Table 1. By the lower bound $\Omega(\max(dk \log(k), \log(dm)))$ for $d \geq 4$ in [7], the new bounds for polygonal curves are tight in each of the parameters k, m and d separately. For the ***Dynamic time warping distance***, we show a new bound of $O(\min(dk^2 \log(m), dkm \log(k)))$. The proofs for Fréchet and DTW are very similar to the ones used for the Hausdorff distance and we discuss them in the full version [4].

3 Analysis for the Hausdorff distance

To bound the VC-dimension of range spaces of the form $\mathcal{R}_{d_H,k}$, we define geometric predicates. The truth values of these predicates have to uniquely determine distance queries with d_H. We give predicates such that the directed Hausdorff distance query $d_{\overrightarrow{H}}(P, Q) \leq \Delta$ is determined by them. The other direction $d_{\overrightarrow{H}}(Q, P) \leq \Delta$ is analogous. We will show that our predicates can be viewed as combinations of simple predicates.

▶ **Definition 3.1.** Let F be a class of functions mapping from $\mathbb{R}^{dm} \times \mathbb{R}^{dk+1}$ to $\mathbb{R}$ so that, for all $f \in F$ the function $(x, y) \to f(x, y)$ is a polynomial of constant degree. Let $\mathcal{P}$ be a function from $\mathbb{R}^{dm} \times \mathbb{R}^{dk+1}$ to $\{0, 1\}$. We say that the predicate $\mathcal{P}$ is ***simple*** if $\mathcal{P}$ is a t-combination of $sgn(F)$ with $t \in O(1)$.

In our proof of the VC-dimension bounds we will use the following corollary to Theorem 1.1.

▶ **Corollary 3.2.** *Suppose that for a given d_ρ there exists a polynomial $p(k, m)$ such that for any $k, m \in \mathbb{N}$ the space $\mathcal{R}_{\rho,k}$ with ground set $\mathbb{R}^{dm}$ is a t-combination of simple predicates with $t = p(k, m)$. Then $VCdim(\mathcal{R}_{\rho,k})$ is in $O(dk \log(km))$.*

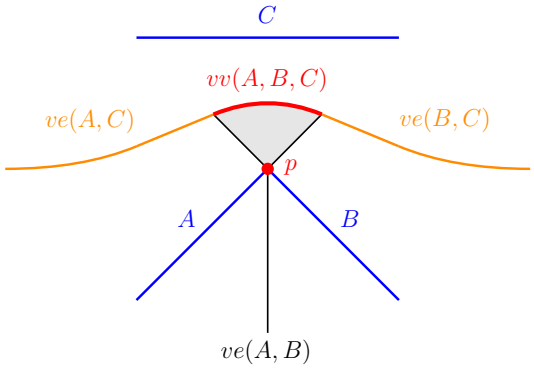


Figure 1 Degenerate case: $vv(A, B, C)$ consist of a whole arc and $ve(A, B)$ contains a region.

Let $P \in \mathbb{X}_m^d$ with vertices $p_1, \ldots, p_m$ and $Q \in \mathbb{X}_k^d$ with vertices $q_1, \ldots, q_k$ be two polygonal curves. Let further $\Delta \in \mathbb{R}_+$. By [7] the directed Hausdorff distance query $d_{\overrightarrow{H}}(P, Q) \leq \Delta$ is uniquely determined by the following predicates.

- $(\mathcal{P}_1)$: Given an edge e_1 of Q and a vertex p of P, this predicate returns true iff there exists a point q on e_1, such that $\|q - p\| \leq \Delta$.
- $(\mathcal{P}_2)$: Given an edge of P, $\overline{p_j p_{j+1}}$, and two edges e_1, e_2 of Q, this predicate is equal to $\ell(\overline{p_j p_{j+1}}) \cap D_{\Delta,2}(e_1, e_2) \neq \emptyset$.

Examples for the predicates $\mathcal{P}_1$ and $\mathcal{P}_2$ are depicted in Figure 3 (they are also used for polygonal regions).

▶ **Lemma 3.3** (Lemma 7.1, [7]). *For any two polygonal curves P, Q, given the truth values of all predicates of the type $\mathcal{P}_1, \mathcal{P}_2$ one can determine whether $d_{\overrightarrow{H}}(P, Q) \leq \Delta$.*

In the case of polygonal regions that may contain holes, we define some of the predicates based on the Voronoi vertices of the edges of the boundary of the polygonal region. Since degenerate situations can occur if Voronoi sites intersect in a point p (see Figure 1), we restrict the predicates to the subset of the Voronoi vertices that are relevant to our analysis.

▶ **Definition 3.4.** Let $a = \overline{a_1 a_2}, b = \overline{b_1 b_2}$ and $c = \overline{c_1 c_2}$ be edges of a polygonal region that may contain holes. Consider their vertices and supporting lines $A = \{\{a_1\}, \{a_2\}, \ell(a)\}$, $B = \{\{b_1\}, \{b_2\}, \ell(b)\}$ and $C = \{\{c_1\}, \{c_2\}, \ell(c)\}$. Let $X \in A$, $Y \in B$ and $Z \in C$. If either X, Y or Z is a subset of one of the others, we set $V_0(X, Y, Z) = \emptyset$ otherwise let

$$V_0(X, Y, Z) = \{v \in \mathbb{R}^2 \mid d_{\overrightarrow{H}}(v, X) = d_{\overrightarrow{H}}(v, Y) = d_{\overrightarrow{H}}(v, Z)\}$$

be the set of points with the same distance to all sets X, Y and Z. The ***set of Voronoi-vertex-candidates*** $V(a, b, c)$ of the line segments a, b and c is defined as

$$V(a, b, c) = \bigcup_{X \in A, Y \in B, Z \in C} V_0(X, Y, Z).$$

By only considering Voronoi-vertex-candidates, we restrict ourselves to a finite set of vertices that includes all relevant Voronoi vertices and does not include the degenerate cases. Let P and Q be two polygonal regions that may contain holes. Let further $\Delta \in \mathbb{R}_+$. The distance $d_{\overrightarrow{H}}(p, Q)$ for points $p \in P$ can be maximized at points in the interior of P or at points at the boundary of P (see Figure 2 for the two cases). Since these cases are different to analyze, we split the query into two general predicates.

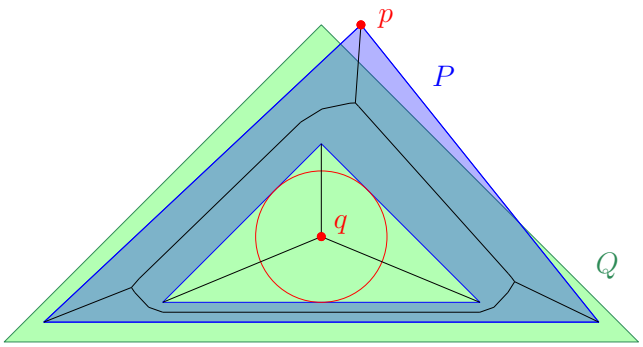


Figure 2 Illustration of the two cases: The point p on the boundary of P maximizes $d_{\overrightarrow{H}}(p, Q)$. The point q in the interior of Q that is a Voronoi vertex of the edges of P maximizes $d_{\overrightarrow{H}}(q, P)$.

- $(\mathcal{B})$ (Boundary): This predicate returns true if and only if $d_{\overrightarrow{H}}(\partial P, Q) \leq \Delta$.
- $(\mathcal{I})$ (Interior): This predicate returns true if $d_{\overrightarrow{H}}(P, Q) \leq \Delta$. This predicate returns false if $d_{\overrightarrow{H}}(P, Q) > d_{\overrightarrow{H}}(\partial P, Q)$ and $d_{\overrightarrow{H}}(P, Q) > \Delta$.

Note that it is not defined what $(\mathcal{I})$ returns if $d_{\overrightarrow{H}}(P, Q) = d_{\overrightarrow{H}}(\partial P, Q)$ and $d_{\overrightarrow{H}}(P, Q) > \Delta$. This does not matter, since the correctness of $d_{\overrightarrow{H}}(P, Q) \leq \Delta$ is still equivalent to both $(\mathcal{B})$ and $(\mathcal{I})$ being true.

Since $(\mathcal{B})$ and $(\mathcal{I})$ are very general, we define more detailed predicates that can be used to determine feasible truth values of $(\mathcal{B})$ and $(\mathcal{I})$. To determine $(\mathcal{B})$, we need the following predicates in combination with $\mathcal{P}_1$ and $\mathcal{P}_2$ (defined earlier for curves):

- $(\mathcal{P}_3)$: Given a vertex p of P, this predicate returns true if and only if $p \in Q$.
- $(\mathcal{P}_4)$: Given an edge e_1 of P and an edge e_2 of Q, this predicate is equal to $e_1 \cap e_2 \neq \emptyset$.
- $(\mathcal{P}_5)$: Given a directed edge e_1 of P and two edges e_2 and e_3 of Q, this predicate is true if and only if $e_1 \cap e_2 \neq \emptyset$, $e_1 \cap e_3 \neq \emptyset$ and e_1 intersects e_2 before or at the same point that it intersects e_3.
- $(\mathcal{P}_6)$: Given a directed edge e_1 of P and two edges e_2 and e_3 of Q, this predicate is true if and only if $e_1 \cap e_2 \neq \emptyset$ and if there exists a point b on e_3 such that $\|a - b\| \leq \Delta$ where a is the first intersection point of $e_1 \cap e_2$.
- $(\mathcal{P}_7)$: Given a directed edge e_1 of P and two edges e_2 and e_3 of Q, this predicate is true if and only if $e_1 \cap e_2 \neq \emptyset$ and if there exists a point b on e_3 such that $\|a - b\| \leq \Delta$ where a is the last intersection point of $e_1 \cap e_2$.

Using Voronoi-vertex-candidates, we define the detailed predicates for determining $(\mathcal{I})$:

- $(\mathcal{P}_8)$: Given 4 edges e_1, e_2, e_3, e_4 of Q and a point v from the set of Voronoi-vertex-candidates $V(e_1, e_2, e_3)$, this predicate returns true if and only if there exists a point $p \in e_4$, such that $\|v - p\| \leq \Delta$.
- $(\mathcal{P}_9)$: Given 3 edges e_1, e_2, e_3 of Q and a point v from the set of Voronoi-vertex-candidates $V(e_1, e_2, e_3)$, this predicate returns true if and only if $v \in Q$.
- $(\mathcal{P}_{10})$: Given 3 edges e_1, e_2, e_3 of Q and a point v from the set of Voronoi-vertex-candidates $V(e_1, e_2, e_3)$, this predicate returns true if and only if $v \in P$.

Examples for the predicates $\mathcal{P}_3, \ldots \mathcal{P}_{10}$ are depicted in Figure 3.

In the full version [4], we show that given the truth values of all these predicates one can determine a feasible truth value for predicates of the type $(\mathcal{B})$ and $(\mathcal{I})$. The proof for $(\mathcal{B})$ is very similiar to the proof of Lemma 7.1 in [7] for polygonal curves. In the proof for $(\mathcal{I})$, we show by contradiction that if the distance is realized only in the interior and at no Voronoi

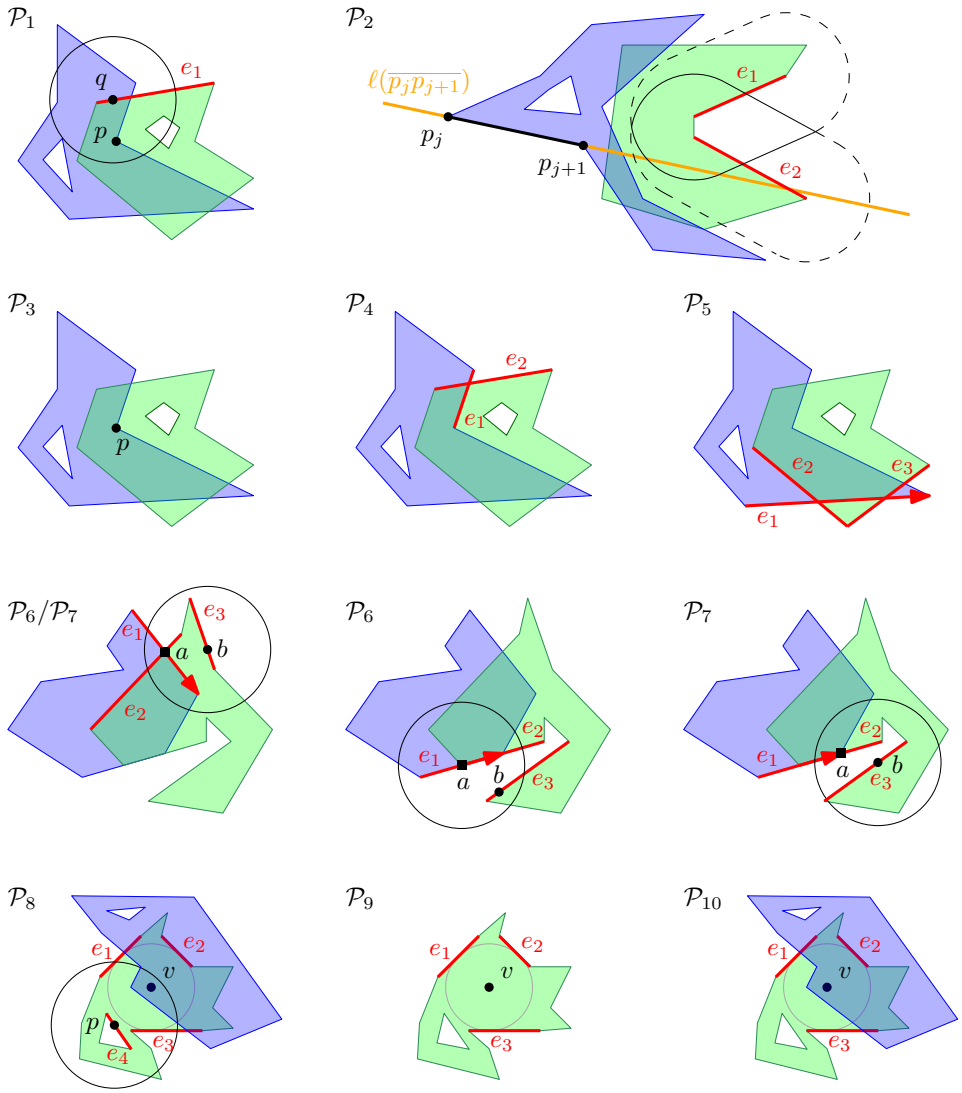


Figure 3 Illustration of the predicates $\mathcal{P}_1, \ldots, \mathcal{P}_{10}$: In all depicted cases the predicates are true.

vertex, then you can always increase the distance of Q to a point p in the interior of P by moving p to a Voronoi vertex or to the boundary.

Furthermore, we give a detailed proof in the full version [4], that all predicates $\mathcal{P}_1, \ldots, \mathcal{P}_{10}$ can be determined by a polynomial number of simple predicates. In that technical proof, we explicitly determine for each predicate, how it can be divided into sign values of polynomials. Corollary 3.2 then implies the following bounds on the VC-dimension.

▶ **Theorem 3.5.** *Let $\mathcal{R}_{d_H,k}$ be the range space of balls centered at polygonal curves in $\mathbb{X}_k^d$ with ground set $\mathbb{X}_m^d$. $VCdim(\mathcal{R}_{d_H,k})$ is in $O(dk\log(km))$.*

▶ **Theorem 3.6.** *Let $\mathcal{R}_{d_H,k}$ be the range space of balls centered at polygonal regions that may contain holes in $(\mathbb{R}^{2+1})^k$ with ground set $(\mathbb{R}^{2+1})^m$. The third dimension encodes a label that associates each vertex with its boundary component. $VCdim(\mathcal{R}_{d_H,k})$ is in $O(k\log(km))$.*

References

1 Hugo Akitaya, Frederik Brüning, Erin Chambers, and Anne Driemel. Subtrajectory Clustering: Finding Set Covers for Set Systems of Subcurves. *Computing in Geometry and Topology*, 2(1):1:1–1:48, Feb. 2023.

2 Martin Anthony and Peter L. Bartlett. *Neural Network Learning: Theoretical Foundations*. Cambridge University Press, 1999.

3 Frederik Brüning, Jacobus Conradi, and Anne Driemel. Faster Approximate Covering of Subcurves Under the Fréchet Distance. In *30th Annual European Symposium on Algorithms (ESA 2022)*, pages 28:1–28:16, 2022.

4 Frederik Brüning and Anne Driemel. Simplified and Improved Bounds on the VC-Dimension for Elastic Distance Measures. *CoRR (arXiv), arXiv.2308.05998*, 2023.

5 Maike Buchin and Dennis Rohde. Coresets for (k,l)-Median Clustering Under the Fréchet Distance. In *Algorithms and Discrete Applied Mathematics - 8th International Conference, CALDAM 2022, Puducherry, India, February 10-12, 2022, Proceedings*, pages 167–180, 2022.

6 Siu-Wing Cheng and Haoqiang Huang. Solving Fréchet Distance Problems by Algebraic Geometric Methods. In *Proceedings of the 2024 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 4502–4513. SIAM, 2024.

7 Anne Driemel, André Nusser, Jeff M. Phillips, and Ioannis Psarros. The VC Dimension of Metric Balls under Fréchet and Hausdorff Distances. *Discrete & Computational Geometry*, 66(4):1351–1381, 2021.

8 Dan Feldman. Introduction to Core-sets: an Updated Survey. *ArXiv*, abs/2011.09384, 2020.

9 Dan Feldman and Michael Langberg. A Unified Framework for Approximating and Clustering Data. In *Proceedings of the Forty-Third Annual ACM Symposium on Theory of Computing*, STOC '11, page 569–578, 2011.

10 Sariel Har-Peled and Micha Sharir. Relative (p, ϵ)-Approximations in Geometry. *Discrete & Computational Geometry*, 45(3):462–496, 2011.

11 David Haussler and Emo Welzl. Epsilon-nets and simplex range queries. *Discrete & Computational Geometry*, 2(2):127–151, 1987.

12 Sarang Joshi, Raj Varma Kommaraji, Jeff M. Phillips, and Suresh Venkatasubramanian. Comparing Distributions and Shapes Using the Kernel Distance. In *Proceedings of the Twenty-Seventh Annual Symposium on Computational Geometry*, SoCG '11, page 47–56, 2011.

13 Marek Karpinski and Angus Macintyre. Polynomial Bounds for VC Dimension of Sigmoidal Neural Networks. In *Proceedings of the Twenty-Seventh Annual ACM Symposium on Theory of Computing*, STOC '95, page 200–208, 1995.

14 Vladimir N. Vapnik. *The Nature of Statistical Learning Theory*. Springer New York, NY, 2000.

15 Vladimir N. Vapnik and Alexey Ya. Chervonenkis. On the Uniform Convergence of Relative Frequencies of Events to Their Probabilities. *Theory of Probability & Its Applications*, 16(2):264–280, 1971.

Optimal In-Place Compaction of Sliding Cubes

Irina Kostitsyna[1,2], Tim Ophelders[*2,3], Irene Parada[4], Tom Peters[2], Willem Sonke[2], and Bettina Speckmann[2]

1 **KBR at NASA Ames Research Center, USA**
irina.kostitsyna@nasa.gov
2 **TU Eindhoven, The Netherlands**
[t.peters1|w.m.sonke|b.speckmann]@tue.nl
3 **Utrecht University, The Netherlands**
t.a.e.ophelders@uu.nl
4 **Universitat Politècnica de Catalunya, Spain**
irene.parada@upc.edu

Abstract

The sliding cubes model is a well-established theoretical framework that supports the analysis of reconfiguration algorithms for modular robots consisting of face-connected cubes. As is common in the literature, we focus on reconfiguration via an intermediate canonical shape. Specifically, we present an in-place algorithm that reconfigures any n-cube configuration into a compact canonical shape using a number of moves proportional to the sum of coordinates of the input cubes. This result is asymptotically optimal and strictly improves on all prior work. Furthermore, our algorithm directly extends to two dimensions and any dimension higher than three.

Related Version A full version of the paper is available at arxiv.org/abs/2312.15096.

1 Introduction

Modular robots consist of a large number of comparatively simple robotic units. These units can attach and detach to and from each other, move relative to each other, and in this way form different shapes or configurations. This shape-shifting ability allows modular robots to robustly adapt to previously unknown environments and tasks. In this paper, we study the *sliding cube model*, a well-established theoretical framework that supports the analysis of reconfiguration algorithms for modular robots consisting of face-connected cubes.

Almost 20 years ago, Dumitrescu and Pach [5] showed that the sliding cube model in 2D (or *sliding square model*) is universally reconfigurable. More precisely, they presented an algorithm that transforms any two given configurations with n squares into each other in $O(n^2)$ moves. This algorithm transforms any given configuration into a canonical shape (a horizontal line) and then reverts the procedure to reach the final configuration. Recently, Akitaya et al. [3] presented Gather&Compact: an input-sensitive in-place algorithm which uses $O(Pn)$ moves, where P is the maximum among the perimeters of the bounding boxes of the initial and final configurations. The authors also show that minimizing the number of moves required to reconfigure is NP-hard.

Until recently, the most efficient algorithm for the reconfiguration problem in 3D was the algorithm by Abel and Kominers [1], which uses $O(n^3)$ moves to transform any n-cube configuration into any other n-cube configuration. As is common in the literature, this algorithm reconfigures the input into an intermediate canonical shape. Stock et al. [7]

* T. Ophelders is partially supported by the Dutch Research Council (NWO) under project no. VI.Veni.212.260.

40th European Workshop on Computational Geometry, Ioannina, Greece, March 13–15, 2024.
This is an extended abstract of a presentation given at EuroCG'24. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

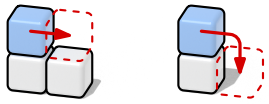


Figure 1 Moves in the sliding cube model: **(a)** slide and **(b)** convex transition.

recently announced a worst-case bound of $O(n^2)$ moves for the Abel and Kominers algorithm. Furthermore, their paper presents an in-place reconfiguration algorithm, which runs in time proportional to a measure of the size of the bounding box times the number of cubes. Specifically, their algorithm requires $O(n(wd+h))$ moves in the worst-case, where w, d, and h are the width, depth, and height of the bounding box, respectively.

In this paper we present an in-place algorithm that reconfigures any n-cube configuration into a compact canonical shape using a number of moves proportional to the sum of coordinates of the input cubes. This result is asymptotically optimal. Furthermore, our algorithm directly extends to squares in two dimensions and to hypercube reconfiguration in dimensions higher than three.

2 Preliminaries

A *configuration* $\mathcal{C}$ is a subset of coordinates in the three-dimensional grid. The elements of $\mathcal{C}$ are called *cubes*. We call two cubes *adjacent* if they lie at unit distance. For a configuration $\mathcal{C}$, denote by $G_{\mathcal{C}}$ the graph with vertex set $\mathcal{C}$, whose edges connect all adjacent cubes. We say a *cell* is a vertex of $G_{\mathbb{Z}^3}$ which is not occupied by a cube in $\mathcal{C}$. We always require a configuration to remain *connected*, that is, $G_{\mathcal{C}}$ must be connected. For ease of exposition we assume $\mathcal{C}$ consists of at least two cubes. We call a configuration $\mathcal{C}$ *nonnegative* if $\mathcal{C} \subseteq \mathbb{N}^3$. Cubes can perform *moves*. A move is an operation that replaces a single cube $c \in \mathcal{C}$ by another cube $c' \notin \mathcal{C}$. Moves come in two types: *slides* and *convex transitions* (see Figure 1). In both cases, we consider a 4-cycle γ in $G_{\mathbb{Z}^3}$. For slides, exactly three cubes of γ are in $\mathcal{C}$; c' is the cell of γ not in $\mathcal{C}$, and c is adjacent to c'. For convex transitions, γ has exactly two adjacent cubes in $\mathcal{C}$; c is one of these two cubes, and c' is the vertex of γ not adjacent to c. The slide or convex transition is a *move* if and only if $\mathcal{C} \setminus \{c\}$ is connected.

Let $\mathcal{C}$ be a nonnegative configuration. Call a cube $c = (x, y, z)$ *finished* if the cuboid spanned by the origin and c is completely in $\mathcal{C}$, that is, if $\{0, \ldots, x\} \times \{0, \ldots, y\} \times \{0, \ldots, z\} \subseteq \mathcal{C}$. We call $\mathcal{C}$ *finished* if all cubes in $\mathcal{C}$ are finished. The *compaction problem* starts with an arbitrary connected configuration $\mathcal{C}$ and is solved when all cubes are finished.

Most of the algorithm works on vertical contiguous strips of cubes in $\mathcal{C}$ called subpillars. More precisely, a *subpillar* is a subset of $\mathcal{C}$ of the form $\{x\} \times \{y\} \times \{z_b, \ldots, z_t\}$. In the remainder of this paper, we denote this subpillar by $\langle x, y, z_b \,..\, z_t \rangle$. The cube (x, y, z_t) is called the *head*, and the remainder $\langle x, y, z_b \,..\, z_t - 1 \rangle$ is called the *support*. A *pillar* is a maximal subpillar, that is, a subpillar that is not contained in any other subpillar. Note that there can be multiple pillars with the same x- and y- coordinate above each other, as long as there is a gap between them. Two sets S and S' of cubes are *adjacent* if S contains a cube adjacent to a cube in S'. All omitted proofs can be found in the full version.

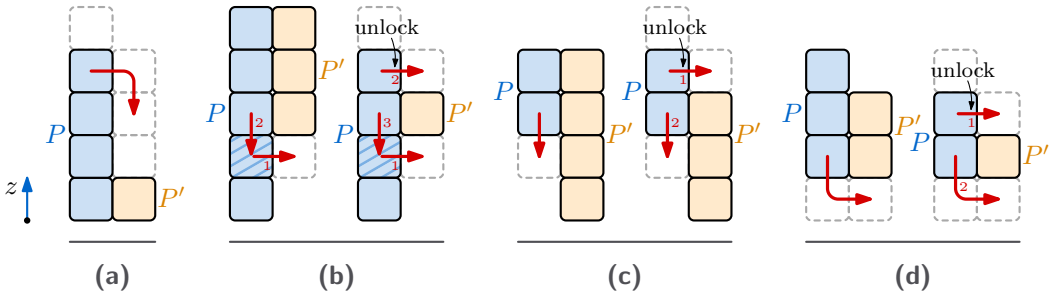


Figure 2 Examples of operations **(a–d)**; hatched cubes are non-cut and dashed outlines indicate cells that must be empty. Each case admits a move sequence that reduces $Z_{\mathcal{C}}$.

3 Algorithm

For a set of cubes $S \subseteq \mathcal{C}$, let (X_S, Y_S, Z_S) denote its *coordinate vector sum* $\sum_{(x,y,z)\in S}(x, y, z)$. Let $\mathcal{C}_{>0}$ be the subset of cubes $(x, y, z) \in \mathcal{C}$ for which $z > 0$, and $\mathcal{C}_0$ be the subset of cubes for which $z = 0$. Let the *potential* of a cube $c = (c_x, c_y, c_z)$ be $\Pi_c = w_c(c_x + 2c_y + 4c_z)$, where the *weight* w_c depends on the coordinates of c in the following way. If $c_z > 1$, then $w_c = 5$; if $c_z = 1$, then $w_c = 4$. If $c_z = 0$, then w_c depends on c_y. If $c_y > 1$, then $w_c = 3$; if $c_y = 1$, then $w_c = 2$; lastly, if $c_z = c_y = 0$, then $w_c = 1$. We aim to minimize the *potential function* $\Pi_{\mathcal{C}} = \sum_{c\in\mathcal{C}} \Pi_c$. From now on, let $\mathcal{C}$ be an unfinished nonnegative configuration. We call a sequence of m moves *safe* if the result is a nonnegative instance $\mathcal{C}'$, such that $\Pi_{\mathcal{C}'} < \Pi_{\mathcal{C}}$ and $m = O(\Pi_{\mathcal{C}} - \Pi_{\mathcal{C}'})$. This means that the sequence of moves reduces the potential by at least some constant fraction of m by going from $\mathcal{C}$ to $\mathcal{C}'$. We show that if $\mathcal{C}$ is unfinished, it always admits a safe move sequence.

The main idea is as follows. For a configuration $\mathcal{C}$, whenever possible, we try to reduce $Z_{\mathcal{C}}$. If that is not possible, the configuration must admit a *pillar shove* where a complete pillar is moved to a different x- and y-coordinate. By reducing either the z-coordinate of cubes, or the x- or y-coordinate, we guarantee that eventually every cube becomes finished.

Local Z reduction. Let $P = \langle x, y, z_b .. z_t\rangle$ be a subpillar of $\mathcal{C}$. We refer to the four coordinates $\{(x-1, y), (x+1, y), (x, y-1), (x, y+1)\}$ as the *sides* of P. On each side, P may have one or more adjacent pillars. We order these by their z-coordinates; as such, we may refer to the top- or bottommost adjacent pillar on a side of P. We say that a set of cubes $S \subseteq \mathcal{C}$ is *non-cut* if $G_{\mathcal{C}\setminus S}$ is connected or empty.

Let $P = \langle x, y, z_b .. z_t\rangle$ be a non-cut subpillar, and let $P' = \langle x', y', z'_b .. z'_t\rangle$ be a pillar adjacent to P. We define a set of operations of at most three moves within P which locally reduce $Z_{\mathcal{C}}$ (see Figure 2). Because P is non-cut, $\mathcal{C} \setminus P$ is connected. Therefore, if cubes of P move in such a way that each component (of cubes originating from P) remains adjacent to a cube of $\mathcal{C} \setminus P$, then the result of that operation is a valid configuration. These different operations **(a–d)** can be seen in Figure 2. For a complete definition of these operations **(a–d)**, see the full version of this paper.

Pillar shoves. Next, we consider a longer move sequence **(e)** that still involves a single subpillar. We call this operation a *pillar shove*, which takes as parameters a subpillar $P = \langle x, y, z_b .. z_t\rangle$ and a side (x', y') of P. The result of the pillar shove is the set of cubes

$$\text{shove}(\mathcal{C}, P, (x', y')) \;:=\; (\mathcal{C} \setminus P) \;\cup\; \langle x', y', z_b .. z_t - 1\rangle \;\cup\; \{(x, y, z_b)\},$$

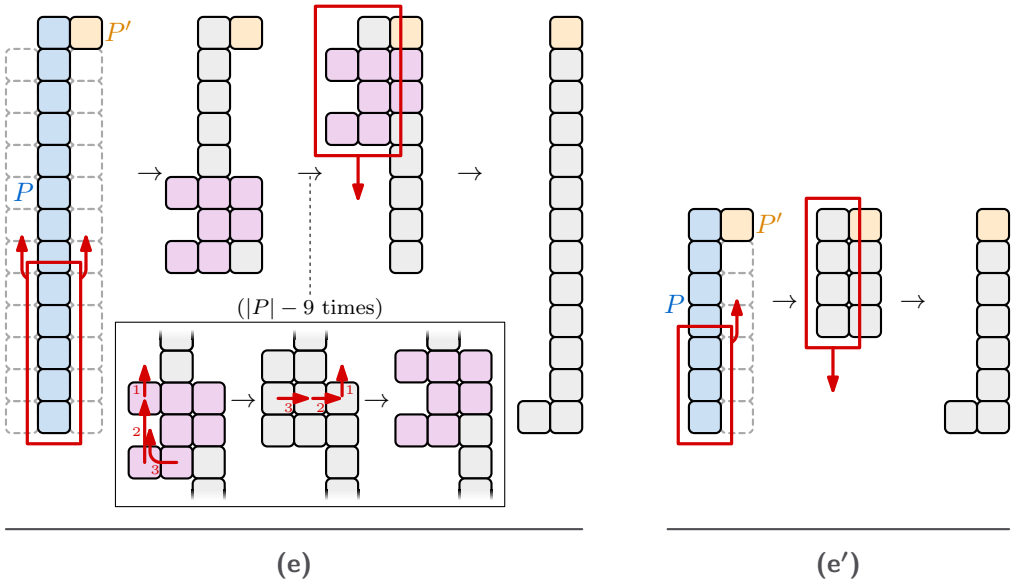


Figure 3 Examples of pillar shoves for a long pillar **(e)** and a short pillar **(e′)**.

in which the support is effectively shifted to the side (x', y'), and the head is effectively moved from (x, y, z_t) to (x, y, z_b). Although shove$(\mathcal{C}, \langle x, y, z_b \,..\, z_t\rangle, (x', y'))$ is well-defined, it is not necessarily a connected configuration, let alone safely reachable from $\mathcal{C}$.

Let $P = \langle x, y, z_b \,..\, z_t\rangle$ be a non-cut subpillar, and assume that on at least two sides (x', y') and (x'', y'') of P, no cube except possibly the head (x, y, z_t) has an adjacent cube. Moreover, assume that $(x', y', z_t) \in \mathcal{C}$. Then the pillar shove can be done without collisions and while keeping connectivity (see Figure 3). There are two cases: one where P has at least 9 cubes, in which case we take $O(|P|)$ moves (left side of Figure 3), and the case where P has fewer than 9 cubes takes $O(1)$ moves and does not require the existence of the second side (x'', y'') (right side of Figure 3). A pillar shove reduces $Z_{\mathcal{C}}$ by $z_t - z_b$ and takes $O(z_t - z_b)$ moves, so it is safe. For a complete definition of **(e)**, see the full version.

Lastly, we define an operation **(f)** that performs any move of $\mathcal{C}$ that moves a cube of $\mathcal{C}_{>0}$, reduces the potential, and results in a nonnegative instance. In summary, the moves **(a–f)** are designed to reduce the z-coordinate of a cube. If this is not directly possible, the pillar shove moves a complete pillar such that the head of that pillar can still reduce its z-coordinate.

Low and high components. Suppose that **(a–f)** do not apply. Let $\mathcal{LH}_{\mathcal{C}}$ be the bipartite graph obtained from $G_{\mathcal{C}}$ by contracting the components of $G_{\mathcal{C}_0}$ and $G_{\mathcal{C}_{>0}}$ to a single vertex (see Figure 4). We call $\mathcal{LH}_{\mathcal{C}}$ the *low-high graph* of $\mathcal{C}$, and we call the vertices of $\mathcal{LH}_{\mathcal{C}}$ that correspond to components of $G_{\mathcal{C}_0}$ and $G_{\mathcal{C}_{>0}}$ *low* and *high* components, respectively. The full version of this paper proves the following lemma.

▶ **Lemma 3.1.** *Assume $\mathcal{C}$ does not admit any operation of type* **(a–f)**. *If H is a high component such that $\mathcal{C} \setminus H$ is connected, then every pillar of H is part of a pillar of $\mathcal{C}$ starting at $z = 0$, H consists entirely of finished cubes, and H contains $(0, 0, 1)$.*

We pick a vertex R of $\mathcal{LH}_{\mathcal{C}}$ that we call the *root* of $\mathcal{LH}_{\mathcal{C}}$. If $(0, 0, 0) \in \mathcal{C}$, pick R to be the low component that contains $(0, 0, 0)$. Otherwise, pick R to be an arbitrary low component.

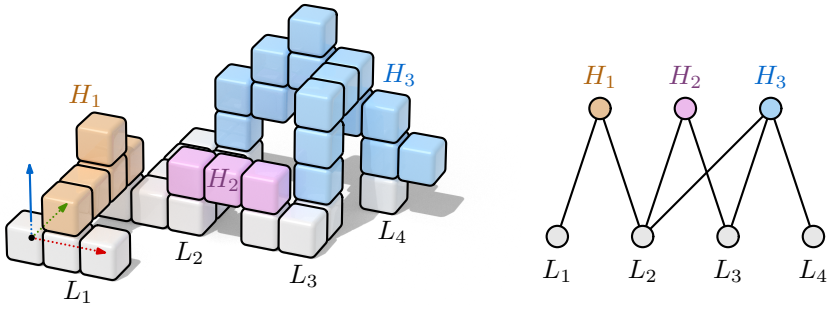


Figure 4 An example configuration $\mathcal{C}$ and its low-high graph $\mathcal{LH}_\mathcal{C}$. This configuration does admit moves of type **(a–f)**

We call a low component L *clear* if $\mathcal{C} \setminus L$ is connected, $L \neq R$, and L is connected to a non-cut pillar P in $\mathcal{C} \setminus L$. We call such a pillar P a *clearing pillar*. The full version proves that such a clear low component always exists, unless $\mathcal{LH}_\mathcal{C}$ contains only the root and possibly a single high component.

We now define operation **(g)**: pick a clear low component L, and perform any move of $\mathcal{C}$ that moves a cube of L, reduces the potential, and results in a nonnegative instance. This does the same as operation **(f)**, but now on $\mathcal{C}_0$ instead of $\mathcal{C}_{>0}$. When operations of type **(g)** are executed, one of three special events could occur:

(1) The component connects to a different low component, merging them.
(2) The component connects to the root, and becomes part of the root.
(3) The component reaches the origin (at which point it becomes the root).

If none of the operations **(a–g)** are available then L is too small to reach the origin. We would like to move the clearing pillar P and do a pillar shove. However, it could be that there are cubes around P, or that moving P would disconnect the low component. For this specific case we define two last operations. Operation **(h)** applies when the bottom of P is completely surrounded. It moves the cube at the bottom of P via $z = -1$ to a positive cell that is closer to the origin. The second operation **(i)** applies when the bottom of P is not sufficiently connected to L and moving it would disconnect L. In this case, we gather cubes from the low component towards P to connect it to the low component, and perform a pillar shove on it, see Figure 5.

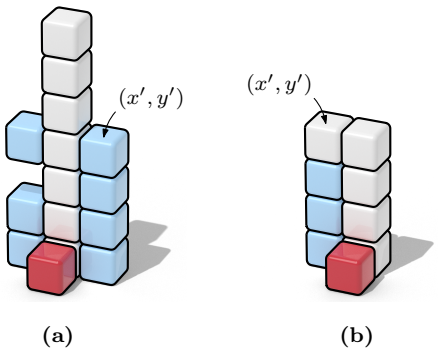


Figure 5 The start configuration for a pillar shove for a clearing pillar. The white pillar is the clearing pillar. The red cube is part of L. The blue cubes are required and need to be gathered. (a): clearing pillar of height at least 5. (b): The configuration for a pillar shove of height at most 4.

This last operation moves cubes that are not part of the clearing pillar $P = \langle x, y, z_b .. z_t \rangle$. However, the move is still safe. Recall that the *potential* of a cube $c = (c_x, c_y, c_z)$ is $\Pi_c = w_c(c_x + 2c_y + 4c_z)$, where w_c is the *weight* of c. The potential of the complete configuration is the sum of potential of the individual cubes. Because the low component cannot reach the origin, its size is at most $O(x + y)$. Therefore, moving a constant number of cubes towards this pillar also only takes $O(x + y)$ moves. This is charged to the head of P: since it goes from $z > 1$ to $z = 1$, or from $z = 1$ to $z = 0$, its weight decreases by 1, paying for the gathering of these cubes.

This algorithm terminates when no clear low component (and hence only the root low component) remains. We are left with two cases. Either no high component remains, or there is at most one high component, which consists of entirely finished cubes.

All of the moves **(a–i)** not only work in 3D, they also work in 2D when instead of prioritizing reducing the z-coordinate, we prioritize reducing the y-coordinate. Moreover, these moves never move the origin. Therefore, we can now run the exact same moves on the bottom layer in 2D, until the root component is finished. If there is still a high component, it stays connected via the origin. We end up with a finished configuration.

Recall that a sequence of m moves on an instance $\mathcal{C}$ is *safe* if the result is a nonnegative instance $\mathcal{C}'$, such that $\Pi_{\mathcal{C}'} < \Pi_{\mathcal{C}}$ and $m = O(\Pi_{\mathcal{C}} - \Pi_{\mathcal{C}'})$. Since each of our operations is safe, the total number of moves our algorithm performs is $O(\Pi_{\mathcal{C}}) = O(X_{\mathcal{C}} + Y_{\mathcal{C}} + Z_{\mathcal{C}})$.

4 Conclusion

We presented an in-place algorithm that reconfigures any configuration of cubes into a compact canonical shape using a number of moves proportional to the sum of coordinates of the input cubes. This result is asymptotically optimal. However, just as many other algorithms in the literature, our bounds are amortized in the sense that we make use of a number of dedicated cubes which help other cubes move by establishing the necessary connectivity in their neighborhood. This is in particular the case with our pillar shoves, that need some additional cubes to gather at the pillar, to then move up and down the pillar to facilitate moves. These extra moves are charged to one cube in the pillar reducing its coordinates. In the literature such cubes are referred to as *helpers*, *seeds*, or even *musketeers* [2, 4, 6, 7]. Hence, an interesting question is whether it is possible to arrive at sum-of-coordinates bounds without amortization?

References

1 Zachary Abel and Scott Duke Kominers. Universal reconfiguration of (hyper-)cubic robots. *ArXiv e-Prints*, 2011. URL: https://arxiv.org/abs/0802.3414v3.

2 Hugo A. Akitaya, Esther M. Arkin, Mirela Damian, Erik D. Demaine, Vida Dujmovic, Robin Y. Flatland, Matias Korman, Belén Palop, Irene Parada, André van Renssen, and Vera Sacristán. Universal reconfiguration of facet-connected modular robots by pivots: The O(1) musketeers. *Algorithmica*, 83(5):1316–1351, 2021. doi:10.1007/S00453-020-00784-6.

3 Hugo A. Akitaya, Erik D. Demaine, Matias Korman, Irina Kostitsyna, Irene Parada, Willem Sonke, Bettina Speckmann, Ryuhei Uehara, and Jules Wulms. Compacting squares: Input-sensitive in-place reconfiguration of sliding squares. In *Proc. 18th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT 2022)*, volume 227 of *LIPIcs*, pages 4:1–4:19, 2022. doi:10.4230/LIPICS.SWAT.2022.4.

4 Matthew Connor and Othon Michail. Centralised connectivity-preserving transformations by rotation: 3 musketeers for all orthogonal convex shapes. In *Proc. 18th International Symposium on Algorithmics of Wireless Networks (ALGOSENSORS 2022)*, volume 13707 of *LNCS*, pages 60–76. Springer, 2022. doi:10.1007/978-3-031-22050-0_5.

5 Adrian Dumitrescu and János Pach. Pushing squares around. *Graphs and Combinatorics*, 22:37–50, 2006. doi:10.1007/s00373-005-0640-1.

6 Othon Michail, George Skretas, and Paul G. Spirakis. On the transformation capability of feasible mechanisms for programmable matter. In *Proc. 44th International Colloquium on Automata, Languages, and Programming (ICALP 2017)*, volume 80 of *LIPIcs*, pages 136:1–136:15, 2017. doi:10.4230/LIPICS.ICALP.2017.136.

7 Frederick Stock, Hugo Akitaya, Matias Korman, Scott Kominers, and Zachary Abel. A universal in-place reconfiguration algorithm for sliding cube-shaped robots in quadratic time. In *Proc. 40th International Symposium on Computational Geometry (SoCG 2024)*, 2024. To appear.

Sibson's formula for higher order Voronoi diagrams

Mercè Claverol[1], Andrea de las Heras-Parrilla[1], Clemens Huemer[1], and Dolores Lara[2]

1 **Universitat Politècnica de Catalunya**
merce.claverol@upc.edu, andrea.de.las.heras@upc.edu, clemens.huemer@upc.edu
2 **Centro de Investigación y de Estudios Avanzados**
dlara@cs.cinvestav.mx

Abstract

Let S be a set of n points in general position in $\mathbb{R}^d$. The order-k Voronoi diagram of S, $V_k(S)$, is a subdivision of $\mathbb{R}^d$ into cells whose points have the same k nearest points of S. Sibson, in his seminal paper from 1980 (A vector identity for the Dirichlet tessellation), gives a formula to express a point Q of S as a convex combination of other points of S by using ratios of volumes of the intersection of cells of $V_2(S)$ and the cell of Q in $V_1(S)$. The natural neighbour interpolation method is based on Sibson's formula. We generalize his result to express Q as a convex combination of other points of S by using ratios of volumes from Voronoi diagrams of any given order.

1 Introduction

Let S be a set of n points in general position in $\mathbb{R}^d$, meaning no m of them lie in a $(m-2)$-dimensional flat for $m = 2, 3, ..., d+1$ and no $d+2$ of them lie in the same d-sphere, and let k be a natural number with $1 \leq k \leq n-1$. Let σ_d denote the Lebesgue measure on $\mathbb{R}^d$, to simplify we just write σ.

The order-k Voronoi diagram of S, $V_k(S)$, is a subdivision of $\mathbb{R}^d$ into cells such that points in the same cell have the same k nearest points of S. Thus, each cell $f(P_k)$ of $V_k(S)$ is defined by a subset P_k of S of k elements, where each point of $f(P_k)$ has P_k as its k closest points from S. Similarly, the ordered Voronoi diagram of order k of S, $OV_k(S)$, can be defined as a subdivision of $\mathbb{R}^d$ into cells such that points in the same cell have the same ordered k nearest points of S. Thus, each cell $f(\langle P_k \rangle)$ of $OV_k(S)$ is defined by an ordered subset $\langle P_k \rangle$ of size k of S, where the points are arranged in order of proximity starting from the closest to the farthest. Note that, by definition, the union of all the cells of $OV_k(S)$ corresponding to the different permutations of a fixed subset of length k of S is the cell, $f(P_k)$, associated to such subset in the (ordinary) order-k Voronoi diagram, $V_k(S)$. See Figure 1.

For the order-k Voronoi diagram of S, the region $R_k(\ell)$ of $Q_\ell \in S$ is defined as the set of cells of $V_k(S)$ that have the point Q_ℓ as one of their k nearest neighbours. See Figure 2. For $OV_k(S)$ we can define these regions in the same way. These regions are not necessarily convex but star-shaped, see [2, 4, 10, 16], and it is known that $R_1(\ell)$ is contained in the kernel of $R_k(\ell)$; see [3]. Also, these regions are related to Brillouin zones. For a given k, the region $R_k(\ell) \setminus R_{k-1}(\ell)$ is known as a Brillouin zone of Q_ℓ. Brillouin zones have been studied mainly for lattices but also for arbitrary discrete sets, see e.g. [6, 17].

Local coordinates based on Voronoi diagrams were introduced by Sibson [13]. He states that, given a set S of n points of $\mathbb{R}^d$ in general position, a point $Q_\ell \in S$ can be expressed as a convex combination of its nearest points of S. This is described next. Cells of $V_2(S)$ that intersect $f(\{Q_\ell\})$ in $V_1(S)$ are of the form $f(\{Q_\ell, Q_j\})$, i.e., cells defined by Q_ℓ and another point Q_j, that we call its natural neighbour. These intersections give ratios of volumes which are the coefficients multiplying the corresponding natural neighbours in the convex

40th European Workshop on Computational Geometry, Ioannina, Greece, March 13–15, 2024.
This is an extended abstract of a presentation given at EuroCG'24. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

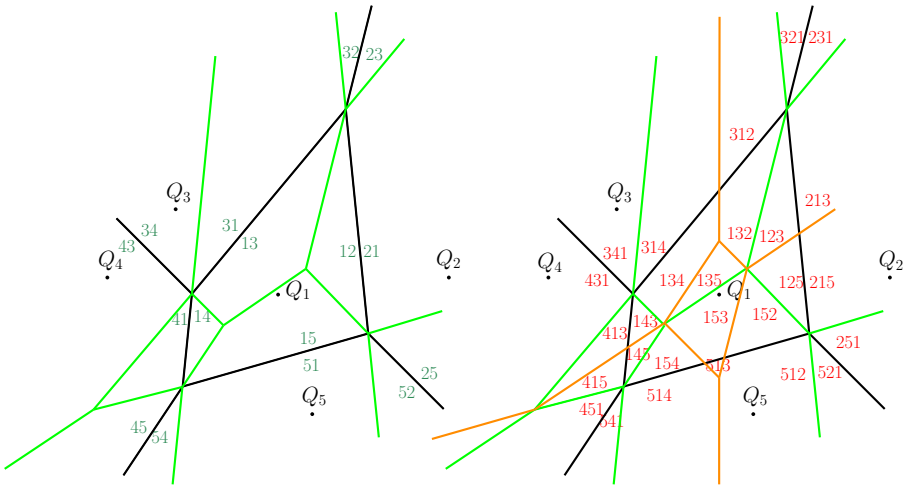


Figure 1 For a set $S = \{Q_1, \cdots, Q_5\}$ of five points in $\mathbb{R}^2$. Each cell of $OV_k(S)$ is labeled by the indices of its k nearest points of S. $V_1(S)$ is shown in black, $V_2(S)$ in green, and $V_3(S)$ in orange colour. Left: The cells of $OV_2(S)$ with the same nearest neighbour Q_i from S form the cell $f(\{Q_i\})$ in $V_1(S)$. The cells of $OV_2(S)$ with the same subset P_2 of two points of S (in any order) form the cell $f(P_2)$ of $V_2(S)$. Right: $OV_3(S)$ is shown together with $V_1(S)$, $V_2(S)$, and $V_3(S)$.

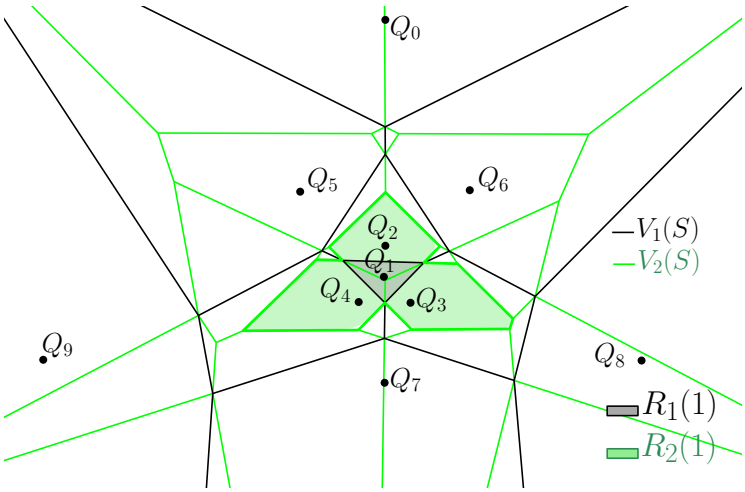


Figure 2 $R_1(1)$ is the cell $f(\{Q_1\})$ in $V_1(S)$. $R_2(1)$ is the union of cells of $V_2(S)$ that have Q_1 as one of its two nearest neighbours. $R_1(1) \subset R_2(1)$.

combination that expresses Q_ℓ. Volumes $\sigma(f(\{Q_\ell, Q_j\}) \cap f(\{Q_\ell\}))$ are equal to the volumes given by the intersection of the cells of $V_1(S \setminus \{Q_\ell\})$ and $f(Q_\ell)$ in $V_1(S)$, see Figure 3.

▶ **Theorem 1.1.** *(Local coordinates property [13]). For a bounded cell $f(\{Q_\ell\})$ of $V_1(S)$,*

$$Q_\ell = \sum_{j \neq \ell} \frac{\sigma(f(\{Q_\ell, Q_j\}) \cap f(\{Q_\ell\}))}{\sigma(f(\{Q_\ell\}))} Q_j \tag{1}$$

Sibson's formula has been used to define the natural neighbour interpolation method [14]. Given a set of points and a function, this interpolation method provides a smooth approximation of new points to the function. Sibson's algorithm uses the closest subset of the input set $S \setminus \{Q_\ell\}$ to interpolate a query point, Q_ℓ, and applies weights based on the ratios of volumes provided by Theorem 1.1. Local coordinates and the natural neighbour interpolation method have been studied e.g. in [5, 11, 15], and they have many applications such as reconstruction of a surface from unstructured data or interpolation of rainfall data, see [9, 15].

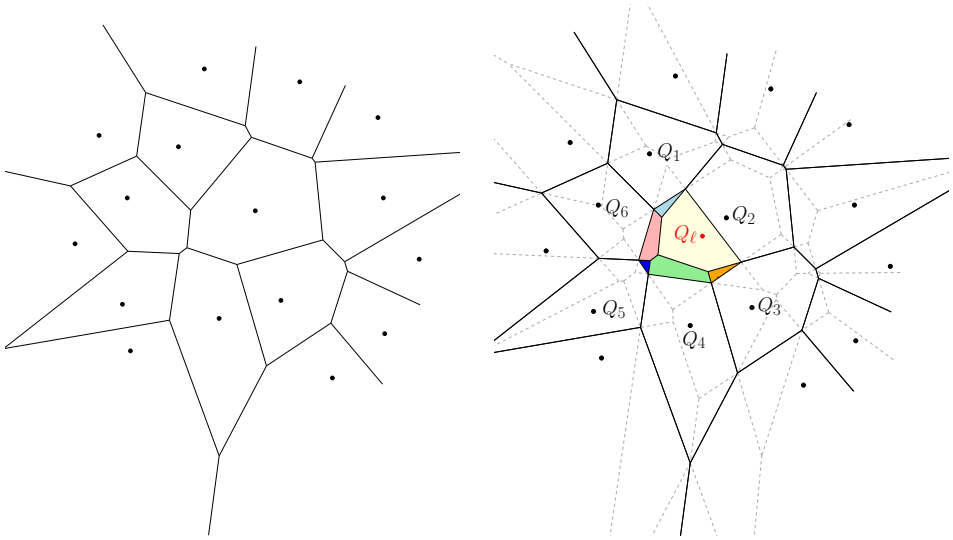


Figure 3 In $\mathbb{R}^2$. Left: The initial Voronoi diagram $V_1(S \setminus \{Q_\ell\})$ without query point Q_ℓ. Right: Colored areas given by the intersections of $f(\{Q_\ell\})$ and the cells of $V_1(S \setminus \{Q_\ell\})$, are the same as the ones given by the intersections of the cells of $V_2(S)$ (shown in dashed) with the cell $f(\{Q_\ell\})$.

Aurenhammer gave a generalization of Sibson's result to Voronoi diagrams of higher order, and more generally to power diagrams, see [1]. Aurenhammer's formula allows to write a point Q_ℓ of S as a linear combination of other points of S. We state this in Theorem 2.1 and Corollary 2.2 below. The formula in Theorem 2.1 is defined in terms of $OV_{k+1}(S)$. It is restated in Corollary 2.2 in terms of intersections of cells of $V_{k-1}(S)$ and $V_{k+1}(S)$ with a cell of $V_k(S)$. This formula works for a bounded cell of $V_k(S)$.

Our main contribution is another generalization of Sibson's result, stated in Theorem 2.3. In this theorem, we express a point $Q_\ell \in S$ as a convex combination of its neighbours of S using ratios of volumes in the region $R_k(\ell)$. Similar to Sibson's formula that required the cell of the point Q_ℓ to be bounded, our formula requires its region $R_k(\ell)$ to be bounded. For the case $k = 1$, Theorem 2.3 coincides with Theorem 1.1.

This paper is organized as follows. Section 2 details the generalization of Sibson's formula. In Section 3 we give a geometric interpretation of the formulas presented in Section 2 for point sets in the plane. Finally, Section 4 is on how the generalization of Sibson's formula could be used for interpolation. Proofs are omitted in this abstract.

2 Coordinates based on Voronoi diagrams

In this section we present a generalization of Sibson's formula that expresses a point using its neighbours of the Voronoi diagram of any given order. For this, we recall results from Aurenhammer [1] in Theorem 2.1 and Corollary 2.2, using a different notation.

Let $F_{k+1}(P_k)$, to simplify $F(P_k)$, be the set of cells of $OV_{k+1}(S)$, $1 \leq k \leq n-2$, whose k nearest neighbours are the points of $P_k \subset S$ in any order, and the $(k+1)$-th nearest neighbour is another point of S not in P_k. Let $f_{i,j}$ denote the union of cells of $OV_{k+1}(S)$ whose k-th nearest neighbour is Q_i and whose $(k+1)$-th nearest neighbour is Q_j.

▶ **Theorem 2.1.** *([1]) If all cells in $F(P_k)$ are bounded in $OV_{k+1}(S)$, then*

$$\sum_{\substack{j \\ f_{i,j} \in F(P_k)}} \sigma(f_{i,j})Q_i = \sum_{\substack{i \\ f_{i,j} \in F(P_k)}} \sigma(f_{i,j})Q_j$$

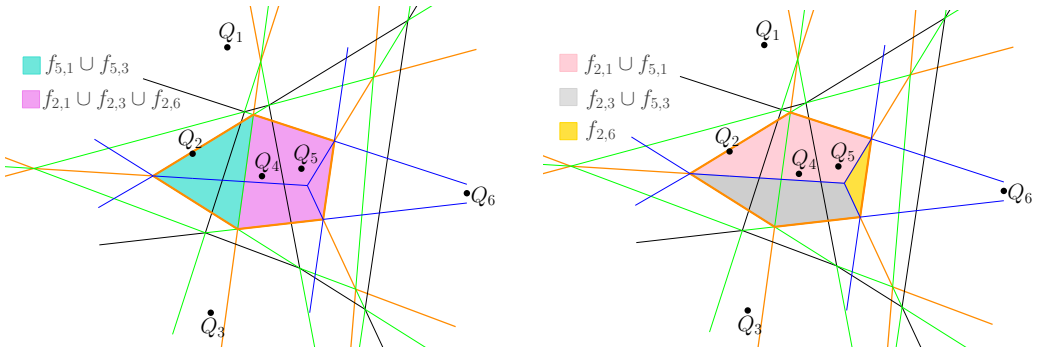


Figure 4 Illustrating Theorem 2.1 for $F(\{Q_2, Q_5\})$ in $OV_3(S)$, where S is a set of six points in $\mathbb{R}^2$. In this case the equation reduces to $\sigma(f_{5,1} \cup f_{5,3})Q_5 + \sigma(f_{2,1} \cup f_{2,3} \cup f_{2,6})Q_2 = \sigma(f_{2,1} \cup f_{5,1})Q_1 + \sigma(f_{2,3} \cup f_{5,3})Q_3 + \sigma(f_{2,6})Q_6$. Left: cells grouped according to its k-nearest neighbour. Right: cells grouped according to its $(k+1)$-nearest neighbour.

Note that, the subdivisions induced by $V_{k-1}(S)$ at the interior of $f(P_k)$ correspond to grouping the cells of $F(P_k)$ in $OV_{k+1}(S)$ that have the same k-nearest neighbour. Also, the subdivisions induced by $V_{k+1}(S)$ at the interior of $f(P_k)$ correspond to grouping the cells of $F(P_k)$ in $OV_{k+1}(S)$ that have the same $(k+1)$-nearest neighbour. See Figure 4.

By these observations, Theorem 2.1 can be stated as follows.

▶ **Corollary 2.2.** *([1]) Let $2 \leq k \leq n-2$ and let $f(P_k)$ be a bounded cell of $V_k(S)$. Then,*

$$\sum_{\substack{f(P_{k-1}) \in V_{k-1}(S) \\ Q_i \in P_k \setminus P_{k-1}}} \sigma(f(P_{k-1}) \cap f(P_k))Q_i = \sum_{\substack{f(P_{k+1}) \in V_{k+1}(S) \\ Q_j \in P_{k+1} \setminus P_k}} \sigma(f(P_{k+1}) \cap f(P_k))Q_j$$

Note that, by the relation between the Voronoi diagrams and the ordered Voronoi diagrams, $R_k(\ell)$ is the set of cells of $OV_{k+1}(S)$ that have Q_ℓ as one of their k nearest neighbours from S, i.e., $R_k(\ell) = \cup_{Q_\ell \in P_k} F(P_k)$. Based on this observation and Theorem 2.1 we can prove the following result.

▶ **Theorem 2.3.** *If $R_k(\ell)$ is a bounded region, then*

$$Q_\ell = \sum_{\substack{i \\ f_{i,j} \in R_k(\ell)}} \frac{\sigma(f_{i,j})}{\sigma(R_k(\ell))} Q_j.$$

▶ **Corollary 2.4.** *Let $1 \leq k \leq n-2$ and let $R_k(\ell)$ be a bounded region. Then,*

$$Q_\ell = \sum_{f(P_k)\in R_k(\ell)} \sum_{\substack{f(P_{k+1})\in V_{k+1}(S)\\ Q_j\in P_{k+1}\setminus P_k}} \frac{\sigma(f(P_{k+1})\cap f(P_k))}{\sigma(R_k(\ell))} Q_j$$

3 A geometric interpretation

In the following we examine the generalization of Sibson's theorem to higher order Voronoi diagrams from Corollary 2.2 in more detail for cells $f(P_k)$ of $V_k(S)$, when S is a point set in $\mathbb{R}^2$. Divide both sides of the equation given in Corollary 2.2 by $\sigma(f(P_k))$; then, each side of the equation describes a point H that is a convex combination of points from S. We have

$$H = \sum_{\substack{f(P_{k-1})\in V_{k-1}(S)\\ Q_i\in P_k\setminus P_{k-1}}} \frac{\sigma(f(P_{k-1})\cap f(P_k))}{\sigma(f(P_k))} Q_i = \sum_{\substack{f(P_{k+1})\in V_{k+1}(S)\\ Q_j\in P_{k+1}\setminus P_k}} \frac{\sigma(f(P_{k+1})\cap f(P_k))}{\sigma(f(P_k))} Q_j \quad (2)$$

What can we say about this point H?

Let $f(P_k)$ be an r-gon. Then S contains r points $Q_1,\ldots,Q_r$, such that each edge of the r-gon lies on a perpendicular bisector between two of these r points, and each vertex, $C_{ij\ell}$, of $f(P_k)$ is the center of a circle passing through three of them, Q_i, Q_j, and Q_ℓ; see e.g. [3].

We denote with $\Delta(ABC)$ the triangle with vertices A, B, and C, and with $\square(ABCD)$ the quadrilateral with vertices A, B, C and D, in cyclic order.

Let us consider the case when $f(P_k)$ is a quadrilateral cell of $V_k(S)$ with vertices C_{123}, C_{124}, C_{134}, and C_{234}, in cyclic order along the boundary of the quadrilateral cell $f(P_k) = \square(C_{123}C_{124}C_{134}C_{234})$. One of the diagonals $C_{123}C_{134}$ and $C_{124}C_{234}$ is an edge of $V_{k-1}(S)$ and the other one of $V_{k+1}(S)$. Figure 5 shows an example. We refer to [3, 7] for a more detailed discussion on the structure of cells of $V_k(S)$. Corollary 2.2 states in this case that

$$\begin{aligned} H &= Q1\cdot\frac{\sigma(\Delta(C_{123}C_{134}C_{234}))}{\sigma(\square(C_{123}C_{124}C_{134}C_{234}))} + Q3\cdot\frac{\sigma(\Delta(C_{123}C_{124}C_{134}))}{\sigma(\square(C_{123}C_{124}C_{134}C_{234}))}\\ &= Q2\cdot\frac{\sigma(\Delta(C_{124}C_{134}C_{234}))}{\sigma(\square(C_{123}C_{124}C_{134}C_{234}))} + Q4\cdot\frac{\sigma(\Delta(C_{124}C_{234}C_{123}))}{\sigma(\square(C_{123}C_{124}C_{134}C_{234}))} \end{aligned} \quad (3)$$

It follows that H is the intersection point of diagonals Q_1Q_3 and Q_2Q_4 of $\square(Q_1Q_2Q_3Q_4)$. This implies that given a quadrilateral cell $\square(C_{123}C_{124}C_{134}C_{234})$ of $V_k(S)$, the four corresponding points of S also form a convex quadrilateral, $\square(Q_1Q_2Q_3Q_4)$. Moreover, we can show that areas of triangles with vertices in $\square(C_{123}C_{124}C_{134}C_{234})$ are proportional to areas of triangles with vertices in $\square(Q_1Q_2Q_3Q_4)$, also see [8, 12].

Let us then consider the case when $f(P_k)$ is a cell of $V_k(S)$ with more than four sides. Equation (2) gives a point H that can be expressed in two ways as convex combination of points of S. Let us look at a pentagonal cell $f(P_k) = \pentagon(C_{123}C_{134}C_{145}C_{245}C_{125})$ of $V_k(S)$; See Figure 6. For $r > 5$ the situation is similar. Corollary 2.2 here gives

$$\begin{aligned} H &= Q_1\cdot\frac{\sigma(\square(C_{123}C_{125}C_{145}C_{134}))}{\sigma(\pentagon(C_{123}C_{134}C_{145}C_{245}C_{125}))} + Q_5\cdot\frac{\sigma(\Delta(C_{125}C_{245}C_{145}))}{\sigma(\pentagon(C_{123}C_{134}C_{145}C_{245}C_{125}))}\\ &= Q_2\cdot\frac{\sigma(\square(C_{245}C_{125}C_{123}C_{234}))}{\sigma(\pentagon(C_{123}C_{134}C_{145}C_{245}C_{125}))} + Q_4\cdot\frac{\sigma(\Delta(C_{245}C_{234}C_{1345}C_{145}))}{\sigma(\pentagon(C_{123}C_{134}C_{145}C_{245}C_{125}))}\\ &+ Q_3\cdot\frac{\sigma(\Delta(C_{123}C_{234}C_{134}))}{\sigma(\pentagon(C_{123}C_{134}C_{145}C_{245}C_{125}))} \end{aligned}$$

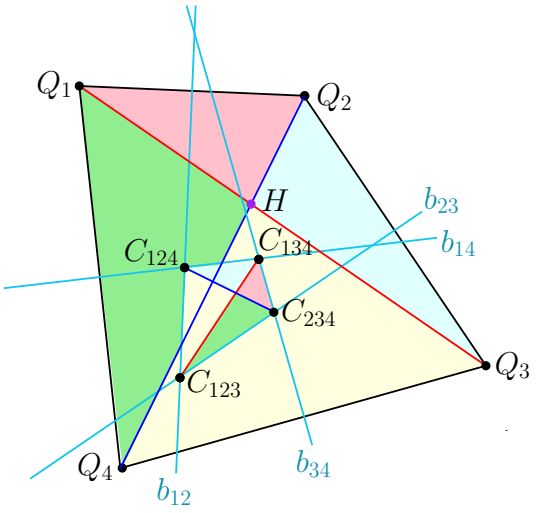


Figure 5 The quadrilateral cell $f(P_k) = \square(C_{123}C_{124}C_{134}C_{234})$ of $V_k(S)$ is obtained by perpendicular bisector construction from $\{Q_1, Q_2, Q_3, Q_4\} \subset S$. Point H given by Equation (2) is the intersection point of diagonals Q_1Q_3 and Q_2Q_4. Triangles with same color have proportional area.

We get that H lies on the segment Q_1Q_5 and inside the triangle $\Delta(Q_2Q_3Q_4)$. Furthermore, H divides the segment Q_1Q_5 in the same proportion as the edge $C_{125}C_{145}$ divides the pentagon $\pentagon(C_{123}C_{134}C_{145}C_{245}C_{125})$ into the the quadrilateral $\square(C_{125}C_{145}C_{134}C_{123})$ and the triangle $\Delta(C_{125}C_{145}C_{245})$. And H divides triangle $\Delta(Q_1Q_2Q_3)$ in the same proportion into triangles $\Delta(Q_3HQ_4)$, $\Delta(Q_2HQ_3)$, and $\Delta(Q_2HQ_4)$ as C_{234} divides $\pentagon(C_{123}C_{134}C_{145}C_{245}C_{125})$ into $\square(C_{245}C_{125}C_{123}C_{234})$, $\square(C_{245}C_{234}C_{134}C_{145})$ and $\Delta(C_{134}C_{234}C_{123})$.

4 Towards higher order natural neighbour interpolation

Sibson's theorem (Theorem 1.1) gave rise to the natural neighbour interpolation method. Given a set of points S and known function values $G(Q_j)$ for $Q_j \in S \setminus \{Q_\ell\}$, the function value $G(Q_\ell)$ of a point Q_ℓ is interpolated by $G(Q_\ell) = \sum_j c_j G(Q_j)$, where the sum is over the natural neighbours Q_j of Q_ℓ in $V_1(S)$. The local coordinates c_j are given by Theorem 1.1. Note that they satisfy $\sum_j c_j = 1$ and $c_j \geq 0$ for all j. Then, Sibson's natural neighour interpolation is given by

$$G(Q_\ell) = \sum_{j \neq \ell} \frac{\sigma(f(\{Q_\ell, Q_j\}) \cap f(\{Q_\ell\}))}{\sigma(f(\{Q_\ell\}))} G(Q_j). \tag{4}$$

The generalization of Sibson's formula given in Theorem 2.3 suggests to approximate the function value $G(Q_i)$ by using the natural neighbours of higher order Voronoi diagrams. By using the region $R_k(\ell)$ for $k > 1$, we can estimate the function value of a point Q_ℓ as

$$G(Q_\ell) = \sum_{\substack{i \\ f_{i,j} \in R_k(\ell)}} \frac{\sigma(f_{i,j})}{\sigma(R_k(\ell))} G(Q_j). \tag{5}$$

Note that $R_1(\ell) = f(\{Q_\ell\})$ in $V_1(S)$, and for $k = 1$ Equations (4) and (5) coincide.

A better estimation can be obtained by using Theorem 2.3 in a combination of different values of k. We explore this for the 1-dimensional case.

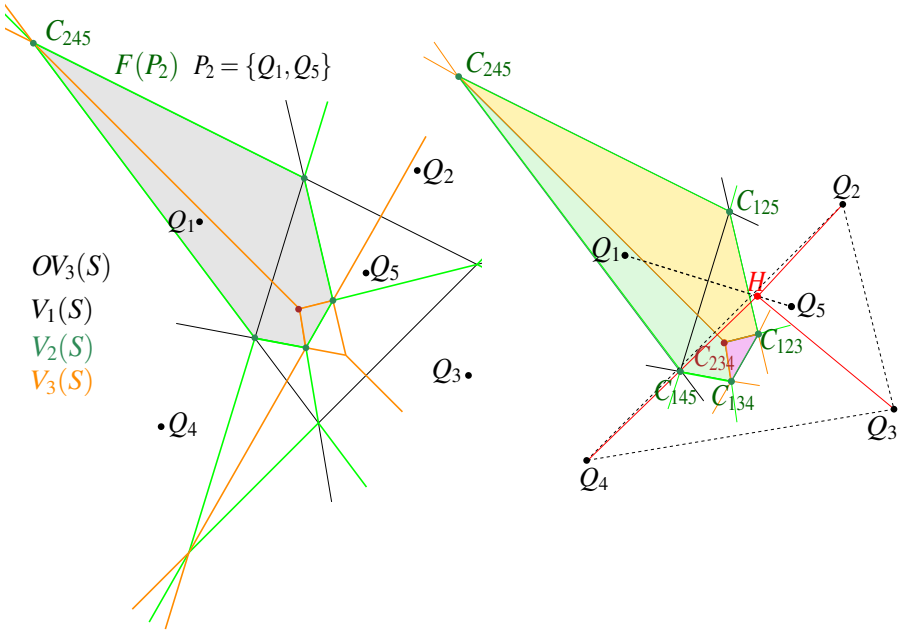


Figure 6 (Left) $OV_3(S)$ for a set of five points $S = \{Q_1, Q_2, Q_3, Q_4, Q_5\}$. For $P_2 = \{Q_1, Q_5\}$, the grey region $F(P_2)$ of $OV_3(S)$ is the pentagonal cell $f(P_2)$ of $V_2(S)$. $f(P_2)$ is divided by an edge of $V_1(S)$ and is also divided by three edges of $V_3(S)$. (Right) The point H lies on the segment Q_1Q_5 and inside the triangle $\Delta(Q_2, Q_3, Q_4)$. Triangle areas of $\Delta(Q_2HQ_3), \Delta(Q_3HQ_4)$ and $\Delta(Q_2HQ_4)$ are proportional to the areas of the three colored regions inside $f(P_2)$, green, yellow, and pink, respectively. The lengths of segments HQ_1 and HQ_5 are proportional to the areas $\sigma(f(P_2) \cap f(\{Q_1\}))$ and $\sigma(f(P_2) \cap f(\{Q_5\}))$, respectively.

Theorem 2.3, respectively Corollary 2.4, for dimension 1 reduces to the following statement.

▶ **Property 4.1.** *Let $S = \{x_0, x_1, \ldots x_{2\ell}\}$ with $x_0 < x_1 < \ldots < x_{2\ell}$ be real numbers. Then,*

$$x_\ell = \frac{1}{x_{2\ell} - x_0} \left(\left(\sum_{i=0}^{\ell-1} x_i (x_{\ell+1+i} - x_{\ell+i}) \right) + \left(\sum_{i=\ell+1}^{2\ell} x_i (x_{i-\ell} - x_{i-\ell-1}) \right) \right). \tag{6}$$

▶ Remark. Property 4.1 has actually a more general statement. The assumption $x_0 < x_1 < \ldots < x_{2\ell}$ is not needed.

We denote points Q_i of S as x_i and their function values $G(Q_i)$ as y_i. When $k = 1$ we have Sibson's classical nearest neighbour interpolation, which for dimension $d = 1$ is piecewise linear interpolation. Let $x_0, x_1, \ldots, x_5$ be six points on the real line in that order. And let $x_2 < x < x_3$ be a query point whose function value $G(x)$ we want to interpolate. To avoid degenerate cases where bisectors between points coincide, we also assume that all midpoints $(x_i + x_j)/2$ with $x_i, x_j \in \{S \cup \{x\}\}$ are different. Sibson's classical formula, Equation (4), uses the two neighbours x_2 and x_3 of x, and gives the interpolation

$$G_1(x) = \frac{1}{x_3 - x_2} \left(y_2(x_3 - x) + y_3(x - x_2) \right), \tag{7}$$

i.e. point $(x, G_1(x))$ lies on the line segment connecting points (x_2, y_2) and (x_3, y_3). This can also be deduced from Property 4.1. Combining Equation (5) for $k = 1$ and $k = 2$, we obtain

$$G_2(x) = \frac{1}{x_4 - x_1 + x_3 - x_2}(y_1(x_3 - x) + y_2(x_4 - x) + y_3(x - x_1) + y_4(x - x_2)). \quad (8)$$

In the same way, combining Equation (5) for $k = 1$, $k = 2$, and $k = 3$, we obtain

$$\begin{aligned} G_3(x) = &\frac{1}{x_5 - x_0 + x_4 - x_1 + x_3 - x_2}(y_0(x_3 - x) + y_1(x_4 - x) + y_2(x_5 - x) \\ &+ y_3(x - x_0) + y_4(x - x_1) + y_5(x - x_2)). \end{aligned} \quad (9)$$

Figure 7 shows an example of the interpolation formulas given in Equations (7), (8), and (9).

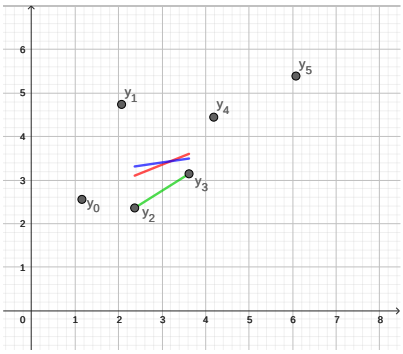


Figure 7 The generalized Sibson interpolation in $\mathbb{R}^1$. In green: Sibson's original interpolation, Equation (7), used only $R_1(x)$. The blue segment shows the interpolation using $R_1(x)$ and $R_2(x)$, given by Equation (8). Four points are used. The red segment shows the interpolation using $R_1(x)$, $R_2(x)$, and $R_3(x)$, given by Equation (9). Six points are used.

We conclude with some comments on the proposed interpolation formulas. First, they appear in a natural way from the generalization of Sibson's formula. This already makes it worth to study such generalized interpolation formulas. In Equations (8) and (9), the coefficients c_j in $G_i(x) = \sum_j c_j y_j$, $i = 2, 3$, satisfy $\sum_j c_j = 1$ and $c_j \geq 0$ for every c_j. We also mention that it can not be guaranteed that $G_i(x)$ coincides with $G_i(x_2)$ or with $G_i(x_3)$, when x coincides with one of the endpoints of the interval, x_2 or x_3, respectively. Though, we observe that in this case, the point farthest away from x on one side, drops from being used in the interpolation formula. This also holds for the classical case $k = 1$.

Finally, we expect that the generalized interpolation formulas can have applications. For instance, when the used values for the interpolation are obtained by measurements and measurement inaccuracy can not be ruled out. Then reliability might be improved by using nearest neighbours from $V_k(S)$ or by using $R_k(x)$, instead of only $V_1(S)$.

Acknowledgements

This research has been supported by projects 2021SGR00266 and PID2019-104129GB-I00/ MCIN/ AEI/ 10.13039/501100011033.

References

1 Franz Aurenhammer. Linear combinations from power domains. *Geometriae dedicata*, 28(1):45–52, 1988.

2 Franz Aurenhammer and Otfried Schwarzkopf. A simple on-line randomized incremental algorithm for computing higher order voronoi diagrams. In *Proceedings of the seventh annual symposium on Computational geometry*, pages 142–151, 1991.

3 Mercè Claverol, Andrea de las Heras Parrilla, Clemens Huemer, and Alejandra Martínez-Moraian. The edge labeling of higher order Voronoi diagrams. 2021. `https://arxiv.org/abs/2109.13002`.

4 Herbert Edelsbrunner and Mabel Iglesias-Ham. Multiple covers with balls i: Inclusion–exclusion. *Computational Geometry*, 68:119–133, 2018.

5 Gerald Farin. Surfaces over Dirichlet tessellations. *Computer aided geometric design*, 7(1-4):281–292, 1990.

6 Gareth A Jones. Geometric and asymptotic properties of Brillouin zones in lattices. *Bulletin of the London Mathematical Society*, 16(3):241–263, 1984.

7 Der-Tsai Lee. On k-nearest neighbor Voronoi diagrams in the plane. *IEEE Transactions on Computers*, C-31(6):478–487, 1982.

8 Maria Flavia Mammana and Biagio Micale. Quadrilaterals of triangle centres. *The Mathematical Gazette*, 92:466–475, 2008.

9 Atsuyuki Okabe, Barry Boots, and Kokichi Sugihara. Nearest neighbourhood operations with generalized Voronoi diagrams: a review. *International Journal of Geographical Information Systems*, 8(1):43–71, 1994.

10 Atsuyuki Okabe, Barry Boots, Kokichi Sugihara, and Sung Nok Chiu. Spatial tessellations: concepts and applications of Voronoi diagrams. 2009.

11 Bruce R. Piper. Properties of local coordinates based on Dirichlet tessellations. In *Geometric modelling*, pages 227–239. Springer, 1993.

12 Olga Radko and Emmanuel Tsukerman. The perpendicular bisector construction, the isotopic point, and the Simson line of a quadrilateral. *Forum Geometricorum*, 12:161–189, 2012.

13 Robin Sibson. A vector identity for the Dirichlet tessellation. In *Mathematical Proceedings of the Cambridge Philosophical Society*, volume 87, pages 151–155. Cambridge University Press, 1980.

14 Robin Sibson. A brief description of natural neighbour interpolation. *Interpreting multivariate data*, pages 21–36, 1981.

15 Kokichi Sugihara. Surface interpolation based on new local coordinates. *Computer-Aided Design*, 31(1):51–58, 1999.

16 G Fejes Tóth. Multiple packing and covering of the plane with circles. *Acta Math. Acad. Sci. Hungar*, 27(1-2):135–140, 1976.

17 JJP Veerman, Mauricio M Peixoto, André C Rocha, and Scott Sutherland. On Brillouin zones. *Communications in Mathematical Physics*, 212:725–744, 2000.

Computing an ε-net of a closed hyperbolic surface

Vincent Despré, Camille Lanuel, and Monique Teillaud

Université de Lorraine, CNRS, Inria, LORIA, F-54000 Nancy, France

Abstract

An ε-net of a metric space X is a set of points P of X such that the balls of radius ε centered at points of P cover X, and two distinct points of P are at least ε apart. We present an algorithm to compute an ε-net of a closed hyperbolic surface and analyze its complexity.

Related Version `https://hal.science/hal-04466350`

1 Introduction

This paper focuses on hyperbolic surfaces, i.e., surfaces with a metric of constant negative curvature. These surfaces have been extensively examined from a mathematical perspective, due to their generic nature: any Riemannian surface of genus at least two can be conformally mapped to a unique hyperbolic surface [16, Section IV.8].

Hyperbolic geometry also plays a key role in computer science. One of the most famous examples is found in the analysis of rotation distance in binary trees [19]. Hyperbolic geometry naturally emerges as a valuable tool for graph representation [14, 15]. The hyperbolic plane also serves as the preferred model for illustrating the universal cover of surfaces with genus at least 2, which has proven to be crucial in the proof of purely topological results [11, 6].

Delaunay triangulations of hyperbolic spaces and surfaces have been studied in the computational geometry community [2, 17, 12, 13]. In this line, we adapt Shewchuk's Delaunay refinement algorithm [18] to construct ε-nets of hyperbolic surfaces, opening the door to the design of efficient approximation algorithms. To the best of our knowledge, this is the first result of this kind.

Let us recall definitions [5]. Let (X, d) be a metric space and $\varepsilon > 0$. A set $P \subset X$ is an *ε-covering* if $\forall x \in X, d(x, P) \leqslant \varepsilon$, i.e., if the closed balls of radius ε centered at each $p \in P$ cover X. It is an *ε-packing* if $\forall p \neq q \in P, d(p, q) \geqslant \varepsilon$, i.e., if the open balls of radius $\varepsilon/2$ centered at each $p \in P$ are pairwise disjoint. An *ε-net* is both an ε-covering and an ε-packing. In this paper, we prove:

▶ **Proposition 1.** *Any ε-packing of a closed hyperbolic surface S of genus g and systole σ contains $N \leqslant 16(g-1)\left(1/\varepsilon^2 + 1/\sigma^2\right)$ points. If $\varepsilon < \sigma$, then $N \leqslant 16(g-1)/\varepsilon^2$.*

The case when $\varepsilon < \sigma$ corresponds to the situation when the surface has no ε-thin part (see Section 2.2).

▶ **Proposition 2.** *The Delaunay refinement algorithm computes an ε-net using at most $\left(10 + C'_h \operatorname{Diam}(S)^{6g-4}\right) N^2 + (N-1)(144g^2 - 104g + 35) - 10$ elementary operations, where C'_h is a constant depending on the metric h of S, and $\operatorname{Diam}(S)$ is the diameter of S.*

For a fixed surface, the complexity is then $O(1/\varepsilon^4)$.

The first result can be regarded as folklore. We prove it in Section 3 for completeness. The second proposition rises interesting obstacles to deal with. In particular, Shewchuk's refinement adds circumcenters of some triangles, which is not straightforward in our context, as locating a new point requires to construct a portion of the universal cover of the surface. We manage to bound the size of this portion.

40th European Workshop on Computational Geometry, Ioannina, Greece, March 13–15, 2024.
This is an extended abstract of a presentation given at EuroCG'24. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

2 Background on hyperbolic surfaces and notation

We refer the reader to textbooks for more details, e.g. [4, 1].

A closed hyperbolic surface can be seen as the quotient $\mathbb{H}^2/\Gamma$ of the hyperbolic plane $\mathbb{H}^2$ under the action of a group Γ of orientation-preserving isometries. Throughout the paper, objects in $\mathbb{H}^2$ are denoted with a tilde $\widetilde{\cdot}$, while objects on S are denoted without. In particular, for an object o on S, $\widetilde{o}$ denotes any of its lifts in $\mathbb{H}^2$. To simplify the language, we often use the term *copy* to refer to an image of an object in $\mathbb{H}^2$ by an element of Γ.

We work with the Poincaré disk model in which the hyperbolic plane $\mathbb{H}^2$ is represented as the unit disk of the complex plane $\mathbb{C}$. The unit circle consists of points at infinity. The geodesics are either diameters of the unit disk, or circular arcs that meet the boundary circle orthogonally. The hyperbolic circles are Euclidean circles (but their hyperbolic and Euclidean centers differ). Orientation-preserving isometries are represented as matrices in $\mathbb{C}^{2\times 2}$.

2.1 Delaunay triangulation and Dirichlet domain

A triangulation T of S is a partition of S into triangles; note that edges may be loops. A triangulation of S is a Delaunay triangulation if for each triangle t of T and any of its lifts $\widetilde{t}$ in $\mathbb{H}^2$, the open disk circumscribing $\widetilde{t}$ contains no vertex of the (infinite) lift of T in $\mathbb{H}^2$ [12]. The Voronoi diagram is the dual of the Delaunay triangulation. The Dirichlet domain $\mathcal{D}_{\widetilde{x}}$ of a point $\widetilde{x} \in \mathbb{H}^2$ is the (closed) cell of $\widetilde{x}$ in the Voronoi diagram of its (infinite) orbit $\Gamma\widetilde{x}$. Unlike the Euclidean case, Γ is non-commutative, and the combinatorics of a Dirichlet domain depends on the point x (Figure 1). The number k of sides of $\mathcal{D}_{\widetilde{x}}$ satisfies $4g \leqslant k \leqslant 12g-6$ (see, e.g., [8]).

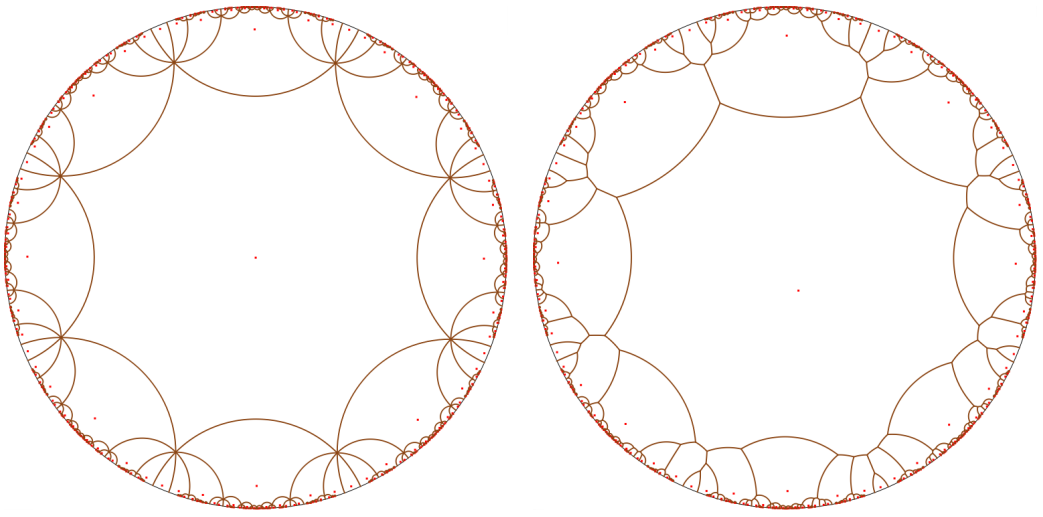


Figure 1 Dirichlet domains for the Bolza surface ($g = 2$). The domain on the left has $4g = 8$ sides and the one on the right has $12g - 6 = 18$ sides. Figure from [3].

In this paper, we assume that the input surface S is given by a Delaunay triangulation having a single vertex b, i.e., all Delaunay edges are loops based in b. The point b is arbitrary. This introduces no restriction, as such a representation can be computed for any closed hyperbolic surface, starting from a standard representation by a fundamental domain and

side pairings [8].[1] The Dirichlet domain $\mathcal{D}_{\widetilde{b}}$ of some lift $\widetilde{b}$ of b can be computed together with the corresponding side pairings, which are generating the group Γ. The sides of $\mathcal{D}_{\widetilde{b}}$ are denoted as $s_i, i = 0, \ldots, k-1$ and the corresponding side pairings as $\gamma_i, i = 0, \ldots, k-1$ (here, side pairings are pairwise inverses).

2.2 Thin and thick parts

The *injectivity radius* $r_x(S)$ of S at a point x is the supremum of all $r > 0$ such that the open ball of radius r centered at x, $B(x, r) = \{y \in S \mid \delta_S(x, y) < r\}$, where δ_S is the distance on S, is isometric to a disk in $\mathbb{H}^2$. In particular, $B(x, r)$ is a topologically embedded disk on S for all $r \leqslant r_x(S)$. The *systole* σ of a surface is the length of its shortest non-contractible curve, which we also denote by σ. The systole is related to the injectivity radius: $\sigma = 2 \cdot \inf \{r_x(S) \mid x \in S\}$.

For any $\varepsilon > 0$, the *ε-thin part* of S is $S_\varepsilon^t = \{x \in S \mid r_x(S) \leqslant \varepsilon/2\}$, and its *$\varepsilon$-thick part* is $S_\varepsilon^T = S \setminus S_\varepsilon^t$. Observe that if $\varepsilon < \sigma$, then there is no ε-thin part.

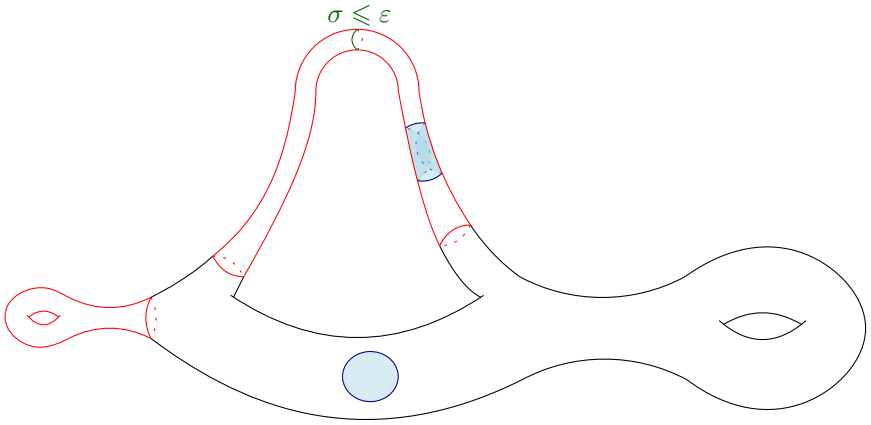


Figure 2 Thick and thin (red) parts of a hyperbolic surface. Disks of radius ε are shown in blue.

3 Proof of proposition 1

Let P be an ε-packing of S. The open balls of radius $\varepsilon/2$ centered at the points of P on the ε-thick part S_ε^T are isometric to disks in $\mathbb{H}^2$ and are pairwise disjoint. The area of such a disk centered at a point p is $\mathcal{A}\left(B\left(p, \frac{\varepsilon}{2}\right)\right) = 4\pi \sinh^2\left(\frac{\varepsilon}{4}\right)$ [1, Theorem 7.2.2]. Since $\sinh(x) \geqslant x$ for all $x \geqslant 0$, we have $\mathcal{A}\left(B\left(p, \frac{\varepsilon}{2}\right)\right) \geqslant \pi\varepsilon^2/4$.

Let N^T be the number of points of P on the ε-thick part S_ε^T. By the Gauss-Bonnet theorem, the area of the surface S is $\mathcal{A}(S) = 4\pi(g-1)$. Summing the above inequality over all the points in $P \cap S_\varepsilon^T$, we obtain $N^T \pi\varepsilon^2/4 \leqslant \sum_{p \in P \cap S_\varepsilon^T} \mathcal{A}\left(B\left(p, \frac{\varepsilon}{2}\right)\right) \leqslant 4\pi(g-1)$, thus

$$N^T \leqslant \frac{16(g-1)}{\varepsilon^2}. \tag{1}$$

The open balls of radius $\varepsilon/2$ in the ε-thin part S_ε^t, if it exists, that is if $\sigma \leqslant \varepsilon$, are also pairwise disjoint, but they are not isometric to disks in $\mathbb{H}^2$. However, by definition, the open balls of radius $\sigma/2$ are isometric to disks in $\mathbb{H}^2$. We can apply the reasoning that led to

[1] The common basepoint is denoted as b'' in [8].

inequality (1) for σ instead of ε, and obtain a bound on the number of points of P on the thin part S^t_ε: $N^t \leqslant 16(g-1)/\sigma^2$. The bound on the total number of points of P follows.

4 Construction of the ε-net

The input of the algorithm consists of the Delaunay triangulation of S with a single vertex b, together with the Dirichlet domain $\mathcal{D}_{\widetilde{b}}$ of a lift $\widetilde{b}$ and the group Γ generated by side-pairings. As mentioned in Section 2.1, this does not induce any loss of generality.

Our algorithm is inspired by Shewchuk's Delaunay refinement [18]. The general idea is to break each Delaunay triangle whose circumcircle has a radius greater than ε by inserting its circumscribing center in the triangulation.

We reuse the data structure proposed by Despré *et al.* for computing the Delaunay triangulation of a surface by edge flips [12]. A triangulation of S is represented by

- its vertices: a vertex p has constant-time access to its lift $\widetilde{p_b}$ in $\mathcal{D}_{\widetilde{b}}$ and one of its incident triangles;
- and its triangles: a triangle Δ has constant-time access to its three vertices $p_0^\Delta, p_1^\Delta, p_2^\Delta$, its three adjacent triangles, and three isometries $\gamma_0^\Delta = \mathbb{1}_\Gamma, \gamma_1^\Delta, \gamma_2^\Delta$ in Γ defined as follows.

A triangle $\Delta = (p_0^\Delta; p_1^\Delta; p_2^\Delta)$ does not always have a lift entirely included in $\mathcal{D}_{\widetilde{b}}$. However, it always has at least one lift with at least one vertex in $\mathcal{D}_{\widetilde{b}}$ (see Figure 3). Let us choose such a lift and denote it as $\widetilde{\Delta_0}$; up to a re-indexing of its vertices, $\widetilde{p_0^\Delta} \in \mathcal{D}_{\widetilde{b}}$. Then γ_1^Δ and γ_2^Δ are the isometries such that the other two vertices of $\widetilde{\Delta_0}$ are $\gamma_1^\Delta \widetilde{p_1^\Delta}$ and $\gamma_2^\Delta \widetilde{p_2^\Delta}$. Note that the other lifts of Δ having at least one vertex in $\mathcal{D}_{\widetilde{b}}$ can be retrieved by applying the inverses of these isometries to $\widetilde{\Delta_0}$. The union, on all triangles of the triangulation of S, of their lifts with at least one vertex in $\mathcal{D}_{\widetilde{b}}$ covers the fundamental domain $\mathcal{D}_{\widetilde{b}}$.

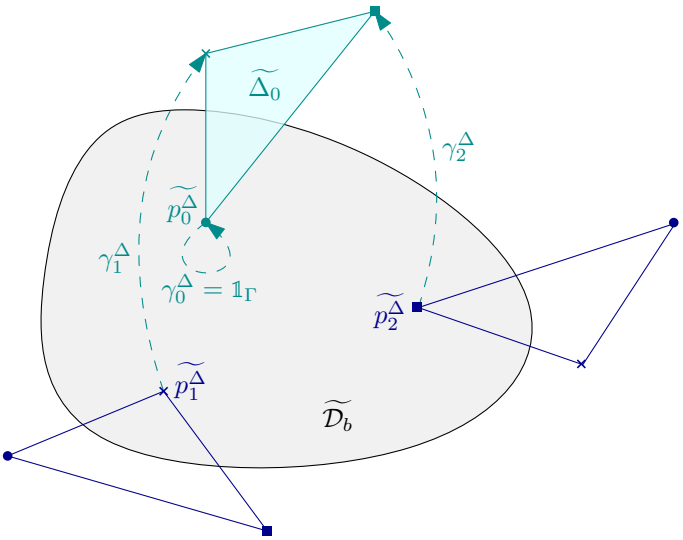


Figure 3 Example of a triangle Δ having three lifts with one vertex in $\mathcal{D}_{\widetilde{b}}$ (the hyperbolic triangles are schematically represented with straight edges).

We denote as $DT(\cdot)$ the Delaunay triangulation of a set of points on S.

Let us fix $\varepsilon > 0$. In a first step, the set of points is initialized as $P_1 = \{b\}$.

At each step $i \geqslant 2$, the algorithm inserts the circumscribing center c of a triangle Δ^ε whose radius is greater than ε. The set of points is updated as $P_i = P_{i-1} \cup \{c\}$ as well as

the Delaunay triangulation $DT(P_i)$. To do so, several operations are needed.

We first compute the radius of $\widetilde{\Delta_0}$ for every triangle Δ of $DT(P_{i-1})$, until a triangle Δ^ε whose radius is at least ε is found.[2] The circumcenter $\widetilde{c}$ of the lift $\widetilde{\Delta_0^\varepsilon}$ is a lift of c, but it does not necessarily lie in $\mathcal{D}_{\widetilde{b}}$. This can be checked by testing whether $\widetilde{b}$ and $\widetilde{c}$ lie on the same side of the supporting line of each side of $\mathcal{D}_{\widetilde{b}}$.

To actually insert c into $DT(P_{i-1})$, we need to find the lift $\widetilde{c}_b$ of c that lies in $\mathcal{D}_{\widetilde{b}}$. If $\widetilde{c}$ lies in $\mathcal{D}_{\widetilde{b}}$, then $\widetilde{c}_b = \widetilde{c}$. Otherwise, the algorithm walks in the tiling $\{\gamma\mathcal{D}_{\widetilde{b}}\}_{\gamma\in\Gamma}$ of $\mathbb{H}^2$ along the geodesic segment $\widetilde{p_0^{\Delta^\varepsilon}\widetilde{c}}$. The first copy of $\mathcal{D}_{\widetilde{b}}$ traversed by $\widetilde{p_0^{\Delta^\varepsilon}\widetilde{c}}$ is found by looking for the side $s_{j_1}, j_1 \in \{0,\dots,k-1\}$ of $\mathcal{D}_{\widetilde{b}}$ intersecting it.[3] The walk along $\widetilde{p_0^{\Delta^\varepsilon}\widetilde{c}}$ continues in $\gamma_{j_1}\mathcal{D}_{\widetilde{b}}$, and so on, until the copy $\gamma_{j_n}\dots\gamma_{j_1}\mathcal{D}_{\widetilde{b}}$ containing $\widetilde{c}$ is found. Then $\widetilde{c}_b = \gamma_{j_1}^{-1}\dots\gamma_{j_n}^{-1}\widetilde{c}$. Note that the walk still works when $\widetilde{p_0^{\Delta^\varepsilon}\widetilde{c}}$ goes through a vertex of a copy of $\mathcal{D}_{\widetilde{b}}$.

The Delaunay triangulation $DT(P_i)$ of $P_i = P_{i-1}\cup\{c\}$ can then be computed. First, the triangle Δ_c of $DT(P_{i-1})$ containing c is found by naively checking if $\widetilde{c}_b$ lies in one of the (at most three) lifts of each triangle Δ in $DT(P_{i-1})$ having a vertex in $\mathcal{D}_{\widetilde{b}}$. This can be done by testing, for each edge, whether $\widetilde{c}_b$ and the third vertex of the triangle lie on the same side of its supporting line. Then Δ_c is split into three by creating an edge between c and its three vertices. In the data structure, the three isometries stored in each new triangle are $\mathbb{1}_\Gamma$ for c, and the corresponding isometries in Δ_c for the other two vertices. Then $DT(P_i)$ is computed with a sequence of flips and the data structure is updated [12].

The termination of the algorithm is quite obvious. At step $i = 1$, the ε-packing P_1 consists of one point. At each step $i \geqslant 2$, the point added to P_i is the circumcenter of a Delaunay triangle whose radius is at least ε. Because no vertex lies in the interior of a Delaunay disk, the center added is at distance at least ε from any point of P_i. By induction, P_i is an ε-packing containing i points. By Proposition 1, the algorithm must terminate after a finite number $N-1$ of insertions. It returns an ε-packing P_N of cardinality N.

It remains to show that P_N is an ε-covering of S. Let x be a point on S. It lies in a triangle Δ of $DT(P_N)$. Let $\widetilde{\Delta}$ be a lift of Δ and $\widetilde{x}$ the lift of x lying in $\widetilde{\Delta}$. The circumdisk of $\widetilde{\Delta}$ has a radius $r \leqslant \varepsilon$. There is a vertex of $\widetilde{\Delta}$ whose distance to $\widetilde{x}$ is at most r (see [10, Lemma 2]). That vertex is a lift of a point of P_N by definition of Δ. It follows that $\delta_S(x, P_N) \leqslant \varepsilon$, therefore P_N is an ε-net. This establishes the first claim of Proposition 2.

5 Algorithm analysis

This section is devoted to proving the complexity announced in Proposition 2.

The following operations take $O(1)$ time in the real RAM model and we consider them as elementary operations:

- Computing $\widetilde{\Delta_0}$ from a triangle Δ of the data structure (see Section 4 for notation);
- Computing the radius or the center of the circumcircle of a triangle in $\mathbb{H}^2$;
- Deciding if a point lies on the right or the left side of an oriented geodesic segment in $\mathbb{H}^2$;
- Flipping an edge of a triangulation [12, Section 4.1].

At the beginning of a step $i \geqslant 2$, P_{i-1} contains $i-1$ points, the Euler characteristic shows that $DT(P_{i-1})$ has $2i+4g-2$ triangles, which gives the cost of finding Δ^ε.

[2] Of course a priority queue could be used to improve the complexity of this search. We accept a linear complexity for simplicity, as this is not the dominant operation in the algorithm.

[3] To check whether two geodesic segments $\widetilde{x}_1\widetilde{x}_2$ and $\widetilde{y}_1\widetilde{y}_2$ intersect, we check whether $\widetilde{x}_1$ and $\widetilde{x}_2$ lie on opposite sides of the supporting line of $\widetilde{y}_1\widetilde{y}_2$, and we run the same test, swapping the roles of x and y.

Recall that the number k of sides of $\mathcal{D}_{\widetilde{b}}$ is at most $12g-6$ (see Section 2). Determining whether $\widetilde{c}$ lies in (a given copy of) $\mathcal{D}_{\widetilde{b}}$ thus requires at most $12g-6$ elementary operations. The algorithm tests the copies of $\mathcal{D}_{\widetilde{b}}$ that intersect the geodesic segment $\widetilde{p_0^{\Delta^\varepsilon}\widetilde{c}}$. Since $\widetilde{\Delta_0^\varepsilon}$ is a triangle of $DT\left(\widetilde{P_{i-1}}\right)$, its circumcircle does not contain any other lift of $p_0^{\Delta^\varepsilon}$, so $\widetilde{p_0^{\Delta^\varepsilon}}$ is the closest lift of $p_0^{\Delta^\varepsilon}$ to $\widetilde{c}$. The geodesic segment $\widetilde{p_0^{\Delta^\varepsilon}\widetilde{c}}$ is thus a lift of a distance path[4] on S, what is called a *distance path* in $\mathbb{H}^2$. By [9, Proposition 14], every side of $\mathcal{D}_{\widetilde{b}}$ is either a distance path, or the concatenation of two distance paths. As two distance paths that do not have a subarc in common, which is the case here, can intersect at most once [9, Lemma 8], $\widetilde{p_0^{\Delta^\varepsilon}\widetilde{c}}$ traverses at most $2k$ sides of copies of $\mathcal{D}_{\widetilde{b}}$. If an intersection occurs at a vertex of degree d of a copy of $\mathcal{D}_{\widetilde{b}}$, then this counts for d intersections. Searching the copy of $\mathcal{D}_{\widetilde{b}}$ containing $\widetilde{c}$ hence requires $k^2 \leqslant (12g-6)^2$ elementary operations. Computing $\widetilde{c_b}$ costs 1 operation.

Finding Δ_c in $DT(P_{i-1})$ when $\widetilde{c_b}$ is known requires at most $9(2i+4g-2)$ elementary operations since it amounts to checking the three edges of at most three lifts of each triangle. The update of the data structure when splitting the triangle containing c into three is done in 8 elementary operations (deleting the triangle that contains c, adding c to the list of vertices, creating 3 triangles and 3 isometries).

Adding the above costs for step i, locating c in $DT(P_{i-1})$ and splitting the triangle containing it costs at most $10(2i+4g-2)+(12g-6)^2+9$ elementary operations.

The flips are counted globally for all steps, which concludes the proof of Proposition 2.

▶ **Lemma 5.1.** *The total number of flipped edges during the execution of the algorithm is at most* $C'_h \operatorname{Diam}(S)^{6g-4}N^2$*, where* C'_h *is a constant depending on the metric* h *of* S*, and* $\operatorname{Diam}(S)$ *is the diameter of* S*.*

The proof of this lemma mimicks the proofs in [12]. The situation is quite different here, as the points are inserted incrementally and the flips are done at each insertion, whereas all points are know in advance in [12], which requires to rewrite a complete proof. Due to lack of space, we refer the reader to [10, Lemma 1].

Note that the bound comes from the best upper bound $O(\operatorname{Diam}(S)^{6g-4})$ known so far for the flip algorithm [12]. The actual complexity of the flip algorithm may be much better [7].

Acknowledgements. The authors wish to thank Hugo Parlier for interesting discussions.

[4] A distance path on S is a shortest path between two points. It is necessarily a geodesic segment, but not all geodesic segments are distance paths since they only locally minimize distances.

References

1. Alan F. Beardon. *The Geometry of Discrete Groups*. Graduate Texts in Mathematics. Springer New York, 1st edition, 1983. doi:10.1007/978-1-4612-1146-4.
2. Mikhail Bogdanov, Olivier Devillers, and Monique Teillaud. Hyperbolic Delaunay complexes and Voronoi diagrams made practical. *Journal of Computational Geometry*, 5(1):56–85, March 2014. doi:10.20382/jocg.v5i1a4.
3. Mikhail Bogdanov, Monique Teillaud, and Gert Vegter. Delaunay triangulations on orientable surfaces of low genus. In *32nd International Symposium on Computational Geometry (SoCG)*, 2016. doi:10.4230/LIPIcs.SoCG.2016.20.
4. Peter Buser. *Geometry and Spectra of Compact Riemann Surfaces*. Modern Birkhäuser Classics. Birkhäuser Boston, 1st edition, 2010. doi:10.1007/978-0-8176-4992-0.
5. Kenneth L. Clarkson. Building triangulations using ϵ-nets. In *38th annual ACM Symposium on Theory of Computing (STOC)*, pages 326–335. Association for Computing Machinery, May 2006. Extended abstract (long paper available at https://kenclarkson.org/enet_tris/p.pdf). doi:10.1145/1132516.1132564.
6. Éric Colin de Verdière, Vincent Despré, and Loïc Dubois. Untangling graphs on surfaces. In *Proceedings of the 2024 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 4909–4941, January 2024. doi:10.1137/1.9781611977912.175.
7. Vincent Despré, Loïc Dubois, Benedikt Kolbe, and Monique Teillaud. Experimental analysis of Delaunay flip algorithms on genus two hyperbolic surfaces, December 2021. URL: https://hal.inria.fr/hal-03462834.
8. Vincent Despré, Benedikt Kolbe, Hugo Parlier, and Monique Teillaud. Computing a Dirichlet domain for a hyperbolic surface. In *39th International Symposium on Computational Geometry (SoCG)*, volume 258, pages 27:1–27:15, 2023. doi:10.4230/LIPIcs.SoCG.2023.27.
9. Vincent Despré, Benedikt Kolbe, and Monique Teillaud. Representing infinite hyperbolic periodic Delaunay triangulations using finitely many Dirichlet domains, July 2021. Preprint. URL: https://hal.science/hal-03045921.
10. Vincent Despré, Camille Lanuel, and Monique Teillaud. Computing an ε-net of a closed hyperbolic surface. Preprint, 2024. URL: https://hal.science/hal-04466350.
11. Vincent Despré and Francis Lazarus. Computing the geometric intersection number of curves. *Journal of the ACM*, 66(6):45:1–45:49, November 2019. doi:10.1145/3363367.
12. Vincent Despré, Jean-Marc Schlenker, and Monique Teillaud. Flipping geometric triangulations on hyperbolic surfaces. In *36th International Symposium on Computational Geometry (SoCG)*, volume 164, pages 35:1–35:16, June 2020. doi:10.4230/LIPIcs.SoCG.2020.35.
13. Matthijs Ebbens, Hugo Parlier, and Gert Vegter. Minimal Delaunay triangulations of hyperbolic surfaces. *Discrete & Computational Geometry*, 69(2):568–592, February 2022. doi:10.1007/s00454-022-00373-0.
14. David Eppstein. Squarepants in a tree: Sum of subtree clustering and hyperbolic pants decomposition. *ACM Transactions on Algorithms*, 5(3):29:1–29:24, July 2009. doi:10.1145/1541885.1541890.
15. David Eppstein. Limitations on realistic hyperbolic graph drawing. In *International Symposium on Graph Drawing and Network Visualization (GD)*, Lecture Notes in Computer Science, pages 343–357. Springer International Publishing, 2021. URL: https://arxiv.org/abs/2108.07441, doi:10.1007/978-3-030-92931-2_25.
16. Hershel M. Farkas and Irwin Kra. *Riemann Surfaces*, volume 71 of *Graduate Texts in Mathematics*. Springer, New York, NY, 1992. doi:10.1007/978-1-4612-2034-3.
17. Iordan Iordanov and Monique Teillaud. Implementing Delaunay triangulations of the Bolza surface. In *33rd International Symposium on Computational Geometry (SoCG)*, pages 44:1–44:15, July 2017. doi:10.4230/LIPIcs.SoCG.2017.44.

18 Jonathan Richard Shewchuk. Delaunay refinement algorithms for triangular mesh generation. *Computational Geometry: Theory and Appplications*, 22(1):21–74, May 2002. doi:10.1016/S0925-7721(01)00047-5.

19 Daniel D. Sleator, Robert E. Tarjan, and William P. Thurston. Rotation distance, triangulations, and hyperbolic geometry. In *18th annual ACM symposium on Theory of computing (STOC 1986)*, pages 122–135. Association for Computing Machinery, November 1986. doi:10.1145/12130.12143.

Barking dogs: A Fréchet distance variant for detour detection

Ivor van der Hoog[1], Fabian Klute[2], Irene Parada[3], and Patrick Schnider[4]

1 **Technical University of Denmark, Denmark**
`idjva@dtu.dk`
2 **Universitat Politècnica de Catalunya, Spain**
`fabian.klute@upc.edu`
3 **Universitat Politècnica de Catalunya, Spain**
`irene.parada@upc.edu`
4 **Department of Computer Science, ETH Zürich, Switzerland**
`patrick.schnider@inf.ethz.ch`

Abstract

Imagine you are a dog behind a fence Q and a hiker is passing by at constant speed along the hiking path P. In order to fulfil your duties as a watchdog, you desire to bark as long as possible at the human. However, your barks can only be heard in a fixed radius ρ and, as a dog, you have bounded speed s. Can you optimize your route along the fence Q in order to maximize the barking time with radius ρ, assuming you can run backwards and forward at speed at most s?

We define the barking distance from a polyline P on n vertices to a polyline Q on m vertices as the time that the hiker stays in your barking radius if you run optimally along Q. This asymmetric similarity measure between two curves can be used to detect outliers in Q compared to P that other established measures like the Fréchet distance and Dynamic Time Warping fail to capture at times. In this extended abstract, we consider this measure in the discrete setting, where the traversals of P and Q are both discrete. In this setting, we show how to compute the barking distance in time $O(nm \log s)$.

Related Version arXiv:2402.13159

1 Introduction

A *curve* is any sequence of points in $\mathbb{R}^d$ where consecutive points are connected by their line segment. Curves may be used to model a variety of real-world input such as trajectories [12], handwriting [11, 17] and even strings [3]. Curves in $\mathbb{R}^1$ may be seen as *time series* which model data such as music samples [10], the financial market [13] and seismologic data [16]. A common way to analyse data that can be modeled as curves is to deploy a curve similarity measure, which for any pair of curves series (P, Q) reports a real number (where the number is lower the more 'similar' P and Q are). Such similarity measures are a building block for common analysis techniques such as clustering [7, 15], classification [1, 8, 9] or simplification [2, 6, 14]. The two most popular similarity measures for curve analysis are the Fréchet distance and the Dynamic Time Warping (DTW) distance. The discrete Fréchet distance for two curves $P = (p_1, \ldots, p_n)$ and $Q = (q_1, \ldots, q_m)$ is illustrated as follows. Imagine a dog walking along

Funding statement: F.K. is supported by a "María Zambrano grant for attracting international talent". I.P. is a Serra Húnter fellow. I.,H.: This project has additionally received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 899987.

40th European Workshop on Computational Geometry, Ioannina, Greece, March 13–15, 2024.
This is an extended abstract of a presentation given at EuroCG'24. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

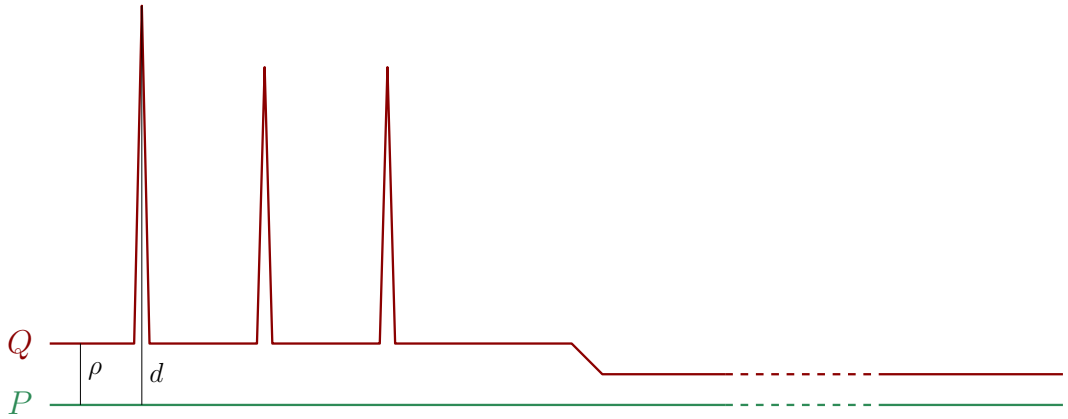


Figure 1 An intended trajectory P and a faulty sample Q of it. The Fréchet distance between P and Q is d and captures the first detour, but fails to capture the others. Continuous DTW, even with a speed bound, cannot distinguish Q from a copy of P translated by ρ if the right part is sufficiently long. Barking distance with barking radius ρ however captures all three detours.

Q and its owner walking along P. Both owner and dog start at the beginning of their curves, and in each step the owner may stay in place or jump to the next point along P and the dog may stay in place or jump to the next vertex along Q, until both of them have reached the end of their curves. Intuitively, the Fréchet distance is the minimal length of the leash between the dog and its owner. The DTW distance is defined analogously but sums over all leash lengths instead.

Both distance measures can be made continuous by defining a traversal as continuous monotone functions $f : [0, 1] \to P$ and $g : [0, 1] \to Q$ which start and end at the respective start and end of the curve. However, for DTW such a direct translation from discrete to continuous traversals invites degenerate behavior. To avoid such degeneracies, Buchin [5] proposed several variants of continuous DTW distances (originally called *average Fréchet* distance) that each penalise the speed of the dog and its owner. The existing curve similarity measures each have their corresponding drawback: The Fréchet distance is not robust versus outliers. The discrete DTW distance is heavily dependent on the sampling rate. The continuous DTW variants are robust to outliers, but they are difficult to compute [4]. Further, all of them fail to capture detours, as can be seen in Figure 1. We present a new curve similarity measure, specifically designed for computing similarities between curves under outliers.

Discrete walks. Given two curves P and Q, we define discrete walks. First, consider the $n \times m$ integer lattice embedded in $\mathbb{R}^2$. We can construct a graph G_{nm} over this lattice where the vertices are all lattice points and two lattice points l_1, l_2 share an edge whenever $d(l_1, l_2) \leq \sqrt{2}$.

▶ **Definition 1.1.** For curves P and Q, a *discrete reparametrization* F is any walk in G_{nm} from $(1, 1)$ to (n, m). F is a curve in $\mathbb{R}^2$ and it is x-monotone whenever its embedding is. The *speed* $\sigma(F)$ is the size $|S|$ for the largest horizontal or vertical subcurve $S \subseteq F$.

Defining Discrete Barking Distance. The barking distance stems from the following illustration, which is again dog-based:[1] assume you are hiking with constant speed along

[1] This illustration is inspired by a dog that some of the authors met while on a hike.

a curve P. A dog is running at bounded speed on a curve Q, constantly barking at you. However, the dogs barks can only be heard within radius $\rho \in \mathbb{R}$. The dog tries to optimize its route in order to maximize the time you hear it. This maximum time is the barking distance of P to Q. Formally, for $\rho \in \mathbb{R}$ we define the *threshold* function as follows:

$$\theta_\rho(p,q) = \begin{cases} 1 & \text{if } d(p,q) > \rho \\ 0 & \text{otherwise.} \end{cases}$$

▶ **Definition 1.2.** For curves P and Q, denote by $\mathbb{F}$ the set of all pairs of discrete x-monotone reparametrizations of (P,Q). For any $\rho, s \in \mathbb{R}$, the discrete barking distance is defined as:

$$\mathbb{D}_B^s(P,Q) = \min_{\substack{F \in \mathbb{F} \\ \sigma(F) \leq s}} \sum_{(i,j)\in F} \theta_\rho(p_i, q_j).$$

2 Computing the Discrete Barking Distance

Let $G_{n,m} = (V,E)$ be a graph defined on top of an $n \times m$ lattice in $\mathbb{R}^2$ where $v_{i,j} \in V$ is identified with the lattice point at coordinate (i,j). We find $(v_{i,j}, v_{i',j'}) \in E$ with distinct $v_{i,j}, v_{i',j'} \in V$ whenever $v_{i,j}$ and $v_{i',j'}$ are identified with points of the lattice at distance $\leq \sqrt{2}$. We say that $v_{i',j'}$ is the *southern*, *south-western*, *western*, *north-western*, or *northern* neighbor of $v_{i,j}$ if $v_{i',j'}$ lies in the corresponding cardinal direction in the lattice.

For $v_{i,j} \in V$ we set $w(v_{i,j}) = \theta(p_i, q_j)$, with p_i the i-th corner of P and q_j the j-th corner of Q. Similarly, we set $w(\pi) = \sum_{a=0}^{k} w(v_{i_a,j_a})$ for a walk $\pi = (v_{i_1,j_1}, \ldots, v_{i_k,j_k})$ in $G_{n,m}$. We say that π is *monotone* if $j_a \leq j_{a'}$ whenever $a \leq a'$ and we define the *length* of π as $|\pi|$, i.e., the number of vertices in the walk. A sub-walk of π is said to be *horizontal* if all its vertices correspond to lattice points with the same y-coordinate and *vertical* if all its vertices correspond to lattice points with the same x-coordinate. Moreover, we say that π has *speed* s if the longest horizontal or vertical sub-walk of π has length at most s. Let $\Pi(s,\rho)$ be the set of all monotone walks in $G_{n,m}$ starting at $v_{1,1}$ and ending at $v_{n,m}$ with speed s and weight function depending on the threshold ρ. The next observation now follows from Definitions 1.1 and 1.2.

▷ Observation 1. Given two polygonal curves P and Q, a threshold ρ, and a speed bound s, let $G_{n,m}$ be defined as above, then $w(\pi) = \mathbb{D}_B^s(P,Q)$ for any $\pi \in \Pi(s,\rho)$ of minimum weight.

By Observation 1 we can restrict our attention to monotone paths from $v_{1,1}$ to $v_{n,m}$ that have speed at most s and are of minimum weight. Our strategy is to compute for each vertex $v_{i,j} \in V$ the weight of such a path from $v_{1,1}$ and to $v_{i,j}$. Our computation will proceed in n rounds, where in each round we consider the m vertices of column j. The challenge is to compute the length of a minimum weight monotone path of speed s in time $O(\log s)$.

Let $R_i(j_1,j_2)$ be the weight of path $(v_{i,j_1+1}, v_{i,j_1+2}, \ldots, v_{i,j_2})$ and $C_j(i_1,i_2)$ be the weight of path $(v_{i_1+1,j}, v_{i_1+2,j}, \ldots, v_{i_2,j})$. Observe, that these values can be computed in constant time if we have arrays containing at position i the length of a path from the first element in the row or column to the i-th element of the row or column. For each row and column and taking either side as the starting vertex. We precompute these arrays for all rows at the beginning and for each column only when we process this column in the computation.

Let $F_\delta(i,j)$ with $\delta \in D = \{\uparrow, \nearrow, \rightarrow, \searrow, \downarrow\}$ be the minimum weight of a monotone path of speed s from $v_{1,1}$ to $v_{i,j}$ where the vertex preceding $v_{i,j}$ on the path is the southern, south-western, western, north-western, or northern neighbor of $v_{i,j}$, respectively. We set $F_\delta(i,j) = \infty$ if $v_{i,j}$ cannot be reached with any monotone path of speed s from $v_{1,1}$.

We then compute the minimum weight monotone path of speed s from $v_{1,1}$ to $v_{i,j}$ as $F(i,j) = \min\{F_\delta(i,j) \mid \delta \in D\}$. To compute $F(i,j)$ from left to right along the columns we maintain the relevant minima of paths F_δ ending at vertices around $v_{i,j}$ for each row and for the current column in separate heaps. Moreover, instead of updating the weights of all heap-elements explicitly for each $v_{i,j}$, we precompute the lengths of paths starting at the beginning or end of a row or column. From this we can in constant time compute the necessary offsets. The runtime of $O(nm \log s)$ then follows as every of the $O(nm)$ elements gets only inserted and deleted from some min-heap a constant number of times and at no point any min-heap contains more than s elements.

For the following proof we rewrite $F_d(i,j)$ as a recurrence taking the speed-bound s into account for $j > 1$. Recall that $w(v_{i,j})$ contributes to the values of C_j and R_i.

$$F_d(i,j) = \begin{cases} \min\{C_j(i-k,i) + F_\delta(i-k,j) \mid \delta \in \{\nearrow, \rightarrow, \searrow\} \wedge k \in [1,s]\} & \text{if } d = \uparrow \\ F(i-1,j-1) + w(v_{i-1,j-1}) & \text{if } d = \nearrow \\ \min\{R_i(j-k,j) + F_\delta(i,j-k) \mid \delta \in \{\uparrow, \nearrow, \searrow, \downarrow\} \wedge k \in [1,s]\} & \text{if } d = \rightarrow \\ F(i+1,j+1) + w(v_{i+1,j+1}) & \text{if } d = \searrow \\ \min\{C_j(i+k,i) + F_\delta(i+k,j) \mid \delta \in \{\nearrow, \rightarrow, \searrow\} \wedge k \in [1,s]\} & \text{if } d = \downarrow \end{cases}$$

▶ **Theorem 2.1.** *Given two polygonal curves P and Q with n and m vertices, respectively, the discrete Barking distance of P to Q can be computed in time $O(nm(\log s))$ where s is the speed bound or time $O(nm \log(nm))$ if $s > n$ or $s > m$.*

Proof. For the first column, i.e., $j = 1$, we can compute $F(i,j)$ as follows. Clearly, $F(1,1) = w(v_{1,1}) = \theta_\rho(p_1, q_1)$. We set $F(i,1) = \infty$ for all $i > s$. Finally, we find that the remaining entries $F(i,1)$ with $i \in [2,s]$ in a bottom-up traversal as the values $C_1(1,i)$. We conclude this step by initializing a min-heap H_i for each row i containing vertex $v_{i,1}$ as its sole element and $F(i,1) = F_\rightarrow(i,1)$ as the key.

Assume now that we want to compute the entries $F_d(i,j)$ for $i \in [1,m]$ where all entries $F_d(i,j')$ with $j' < j$ are already computed and for row i we have a min-heap H_i containing all $F_\rightarrow(i,j-k)$ for $k \in [1,s]$ ordered by key $F_\rightarrow(i,j-k) + R_i(j-k,j-1)$. From this information we can for each i immediately compute $F_\rightarrow(i,j)$ as the minimum of H_i, say $v_{i,j'}$ plus $R_i(j-j',j)$. We then update H_i by deleting all entries for $v_{i,j-s}$ and then inserting $(v_{i,j}, F_\uparrow(i,j))$, $(v_{1,j}, F_\downarrow(i,j))$, $(v_{1,j}, F_\searrow(i,j))$, and $(v_{1,j}, F_\nearrow(i,j))$ using as key for comparison $F.(i,j) + R_i(j-k,j)$ in the insertion. Note that since for all elements already present in H_i their keys change only by $w(v_{i,j})$ and hence their order remains the same. Moreover, we can directly compute $F_\nearrow(i,j)$ and $F_\searrow(i,j)$ for each i.

It remains to compute $F_\uparrow(i,j)$ and $F_\downarrow(i,j)$ for each $i \in [1,m]$ in column j. We describe how to compute $F_\uparrow(i,j)$, $F_\downarrow(i,j)$ can be computed symmetrically. We start from $v_{1,j}$. Clearly, $F_\uparrow(1,j) = \infty$. We also initialize a min-heap H and insert $(v_{1,j}, F_\rightarrow(1,j))$, $(v_{1,j}, F_\searrow(1,j))$, and $(v_{1,j}, F_\nearrow(1,j) = \infty)$ where the second element is used as key. Assume that we now want to compute $F_\uparrow(i,j)$ and that we have a heap H containing for $k \in [1,s]$ the elements $(v_{i-k,j}, F_\rightarrow(i-k,j))$, $(v_{i-k,j}, F_\searrow(i-k,j))$, and $(v_{i-k,j}, F_\nearrow(i-k,j))$ ordered by key $F.(i-k,j) + C_j(i-k,i-1)$. This can be done as for the row by just extracting the minimum element from the heap H, say $(v_{i',j}, F_\delta(i',j))$, and setting $F_\uparrow(i,j) = F_\delta(i',j) + C_j(i',i)$. We update the heap as in the case of H_i, with the only difference being that we need to insert the three elements $(v_{i,j}, F_\rightarrow(i,j))$, $(v_{1,j}, F_\searrow(i,j))$, and $(v_{1,j}, F_\nearrow(i,j))$

Correctness follows since the algorithm computes directly the above recurrence. Moreover, since every element vertex and partial weight combination gets deleted and inserted at

most once from some heap over the whole computation and no heap contains more than $3s$ elements at a time, we obtain the claimed running time of $O(nm \log s)$. Note, that if $s > m$ or $s > n$ we obtain a runtime of $O(nm(\log(m) + \log(n)))$ since again never more than $O(s)$ elements are contained in a heap and no more than $O(nm)$ elements can be inserted or deleted. ◀

3 Outlook and Conclusion

In the full version of this paper, we also study the barking distance in two other settings, namely the *semi-discrete* and the *continuous* setting. In the semi-discrete setting, the traversal of Q is continuous while the one of P is again discrete. We show the following.

▶ **Theorem 3.1.** *Given two polygonal curves P and Q with n and m vertices, respectively, the semi-discrete Barking distance of P to Q can be computed in time $O(nm \log(nm))$.*

In the continuous setting, both traversals are continuous. Here our algorithm is slower, but still polynomial.

▶ **Theorem 3.2.** *Given two polygonal curves P and Q with n and m vertices, respectively, the continuous Barking distance of Q to P can be computed in time $O(n^4 m^3 \log(nm))$.*

For all the settings we show that, assuming the Strong Exponential Time Hypotheis (SETH), no truly subquadratic algorithm can exist.

▶ **Theorem 3.3.** *Let P and Q be two disjoint polygonal curves with n vertices. Assuming OVC, solving the barking decision problem where the maximum speed of the dog matches the speed of the hiker and with constant barking radius ρ requires $\Omega(n^{2-\epsilon})$ time for any $\epsilon > 0$.*

In the discrete and semi-discrete setting, the runtime of our algorithms match the lower bound up to logarithmic factors. For the continuous setting we give an algorithm that is likely not optimal. We believe that using techniques as for the proof of 3.1 we can improve the runtime to $O(nm^3 \log m)$, but this would still leave a gap between upper and lower bound. While we conjecture that it is possible to obtain an $O(nm \log(nm))$ algorithm, it is likely that new ideas are necessary for this. It would also be interesting to find more efficient algorithms in the continuous setting for restricted types of curves such as time series. Throughout our paper, we assumed that the barking radius ρ and the speed bound s are fixed. Considering them as variables leads to other interesting algorithmic problems where we ask for the minimal speed or barking radius required for the dog such that the hiker can hear it the entire time. We leave the study of these problems for future work.

References

1 Kevin Adistambha, Christian H. Ritz, and Ian S. Burnett. Motion classification using dynamic time warping. In *Proceedings of the 2008 IEEE 10th Workshop on Multimedia Signal Processing (MMSP'08)*, pages 622–627. IEEE, 2008. doi:10.1109/MMSP.2008.4665151.

2 Pankaj K. Agarwal, Sariel Har-Peled, Nabil H. Mustafa, and Yusu Wang. Near-linear time approximation algorithms for curve simplification. *Algorithmica*, 42:203–219, 2005. doi:10.1007/S00453-005-1165-Y.

3 Karl Bringmann and Marvin Künnemann. Quadratic conditional lower bounds for string problems and dynamic time warping. In *Proceedings of the IEEE 56th Annual Symposium on Foundations of Computer Science (FOCS 2015)*, pages 79–97. IEEE, 2015. doi:10.1109/focs.2015.15.

4 Kevin Buchin, André Nusser, and Sampson Wong. Computing continuous dynamic time warping of time series in polynomial time. In *Proceedings of the 38th International Symposium on Computational Geometry (SoCG'22)*, volume 224 of *LIPIcs*, pages 22:1–22:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.SoCG.2022.22.

5 Maike Buchin. *On the computability of the Fréchet distance between triangulated surfaces*. PhD thesis, FU Berlin, 2007. doi:10.17169/refubium-6111.

6 Siu-Wing Cheng and Haoqiang Huang. Curve simplification and clustering under Fréchet distance. In *Proceedings of the 2023 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'23)*, pages 1414–1432. SIAM, 2023. doi:10.1137/1.9781611977554.CH51.

7 Anne Driemel, Amer Krivošija, and Christian Sohler. Clustering time series under the Fréchet distance. In *Proceedings of the 2016 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'16)*, pages 766–785. SIAM, 2016. doi:10.1137/1.9781611974331.CH55.

8 Young-Seon Jeong, Myong K. Jeong, and Olufemi A Omitaomu. Weighted dynamic time warping for time series classification. *Pattern recognition*, 44(9):2231–2240, 2011. doi:10.1016/J.PATCOG.2010.09.022.

9 Rohit J. Kate. Using dynamic time warping distances as features for improved time series classification. *Data Mining and Knowledge Discovery*, 30:283–312, 2016. doi:10.1007/S10618-015-0418-X.

10 Joan Serra, Emilia Gómez, Perfecto Herrera, and Xavier Serra. Chroma binary similarity and local alignment applied to cover song identification. *IEEE Transactions on Audio, Speech, and Language Processing*, 16(6):1138–1151, 2008. doi:10.1109/TASL.2008.924595.

11 E Sriraghavendra, K Karthik, and Chiranjib Bhattacharyya. Fréchet distance based approach for searching online handwritten documents. In *Proceedings of the 9th international conference on document analysis and recognition (ICDAR'07)*, volume 1, pages 461–465. IEEE, 2007. doi:10.1109/ICDAR.2007.4378752.

12 Koh Takeuchi, Masaaki Imaizumi, Shunsuke Kanda, Yasuo Tabei, Keisuke Fujii, Ken Yoda, Masakazu Ishihata, and Takuya Maekawa. Fréchet kernel for trajectory data analysis. In *Proceedings of the 29th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (SIGSPATIAL'21)*, pages 221–224, 2021. doi:10.1145/3474717.3483949.

13 Stephen J. Taylor. *Modelling financial time series*. World cientific, 2008.

14 Mees van de Kerkhof, Irina Kostitsyna, Maarten Löffler, Majid Mirzanezhad, and Carola Wenk. Global curve simplification. In *Proceedings of the 27th Annual European Symposium on Algorithms (ESA'19)*, volume 144 of *LIPIcs*, pages 67:1–67:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPICS.ESA.2019.67.

15 Weizeng Wang, Gaofan Lyu, Yuliang Shi, and Xun Liang. Time series clustering based on dynamic time warping. In *Proceedings of the 2018 IEEE 9th International Conference on Software Engineering and Service Science (ICSESS'18)*, pages 487–490. IEEE, 2018. doi:10.1109/ICSESS.2018.8663857.

16 Öz Yilmaz. *Seismic data analysis: Processing, inversion, and interpretation of seismic data*. Society of Exploration Geophysicists, 2001. doi:10.1190/1.9781560801580.fm.

17 Jianbin Zheng, Xiaolei Gao, Enqi Zhan, and Zhangcan Huang. Algorithm of on-line handwriting signature verification based on discrete Fréchet distance. In *Proceedings of the 3rd International Symposium on Intelligence Computation and Applications: Advances in Computation and Intelligence (ISICA'08)*, volume 5370 of *LNCS*, pages 461–469. Springer, 2008. doi:10.1007/978-3-540-92137-0_51.

The Number of Non-overlapping Edge Unfoldings in Convex Regular-faced Polyhedra

Takumi Shiota[1], Yudai Enomoto[2], Takashi Horiyama[2], and Toshiki Saitoh[1]

1 **Kyushu Institute of Technology, Fukuoka, Japan**
shiota.takumi779@mail.kyutech.jp, toshikis@ai.kyutech.ac.jp
2 **Hokkaido University, Hokkaido, Japan**
yudai1006@gmail.com, horiyama@ist.hokudai.ac.jp

Abstract

An edge unfolding of a polyhedron is a flat polygon obtained by cutting along the polyhedron's edges and unfolding the polygon onto a plane. It is known that the number of edge unfoldings is equal to the number of spanning trees formed by the cutting edges of the polyhedron. However, some edge unfoldings overlap, i.e., two distinct faces in the edge unfoldings overlap, so they cannot be embedded in the plane. We do not know the percentage of the overlapping edge unfoldings for almost all polyhedra. In particular, there exists an interesting and well-known open problem of whether or not all convex polyhedrons have a non-overlapping edge unfolding. Horiyama et al. proposed an enumeration algorithm for edge unfoldings using zero-suppressed binary decision diagrams (ZDDs), which are compact data structures for families of sets. The ZDDs have attractive family algebraic operations; for example, we can extract sets satisfying some constraints from the family of sets over ZDDs efficiently. In this study, we propose an enumeration algorithm for non-overlapping edge unfoldings in a polyhedron using ZDDs and their operations. The algorithm first enumerates the minimal overlapping partial edge unfoldings (MOPEs) obtained through the "rotational unfolding" by Shiota and Saitoh. Then, we subtract the overlapping edge unfoldings containing the MOPEs from all edge unfoldings over ZDDs. We apply the algorithm to convex regular-faced polyhedra (including three types of Archimedean solids, twenty types of Johnson solids, nineteen types of Archimedean prisms, and twenty-one types of Archimedean antiprisms) and show the number of non-overlapping edge unfoldings for each type of polyhedron.

1 Introduction

An edge unfolding of a polyhedron is a flat polygon obtained by cutting along the polyhedron's edges and unfolding the polygon onto a plane. The origin of edge unfoldings is recognized as the illustrations found in Albrecht Dürer's "Underweysung der messung mit dem zirckel un richt scheyt" [5] published in 1525 [3]. However, the edge unfoldings can sometimes result in overlapping polygons, i.e., two distinct faces overlap, or their boundaries are in contact (Figure 1). In Dürer's book, all the polyhedra are drawn as edge unfoldings

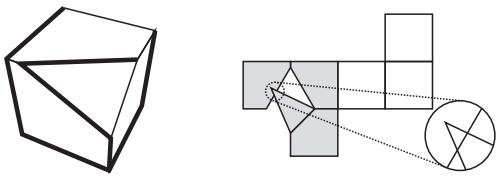


Figure 1 A cube with cut-off corners and its overlapping edge unfolding. The faces shown in gray are a MOPE.

40th European Workshop on Computational Geometry, Ioannina, Greece, March 13–15, 2024.
This is an extended abstract of a presentation given at EuroCG'24. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

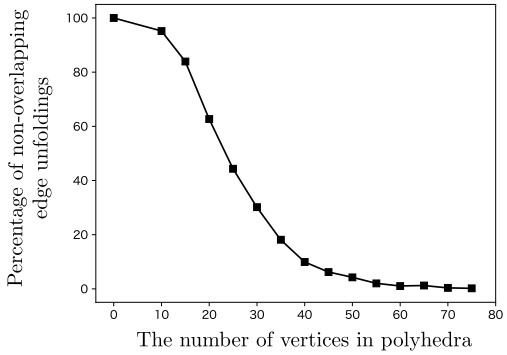


Figure 2 The line graph represents the percentage of non-overlapping edge unfoldings among 1000 randomly selected edge unfoldings. Each point on the graph represents the average values for five randomly generated convex polyhedra [15].

Table 1 Overlapping edge unfoldings for convex regular-faced polyhedra

Convex regular-faced polyhedra	Is there an overlapping edge unfolding?
Platonic solids (Total 5 types)	No [8]
Archimedean solids (Total 13 types)	No (7 types) / Yes (6 types) [2, 8, 6, 18]
Johnson solids (Total 92 types)	No (48 types) / Yes (44 types) [17]
n-gonal Archimedean prisms ($n \geq 3$)	No ($3 \leq n \leq 23$) / Yes ($n \geq 24$) [18]
m-gonal Archimedean antiprisms ($m \geq 3$)	No ($3 \leq m \leq 11$) / Yes ($m \geq 12$) [18]

without overlaps. Focusing on this point, Shephard proposed the following conjecture.

▶ **Conjecture 1.1** ([16]). *For any convex polyhedron, at least one non-overlapping edge unfolding exists.*

This conjecture is still unsolved, and some studies to solve it are ongoing. One of the studies is Schevon's experiment on randomly generated convex polyhedra [15]. She showed that the percentage of non-overlapping edge unfoldings decreases as the number of vertices increases (Figure 2). Some studies have reported the existence of an overlapping edge unfolding in a given polyhedron. Shiota and Saitoh presented an algorithm "rotational unfolding" that can quickly find an overlapping edge unfolding of a polyhedron, and they showed the existence of overlapping edge unfoldings for convex regular-faced polyhedra (Table 1).

It is known that the number of edge unfoldings is equal to the number of spanning trees formed by the cutting edges of the polyhedron. We can count the number of spanning trees using Kirchhoff's theorem [13] or a data structure called binary decision diagrams (BDD) [1] / zero-suppressed binary decision diagram (ZDD) [14]. The BDDs/ZDDs represent compact data structures for families of sets and have family algebraic operations (i.e., union, intersection, and set difference). In addition, BDDs/ZDDs allow for the counting, enumerating, and extracting of optimal families of sets. BDDs/ZDDs have been applied to enumerate specific structures on graphs [12]. Horiyama et al. enumerated spanning trees using BDDs/ZDDs and counted the number of convex regular-faced polyhedra [9, 7]. Horiyama and Shoji proposed a method for counting the number of non-overlapping edge unfolding for Platonic solids by extracting each spanning tree one by one from BDDs [8]. However, this method only applies to the polyhedra with few edge unfoldings. For example, the truncated icosahedron (Figure 3) has $375,291,866,372,898,816,000$ (approximately 3.75×10^{20}) edge

unfoldings [9], so checking each unfolding individually, it would take over ten thousand years with current computers.

In this study, we propose an enumeration algorithm for non-overlapping edge unfoldings in a polyhedron using ZDDs and their operations. The algorithm first enumerates the minimal overlapping partial edge unfoldings (MOPEs), which are the minimal units of edge unfoldings with overlaps obtained through the rotational unfolding [18]. Then, we subtract the overlapping edge unfoldings containing the MOPEs from all edge unfoldings over ZDDs.

We apply this counting method to convex regular-faced polyhedra, including three types of Archimedean solids, twenty types of Johnson solids, nineteen types of Archimedean prisms, and twenty-one types of Archimedean antiprisms, and show the number of non-overlapping edge unfoldings for each type of polyhedron.

2 Preliminaries

A *polyhedron* is a three-dimensional object consisting of at least four polygons, called *faces*, joined at their edges. A *convex polyhedron* is a polyhedron with the interior angles between any two adjacent faces less than π. A *convex regular-faced polyhedron* is a convex polyhedron with all faces being regular polygons. A *Platonic solid* is a convex regular-faced polyhedron with all faces composed of congruent regular polygons. An n *prism*, where $n \geq 3$, is a polyhedron composed of two identical n-sided polygons, called *bases*, facing each other, and n parallelograms, called *side faces*, connecting the corresponding edges of the two bases. An m *antiprism*, where $m \geq 3$, is a polyhedron composed of two congruent m-sided polygonal bases and $2m$ triangular side faces alternating around the bases. An *n-gonal (anti)prism* is an n (anti)prism if the bases are n-sided regular polygons. An *n-gonal Archimedean (anti)prism* is an n-gonal (anti)prism if it is a convex regular-faced polyhedron (i.e., the side faces are regular squares (or triangles)). An *Archimedean solid* is a convex regular-faced polyhedron composed of regular polygons with the same type and order of regular polygons gathered at the vertices, except for Platonic solids and Archimedean (anti)prisms. A *Johnson solid* is a convex regular-faced polyhedron, except Platonic solids, Archimedean solids, and Archimedean (anti)prisms. There are 92 Johnson solids [11].

Let Q be a polyhedron. Two faces in Q are *neighbors* if they contain a common edge. An *unfolding* (also called a net, a development, or a general unfolding) of the polyhedron Q is a flat polygon formed by cutting Q's edges or faces and unfolding it into a plane. An *edge unfolding* of Q is an unfolding formed by cutting only Q's edges. Q can be viewed as a graph $G_Q = (V_Q, E_Q)$, where V_Q is a set of faces in the polyhedron, and E_Q is a set of edges such that two vertices are adjacent if and only if the corresponding two faces are neighbors. The following lemma applies to an edge unfolding of Q.

▶ **Lemma 2.1** (see e.g., [19], Theorem 2.2.1 and its proof)**.** *The set of non-cutting edges for*

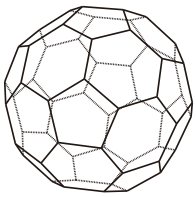


Figure 3 A truncated icosahedron (a type of Archimedean solid)

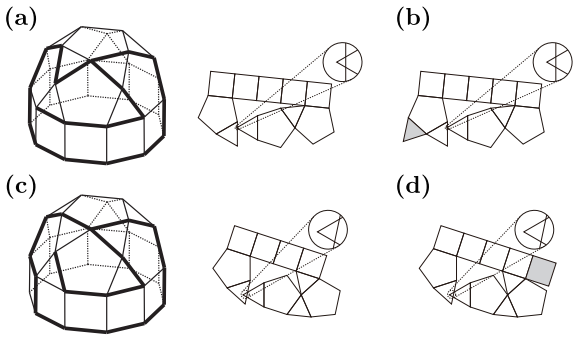


Figure 4 (a),(c) MOPEs in J21 (a type of Johnson solid). Removing any face from each MOPE results in non-connected structures, contradicting the definition of partial edge unfoldings. (b),(d) Partial edge unfoldings in J21 that are not MOPE. Removing the gray faces results in MOPEs.

an edge unfolding of Q forms a spanning tree of G_Q.

This lemma implies that counting the spanning trees of G_Q is equal to counting the edge unfoldings of Q. A *partial edge unfolding* is a flat polygon formed from a set of faces corresponding to a connected induced subgraph of G_Q.

Two distinct polygons *overlap* if there exists a point p contained in two polygons. Note that any point on a boundary is included in the polygons in this paper; the polygons overlap if they touch at the boundaries. An unfolding is *overlapping* if a pair of distinct faces exists such that the faces overlap. Herein, neighbor faces are not overlapping. The algorithm *rotational unfolding* has been developed to efficiently determine overlaps in a given polyhedron based on [4, 6] ideas [18]. Rotational unfolding can enumerate *minimal overlapping partial edge unfoldings (MOPEs)*, a partial edge unfolding with the minimal number of faces required to connect two overlapping faces. Figure 4 shows the example of MOPEs and non-MOPEs.

One method for counting spanning trees in a graph is using a *Zero-suppressed Decision Diagram (ZDD)*. A ZDD is a data structure representing families of sets compactly as a directed acyclic graph. In a ZDD, there are two types of nodes: *terminal nodes* with the out-degree zero $\top$, $\bot$, and *branching nodes*. Branching nodes are labeled by elements of the set, and each has two outgoing edges: a 1-*edge* and a 0-*edge*. The 1-edge indicates the inclusion of the labeled element, while the 0-edge indicates the exclusion of the element. In a ZDD, there is a *root node* with no incoming edges. For example, the ZDD, which represents a spanning tree as shown in Figure 5, and a path from the root node (labeled e_0) following a 1-edge, a 1-edge, a 0-edge, and a 1-edge leading to $\top$ means that the set $\{e_0, e_1, e_3\}$ forms a spanning tree. ZDDs have some operations, such as computing the union or intersection of two ZDDs [14].

3 Counting algorithm for non-overlapping edge unfoldings

In this section, we describe an algorithm counting the number of non-overlapping edge unfoldings for any polyhedron Q. Let P be a (partial) edge unfolding of Q, and let $G(P) = (V(P), E(P))$ be the graph corresponding to P. The following lemma holds.

▶ **Lemma 3.1.** *For any overlapping edge unfolding U, there exists a MOPE C such that $E(C) \subseteq E(U)$.*

Let C_i $(1 \leq i \leq k)$ be MOPEs of a polyhedron, where k is the number of MOPEs. From Lemma 3.1, the following claim holds.

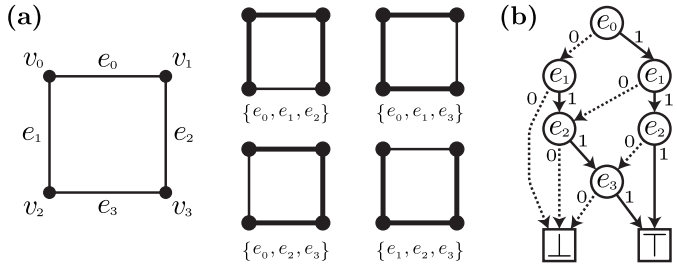


Figure 5 (a) An example of the graph C_4 and its spanning trees. (b) A ZDD representing the spanning trees of C_4. Circles represent branching nodes, labels are inside the circles, solid lines represent 1-edges, and dashed lines represent 0-edges.

▶ Claim 3.2. Let U be a non-overlapping edge unfolding. For any MOPE C_i, $E(C_i) \nsubseteq E(U)$.

The number of edge unfoldings can be counted by constructing ZDD $\mathcal{Z}_S$ [12]. However, it contains overlapping edge unfoldings. To exclude these overlapping unfoldings, we employ the *subsetting method*, an operation over ZDDs [10]. For a ZDD $\mathcal{Z}$, the subsetting method generates a new ZDD $\mathcal{Z}_C$ by extracting combinations satisfying a constraint C from $\mathcal{Z}$.

We can count the number of non-overlapping edge unfoldings by following three steps.

Step 1 Generate a ZDD $\mathcal{Z}_S$.

Step 2 For each MOPE C_i, generate a new ZDD $\mathcal{Z}_{C_i}$ representing combinations that do not simultaneously include all elements of $E(C_i)$. Herein, a ZDD Z_{C_i} serves as a filter to exclude overlapping edge unfoldings.

Step 3 Generate a new ZDD $\mathcal{Z}_C$ by extracting combinations that satisfy all $\mathcal{Z}_{C_i}$ from $\mathcal{Z}_S$ using the subsetting method.

4 The number of non-overlapping edge unfoldings in convex regular-faced polyhedra

In this section, we apply the counting algorithm described in Section 3 for non-overlapping edge unfoldings with regular convex regular-faced polyhedra (including three types of Archimedean solids, twenty types of Johnson solids, nineteen types of Archimedean prisms, and twenty-one types of Archimedean antiprisms). We use TdZdd library* for constructing ZDD $\mathcal{Z}_S$ and $\mathcal{Z}_{C_i}$. Experiments were conducted on a Mac OS Venture computer with an Apple M1 Max chip and 64GB of memory. To enumerate the MOPEs for Archimedean solids, Johnson solids, and Archimedean (anti)prism, we used rotational unfolding [18, 17].

We show the number of non-overlapping edge unfoldings for three types of Archimedean solids (Table 2), twenty types of Johnson solids, nineteen types of Archimedean prisms, and twenty-one types of Archimedean antiprisms†. Figure 6 shows the percentage of non-overlapping edge unfoldings from all edge unfoldings for Archimedean (anti)prisms.

From the results of these experiments, we observe the following: In Archimedean solids (Table 2), both the truncated dodecahedron and truncated icosahedron have the same number of vertices, edges, and faces, yet the truncated icosahedron has more MOPEs than that of truncated dodecahedron. Despite this, the results indicate that the percentage of

* `https://github.com/kunisura/TdZdd`

† See `https://shiotatakumi.github.io/MyPage/contents/240313-EuroCG-2024.html` for the number and percentage of non-overlapping edge unfoldings in Johnson solids, and Archimedean (anti)prisms.

Table 2 The number and percentage of non-overlapping edge unfoldings in Archimedean solids.

Archimedean solids	$\|V\|$	$\|E\|$	$\|F\|$	$\#(MOPE)$	#(Edge unfolding) [9]	#(Non-overlapping edge unfolding)	%
Sunb cube	24	60	38	72	89,904,012,853,248	85,967,688,920,076	95.6
Truncated dodecahedron	60	90	32	120	4,982,259,375,000,000,000	931,603,573,888,462,350	18.6
Truncated Icosahedron	60	90	32	240	375,291,866,372,898,816,000	366,359,657,802,290,909,354	97.6

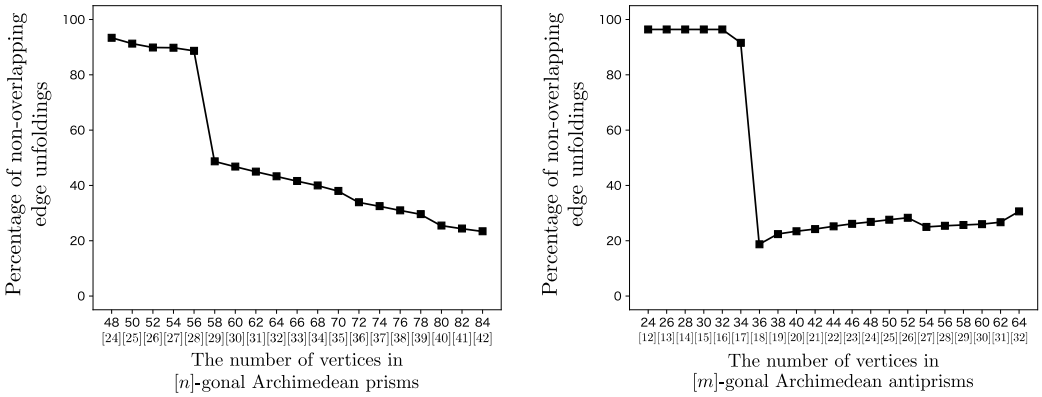


Figure 6 The relationship between the number of vertices and the percentage of non-overlapping edge unfoldings in n-gonal Archimedean prisms and m-gonal Archimedean antiprisms.

non-overlapping edge unfoldings in the truncated dodecahedron is lower than that of the truncated icosahedron. The truncated icosahedron's MOPEs consist of 8 or 9 faces (Figure 7), while the truncated dodecahedron's MOPEs have 4 faces (Figure 8). Therefore, we observe that the number of faces constituting each MOPE, rather than the number of MOPEs, influences the percentage of non-overlapping edge unfoldings. The same observation also applies to Archimedean (anti)prisms. In n-gonal Archimedean prisms, the percentage of non-overlapping edge unfoldings significantly decreases at $n = 29$, as shown in Figure 6 (left). This decrease may be attributed to the appearance of two new types of MOPEs, consisting of 4 faces for $n \geq 29$, as illustrated in Figure 10 (for $n \leq 28$, the MOPEs consist of 6, 7, or 8 faces (Figure 9)). Similarly, in m-gonal Archimedean antiprisms, the percentage of non-overlapping edge unfoldings significantly decreases at $m = 18$, as shown in Figure 6 (right). This decrease may be attributed to the appearance of three new types of MOPEs, consisting of 6 faces for $m \geq 18$, as illustrated in Figure 12 (for $m \leq 17$, the MOPEs consist of 8 faces (Figure 11)).

Acknowledgments. This work was supported in part by JSPS KAKENHI Grant Numbers JP18H04091, JP19K12098, and JP22H03549, by MEXT KAKENHI Grant Number JP20H05964, and by JST SPRING Grant Number JPMJSP2154.

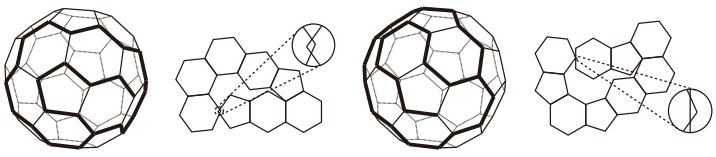


Figure 7 MOPEs in the truncated icosahedron [9, 18]

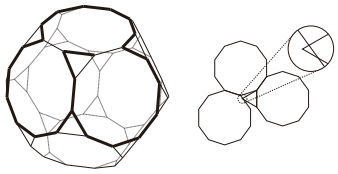


Figure 8 A MOPE in the truncated dodecahedron [9]

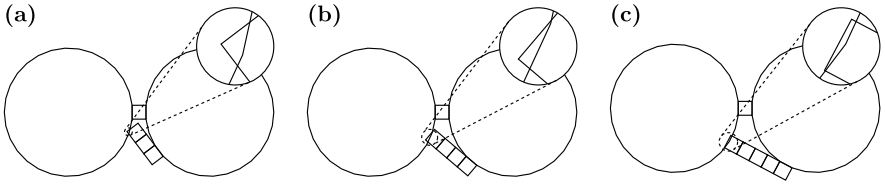


Figure 9 MOPEs in n-gonal Archimedean prisms for (a) $n \geq 24$, (b) $n \geq 26$, and (c) $n \geq 28$.

References

1 Randal E. Bryant. Graph-based algorithms for Boolean function manipulation. *IEEE Trans. Computers*, 35(8):677–691, 1986.

2 Hallard T. Croft, Kenneth J. Falconer, and Richard K. Guy. *Unsolved Problems in Geometry*. Springer-Verlag, reissue edition, 1991.

3 Erik D. Demaine and Joseph O'Rourke. *Geometric Folding Algorithms: Linkages, Origami, Polyhedra*. Cambridge University Press, 2007.

4 Kristin DeSplinter, Satyan L. Devadoss, Jordan Readyhough, and Bryce Wimberly. Nets of higher-dimensional cubes. In *32nd Canadian Conference on Computational Geometry*, 2020. URL: `https://cccg.ca/proceedings/2020/proceedings.pdf`.

5 Albrecht Dürer. Underweysung der messung, mit dem zirckel und richtscheyt in linien ebenen unnd gantzen corporen, 1525.

6 Kenta Hirose. Hanseitamentai no tenkaizu no kasanari ni tsuite (On the overlap of Archimedean solids), in Japanese, 2015. Saitama Univ. graduation thesis. Supervisor : Takashi Horiyama.

7 Takashi Horiyama, Masahiro Miyasaki, and Riku Sasaki. Isomorphism elimination by zero-suppressed binary decision diagrams. In *30th Canadian Conference on Computational Geometry*, 2018. URL: `https://home.cs.umanitoba.ca/~cccg2018/papers/session7B-p2.pdf`.

8 Takashi Horiyama and Wataru Shoji. Edge unfoldings of Platonic solids never overlap. In *23rd Canadian Conference on Computational Geometry*, 2011. URL: `http://www.cccg.ca/proceedings/2011/papers/paper107.pdf`.

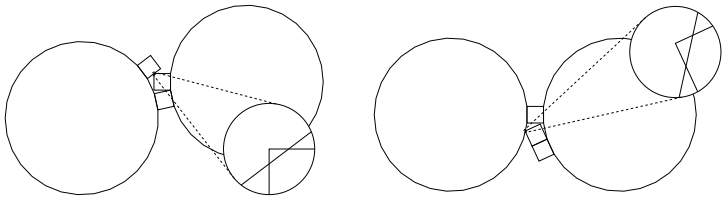


Figure 10 New MOPEs in n-gonal Archimedean prisms in $n = 29$.

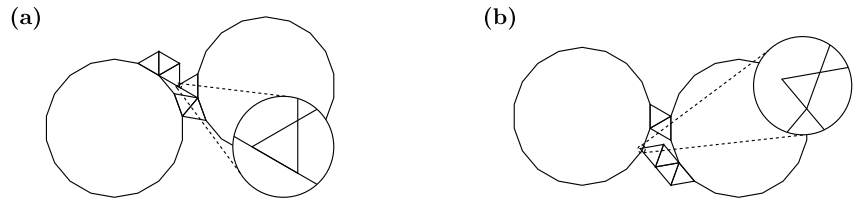


Figure 11 MOPEs in m-gonal Archimedean antiprisms for (a) $m \geq 12$, and (b) $m \geq 17$.

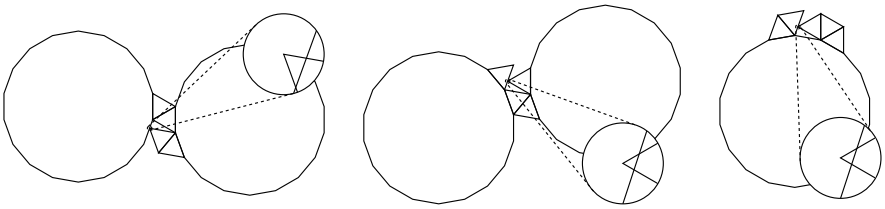


Figure 12 New MOPEs in m-gonal Archimedean antiprisms in $m = 18$.

9 Takashi Horiyama and Wataru Shoji. The number of different unfoldings of polyhedra. In *24th International Symposium on Algorithms and Computation*, volume 8283 of *LNCS*, pages 623–633. Springer, 2013.

10 Hiroaki Iwashita and Shinichi Minato. Efficient top-down ZDD construction techniques using recursive specifications. Technical Report TCS-TRA-1369, Graduate School of Information Science and Technology, Hokkaido University, 2013.

11 Norman Johnson. Convex polyhedra with regular faces. *Canadian Journal of Mathematics*, 18:169–200, 01 1966.

12 Jun Kawahara, Takeru Inoue, Hiroaki Iwashita, and Shin-ichi Minato. Frontier-based search for enumerating all constrained subgraphs with compressed representation. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, E100-A(9):1773–1784, 2017.

13 Mordechai Lewin. A generalization of the matrix-tree theorem. *Mathematische Zeitschrift*, 181(1), 1982.

14 Shin-ichi Minato. Zero-suppressed BDDs for set manipulation in combinatorial problems. In *Proceedings of the 30th Design Automation Conference*, pages 272–277. ACM Press, 1993.

15 Catherine Anne Schevon. *Algorithms for geodesics on convex polytopes*. PhD thesis, The Johns Hopkins University, 1989.

16 G. C. Shephard. Convex polytopes with convex nets. *Mathematical Proceedings of the Cambridge Philosophical Society*, 78(3):389–403, 1975.

17 Takumi Shiota. Overlapping edge unfoldings for convex regular-faced polyhedrons, 2023. Kyushu Institute of Technology master's thesis. Supervisor : Toshiki Saitoh.

18 Takumi Shiota and Toshiki Saitoh. Overlapping edge unfoldings for Archimedean solids and (anti)prisms. In *17th International Conference and Workshops of Algorithms and Computation*, volume 13973 of *LNCS*, pages 36–48. Springer, 2023.

19 Ryuhei Uehara. *Introduction to Computational Origami: The World of New Computational Geometry*. Springer, 2020.

The Complexity of the Lower Envelope of Collections of Various Geometric Shapes

Carlos Alegría[1], Anna Brötzner[2], Bengt J. Nilsson[2], Christiane Schmidt[3], and Carlos Seara[4]

1 **Dipartimento di Ingegneria, Università Roma Tre, Italy**
carlos.alegria@uniroma3.it

2 **Department of Computer Science and Media Technology, Malmö University, Malmö, Sweden**
{anna.brotzner,bengt.nilsson.TS}@mau.se

3 **Department of Science and Technology, Linköping University, Campus Norrköping, Sweden**
christiane.schmidt@liu.se

4 **Department of Mathematics, Universidad Politécnica de Catalunya, Spain.**
carlos.seara@upc.edu

Abstract

We study the problem of determining the complexity of the lower envelope of a collection of n geometric objects. For collections of rays; unit length line segments; and collections of unit squares to which we apply at most two transformations from translation, rotation, and scaling, we prove a complexity of $\Theta(n)$. If all three transformations are applied to unit squares, then we show the complexity becomes $\Theta\big(n\alpha(n)\big)$, where $\alpha(n)$ is the slowly growing inverse of Ackermann's function.

1 Introduction

Consider a set of n line segments (segments for short) in the plane. It is known that the complexity of their lower envelope is at least $\Omega\big(n\alpha(n)\big)$, where $\alpha(n)$ is the extremely slowly growing inverse of Ackermann's function. The lower bound was proved by Wiernik and Sharir [4], a matching upper bound of $O\big(n\alpha(n)\big)$ was proved by Hart and Sharir [1], and an $O(n \log n)$ time and $O\big(n\alpha(n)\big)$ space algorithm to find such a lower envelope was described by Hershberger [2]. The motivation of this work is to determine under which geometric properties of a given set of n geometric objects we can ensure that their lower envelope has a tight complexity, e.g., linear or $\Theta\big(n\alpha(n)\big)$.

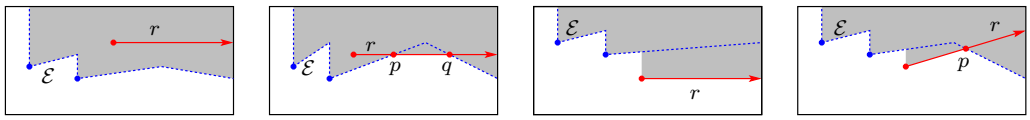
For many of the results, we will make extensive use of Observation 1.1.

▶ **Observation 1.1.** *Let S_1 and S_2 be two sets of n_1 and n_2 planar geometric objects whose lower envelopes have complexity $O\big(f_1(n_1)\big)$ and $O\big(f_2(n_2)\big)$, respectively.*

If any pair of objects in the set $S_1 \cup S_2$ intersect at most $O(1)$ times, then the union of the lower envelopes of S_1 and S_2 has complexity $O\big(f_1(n) + f_2(n)\big)$, where $n = n_1 + n_2$.

The observation follows by merging the two sequences of intervals generated by the corresponding two envelopes, since any two objects will appear at most a constant number of times in the lower envelope where they intersect. Thus, the complexity becomes as stated.

40th European Workshop on Computational Geometry, Ioannina, Greece, March 13–15, 2024.
This is an extended abstract of a presentation given at EuroCG'24. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.



(a) (b) (c) (d)

Figure 1 Illustrating the proof of Theorem 2.1.

2 Collections of Rays

▶ **Theorem 2.1.** *The lower envelope of a set of n rays has a complexity of $\Theta(n)$.*[1]

Proof. Given a ray r, let $s(r)$ and $\ell(r)$ denote the starting point and the supporting line of r, respectively. Without loss of generality, we assume in the following that no ray is vertical, since the lower envelope of any such ray is a single point.

To see the lower bound, consider n rays that move horizontally to the right, with starting points $(1, n), (2, n-1), \ldots, (n, 1)$. The lower envelope has vertices at each integer x-coordinate from 1 to n and therefore complexity $\Omega(n)$.

To see the upper bound, we argue as follows. Let R be the subset of rays that have no point to the left of their starting point and let L be the subset of remaining rays. We show next that the complexity of the lower envelope of R is $O(n)$. By symmetry (mirroring the rays in L along the line $x = 0$), the complexity of the lower envelope of L is also $O(n)$. The upper bound follows by combining these results and Observation 1.1.

For the rays in R, our proof makes use of the following observation.

▶ **Observation 2.2.** *Given two rays in R that intersect, the ray that lies above the other after their intersection point will never again be included in the lower envelope after that point.*

Consider the set of rays in R and sort them by the x-coordinate of the endpoint in order from left to right. By greedily inserting a ray r in this order into the lower envelope $\mathcal{E}$ of the previously added rays we argue that the number of intersection points in the lower envelope can increase by at most two. We have the following cases.

1. The starting point of r lies above $\mathcal{E}$ and r never intersects it, then r is never seen from below and the lower envelope does not change; see Figure 1(a),
2. The starting point of r lies above $\mathcal{E}$ and the ray intersects it at p, then p is a vertex of the new lower envelope. If r does not intersect the current lower envelope to the right of p, then r is the only object seen to the right of p. If r intersects $\mathcal{E}$ again at q, by Observation 2.2, the ray r will not be included in the lower envelope again, and furthermore, since two rays can only intersect once, between p and q, there must be at least one vertex of $\mathcal{E}$. Hence, either one vertex is added to $\mathcal{E}$ or two are added and at least one vertex must also be removed; see Figure 1(b),
3. The starting point of r lies below $\mathcal{E}$ and r never intersects $\mathcal{E}$, then a vertex in the lower envelope is introduced at the x-coordinate of the starting point of r and the ray is the only object seen to the right of this point; see Figure 1(c),
4. The starting point of r lies below $\mathcal{E}$ and r intersects it at p, then a new vertex in the lower envelope is introduced at the x-coordinate of the starting point of r and another one at p. By Observation 2.2, r can never appear again in the lower envelope; see Figure 1(d).

Thus, at most two new vertices are introduced to the lower envelope when we insert a ray. ◀

[1] This result was stated without a proof by Sharir and Agarwal in [3, page 112].

3 Collections of Line Segments with Unit Length

▶ **Theorem 3.1.** *The lower envelope of a set of n unit length segments has complexity $\Theta(n)$.*

Proof. To prove the lower bound, consider the lower envelope of the n unit length segments $[(1,0),(2,0)],[(3,0),(4,0)],\ldots,[(2n-1,0),(2n,0)]$ with $2n$ vertices, establishing the claim.

To prove the upper bound, consider a square grid covering the plane whose cells have side length $3/5$. We denote by $S_{i,j}$ the grid cell at row i and column j. Let $L_{i,j}$ be the set of line segments obtained by intersecting the input set of line segments with the region bounded by $S_{i,j}$, and let $n_{i,j}$ be the number of line segments in $L_{i,j}$.

The lower envelope of $L_{i,j}$ has complexity $O(n_{i,j})$, since the input segments have length 1 and the grid cells have side length $3/5$. The segments of $L_{i,j}$ have either a single endpoint or no endpoints in the interior of $S_{i,j}$. Observe that the subset of segments of $L_{i,j}$ with no endpoints behave as lines inside $S_{i,j}$ and each such segment has at most one connected piece on the lower envelope of $L_{i,j}$. Hence, the lower envelope of this subset has linear complexity [3]. On the other hand, note that the subset of segments of $L_{i,j}$ with a single endpoint behave as rays inside $S_{i,j}$. In particular, Observation 2.2 holds. Therefore, by Theorem 2.1 the lower envelope of this subset has also linear complexity. Combining these observations with Observation 1.1, we conclude that the lower envelope of $L_{i,j}$ has complexity $O(n_{i,j})$.

Let $S_j^{(k)} = \bigcup_{i=-\infty}^{\infty} S_{3i+k,j}$, for $k \in \{0,1,2\}$, be the union of the cells in a grid column that are three cells apart. We say that $S_j^{(k)}$ is the k^{th} *sub-strip* of the j^{th} grid column. We denote by $L_j^{(k)} = \bigcup_{i=-\infty}^{\infty} L_{3i+k,j}$ the set of line segments resulting from intersecting the input set of line segments with $S_j^{(k)}$. Let the number of segments of each set $L_j^{(k)}$ be $n_j^{(k)}$.

No two grid cells $S_{3i+k,j}$ and $S_{3i'+k,j}$, with $i \neq i'$, in sub-strip $S_j^{(k)}$, contain a common segment since they are more than one unit apart vertically. The lower envelope of a sub-strip $S_j^{(k)}$ is the lower envelope of the lower envelopes for the squares included in $S_j^{(k)}$ and therefore any vertex in the lower envelope in a square is either included or excluded in the lower envelope of $S_j^{(k)}$, whereby the complexity is $\sum_{i=-\infty}^{\infty} O(n_{3i+k,j}) = O\left(n_j^{(k)}\right)$; see Figure 2(a).

We now consider the lower envelope of sub-strips that are three squares apart horizontally. We denote the union of such sub-strips by $S^{(k,l)} \stackrel{\text{def}}{=} \bigcup_{j=-\infty}^{\infty} S_{3j+l}^{(k)}$, for $l \in \{0,1,2\}$. Let $L^{(k,l)} = \bigcup_{j=-\infty}^{\infty} L_{3j+l}^{(k)}$ be the set of line segments resulting from intersecting the input set of line segments with $S^{(k,l)}$. Let the number of segments of each set $L^{(k,l)}$ be $n^{(k,l)}$.

Any two sets $L_{3j+l}^{(k)}$ and $L_{3j'+l}^{(k)}$, with $j \neq j'$, must have empty intersection since the corresponding sub-strips $S_{3j+l}^{(k)}$ and $S_{3j'+l}^{(k)}$ are more than one unit apart. Hence, no segment occurs in the two sub-strips whereby the lower envelope of the segments in $S^{(k,l)}$ has complexity $\sum_{j=-\infty}^{\infty} O\left(n_{3j+l}^{(k)}\right) = O\left(n^{(k,l)}\right) \subseteq O(n)$; see Figure 2(b).

The complexity of the lower envelope of $\bigcup_{\substack{0 \leq k \leq 2 \\ 0 \leq l \leq 2}} L^{(k,l)}$, i.e., the whole domain, is then

$$\sum_{\substack{0 \leq k \leq 2 \\ 0 \leq l \leq 2}} O\left(n^{(k,l)}\right) \subseteq \sum_{\substack{0 \leq k \leq 2 \\ 0 \leq l \leq 2}} O(n) = O(n), \tag{1}$$

using Observation 1.1 since we are summing over nine linear sized subsets. ◀

3.1 Segments Traced by Moving Points with Constant Speed

Let P be a set of n points in the plane, each moving at the same constant speed along a different line. The points start simultaneously moving at an instant $t = 0$, so at any given

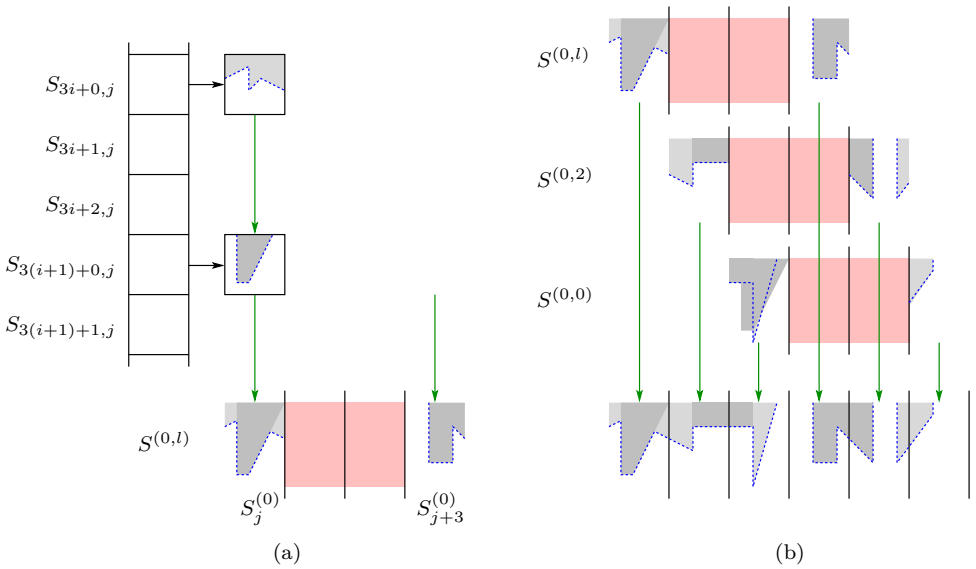


Figure 2 Illustrating the proof of Theorem 3.1.

instant $t > 0$, the points have traced a set L_t of n line segments, all with equal length; see Figure 3. By Theorem 3.1, and combining Corollary 3.2 with the algorithm by Hershberger [2], we obtain the following result.

▶ **Corollary 3.2.** *For any fixed $t > 0$, the lower envelope of L_t has a complexity of $\Theta(n)$ and can be computed in $O(n \log n)$ time and $O(n)$ space.*

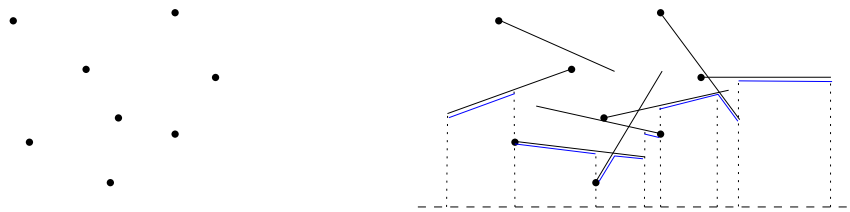


Figure 3 (a) A set P of eight points in the plane at an instant $t = 0$. (b) The lower envelope of the segments traced by moving the points of P along linear trajectories, at an instant $t > 0$.

4 Collections of Unit Squares Under Linear Transformations

We consider unit squares and allow rotation, translation, and scaling. We settle the complexity of the lower envelope for all possible combinations of these transformations, see Table 1. More specifically, we consider n copies of the unit square with corners at $[0, 0], [0, 1], [1, 0]$, and $[1, 1]$, and apply a subset of the transformations to these n unit squares.

If we do not allow scaling, each square has four unit-length segments, out of which at most two appear on the lower envelope. Combining Theorem 3.1 and Observation 1.1, we get

▶ **Corollary 4.1** (Cases 2, 5, and 6)**.** *The lower envelope of a set of n unit squares that can be rotated and/or translated has a complexity of $\Theta(n)$.*

Case	Rotation	Translation	Scaling	Complexity
1	×	×	×	$\Theta\big(n\alpha(n)\big)$
2	×	×		$\Theta(n)$
3	×		×	$\Theta(n)$
4		×	×	$\Theta(n)$
5	×			$\Theta(n)$
6		×		$\Theta(n)$
7			×	$\Theta(1)$

Table 1 Complexity of the lower envelope of unit squares under various linear transformations.

On the other hand, if we only allow scaling, we can only achieve constant complexity.

▶ **Lemma 4.2** (Case 7). *The lower envelope of a set of n unit squares that can be scaled has a complexity of $\Theta(1)$.*

▶ **Lemma 4.3** (Case 4). *The lower envelope of a set of n unit squares that can be translated and scaled has a complexity of $\Theta(n)$.*

Proof. For the lower bound, consider the lower envelope of the n axis-aligned squares with base edges $[(1,0),(2,0)],[(3,0),(4,0)],\ldots,[(n-1,0),(n,0)]$. Since such a lower envelope has $2n$ vertices, we have established the $\Omega(n)$ bound.

For the upper bound, we split the segments of the squares into two groups: All n squares have the same rotation, hence, for each square either a single horizontal line segment (if the squares are axis-aligned) or two line segments with two coinciding slopes for all squares may appear on the lower envelope. We consider the line segments with non-negative and those with negative slope separately. Let L^+ and L^- be the set of line segments with non-negative and negative slope, respectively. (L^- may be empty.) All line segments in L^+ have the same slope, hence, no two line segments from the set can intersect. We order the segments in L^+ by their left endpoint and insert them one after another. When we insert a new line segment $\ell_i \in L^+$, we introduce at most two vertices to the lower envelope:

1. If the left endpoint of ℓ_i is to the right of all previously inserted segments, we introduce two new vertices to the lower envelope.
2. If the left endpoint of ℓ_i is to the left of some right endpoints of previously inserted line segments, but its right endpoint is to the right of all previously inserted segments, we introduce at most two vertices to the lower envelope: both of ℓ_i's endpoints or its right endpoint (the other vertex is in that case an already introduced endpoint of a segment).
3. If the complete segment ℓ_i is within the interval of x-coordinates of previously inserted segments, no left endpoint of a segment $\ell_j, j < i$ can be to the right of ℓ_i's left endpoint, thus, depending on the y-coordinates of the so far introduced segments, we introduce either ℓ_i's endpoints, its right endpoint, or no point to the lower envelope.

Hence, the lower envelope of the line segments in L^+ has complexity $O(n)$. An analogous argument yields the same bound for the lower envelope of line segments in L^-. Hence, with Observation 1.1, we yield the upper bound. ◀

▶ **Lemma 4.4** (Case 3). *The lower envelope of a set of n unit squares that can be scaled and rotated has complexity $\Theta(n)$.*

While in all previous cases, we could give a linear bound on the complexity of the lower envelope, this does not hold if we allow all three linear transformations.

▶ **Theorem 4.5** (Case 1). *The lower envelope of a set of n unit squares that can be rotated, translated, and scaled has complexity $\Theta\big(n\alpha(n)\big)$.*

Proof. For a collection of n line segments in the plane (none of which are vertical), Hart and Sharir [1] showed that the complexity of the lower envelope can be at most $O\big(n\alpha(n)\big)$. For each of the n squares at most two line segments appear on the lower envelope. Thus, the result by Hart and Sharir in combination with Observation 1.1 yields an upper bound of $O\big(n\alpha(n)\big)$ on the complexity of the lower envelope of the squares.

Let $G = \{\ell_1, \ldots, \ell_n\}$ be the set of line segments from the lower-bound construction of Wiernik and Sharir [4]. We construct a new set of line segments $G' = \{\ell'_1, \ldots, \ell'_n\}$ from G: for a large constant M, we substitute each endpoint (e_x, e_y) of a segment by the endpoint $(e_x, \frac{e_y}{M})$. For an example, see Figure 4. If any two segments ℓ_i and ℓ_j intersect in the point (p_x, p_y), then ℓ'_i and ℓ'_j intersect in the point $(p_x, \frac{p_y}{M})$. Thus, if ℓ_i is on the lower envelope of G for the interval (x_1, x_2), then ℓ'_i is on the lower envelope of G' for the same interval (x_1, x_2). Consequently, the complexity of the lower envelope of G' equals that of G.

We use the new ("nearly" horizontal) line segments as the base edge of our squares S, see Figure 4: Let ℓ'_i have endpoints (l^i_x, l^i_y) and (r^i_x, r^i_y), we construct a square s_i with side length $||\ell'_i||$. For each of these squares at most two edges are on the lower envelope.

For M being large enough, the square edges appearing on the lower envelope that do not stem from the line segments from the construction by Wiernik and Sharir have a very large absolute value of slope (that is, they are "nearly" vertical). We can choose M large enough such that for ℓ_i, ℓ_j, ℓ_k appearing on the lower envelope in that order, the vertical edges of s_i and s_k cannot block s_j (with higher y-coordinate) from appearing on the lower envelope: Let ℓ_j appear on the lower envelope of G within the interval $I_j = [r^i_x, l^k_x]$. We split I_j into three equal-length closed intervals I_{jl}, I_{jm} and I_{jr}, aiming that s_i and s_k may block at most I_{jl} and I_{jr}, respectively—which yields the claim. We consider the case that ℓ_i has negative slope (with positive slope, s_i does not block any of I_{jl}). For simplicity, assume that ℓ_j is horizontal (similar arguments hold in the other cases). We can consider ℓ_j, because if ℓ_j is not blocked in I_{jm} then ℓ'_j (with smaller y-coordinates) is not blocked either. Assume, we construct a square σ_i with ℓ_i as base edge (instead of ℓ'_i for s_i). Let e_i be σ_i's second edge that may appear on the lower envelope, and let $p^{ij} = (p^{ij}_x, p^{ij}_y)$ be the intersection point of e_i and ℓ_j. If $p^{ij} \in I_{jl}$, we are done for any value of M, hence, let $p^{ij} \in I_{jm} \cup I_{jr}$. The slope of e_i is $\frac{p^{ij}_y - r^i_y}{p^{ij}_x - r^i_x}$, the equivalent edge of s_i should have slope at least $\frac{p^{ij}_y - r^i_y}{r^i_x + 1/3(l^k_x - r^i_x) - r^i_x} = \frac{p^{ij}_y - r^i_y}{1/3(l^k_x - r^i_x)}$, such that we achieve $p^{ij} \in I_{jl}$. Hence, we need to choose $M \geq \frac{1/3(l^k_x - r^i_x)}{p^{ij}_x - r^i_x}$. By choosing M larger than these constraints for all pairs of line segments appearing consecutively on the lower envelope, we can ensure that while s_i and ℓ_i will not appear on the exact same interval on the lower envelope for S and G, the sequence of the s_i appearing on the lower envelope for S coincides with the sequence of the ℓ_i appearing on the lower envelope for G. Hence, the lower bound for line segments established by Wiernik and Sharir [4] translates to unit squares that can be rotated, translated and scaled. ◀

5 Open Questions

The complexity for other geometric shapes, such as differently oriented parabolae, ellipses, and fat objects in general would be of great interest to settle. In particular, we are interested in knowing if the complexity can be proved with purely geometric arguments.

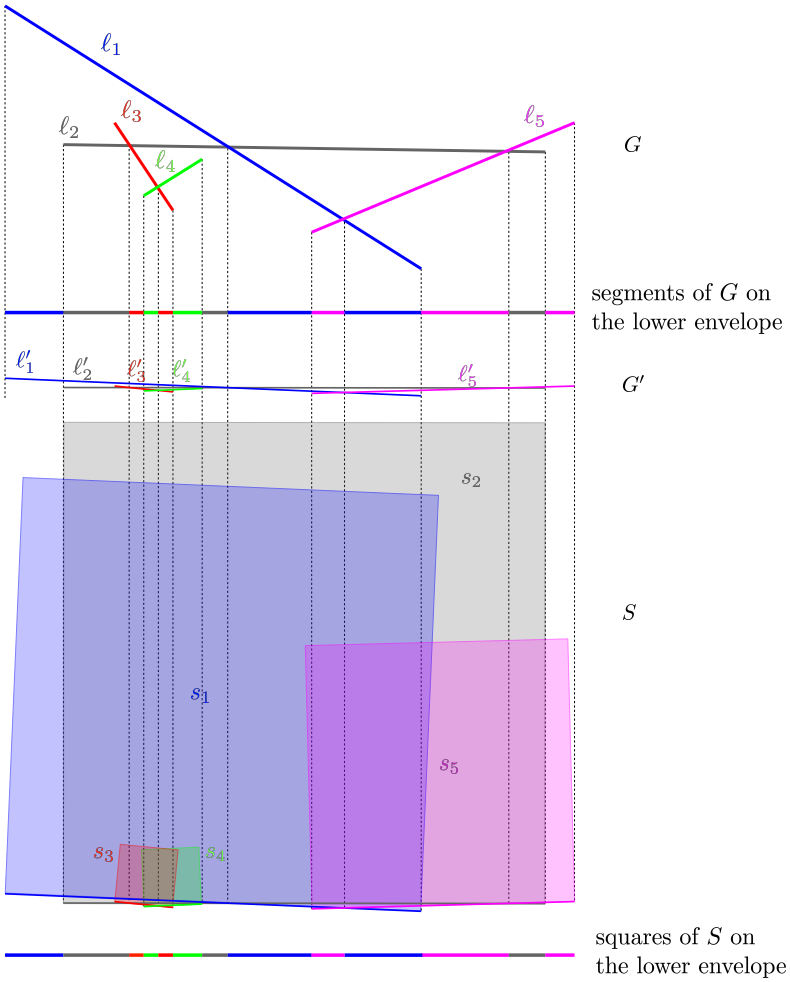


Figure 4 Example for the construction from the proof of Theorem 4.5. Top: Set of line segments $G = \{\ell_1, \ldots, \ell_5\}$; second to top: the sequence of segments of G appearing on the lower envelope; middle: new set of line segments $G' = \{\ell'_1, \ldots, \ell'_5\}$; second to bottom: set of squares $S = \{s_1, \ldots, s_5\}$; bottom: sequence of squares of S appearing on the lower envelope.

References

1 Sergiu Hart and Micha Sharir. Nonlinearity of Davenport-Schinzel sequences and of generalized path compression schemes. *Combinatorica*, 6(2):151–177, 1986. doi:10.1007/BF02579170.

2 John Hershberger. Finding the upper envelope of n line segments in $O(n \log n)$ time. *Information Processing Letters*, 33(4):169–174, 1989. doi:10.1016/0020-0190(89)90136-1.

3 Micha Sharir and Pankaj K. Agarwal. *Davenport–Schinzel Sequences and their Geometric Applications*. Cambridge University Press, 1995.

4 Ady Wiernik and Micha Sharir. Planar realizations of nonlinear Davenport-Schinzel sequences by segments. *Discrete & Computational Geometry*, 3(1):15–47, 1988. doi:10.1007/BF02187894.

Unit Interval Graphs & Maximum c-Independent Sets Maximizing the Number of Isolated Vertices

Linda Kleist[1] **and Kai Kobbe**[1]

1 **Technische Universität Braunschweig**
{kleist,kobbe}@ibr.cs.tu-bs.de

Abstract

We study maximum c-independent sets that maximize the number of isolated vertices and present an algorithm that computes such subgraphs for unit interval graphs in linear time. The algorithm is based on a simple test that gives a certificate whether a specific vertex can be isolated. While the crucial property seems straight-forward, its proof requires a careful analysis of the structure of c-independent sets in unit interval graphs. Surprisingly, the techniques do not generalize to interval graphs and the algorithm does not even yield an approximation on general interval graphs.

1 Introduction

Computing maximum independent sets is a fundamental problem and appears on the list of Karp's 21 NP-complete problems. Maximum independent sets and also its generalizations of maximum c-independent sets find a variety of use cases across various fields in modelling and solving real-world problems, e.g., wireless sensor networks [3], DNA sequencing in bioinformatics [13, 19], VLSI design [13, 23], job scheduling [6, 11] and resource allocation [20], as well as identifying independent strategies in game theory [28]. For $c \in \mathbb{N}$, a *c-independent set* (c-IS) of a graph is the union of c independent sets.

While the special case of $c = 1$ is the *Maximum Independent Set Problem*, the special case of $c = 2$ is also known as *Maximum Bipartite Subgraph*, *Graph Bipartization*, or *Odd Cycle Transversal*. Not only that computing a maximum c-IS is NP-complete for general graphs, it has also been shown that there is no approximation algorithm with a factor in $O(n^{1-\varepsilon})$ for any $\varepsilon > 0$ and (possibly fixed) $c \in \mathbb{N}_{\geq 1}$, unless P = NP [18, 22]. In contrast, a maximum c-IS can be computed in polynomial time for special graph classes; including interval graphs, even if c is part of the input [29].

We consider the problem of computing a maximum c-IS for unit interval graphs with the additional property of maximizing the number of isolated vertices (among all maximum c-ISs). To this end, we say that a maximum c-IS is *max-iso*, if no other maximum c-IS has more isolated vertices. For an example, consider the unit interval graph depicted in Figure 1, where the thick intervals correspond to a maximum 2-IS with one isolated vertex. Is there a maximum 2-IS with more isolated vertices?

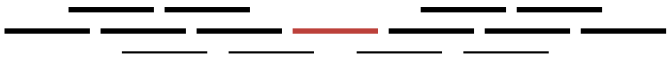


Figure 1 A unit interval graph where the subset of thick intervals is a maximum 2-IS with one isolated vertex. Note that exchanging the top four intervals with the bottom four intervals yields a maximum 2-IS with two isolated vertices, namely the first and last vertex.

Our main result is as follows.

▶ **Theorem 1.1.** *There exists an algorithm that computes a max-iso c-IS for every unit interval graph on n vertices with a running time in $O(n)$, even if c is part of the input.*

40th European Workshop on Computational Geometry, Ioannina, Greece, March 13–15, 2024.
This is an extended abstract of a presentation given at EuroCG'24. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

Our key motivation is a scheduling problem where conflicts between machines appear due to shared resources or spatial proximity [7]. In this variant, each job has phases of pre- and post-processing, and no two jobs on conflicting machines may overlap in such phases. For the case that machine conflicts can be expressed by a unit interval graph, we expect that maximum c-ISs with many isolated vertices are crucial to obtain good schedules.

The rest of our paper is organized as follows. We review related work in Section 1.1, introduce fundamental concepts in Section 1.2 and discuss the algorithm in Section 2.

1.1 Related work

Finding a maximum c-IS is known to be NP-complete [15] and, for graphs on n vertices, there does not even exist an $O(n^{1-\varepsilon})$-approximation for any $c \in \mathbb{N}_{\geq 1}$ and $\varepsilon > 0$, unless P = NP [18, 22, 31]. Moreover, when considering general graphs, computing a maximum c-IS is equivalent to computing a maximum 1-IS, under polynomial-time reductions [25].

Considering $c = 1$, the maximum independent set (MIS) is relevant in many applications and therefore well studied, also for special graph classes. In this sense, constant factor approximations exist for bounded degree graphs and families of geometric intersection graphs [8]. Furthermore, polynomial-time approximation schemes (PTAS) exist for families of graphs that are closed under taking minors [16], e.g., planar graphs [5]. For many other graph classes, a maximum weight independent set may be found in polynomial time. Famous examples include claw-free graphs [12, 24], P_5-free graphs [21] and perfect graphs [17]. For chordal graphs, and thus in particular for interval graphs, maximum weight independent sets can be computed in linear time [14]. More generally, the maximum weight independent set can be computed in polynomial time for graphs that do not contain a k-prism or an induced C_n with $n \geq 5$ [10].

For $c = 2$, the problem is also known as Maximum (Induced) Bipartite Subgraph, Graph Bipartization, or Odd Cycle Transversal. A maximum 2-IS can be found in polynomial time in planar graphs with maximum degree three, while it is NP-complete for cubic graphs and planar graphs with maximum degree four [4, 9]. Additionally, polynomial-time algorithms exist for many other graph classes; including split graphs, permutation graphs, tolerance graphs and circular-arc graphs [25, 30]. We emphasize that there are graph classes, for which a maximum 1-IS can be computed in polynomial time, while finding a maximum 2-IS is NP-hard. Examples are circle graphs [2, 26, 27] and perfect graphs, as computing a maximum 2-IS is NP-hard for the subclass of clique-separable graphs [1].

For general c, it is known that the class of chordal graphs allows for a polynomial-time algorithm if c is not part of the input [29]. If c is part of the input, polynomial-time algorithms exist for the two subfamilies of perfect graphs of i-triangulated graphs [1] and interval graphs [29].

1.2 Fundamental Concepts

Let $G = (V, E)$ be a graph. As usual, we denote the *(open) neighborhood* of a vertex v by $N(v) := \{u \in V \mid uv \in E\}$ and the *closed neighborhood* by $N[v] := N(v) \cup \{v\}$. For a subset $U \subset V$, $G[U]$ denotes the subgraph induced by U.

Independent sets. A subset of vertices $U \subset V$ is an *independent set* of G, if no two vertices of U share an edge in G. For a fixed $c \in \mathbb{N}$, a *c-independent set* (c-IS) $\mathcal{I}$ of G is a union of c independent sets $\mathcal{I}_1, ..., \mathcal{I}_c$ of G; its size is given by the number of vertices, i.e., $|\mathcal{I}| := |\bigcup_i \mathcal{I}_i|$. A c-IS is *maximum* if no other c-IS has larger size. We denote the size of a maximum c-IS of

G by $\alpha_c(G)$. As mentioned before, we are particularly interested in c-ISs with many isolated vertices. To this end, we call a maximum c-IS $\mathcal{I}$ of G *max-iso*, if G has no maximum c-IS with more isolated vertices.

Interval graphs. In a (unit) interval representation of a graph $G = (V, E)$, every vertex is represented by a (unit) interval in $\mathbb{R}^1$ such that two intervals intersect if and only if the corresponding vertices share an edge. If a graph has a (unit) interval representation, then it is a (unit) interval graph. Without loss of generality, we assume that no two intervals are identical. A unit interval representation has a natural ordering of the intervals by their start points. If the interval of a vertex v starts before the interval of w, then we say v is smaller than w; we write $v \prec w$. We write $v \preceq w$ if either $v \prec w$, or $v = w$, i.e., v and w refer to the same vertex. An ordering $v_1, \dots, v_n$ of the vertices of G is a *left-right-order* if $v_i \prec v_{i+1}$ for all i. We shortly recall a simple greedy algorithm to compute a maximum c-IS for a unit interval graph $G = (V, E)$ with left-right-order $v_1, \dots, v_n$ with a runtime in $O(n)$ [29]: We start with $\mathcal{I} = \emptyset$ and consider the vertices by increasing index. Vertex v_i is added to $\mathcal{I}$, if this maintains a c-IS in G. A useful property of the greedy solution $\mathcal{I}$ for G is the following: For every vertex $u \in V$, it holds that $\mathcal{I}' := \{v \in \mathcal{I} \mid v \prec u\}$ is a maximum c-IS in $G[\{v \in V \mid v \prec u\}]$. Clearly, an analogous statement holds when using the reversed (right-left) order of the vertices.

2 The Algorithm

In this section, we study the computation of max-iso c-ISs for unit interval graphs. We start by establishing a simple test that states if it is reasonable to isolate a specific vertex for our purposes. A straight-forward implementation would yield a quadratic algorithm. We additionally show how to derive a linear-time algorithm.

▶ **Lemma 2.1.** *Let G be a connected unit interval graph, $\mathcal{I}_w$ a max-iso c-IS of G and w be the smallest isolated vertex in $\mathcal{I}_w$. Consider a vertex $v \prec w$. If $\alpha_c(G - N(v)) = \alpha_c(G)$, then $G - N(v)$ contains a max-iso c-IS of G.*

Proof-Sketch. We consider a maximum c-IS $\mathcal{I}_v$ in $G - N(v)$, i.e., $\mathcal{I}_v$ has size $\alpha_c(G)$. If $w \in \mathcal{I}_v$, we observe that exchanging the sets of vertices that are larger than w in $\mathcal{I}_v$ and $\mathcal{I}_w$ (denoted by $\mathcal{I}_v^{\succ}, \mathcal{I}_w^{\succ}$) by each other, yields a c-IS; here we use the fact that G is a unit interval graph. For a schematic illustration consider Figure 2. Therefore, it holds that $|\mathcal{I}_v^{\succ}| = |\mathcal{I}_w^{\succ}|$ and consequently $\mathcal{I}' := (\mathcal{I}_v \setminus \mathcal{I}_v^{\succ}) \cup \mathcal{I}_w^{\succ}$ is a maximum c-IS in G, where the isolated vertices are the same as in $\mathcal{I}_w$, except for isolating now v instead of w. Thus, $\mathcal{I}'$ is max-iso.

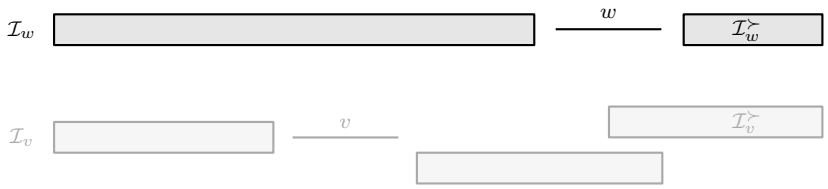


Figure 2 Illustration for the proof of Lemma 2.1. If $w \in \mathcal{I}_w, \mathcal{I}_v$, then replacing $\mathcal{I}_w^{\succ}$ and $\mathcal{I}_v^{\succ}$ by each other in $\mathcal{I}_w$ or $\mathcal{I}_v$ maintains a c-IS.

Afterwards, we show that there exists a maximum c-IS in $G - N(v)$ that contains w. In particular, this implies that $v \notin N(w)$. To this end, we carefully analyse the structure of $G[\mathcal{I}_v \cup \mathcal{I}_w]$ and, assuming that there is no such maximum c-IS, obtain a contradiction. ◀

▶ Remark. Lemma 2.1 does not hold for interval graphs in general: Consider the interval graph G depicted in Figure 3. Note that it contains several c-cliques and a unique $(2c-1)$-clique $C = \{u, b_1, \ldots, b_{c-1}, r_1, \ldots, r_{c-1}\}$. In order to obtain a maximum c-IS, we have to delete $c-1$ vertices of C. There exist various choices. However, when we keep at least one b_i and one r_j, then the resulting c-IS has no isolated vertex. Deleting the vertices $b_1, \ldots, b_{c-1}$ isolates vertex v, while deleting the vertices $r_1, \ldots, r_{c-1}$ isolates the vertices $w_1, \ldots, w_k$ and thus yields the unique max-iso c-IS of G. Consequently, as $N(v) = \{b_1, \ldots, b_{c-1}\}$, the graph $G - N(v)$ contains a maximum c-IS of G but no max-iso c-IS. This implies that the decision of isolating v is not only suboptimal but even arbitrarily bad as the number of isolated vertices is 1 vs k in the optimum. Consequently, Lemma 2.1 cannot be used to derive a constant-factor approximation algorithm for general interval graphs.

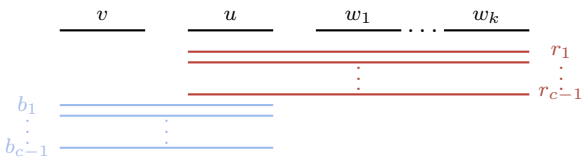


Figure 3 An interval graph G and its first vertex v such that $\alpha_c(G - N(v)) = \alpha_c(G)$, but $G - N(v)$ contains no max-iso c-IS of G. Thus, Lemma 2.1 does not generalize to interval graphs.

▶ Remark. It is crucial in Lemma 2.1 that v is a valid candidate for a smallest isolated vertex. In other words, even if $\alpha_c(G-N(v)) = \alpha_c(G)$ for some vertex v, it is not necessarily true that $G - N(v)$ contains a max-iso c-IS of G. For an example, consider the unit interval graph G depicted in Figure 4. Observe that there are two disjoint $(2c-1)$-cliques in G, namely, $C_i = \{u_i, b_1^i, \ldots, b_{c-1}^i, r_1^i, \ldots, r_{c-1}^i\}$ for $i \in \{1, 2\}$. In order to obtain a maximum c-IS, we have to delete $c-1$ vertices from C_1 and C_2, respectively. The unique maximum c-IS $\mathcal{I}$ that isolates v is obtained by deleting all r_j^i; note that this implies that there is no other isolated vertex in $\mathcal{I}$. In contrast, the maximum c-IS $\mathcal{I}'$ obtained by deleting all b_j^i has two isolated vertices, namely w_1 and w_2. Thus, when applying Lemma 2.1, it is crucial that v is a candidate for a smallest isolated vertex (in its connected component).

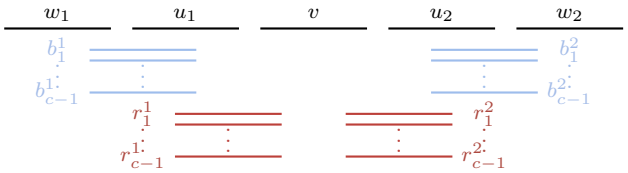


Figure 4 A unit interval graph G and vertex v with $\alpha_c(G - N(v)) = \alpha_c(G)$, but $G - N(v)$ contains no max-iso c-IS of G.

As mentioned before, Lemma 2.1 yields a simple quadratic algorithm: For a given left-right-order and increasing i, we check iteratively whether $\alpha_c(G) = \alpha_c(G - N(v_i))$. If so, we delete $N(v_i)$ from G and repeat with incremented i. In each step, we use Lemma 2.1 for the connected component of v_i. Finally, we compute a maximum c-IS of the modified graph and return it. As computing a maximum c-IS takes linear time (for a given left-right-order), we obtain a simple quadratic algorithm.

In the following, we show how to obtain a linear-time algorithm for unit interval graphs.

▶ **Lemma 2.2.** *For every unit interval graph on n vertices with given left-right-order, Algorithm 1 can be implemented to compute a max-iso c-IS with a running time in $O(n)$.*

Algorithm 1 Algorithm for computing a max-iso c-IS in unit interval graphs

Require: Unit interval Graph $G = (V, E)$ with left-right-order $v_1, \dots, v_n$
Ensure: max-iso c-IS of G

1: $U := V$
2: $L := \emptyset$
3: $R :=$ Greedy maximum c-IS with reversed order $v_n, \dots, v_1$
4: **while** $U \neq \emptyset$ **do**
5: Identify the smallest vertex v from U and delete it from U
6: **if** $|(L \cup R) \cap N[v]| = 1$ **then** ▷ if true, v will be isolated
7: $U = U \setminus N[v]$
8: $L = (L \setminus N[v]) \cup \{v\}$
9: $R = R \setminus N[v]$
10: **else if** $|L \cap N[v]| < c$ **then**
11: $L = L \cup \{v\}$
12: Delete the smallest vertex from R
13: **end if**
14: **end while**
15: **return** L

Proof-Sketch. The key idea of the algorithm is to maintain a leftmost partial solution L (before the current vertex) and a rightmost partial (greedy) solution R (after the current vertex) which allows for constant update time in each iteration. This is schematically depicted in Figure 5. When the algorithm ends, it holds that L is a max-iso c-IS for the given graph. More precisely, in each iteration of the while-loop, we test whether the smallest yet unconsidered vertex v (selected in line 5) can be isolated (in line 6). If so, we delete its neighborhood to guarantee that it is isolated and add v to our partial solution L (in lines 7-9). If not, we test whether it can be added greedily to L (in line 10).

Throughout the algorithm, we maintain a set of invariants. To this end, consider an iteration where v is selected in line 5 and let S denote the set of vertices for which line 6 evaluated to true so far. Before the iteration, the following invariants hold:

- $L \cup R$ is a maximum c-IS in G in which vertices from S are isolated.
- $L \subset \{u \in V \mid u \prec v\} =: V_L$ and $R \subset \{u \in V \mid v \preceq u\} =: V_R$
- $L \setminus N[v]$ is a maximum c-IS in $G[V_L - N[v]]$ (in which vertices from S are isolated); moreover, after the largest isolated vertex $u \in L$, vertices are added greedily to L.
- $R \setminus N[v]$ is a (greedy) maximum c-IS in $G[V_R - N[v]]$ (for right-left order).

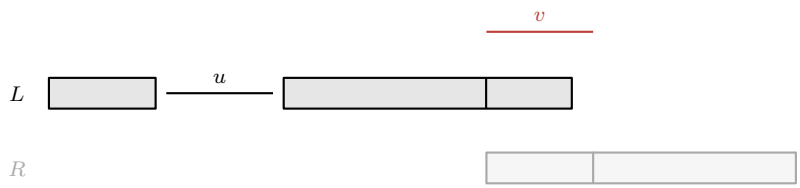


Figure 5 State of L and R before the iteration where v is selected in line 5 of Algorithm 1.

These properties allow us to check efficiently the size of a largest c-IS where the vertices $S \cup \{v\}$ are isolated, because $(L \cup R \cup \{v\}) \setminus N(v)$ is such a largest c-IS. In line 6, we test whether this size is equal to $\alpha_c(G) = |L \cup R|$ and if so, we isolate v. ◀

References

1 Louigi Addario-Berry, W.S. Kennedy, Andrew D. King, Zhentao Li, and Bruce Reed. Finding a maximum-weight induced k-partite subgraph of an i-triangulated graph. *Discrete Applied Mathematics*, 158(7):765–770, 2010. doi:10.1016/j.dam.2008.08.020.

2 Alberto Apostolico, Mikhail J. Atallah, and Susanne E. Hambrusch. New clique and independent set algorithms for circle graphs. *Discrete Applied Mathematics*, 36(1):1–24, 1992. doi:10.1016/0166-218X(92)90200-T.

3 Ozkan Arapoglu and Orhan Dagdeviren. An asynchronous self-stabilizing maximal independent set algorithm in wireless sensor networks using two-hop information. In *2019 International Symposium on Networks, Computers and Communications (ISNCC)*, pages 1–5, 2019. doi:10.1109/ISNCC.2019.8909189.

4 Mourad BaÏou and Francisco Barahona. Maximum weighted induced bipartite subgraphs and acyclic subgraphs of planar cubic graphs. *SIAM Journal on Discrete Mathematics*, 30(2):1290–1301, 2016. doi:10.1137/140980053.

5 Brenda S. Baker. Approximation algorithms for NP-complete problems on planar graphs. *J. ACM*, 41(1):153–180, 1994. doi:10.1145/174644.174650.

6 Matthias Bentert, René van Bevern, and Rolf Niedermeier. Inductive k-independent graphs and c-colorable subgraphs in scheduling: a review. *Journal of Scheduling*, 22(1):3–20, 2018. doi:10.1007/s10951-018-0595-8.

7 Moritz Buchem, Linda Kleist, and Daniel Schmidt genannt Waldschmidt. Scheduling with machine conflicts. In *Approximation and Online Algorithms: 20th International Workshop, (WAOA 2022)*, page 36–60, 2022. doi:10.1007/978-3-031-18367-6_3.

8 Timothy M Chan and Sariel Har-Peled. Approximation algorithms for maximum independent set of pseudo-disks. In *Proceedings of the twenty-fifth annual symposium on Computational geometry*, pages 333–340, 2009. doi:10.1145/1542362.1542420.

9 Hyeong-Ah Choi, Kazuo Nakajima, and Chong S. Rim. Graph bipartization and via minimization. *SIAM Journal on Discrete Mathematics*, 2(1):38–47, 1989. doi:10.1137/0402004.

10 Maria Chudnovsky, Marcin Pilipczuk, Michał Pilipczuk, and Stéphan Thomassé. On the maximum weight independent set problem in graphs without induced cycles of length at least five. *SIAM Journal on Discrete Mathematics*, 34(2):1472–1483, 2020. doi:10.1137/19M1249473.

11 Duncan Eddy and Mykel J. Kochenderfer. A maximum independent set method for scheduling earth observing satellite constellations, 2020. arXiv:2008.08446.

12 Yuri Faenza, Gianpaolo Oriolo, and Gautier Stauffer. Solving the weighted stable set problem in claw-free graphs via decomposition. *Journal of the ACM (JACM)*, 61(4):1–41, 2014. doi:10.1145/2629600.

13 Pierre Fouilhoux and A. Ridha Mahjoub. Solving vlsi design and dna sequencing problems using bipartization of graphs. *Computational Optimization and Applications*, 51(2):749–781, 2012. doi:10.1007/s10589-010-9355-1.

14 András Frank. Some polynomial algorithms for certain graphs and hypergraphs. In *Proceedings of the 5th British combinatorial conference, congressus numerantium*, volume XV, pages 211—-226, 1975.

15 M. R. Garey and D. S. Johnson. "Strong" NP-completeness results: Motivation, examples, and implications. *J. ACM*, 25(3):499–508, 1978. doi:10.1145/322077.322090.

16 Martin Grohe. Local tree-width, excluded minors, and approximation algorithms. *Combinatorica*, 23(4):613–632, 2003. doi:10.1007/s00493-003-0037-9.

17 M. Grötschel, L. Lovász, and A. Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1(2):169–197, 1981. doi:10.1007/BF02579273.

18 Johan Håstad. Clique is hard to approximate within $n^{1-\epsilon}$. *Acta Mathematica*, 182(1):105–142, 1999. doi:10.1007/BF02392825.

19 Deborah Joseph, Joao Meidanis, and Prasoon Tiwari. Determining dna sequence similarity using maximum independent set algorithms for interval graphs. In *Algorithm Theory — SWAT '92*, pages 326–337, 1992. doi:10.1007/3-540-55706-7_29.

20 Alper Köse and Berna Özbek. Resource allocation for underlaying device-to-device communications using maximal independent sets and knapsack algorithm. In *IEEE 29th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, pages 1–5, 2018. doi:10.1109/PIMRC.2018.8580784.

21 Daniel Lokshantov, Martin Vatshelle, and Yngve Villanger. Independent set in p_5-free graphs in polynomial time. In *Symposium on discrete algorithms (SODA)*, pages 570–581, 2014. doi:10.1137/1.9781611973402.4.

22 Carsten Lund and Mihalis Yannakakis. The approximation of maximum subgraph problems. In *Automata, Languages and Programming*, pages 40–51, 1993. doi:10.1007/3-540-56939-1_60.

23 M. Marek-Sadowska. An unconstrained topological via minimization problem for two-layer routing. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 3(3):184–190, 1984. doi:10.1109/TCAD.1984.1270074.

24 George J. Minty. On maximal independent sets of vertices in claw-free graphs. *Journal of Combinatorial Theory, Series B*, 28(3):284–304, 1980. doi:10.1016/0095-8956(80)90074-X.

25 G. Narasimhan. *The Maximum k-Colorable Subgraph Problem*. PhD thesis, University of Wisconsin-Madison, 1989. URL: https://research.cs.wisc.edu/techreports/1989/TR864.pdf.

26 Nicholas Nash and David Gregg. An output sensitive algorithm for computing a maximum independent set of a circle graph. *Information Processing Letters*, 110(16):630–634, 2010. doi:10.1016/j.ipl.2010.05.016.

27 M. Sarrafzadeh and D.T. Lee. A new approach to topological via minimization. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 8(8):890–900, 1989. doi:10.1109/43.31548.

28 Christof Spanring. Axiom of Choice, Maximal Independent Sets, Argumentation and Dialogue Games. In *2014 Imperial College Computing Student Workshop*, volume 43 of *Open Access Series in Informatics (OASIcs)*, pages 91–98, 2014. doi:10.4230/OASIcs.ICCSW.2014.91.

29 Mihalis Yannakakis and Fanica Gavril. The maximum k-colorable subgraph problem for chordal graphs. *Information Processing Letters*, 24(2):133–137, 1987. doi:10.1016/0020-0190(87)90107-4.

30 Susan S. Yeh and Andrea S. LaPaugh. Algorithms for finding a maximum bipartite subgraph for special classes of graphs. 1988. technical report. URL: https://www.cs.princeton.edu/research/techreps/TR-149-88.

31 David Zuckerman. Linear degree extractors and the inapproximability of max clique and chromatic number. In *Proceedings of the Symposium on Theory of Computing (STOC)*, pages 681–690, 2006. doi:10.1145/1132516.1132612.

Flip Graphs of Pseudo-Triangulations With Face Degree at Most Four*

Maarten Löffler[1], Tamara Mchedlidze[1], David Orden[2], Josef Tkadlec[3], and Jules Wulms[4]

1 **Utrecht University, the Netherlands**
[m.loffler,t.mtsentlintze]@uu.nl
2 **University of Alcalá, Spain**
david.orden@uah.es
3 **Charles University, Czech Republic**
josef.tkadlec@iuuk.mff.cuni.cz
4 **TU Eindhoven, the Netherlands**
j.j.h.m.wulms@tue.nl

Abstract

A *pseudo-triangle* is a simple polygon with exactly three convex vertices. A *pseudo-triangulation* $\mathcal{T}$ of a point set P in $\mathbb{R}^2$ is a partitioning of the convex hull of P into pseudo-triangles, such that the union of the vertices of the pseudo-triangles is exactly P. We call a size-4 pseudo-triangle a *dart*. For a fixed $k \geq 1$, we study k-dart pseudo-triangulations (k-DPTs), that is, pseudo-triangulations in which exactly k faces are darts and all other faces are triangles. Our results are as follows. We prove that the flip graph of 1-DPTs is generally not connected, and show how to compute its connected components. Furthermore, for k-DPTs on a point configuration called the *double chain* we analyze the structure of the flip graph on a more fine-grained level.

Related Version A full version of the paper is available at arxiv.org/abs/2402.12357

1 Introduction

Pseudo-triangulations were introduced in the early 1990's by Pocchiola and Vegter to study the visibility complex of disjoint convex regions [13] and by Chazelle et al. for ray shooting in polygons [6]. It was in the early 2000's that pseudo-triangulations of point sets became popular, when Streinu showed that *pointed* pseudo-triangulations of point sets, those in which every vertex is *pointed*, i.e., incident to an angle larger than π, are minimally rigid [15] and used this for a solution of the Carpenter's Rule Problem. The converse statement that every planar minimally rigid graph admits a drawing as a pointed pseudo-triangulation was proved by Haas et al. [8], later generalized to non-minimally rigid and non-pointed pseudo-triangulations by Orden et al. [12] using the notion of *combinatorial pseudo-triangulation*, an embedding of a planar graph together with a labelling of the angles mimicking the properties of angles in a geometric pseudo-triangulation.

Among the many other results on pseudo-triangulations, for which we refer to the survey by Rote et al. [14], let us highlight the notion of a *flip* [1, 11]. There are three types of flips. The first one follows the spirit of flips in triangulations, exchanging the only interior edge in the two geodesic diagonals of a pseudo-quadrilateral, as in Figure 1a-b. This includes

* D.O. was supported by Project PID2019-104129GB-I00 funded by MCIN/ AEI /10.13039/501100011033. J.T. was supported by the Center for Foundations of Modern Computer Science (Charles University project UNCE/SCI/004) and by project PRIMUS/24/SCI/012 from Charles University. This work was initiated during the GG Week 2023, held at the University of Alcalá, Spain.

40th European Workshop on Computational Geometry, Ioannina, Greece, March 13–15, 2024.
This is an extended abstract of a presentation given at EuroCG'24. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

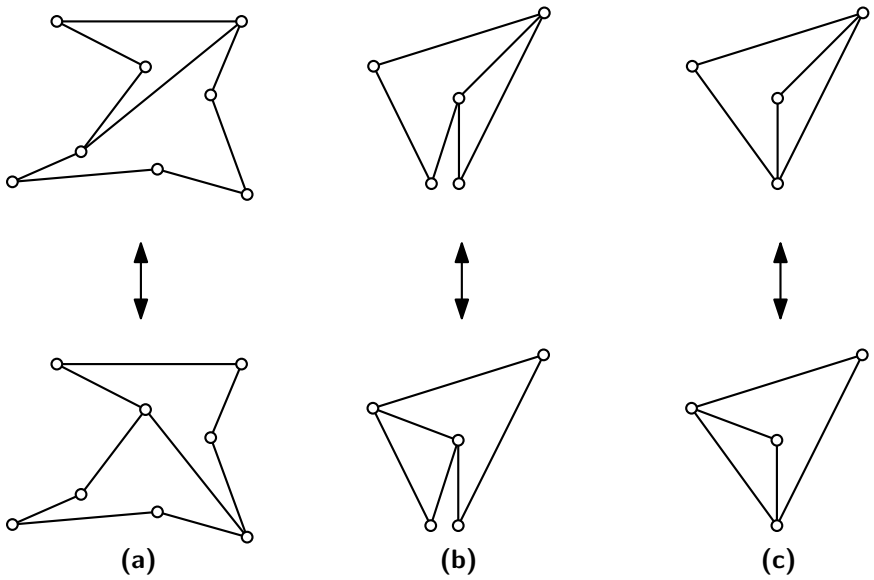


Figure 1 Some flips in pseudo-triangulations.

the case of a degenerate pseudo-quadrilateral, with two consecutive corners merged into a single one, as in Figure 1c where the two lowest vertices from Figure 1b have been merged. The two remaining types of flip insert or remove an interior edge to obtain another pseudo-triangulation, respectively increasing or decreasing by one the number of pointed vertices (this will be fixed in the present work, so such types of flip will not appear). The flip graph for pseudo-triangulations turns out to be connected and have diameter in $O(n \log n)$ [5].

The fact that a pseudo-triangle can have linear size and, therefore, the flip operation cannot be computed in constant time as for triangulations, led to the consideration of pseudo-triangulations with bounded size of the internal faces. In particular, Kettner et al. [9] showed that for every point set there is a pointed pseudo-triangulation with internal faces of size 3 or 4, called a *4-pointed pseudo-triangulation* or *4-PPT*. These 4-PPTs fulfill nice properties, like being properly 3-colorable while that question is NP-complete for general pseudo-triangulations [2]. By contrast, some properties known for general pseudo-triangulations turned out to be elusive for 4-PPTs. In particular, a long-standing open problem is whether the flip graph of 4-PPTs is connected, which has only been proved for combinatorial 4-PPTs [3]. The aim of this work is to generalize this problem and prove results on cases that can provide additional insight towards solving that open problem.

We consider flips in *4-pseudo-triangulations* or *4-PT*s, which are defined as general, not necessarily pointed, pseudo-triangulations with internal faces of size 3 or 4. We call a size-4 pseudo-triangle a *dart*, with its *tail* being the concave (also called reflex) vertex, its *tip* being the vertex not adjacent to the tail, and its two *wings* being the remaining two vertices. The segment between the tip and tail of a dart will be referred to as its *spine*, though such a segment is necessarily not an edge and therefore missing in a dart. In a 4-PT, each interior pointed vertex is the tail of a dart, and 4-PTs with k interior pointed vertices are 4-PTs with k darts or *k-dart 4-PTs*, which will be denoted as k-DPTs. See Figure 2.

For a size-n point set P with a convex hull of size $h \leq n$, the maximum number of interior pointed vertices is $n - h$, and therefore the maximum number of darts in a 4-PT is $n - h$ as well. In particular, 4-PPTs coincide with k-DPTs for $k = n - h$, since they are those 4-PTs in which every interior vertex is pointed. Thus, the aforementioned open problem in [3] asks

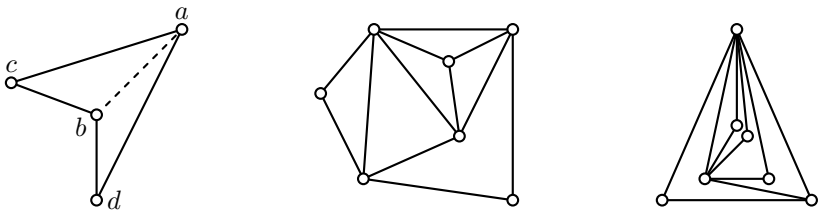


Figure 2 (a) A dart with tip a, tail b, wings c and d, and a dashed spine. **(b)** A 1-DPT on 7 points. **(c)** An $(n-h)$-DPT with $n=7$ and $h=3$; each vertex not on the convex hull is a dart tail.

about the connectivity of the flip graph of k-DPTs for the largest possible value of k, that is $k = n - h$. Our first goal is to look at the opposite end of the range and study the flip graph of k-DPTs for the smallest possible value of k, that is $k = 1$. This corresponds to 4-PTs with only one dart, i.e., only one interior pointed vertex. We show that the resulting flip-graph of 1-DPTs is not connected and we show how to compute its connected components. Furthermore, for k-DPTs on a frequently-studied point configuration, the double chain [4], we analyze the structure of the flip graph on a more fine-grained level.

Due to space constraints, most proofs are omitted and can be found in the full version [10].

2 Components of the Flip Graph for 1-DPTs

To compute the number of components of the flip graph in the presence of a single dart, we first partition the class of 1-DPTs on a point set P into separate classes $\mathcal{G}_p$ where $p \in P$. Each such class $\mathcal{G}_p$ consists of all those 1-DPTs that have the tail of the dart located at p. We show that all 1-DPTs in $\mathcal{G}_p$ are in the same connected component of the flip graph. The proof consists of three steps: First, we show that for any dart d on P there exists a *dart triangle*, defined as a triangle on the three corners of a dart containing no more points of P than the tail of that dart (see Figure 3a), with the property that such a dart triangle shares the tip and tail with the original dart d (but may have other wings, see Figures 3b-d).

▶ **Lemma 2.1.** *For any 1-DPT of a point set P containing an arbitrary dart d there is a flip sequence to a 1-DPT with a dart triangle that has the same tip and tail as d.*

Proof sketch. We define a flip sequence that creates a dart triangle sharing the tip and tail with d. We first flip to a triangulation where the wings of d are connected by an edge (see Figure 3a). Such a flip sequence exists, since Dyn et al. [7] proved that a triangulation can be flipped to any other triangulation, without ever flipping an edge in a set D of constrained edges. We insert the spine edge and add it to D along with all other edges of d. If there are points in the dart triangle (see Figure 3b), we flip to a triangulation on those points that can easily be flipped to an empty dart triangle (see Figures 3c-d), and remove the spine edge. ◀

Second, we show that for a 1-DPT with the tail of the dart at point $p_d \in P$, the tip can be flipped to any point $p \in P \setminus \{p_d\}$ if this allocation of tip and tail permits a 1-DPT of P.

▶ **Lemma 2.2.** *If a point $p_d \in P$ is the tail of the single dart in two 1-DPTs $\mathcal{T}_1$ and $\mathcal{T}_2$ on point set P, then there is a flip sequence between $\mathcal{T}_1$ and $\mathcal{T}_2$.*

Proof sketch. We apply Lemma 2.1 to $\mathcal{T}_1$ and $\mathcal{T}_2$ to ensure the respective darts are in dart triangles, and use the obtained flip sequences as prefixes/suffixes for our final sequence. We

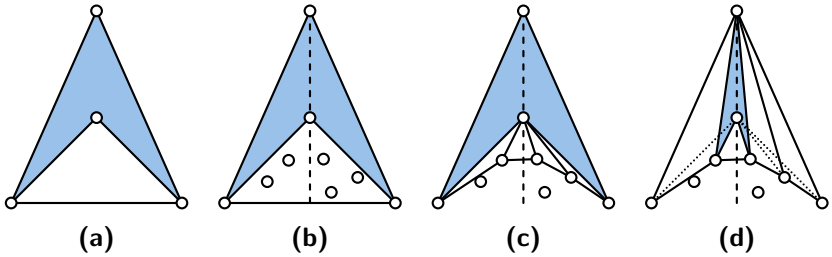


Figure 3 **(a)** A dart triangle. **(b)** A dart with an edge connecting the wings. The vertices in the bottom face are split by the extension of the (dashed) spine. **(c)** The specific triangulation between the dart and the upper envelop of the subset P' of points of P in the triangle formed by the wings and tail of d, together with the wings of d. **(d)** A number of flips linear in $|P'|$ creates a dart triangle.

remove the tail vertex from the obtained triangulations and mark the new faces as special faces. This results in two (proper) triangulations $\mathcal{T}_1'$ and $\mathcal{T}_2'$, for which we can find a flip sequence transforming one into the other. We apply this flip sequence to flip from $\mathcal{T}_1$ and $\mathcal{T}_2$ with one change: any flip involving an edge adjacent to a special face is substituted for a short flip sequence as shown in Figure 4, to preserve the dart during the flip sequence. ◀

The flip sequence of Lemma 2.2 implies that all 1-DPTs in the class $\mathcal{G}_p$ of 1-DPTs that have the tail of the dart located at p reside in the same connected component of the flip graph. As the third and final step in finding the connected components of the flip graph for 1-DPTs on P, we still have to check whether the tail of the dart can move from one vertex $p \in P$ to $q \in P$, i.e., whether the 1-DPTs in $\mathcal{G}_p$ and $\mathcal{G}_q$ all reside in the same connected component. To do so, we consider, for each pair of vertices $p, q \in P$, each triple of points in P distinct from p and q and check whether no other points are located in the faces of any of the small configurations in Figure 5a-d. Observe that these five vertices must admit two overlapping darts with different tails. If this is the case, then by Lemma 2.2 we can flip to the 1-DPTs where the respective darts have their tips in the position prescribed in the small configuration, and perform the flip in the configuration to swap the tails. To complete the argument, we can use a finite case analysis to prove that we have to check only the configurations in Figure 5a-d; no other ways to swap tails in 1-DPTs exist.

▶ **Lemma 2.3.** *There exist exactly four configurations of five points, that allow a dart to move its tail with an edge flip, as illustrated in Figure 5a-d.*

▶ **Theorem 2.4.** *For 1-DPTs on a set P of n points with h points on the convex hull, there are at most $n - h$ components in the flip graph. The exact components can be determined by checking every quintuple of points that have a triangular convex hull.*

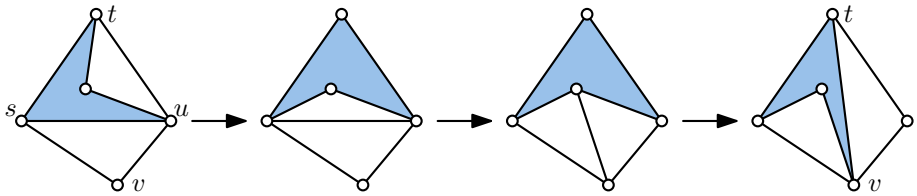


Figure 4 A flip sequence to flip an edge incident with the face containing the tail p_d of a dart.

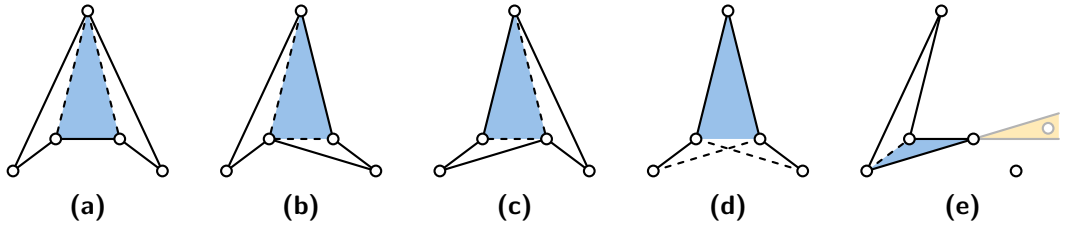


Figure 5 (a)-(d) All dart configurations of five points that allow us to move the tail of a dart using an edge flip. The darts share the blue triangle and each require one dashed edge. **(e)** Other triangles cannot use both middle vertices as tails, without a (non-existing) point in the yellow area.

3 Characterizing the Flip Graph for the Double Chain

In this section we consider the double chain, a point set consisting of a convex 4-gon being the hull of two concave chains of points next to opposite edges of the 4-gon, $P_{\asymp} = P_1 \cup P_2$, such that these concave chains do not cross the diagonals of the 4-gon, see Figure 6a. We can completely characterize the flip graph of k-DPTs on $P_{\asymp}$, for any possible k.

A k-DPT on a point set $P_{\asymp}$ admits two kinds of darts, *aligned darts* for which the spine connects two adjacent vertices of one concave chain, and *crossing darts* which have tip and tail in opposite concave chains. In this section we prove that the tail of a dart cannot swap between concave chains, and we say that a dart is *designated* to the chain where the tail is located. Additionally, we show that we can use edge flips to flip any k-DPT of an instance $P_{\asymp}$ to a *canonical* k-DPT. In such a k-DPT, all darts are aligned, and flipped as far left as possible (see Figure 6b); for darts designated to P_1 the wings on the opposite chain are at the leftmost point on P_2, while for darts designated to P_2 the analogous wings are at the rightmost tail on P_1. Furthermore, all wings inside the convex hulls of their designated chains are located at the rightmost point of the chain. By analyzing the number of canonical k-DPTs, we will analyze the number of connected components of the flip graph.

We first prove a few properties of aligned and crossing darts using geometric observations.

▶ **Lemma 3.1.** *In a k-DPT of point set $P_{\asymp}$, any aligned dart has one wing on the opposite concave chain, and for any choice of wings on the respective concave chains, a dart exists.*

▶ **Lemma 3.2.** *In a k-DPT of point set $P_{\asymp}$, the wings and tail of any crossing dart are consecutive on one concave chain; for any choice of tip on the opposite chain, a dart exists.*

▶ **Lemma 3.3.** *A k-DPT of point set $P_{\asymp}$ admits only crossing and aligned darts.*

Again using quintuples of points, we prove that tails cannot swap between concave chains.

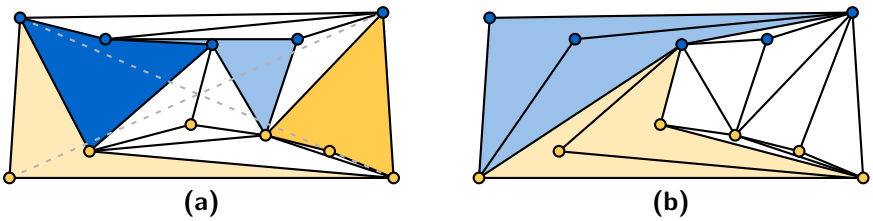


Figure 6 (a) A k-DPT for concave chains P_1 (blue) and P_2 (yellow), with $k = 4$. The aligned and crossing darts are light and dark colored, respectively. Note that P_1 and P_2 do not cross the dashed-grey diagonals. **(b)** The canonical k-DPT of **(a)**; white faces are triangulated arbitrarily.

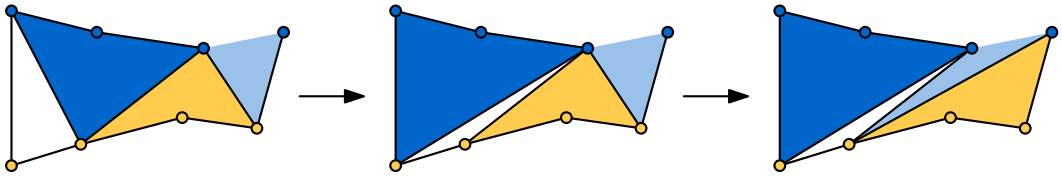


Figure 7 Moving the tip of a crossing dart, followed by moving the wing of an aligned dart. Faces (or parts thereof) inside the convex hull of either chain remain unchanged and are hence not drawn.

▶ **Lemma 3.4.** *In any k-DPT of point set $P_{\asymp} = P_1 \cup P_2$ the tail of a dart cannot swap between P_1 and P_2 through an edge flip.*

Next we show how to flip any k-DPT on an instance $P_{\asymp}$ to the canonical k-DPT for $P_{\asymp}$.

▶ **Lemma 3.5.** *Any k-DPT of point set $P_{\asymp}$ can be flipped to the canonical k-DPT.*

Proof sketch. We first consider all darts with tails on P_1 and flip them to be aligned darts on the left of the k-DPT, followed by all darts with tails on P_2. We move the vertices on the opposite chain to their (final) leftmost position, as in Figure 7. Then we move the spines in place by swapping between crossing and aligned darts with a single flip per swap, as in Figure 8. Some tails may not be located in the leftmost position, but they can move leftward when their dart is crossing, as in Figure 9. The end result can be seen in Figure 6b. ◀

Since any k-DPT on a set $P_{\asymp}$ can flip to a canonical k-DPT, we can now prove that:

▶ **Theorem 3.6.** *The number of connected components of the flip graph of k-DPTs on a point set $P_{\asymp} = P_1 \cup P_2$ is equal to the number of ways to designate k darts to P_1 or P_2.*

Proof. By Lemma 3.4 we know that the tails of darts cannot swap between convex chains. Furthermore, Lemma 3.5 tells us that all such k-DPTs can be flipped to a canonical k-DPT. Now observe that the number of darts designated to P_1 and P_2 completely determines the canonical k-DPT. Thus all k-DPTs with the same designation of darts to the concave chains are part of the same connected component of the flip graph. ◀

If P_1 and P_2 have $a+2$ and $b+2$ points, respectively, with $a, b \geq 0$, then there are $a+b$ points that can be a tail of a dart, and thus $k \leq a+b$. Distinguishing several cases depending on a, b, and k, we arrive at a unified formula for the number of components of the flip graph: Intuitively, we distribute k indistinguishable balls over 2 distinguishable (fixed-size) bins.

▶ **Corollary 3.7.** *The number of connected components of the flip graph of k-DPTs on a point set $P_{\asymp} = P_1 \cup P_2$, with $|P_1| = a+2$ and $|P_2| = b+2$, is equal to $\min\{a, b, k, a+b-k\} + 1$ for $0 \leq k \leq a+b$.*

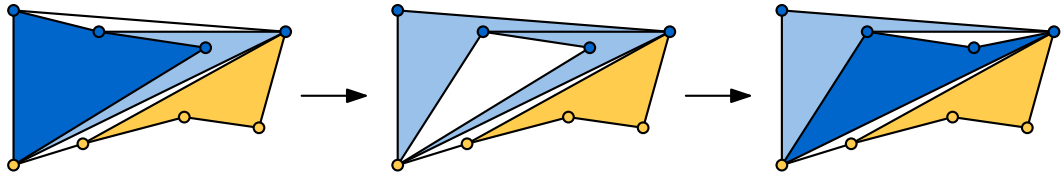


Figure 8 Flipping a crossing dart to be aligned, followed by a flip from an aligned to a crossing dart.

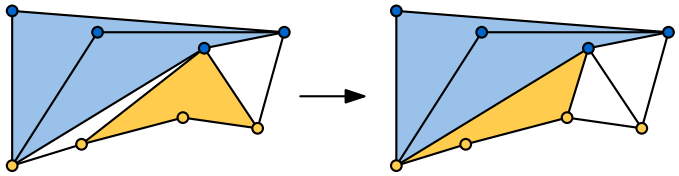


Figure 9 Moving a crossing dart designated to P_2 leftwards.

4 Conclusion

We studied the flip graph of pseudo-triangulations with faces of size 3 (triangles) and a bounded number k of size-4 faces (darts). For $k = 1$ in any point configuration, and for any k in the double chain point configuration, we showed how to find the connected components of the flip graph. For general point configurations, we conjecture that a similar approach will work for slightly higher values of k, such as $k \in \{2, 3\}$. Our goal in studying these special configurations is to obtain new insights into the flip graph of pointed pseudo-triangulation with faces of size 3 or 4. However, our current findings do not seem to allow us to reach much further than low values of k, as the required number of cases for higher k becomes infeasible.

References

1 Oswin Aichholzer, Franz Aurenhammer, Peter Brass, and Hannes Krasser. Pseudo-triangulations from surfaces and a novel type of edge flip. *SIAM Journal on Computing*, 32:1621–1653, 2003.

2 Oswin Aichholzer, Franz Aurenhammer, Thomas Hackl, Clemens Huemer, Alexander Pilz, and Birgit Vogtenhuber. 3-colorability of pseudo-triangulations. *International Journal of Computational Geometry & Applications*, 25(04):283–298, 2015.

3 Oswin Aichholzer, Thomas Hackl, David Orden, Alexander Pilz, Maria Saumell, and Birgit Vogtenhuber. Flips in combinatorial pointed pseudo-triangulations with face degree at most four. *International Journal of Computational Geometry & Applications*, 24(03):197–224, 2014.

4 Oswin Aichholzer, David Orden, Francisco Santos, and Bettina Speckmann. On the number of pseudo-triangulations of certain point sets. *Journal of Combinatorial Theory, Series A*, 115(2):254–278, 2008.

5 Sergey Bereg. Transforming pseudo-triangulations. *Information Processing Letters*, 90(3):141–145, 2004.

6 Bernard Chazelle, Herbert Edelsbrunner, Michelangelo Grigni, Leonidas Guibas, John Hershberger, Micha Sharir, and Jack Snoeyink. Ray shooting in polygons using geodesic triangulations. *Algorithmica*, 12(1):54–68, 1994.

7 Nira Dyn, Ifat Goren, and Shmuel Rippa. Transforming triangulations in polygonal domains. *Computer Aided Geometric Design*, 10(6):531–536, 1993.

8 Ruth Haas, David Orden, Günter Rote, Francisco Santos, Brigitte Servatius, Hermann Servatius, Diane Souvaine, Ileana Streinu, and Walter Whiteley. Planar minimally rigid graphs and pseudo-triangulations. In *Proc. 19th Symposium on Computational Geometry (SoCG)*, pages 154–163, 2003.

9 Lutz Kettner, David Kirkpatrick, Andrea Mantler, Jack Snoeyink, Bettina Speckmann, and Fumihiko Takeuchi. Tight degree bounds for pseudo-triangulations of points. *Computational Geometry*, 25(1-2):3–12, 2003.

10 Maarten Löffler, Tamara Mchedlidze, David Orden, Josef Tkadlec, and Jules Wulms. Flip graphs of pseudo-triangulations with face degree at most four. *CoRR*, abs/2402.12357, 2024. arXiv:2402.12357, doi:10.48550/ARXIV.2402.12357.

11 David Orden and Francisco Santos. The polytope of non-crossing graphs on a planar point set. *Discrete & Computational Geometry*, 33(2):275–305, 2005.

12 David Orden, Francisco Santos, Brigitte Servatius, and Herman Servatius. Combinatorial pseudo-triangulations. *Discrete Mathematics*, 307(3-5):554–566, 2007.

13 Michel Pocchiola and Gert Vegter. The visibility complex. *International Journal of Computational Geometry & Applications*, 6(3):279–308, 1996.

14 Gunter Rote, Francisco Santos, and Ileana Streinu. Pseudo-triangulations — a survey. *Contemporary Mathematics*, 453:343–410, 2008.

15 Ileana Streinu. Pseudo-triangulations, rigidity and motion planning. *Discrete and Computational Geometry*, 34(4):587, 2005.

Lower Bounding Minimal Faithful Sets of Verbose Persistence Diagrams

Brittany Terese Fasy[1,2], David L. Millman[2], and Anna Schenfisch[3]

1 **Montana State University**
`brittany.fasy@montana.edu`

2 **Blocky Inc.**
`david@blocky.rocks`

3 **TU Eindhoven, Eindhoven**
`a.k.schenfisch@tue.nl`

Abstract

An important aim of inverse problems in topological data analysis is to better understand sets of directional topological descriptors that uniquely correspond to an underlying shape; such sets are called *faithful* for the shape. Here, we specifically focus on sets of verbose persistence diagrams that arise from lower-star filtrations of geometric simplicial complexes. While explicit constructions of finite faithful sets in this setting exist in the literature, they do not come with any guarantees of optimality in terms of cardinality. To better understand faithful sets with low cardinality, we first establish a tight lower bound on the size of any faithful set. Then, we construct a family of simplicial complexes for which faithful sets must have size at least linear in the number of vertices.

1 Introduction

The persistent homology transform of a shape in Euclidean space was first explored in [15], and is the set of persistence diagrams corresponding to lower-star filtrations in every possible direction. Importantly, [15] establishes that this uncountably infinite set of persistence diagrams uniquely represents the underlying shape, i.e., it is *faithful* for the underlying shape. Since then, related theoretical work has focused on finding finite sets of persistence diagrams or other topological descriptors (such as Euler Characteristic functions or Betti functions) that are faithful [1, 3, 7, 14]. A key parameter in such work is whether or not the descriptors are assumed to be *verbose* or *concise*, i.e., if they contain information with a trivial lifespan. The relevance of verbose or concise descriptors was explored in [5, 13, 18], although with slightly different language.

In the full version of this work [8], we develop a framework for comparing the relative strengths of different topological descriptor types through the cardinality of faithful sets. Roughly speaking, if faithful sets of a particular topological descriptor type are always larger than faithful sets of another type, it is *weaker* than that other type. Thus, in such quantitative comparisons, it can be vital to understand minimum faithful sets. While explicitly identifying a minimum faithful set is a difficult problem in general, we are able to provide lower bounds on the cardinality of minimum faithful sets, both in general and in a worst-case construction. In what follows, we focus on the specific topological descriptor type of *verbose persistence diagrams*; we refer the reader to our full version to see how this and similar constructions apply to other common descriptor types.

40th European Workshop on Computational Geometry, Ioannina, Greece, March 13–15, 2024.
This is an extended abstract of a presentation given at EuroCG'24. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

2 Preliminary Considerations

We assume the reader has familiarity with ideas from topology, including homology and simplicial complexes. See [4, 9] for further details. We always take $\mathbb{N}$ to include zero. For a simplicial complex K, we use the notation K_i for its i-skeleton and n_i as the number of i-simplices. We always assume our simplicial complexes are geometric and finite. A *filter* of K is a map $f\colon K \to \mathbb{R}$ such that, for $\tau, \sigma \in K$, whenever τ is a face of σ, then we have $f(\tau) \leq f(\sigma)$. Then, letting $F(t) := f^{-1}(-\infty, t]$, the sequence $\{F(t)\}_{t\in\mathbb{R}}$ is the *filtration* associated to f; in particular, the filtration is a sequence of nested subcomplexes along with inclusion maps $F(s) \hookrightarrow F(t)$ for every $s \leq t$. Moreover, for each $k \in \mathbb{N}$, the inclusion $F(s) \hookrightarrow F(t)$ induces a linear map on homology, $H_k(F(s)) \to H_k(F(t))$. We write $\beta_k^{s,t}(K, f)$ to mean rank of this map, or simply $\beta_k^{s,t}$ if K and f are clear from context.

In particular, the *lower-star filter* of a simplicial complex K immersed in $\mathbb{R}^d$ with respect to some direction $s \in \mathbb{S}^{d-1}$, is the map $f_s : K \to \mathbb{R}$ defined by $f_s(\sigma) = \max\{s\cdot v \mid v \in K_0 \cap \sigma\}$. Note that s defines a preorder on K_0, $v_0 \leq v_1 \leq \ldots \leq v_{n_0-1}$. Then, the *lower-star filtration* of K with respect to s is

$$\emptyset \subset f_s^{-1}(-\infty, s\cdot v_0] \subseteq f_s^{-1}(-\infty, s\cdot v_1] \subseteq \ldots \subseteq f_s^{-1}(-\infty, s\cdot v_{n_0-1}] = K.$$

Any filter function has *compatible index filters*, which are functions $f' : K \to \mathbb{R}$ such that f' orders all the simplices of K uniquely and if $f(\tau) \leq f(\sigma)$, then $f'(\tau) \leq f'(\sigma)$. We say their corresponding filtrations are *compatible index filtrations*.

Our principal objects of study are *verbose* persistence diagrams. As they are closely related to the more familiar *concise* persistence, we begin with the following definition.

▶ **Definition 2.1** (Concise Persistence Diagram, ρ)**.** Let $f : K \to \mathbb{R}$ be a filter function. We define the *kth-dimensional persistence diagram* as the following multiset:

$$\rho_k^f := \left\{(i,j)^{\mu^{(i,j)}} \text{ s.t. } (i,j) \in \overline{\mathbb{R}}^2 \text{ and } \mu^{(i,j)} = \beta_k^{i,j-1} - \beta_k^{i,j} - \beta_k^{i-1,j-1} + \beta_k^{i-1,j}\right\},$$

where $\overline{\mathbb{R}} = \mathbb{R}\cup\{\pm\infty\}$ and $(i,j)^m$ denotes m copies of the point (i,j). The *persistence diagram* of f, denoted ρ^f, is the union of all k-dimensional persistence diagrams $\rho^f := \cup_{k\in\mathbb{N}}\rho_k^f$.

Since simplices can appear at the same parameter value in a general filtration, not all cycles are represented in the persistence diagram. However, having every simplex "appear" in the persistence diagram is helpful, in addition to being natural. Thus, we introduce *verbose persistence diagrams*, which contain this information.

▶ **Definition 2.2** (Verbose Persistence Diagram, $\hat{\rho}$)**.** Let $f : K \to \mathbb{R}$ be a filter and let f' be a compatible index filter. For $k \in \mathbb{N}$, the *k-dimensional verbose persistence diagram* is the following multiset:

$$\hat{\rho}_k^f := \left\{(f(\sigma_i), f(\sigma_j)) \;\mid\; (i,j) \in \rho_k^{f'}\right\}. \tag{1}$$

The *verbose persistence diagram* of f, denoted $\hat{\rho}^f$, is the union of all $\hat{\rho}_k^f$.

While concise persistence diagrams may feel more familiar, the idea of verbose persistence diagrams is not new. Indeed, many typical algorithms for computing persistence (e.g., [4, Chapter VII]), explicitly compute topological events with trivial lifespan but then discard them from the output. In [11], persistence diagrams are the same as our definition of $\hat{\rho}$. In [16], we see filtered chain complexes as a source of verbose persistence; [2, 12, 13, 17] also take this view.

If a verbose persistence diagram corresponds to a direction s in addition to a simplicial complex K, we use the notation $\hat{\rho}(K, s)$, or $\hat{\rho}(s)$ when K is clear from context. Furthermore, we refer to the set of verbose persistence diagrams parameterized by S as $\hat{\rho}(K, S)$. We denote the ith standard basis vector by e_i, so e_1 is the unit vector in the x-direction, etc. We say a (parameterized) set of verbose diagrams is *faithful* for a simplicial complex K if only K could have generated that same set of diagrams.

▶ **Definition 2.3** (Faithful Set)**.** Let K be simplicial complex in $\mathbb{R}^d$ and let $S \subseteq \mathbb{S}^{d-1}$. We say that $\hat{\rho}(K, S)$ *is faithful* if, for any K' we have the equality $\hat{\rho}(K', S) = \hat{\rho}(K, S)$ if and only if $K' = K$.

See Figure 1 for an example of a verbose persistence diagram and a non-faithful set.

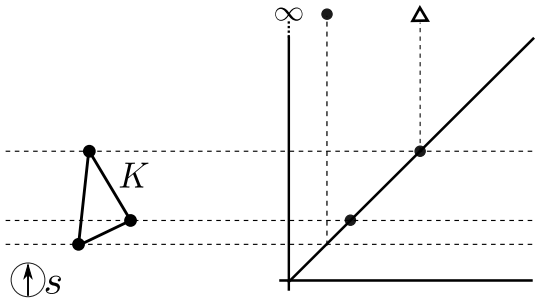


Figure 1 The verbose persistence diagram shown above, $\hat{\rho}(K, s)$, is identical to the corresponding concise persistence diagram, except for the on-diagonal points. Notice that the singelton set $\hat{\rho}(K, s)$ is not faithful; any cycle with vertices at the same heights as the vertices of K produces the same verbose persistence diagram.

3 Bounds on Faithful Sets

This section provides lower bounds on the size of faithful sets of verbose persistence diagrams. We begin with a tight lower bound.

▶ **Lemma 3.1** (Tight Lower Bound)**.** *Let K be a simplicial complex in $\mathbb{R}^d$. Suppose that $\hat{\rho}(K, S)$ is faithful. Then $|S| \geq d$, and this bound is tight.*

Proof. No vertex in K can be described using fewer than d coordinates. Thus, a set of verbose persistence diagrams with cardinality less than d cannot be faithful for K_0, let alone K. To see that this bound is tight, consider the case where K is a single vertex; verbose descriptors generated by any d pairwise linearly independent directions form a faithful set for the vertex (e.g., $\hat{\rho}(K, \{e_1, e_2, \ldots, e_d\})$). ◀

In many examples, we find that minimum faithful sets of verbose descriptors for simple simplicial complexes in $\mathbb{R}^d$ often have cardinality $d + 1$. Precise statements characterizing simplicial complexes with faithful sets of size $d + 1$ remain as ongoing work. However, frequently encountering faithful sets with a cardinality independent from the size of the simplicial complex is not at all surprising, since (unlike concise descriptors), verbose descriptors always have events corresponding to each simplex, regardless of how the complex is imbedded or immersed. We did not expect to find a simplicial complex whose minimum faithful set depended on the size of the complex; in fact, it was through trying to *disprove* the existence of such a complex that we came across the construction in this section.

In particular, in this section, we identify a family of simplicial complexes for which minimum faithful sets of verbose persistence diagrams are linear in the number of vertices. We use $\alpha_{i,j}$ to denote the angle that vector $v_j - v_i$ makes with the x-axis. We assume angles take value in $[0, 2\pi)$. We establish a preliminary observation, a specific instance of the general phenomenon that a simplicial complex stratifies the sphere of directions based on vertex order [3, 10].

▶ **Observation 1.** *Suppose that a simplicial complex K in $\mathbb{R}^2$ contains an edge $[v_1, v_2]$ such that v_1 and v_2 have degree one. Then a birth event occurs at the height of v_1 in $\hat{\rho}(K, s)$ for all s in the half of $\mathbb{S}^1$ defined by the open interval $H = (\alpha_{1,2} - \pi, \alpha_{1,2} + \pi)$ (i.e., all s so that $s \cdot v_1 > s \cdot v_2$) and as an instantaneous event for $s \in H^C$ (i.e., all s so that $s \cdot v_1 \leq s \cdot v_2$).*

We also establish the following elementary lemma.

▶ **Lemma 3.2.** *Consider a pair of nested triangles as in Figure 2. Then angle A is larger than θ, $\phi - B$, and $\psi - C$.*

Proof. Adding angles in the larger triangle, we see $\theta + \phi + \psi = \pi$. Then,

$$\begin{aligned} \theta + (\phi - B) + B + (\psi - C) + C &= \pi \\ (A + B + C) + \theta + (\phi - B) + (\psi - C) &= A + \pi \\ \pi + \theta + (\phi - B) + (\psi - C) &= A + \pi \\ \theta + (\phi - B) + (\psi - C) &= A. \end{aligned}$$

All the terms in the last line are positive, meaning A is larger than θ, $\phi - B$, and $\psi - C$. ◀

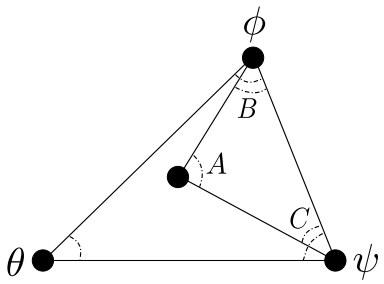


Figure 2 Nested triangles as discussed in Lemma 3.2

We now construct the building block that forms the complexes used in our bound.

▶ **Construction 1** (Clothespin Motif). *Let K be a simplicial complex in $\mathbb{R}^2$ with a vertex set $\{v_1, v_2, v_3, v_4\}$. Suppose that only v_3 is in the interior of the convex hull of $\{v_1, v_2, v_4\}$, and that the edge set consists of $[v_1, v_2]$ and $[v_3, v_4]$. See the left image in Figure 3.*

Construction 1 was built specifically for the following necessary condition for faithful sets of verbose descriptors. We state this condition in terms of $\hat{\rho}$'s in the following lemma.

▶ **Lemma 3.3** (Clothespin Repesentability). *Let K be a clothespin motif, as in Construction 1, and suppose that $\hat{\rho}(K, S)$ is faithful. Then we have at least one direction $s \in S$ such that the angle formed between s and $e_1 = (1, 0)$ lies in the region $[\alpha_{3,2} - \pi, \alpha_{3,4} - \pi] \cup [\alpha_{3,2} + \pi, \alpha_{3,4} + \pi]$.*

Proof. Let K' be a simplicial complex in $\mathbb{R}^2$ with the same vertex set as K, but with edges $[v_1, v_4]$ and $[v_2, v_3]$ (see the left side of Figure 3). Recall that, since $\hat{\rho}(K, S)$ is faithful, the set S must contain some s so that $\hat{\rho}(K, s) \neq \hat{\rho}(K', s)$.

Figure 3 The two simplicial complexes considered in the proof of Lemma 3.3.

Each vertex corresponds to either a birth event or an instantaneous event depending on the direction of filtration. We proceed by considering each vertex v_i individually and determining subsets $R_i \subset \mathbb{S}^1$ such that, whenever $s \in R_i$, the event at $s \cdot v_i$ is different when filtering over K versus K', but for $s_* \notin R_i$, the type of event at $s_* \cdot v_i$ is the same between the two graphs. Figure 4 shows these regions, and in what follows, we define them precisely.

First, consider v_1. By Observation 1, $v_1 \in K$ corresponds to a birth event for all directions in the interval $B = (\alpha_{1,2} - \pi, \alpha_{1,2} + \pi)$ and $v_1 \in K'$ corresponds to a birth event for all directions in the interval $B' = (\alpha_{1,4} - \pi, \alpha_{1,4} + \pi)$. Then we write $R_1 = (B \setminus B') \cup (B' \setminus B)$, which is the wedge-shaped region such that for any $s \in R_1$, the type of event associated to $v_1 \in K$ and $v_1 \in K'$ differ, meaning $\hat{\rho}(K, s) \neq \hat{\rho}(K', s)$.

Using this same notation, identify the wedge shaped region R_i for vertex $i \in [2, 3, 4]$ such that any direction from R_i generates $\hat{\rho}$'s that have different event types at vertex v_i when filtering over K versus K'. Similar arguments for $i \in [2, 3, 4]$ give us the complete list;

$$\begin{aligned}
R_1 &= (\alpha_{1,2} - \pi, \alpha_{1,4} - \pi] \cup [\alpha_{1,2} + \pi, \alpha_{1,4} + \pi) \\
R_2 &= (\alpha_{2,3} - \pi, \alpha_{2,1} - \pi] \cup [\alpha_{2,3} + \pi, \alpha_{2,1} + \pi) \\
R_3 &= (\alpha_{3,2} - \pi, \alpha_{3,4} - \pi] \cup [\alpha_{3,2} + \pi, \alpha_{3,4} + \pi) \\
R_4 &= (\alpha_{1,4} - \pi, \alpha_{3,4} - \pi] \cup [\alpha_{1,4} + \pi, \alpha_{3,4} + \pi)
\end{aligned}$$

Let $W = \cup_{i=1}^{4} R_i$. Then, for any $s \in W$, we have $\hat{\rho}(K, s) \neq \hat{\rho}(K', s)$, and for any $s_* \in W^C$, we have $\hat{\rho}(K, s_*) = \hat{\rho}(K', s_*)$.

Finally, we claim that W is the closure of R_3, denoted $\overline{R}_3$, i.e., exactly the region described in the lemma statement. This is a direct corollary to Lemma 3.2; the angles swept out by each regions correspond to the angles formed by pairs of edges in K and K'; in particular, the angle $\measuredangle v_2 v_3 v_4$ is the largest and geometrically contains the others. This means the extremal boundaries over all R_i's are formed by the angles $\alpha_{2,3} \pm \pi$ and $\alpha_{3,4} \pm \pi$, the defining angles of R_3. Observing that each of these four angles appears as an included endpoint for some R_i, we see $R_1, R_2, R_4 \subseteq \overline{R}_3 = W$ (see Figure 4), as desired. ◀

To get a deeper intuition for this result, observe that the verbose diagrams corresponding to K and K' of Figure 3 are identical when we filter in direction e_1, but when we filter in direction e_2, they are distinct. We refer to the wedge shaped region of directions for which the corresponding verbose diagrams have this distinction as a clothespin's *region of observability* (similar to observability for χ's discussed in [6, 3]). We notate the region as $W = [\alpha_{3,4} - \pi, \alpha_{2,3} - \pi] \cup [\alpha_{3,4} + \pi, \alpha_{2,3} + \pi]$. Crucially, W is defined by the angle $v_2 v_3 v_4$, so a different embedding of K could result in a smaller region.

▶ Remark (W Can be Arbitrarily Small). As the angle $\measuredangle v_2 v_3 v_4$ approaches zero, the region of observability, W, described in the proof of Lemma 3.3 also approaches zero.

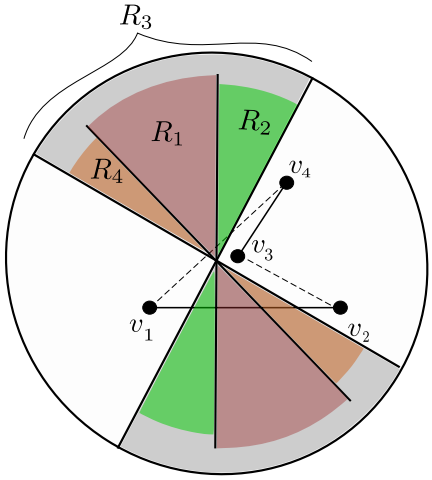


Figure 4 The regions described in the proof of Lemma 3.3. K is shown as solid black edges and K' as dashed edges. For any lower-star filtration in a direction contained in R_i, the event at vertex v_i differs when considering K or K', thus, such directions are able to distinguish K from K'. Note that any direction outside the regions of observability (i.e., the non-shaded portions of the circle) is not able to distinguish K from K'.

To construct a family of simplicial complexes, each of which must have at least $\Theta(n_0)$ verbose descriptors to form a faithful set, we use the preceeding remark to knit together clothespin motifs (Construction 1) in the following way.

▶ **Construction 2** (Clothespins on a Semicircular Clothesline). *Let K_m be a simplicial complex in $\mathbb{R}^2$ formed by m clothespins (Construction 1) such that the regions of observability for each clothespin do not overlap. Note this is possible for any m by the remark above.*

See Figure 5 for an example of K_m for $m = 4$. This construction implies a lower bound on the number of $\hat{\rho}$'s needed to fully represent a simplicial complex.

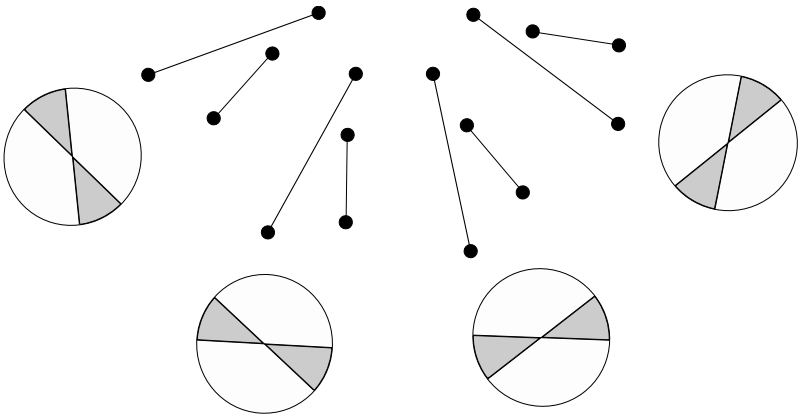


Figure 5 An example of Construction 2 for $m = 4$. The regions of observability are shown below each clothespin. By construction, each of these four double wedges define disjoint regions of $\mathbb{S}^2$.

▶ **Theorem 3.4** (Lower Bound for Worst-Case $\hat{\rho}$'s Complexity). *Let K_m be as in Construction 2 and suppose $\hat{\rho}(K_m, S)$ is a faithful set. Then S must contain at least one direction in each*

of the m regions of observability, meaning that $|S| \geq m = n_0/4$. Thus, the size of a faithful set of $\hat{\rho}$'s for K_m is $\Omega(n_0)$.

References

1. Robin Lynne Belton, Brittany Terese Fasy, Rostik Mertz, Samuel Micka, David L. Millman, Daniel Salinas, Anna Schenfisch, Jordan Schupbach, and Lucia Williams. Reconstructing embedded graphs from persistence diagrams. *Computational Geometry: Theory and Applications*, 90, October 2020.
2. Wojciech Chachólski, Barbara Giunti, Alvin Jin, and Claudia Landi. Decomposing filtered chain complexes: Geometry behind barcoding algorithms. *Computational Geometry*, 109:101938, 2023.
3. Justin Curry, Sayan Mukherjee, and Katharine Turner. How many directions determine a shape and other sufficiency results for two topological transforms. *Transactions of the American Mathematical Society, Series B*, 9(32):1006–1043, 2022.
4. Herbert Edelsbrunner and John Harer. *Computational Topology: An Introduction*. American Mathematical Society, 2010.
5. Brittany Terese Fasy, Samuel Micka, David L. Millman, Anna Schenfisch, and Lucia Williams. Challenges in reconstructing shapes from Euler characteristic curves, 2018. arXiv:1811.11337.
6. Brittany Terese Fasy, Samuel Micka, David L. Millman, Anna Schenfisch, and Lucia Williams. Challenges in reconstructing shapes from Euler characteristic curves. *Fall Workshop Computational Geometry*, 2018.
7. Brittany Terese Fasy, Samuel Micka, David L. Millman, Anna Schenfisch, and Lucia Williams. Efficient graph reconstruction and representation using augmented persistence diagrams. *Proceedings of the 34th Annual Canadian Conference on Computational Geometry*, 2022.
8. Brittany Terese Fasy, David L. Millman, and Anna Schenfisch. Ordering topological descriptors, 2024. `arXiv:2402.13632`.
9. Allen Hatcher. Algebraic topology, Cambridge Univ. *Press, Cambridge*, 2002.
10. Jacob Leygonie, Steve Oudot, and Ulrike Tillmann. A framework for differential calculus on persistence barcodes. *Foundations of Computational Mathematics*, pages 1–63, 2021.
11. Alexander McCleary and Amit Patel. Edit distance and persistence diagrams over lattices. *SIAM Journal on Applied Algebra and Geometry*, 6(2):134–155, 2022.
12. Facundo Mémoli and Ling Zhou. Stability of filtered chain complexes, 2022. arXiv:2208.11770.
13. Facundo Mémoli and Ling Zhou. Ephemeral persistence features and the stability of filtered chain complexes. In *39th International Symposium on Computational Geometry (SoCG 2023)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2023.
14. Samuel Micka. *Algorithms to Find Topological Descriptors for Shape Reconstruction and How to Search Them*. PhD thesis, Montana State University, 2020.
15. Katharine Turner, Sayan Mukherjee, and Doug M. Boyer. Persistent homology transform for modeling shapes and surfaces. *Information and Inference: A Journal of the IMA*, 3(4):310–344, 2014.
16. Michael Usher and Jun Zhang. Persistent homology and Floer–Novikov theory. *Geometry & Topology*, 20(6):3333–3430, 2016.
17. Ling Zhou. *Beyond Persistent Homology: More Discriminative Persistent Invariants*. PhD thesis, The Ohio State University, 2023.
18. Zhen Zhou, Yongzhen Huang, Liang Wang, and Tieniu Tan. Exploring generalized shape analysis by topological representations. *Pattern Recognition Letters*, 87:177–185, 2017.

On exact covering with unit disks

Ji Hoon Chun[1], Christian Kipp[1], and Sandro Roch[1]

1 **Technische Universität Berlin**
lastname@math.tu-berlin.de

Abstract

We study the problem of covering a given point set in the plane by unit disks so that each point is covered exactly once. We prove that 17 points can always be exactly covered. On the other hand, we construct a set of 657 points where an exact cover is not possible.

Related Version arXiv:2401.15821

1 Introduction

In 2008, Inaba [10] gave the following puzzle about covering sets of points in the plane:

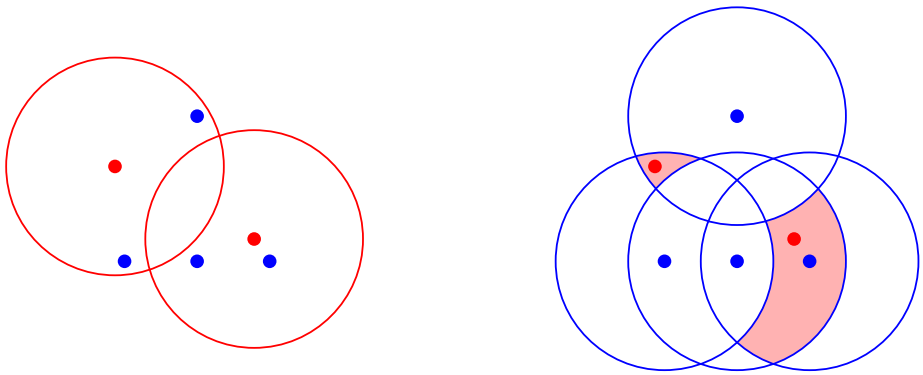
Show that any set of 10 points in $\mathbb{R}^2$ can be covered by nonoverlapping unit disks.

Inaba solved this puzzle [11, 17] with an elegant probabilistic argument (a deterministic proof is also possible). In this article, we study a relaxed version of this covering problem. Given a point set $X \subset \mathbb{R}^2$, can we find a family $\mathcal{D}$ of not necessarily disjoint unit disks so that each point $\mathbf{x} \in X$ is contained in exactly one disk $D \in \mathcal{D}$? We call such a family an *exact cover* of X. For example, in Figure 1a, the two red disks form an exact cover of the four blue points.

Let $B^2 \coloneqq \{\mathbf{x} \in \mathbb{R}^2 \mid \|\mathbf{x}\| < 1\}$, where $\|\cdot\|$ denotes the Euclidean norm. We define an (open) *disk* with *center* $\mathbf{c} \in \mathbb{R}^2$ and *radius* $r > 0$ as the set $D_{\mathbf{c},r} \coloneqq \mathbf{c} + rB^2$; if $r = 1$ we call it the *unit disk* and write $D_{\mathbf{c}}$.

▶ **Definition 1.1.** Let σ_2 be the largest $n \in \mathbb{N}$ such that any set of n points in the plane can be covered by disjoint unit disks. Let $\widehat{\sigma}_2 \in \mathbb{N}$ be the corresponding number for the relaxed problem involving exact covers.

As a covering using disjoint disks is also an exact covering, we have the basic relationship $\widehat{\sigma}_2 \geq \sigma_2$. The current best known bounds for σ_2 are $12 \leq \sigma_2 \leq 44$ [1]. Aloupis, Hearn, Iwasawa, and Uehara (2012) [1] improved Inaba's lower bound to $\sigma_2 \geq 12$ through a careful



(a) **(b)**

Figure 1 Left: primal solution (exact covering). Right: dual solution (exact hitting set).

40th European Workshop on Computational Geometry, Ioannina, Greece, March 13–15, 2024.
This is an extended abstract of a presentation given at EuroCG'24. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

analysis of the probabilistic method on one-dimensional slices of the plane. In the other direction, σ_2 is finite: Intuitively, with a dense enough arrangement of points, this problem becomes similar to the problem of covering the entire set $\operatorname{conv} X$, which is impossible using disjoint disks. Specific upper bounds were reduced in rapid succession from $\sigma_2 < 60$ by Winkler (2010) [17] to $\sigma_2 < 55$ by Elser (2011) [7] and $\sigma_2 < 53$ by Okayama, Kiyomi, and Uehara (2012) [16]. Most recently, Aloupis, Hearn, Iwasawa, and Uehara (2012) [1] proved $\sigma_2 < 50$ "by hand" and demonstrated $\sigma_2 < 45$ using computer calculations.

1.1 Results

In Section 2, we build on some of the mentioned works on lower bounds to establish the following lower bound on $\widehat{\sigma}_2$:

▶ **Theorem 1.2.** *We have* $\widehat{\sigma}_2 \geq 17$.

The finiteness of $\widehat{\sigma}_2$ can be deduced by a similar argument as the finiteness of of σ_2. In Section 3 we construct a close arrangement of points that cannot be exactly covered, leading to the following (rather weak) upper bound on $\widehat{\sigma}_2$:

▶ **Theorem 1.3.** *We have* $\widehat{\sigma}_2 < 657$.

For the full proofs of Theorem 1.2 and Theorem 1.3 we refer to the appendix; nevertheless, we provide sketches of the proofs below.

1.2 Relation between exact covering and exact hitting

We denote by $\mathcal{X}$ the collection of all finite point sets in $\mathbb{R}^2$. A point $\mathbf{x} \in \mathbb{R}^2$ is contained in a unit disk $D_{\mathbf{c}}$ centered at $\mathbf{c} \in \mathbb{R}^2$ if and only if $\mathbf{c}$ is contained in the unit disk $D_{\mathbf{x}}$ centered at $\mathbf{x}$. By this simple observation, the problem of exactly covering some given $X \in \mathcal{X}$ by unit disks (*primal problem*) becomes equivalent to the following *dual problem*: Let $\mathcal{D}_X \coloneqq \{D_{\mathbf{x}} \mid \mathbf{x} \in X\}$; find a $P \in \mathcal{X}$ such that each disk $D \in \mathcal{D}_X$ contains exactly one point $\mathbf{p} \in P$. See Figure 1b for an example of the dual problem. In the literature, such a set P is also called an *exact hitting set*. A dual solution P yields the solution $\mathcal{D} \coloneqq \{D_{\mathbf{p}} \mid \mathbf{p} \in P\}$ to the exact covering problem. Vice versa, a solution $\mathcal{D}$ to the exact covering problem gives a solution to the dual problem by taking the disk centers.

In the dual perspective, the boundary circles of the disks $\mathcal{D}_X$ decompose the plane into *cells*. Observe that all points in a given cell are contained in the same set of disks, so the exact position of a dual solution point $\mathbf{p} \in P$ is irrelevant. Hence, it is sufficient to select a set of cells so that each $D \in \mathcal{D}_X$ contains exactly one selected cell. In the example of Figure 1b, the two red shaded cells form a solution. This observation shows that the solution space for the dual problem and for the exact covering problem is in fact discrete, and methods such as Knuth's Algorithm X (see [13] or Section 7.2.2.1 in [14]), integer programming, or SAT solvers (see [12]) can be used.

2 A lower bound

We exclude the following trivial case from our proofs in this section: If X lies on a line then X can be covered by disjoint disks. Denote by $\mathcal{X}'$ the subset of $\mathcal{X}$ that excludes every point set on a line. To prove Theorem 1.2, we have to show that all $X \in \mathcal{X}'$ with $|X| \leq 17$ can be exactly covered. We combine three separate components on top of Inaba's original probabilistic proof. In Subsection 2.1 we show that $\widehat{\sigma}_2 \geq \sigma_2 + 4$. In Subsection 2.2 we

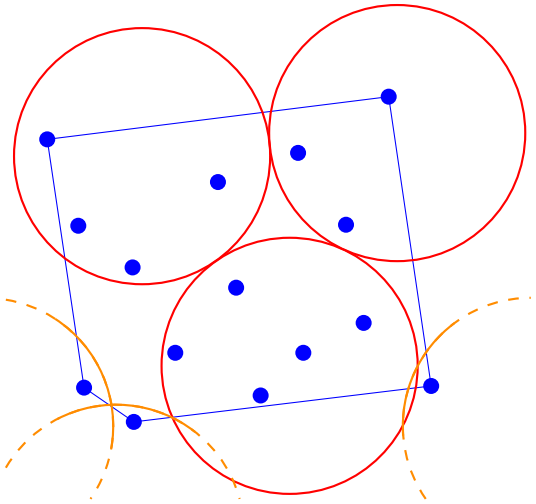


Figure 2 Extending the red disjoint disk covering of the non-boundary points by adding a new orange disk at each uncovered boundary point.

obtain $\widehat{\sigma}_2 \geq 16$ using a covering version of Betke, Henk, and Wills's parametric density [3] and $\widehat{\sigma}_2 \geq 17$ by showing that in some cases, a disk D that overlaps with $\operatorname{conv} X$ can be removed from an exact cover $\mathcal{D}$ of X so that $\mathcal{D} \setminus \{D\}$ is still an exact cover of X.

2.1 Boundary points

▶ **Definition 2.1.** Let $X \in \mathcal{X}$ and $\mathbf{v} \in X$. The point $\mathbf{v}$ is a *boundary point of* X if $\mathbf{v}$ is on the boundary of $\operatorname{conv} X$.

Let $X \in \mathcal{X}$ and $\mathbf{v}^1, \ldots, \mathbf{v}^k$ be the boundary points of X. László Kozma (private communication) observed that a covering $\mathcal{D}'$ of the non-boundary points $X \setminus \{\mathbf{v}^1, \ldots, \mathbf{v}^k\}$ by *disjoint* disks can always be extended to an exact cover of X. A boundary point $\mathbf{v}^i$ is covered by at most one disk in $\mathcal{D}'$ because the disks are disjoint. If $\mathbf{v}^i$ is not already covered by $\mathcal{D}'$, then it can be covered by a new disk which contains $\mathbf{v}^i$ but no other point of X. The resulting disk configuration yields an exact cover $\mathcal{D}$ of X (Figure 2). In particular, if $|X| \leq \sigma_2 + k$ then X can be exactly covered. We refer to this strategy as the *Extension Argument*:

▶ **Lemma 2.2** (Extension Argument). *Let $X \in \mathcal{X}$ and k be the number of boundary points of* $\operatorname{conv} X$.

1. *If $|X| \leq \sigma_2 + k$ then X can be exactly covered.*
2. *If $k \leq 2$ then X can be exactly covered regardless of $|X|$.*
3. *We have $\widehat{\sigma}_2 \geq \sigma_2 + 3$.*

As we assume that X does not lie on a line, we have $k \geq 3$, and the Extension Argument improves the basic inequality $\widehat{\sigma}_2 \geq \sigma_2$ to $\widehat{\sigma}_2 \geq \sigma_2 + 3$. This lower bound is limited by the case where $\operatorname{conv} X$ is a triangle, since otherwise X has at least four boundary points. Therefore, we wish to relax Definition 2.1 so that *every* $X \in \mathcal{X}'$ has at least four "generalized boundary points" that behave like boundary points.

▶ **Definition 2.3.** Let $X \in \mathcal{X}$ and $\mathbf{b} \in X$. The point $\mathbf{b}$ is a *generalized boundary point* of X if there exists a $\mathbf{c} \in \mathbb{R}^2$ such that $X \cap D_{\mathbf{c}} = \{\mathbf{b}\}$.

All vertices and boundary points of X are generalized boundary points of X. In the full version of this paper we prove the following generalization of the Extension Argument.

▶ **Lemma 2.4** (Generalized Extension Argument). *Let $X \in \mathcal{X}'$ and k be the number of generalized boundary points of* $\operatorname{conv} X$.

1. *If $|X| \leq \sigma_2 + k$ then X can be exactly covered. (That is, $\widehat{\sigma}_2(k) \geq \sigma_2 + k$.)*
2. *If $k \leq 3$ then X can be exactly covered regardless of $|X|$.*
3. *We have $\widehat{\sigma}_2 \geq \sigma_2 + 4$.*

We show that any $X \in \mathcal{X}'$ with a triangular convex hull contains at least four generalized boundary points or can be exactly covered regardless of the number of points. The fourth point is often, but not always, the closest non-vertex of X to the longest edge of $\operatorname{conv} X$.

Lemma 2.4 combined with Aloupis, Hearn, Iwasawa, and Uehara's [1] lower bound of $\sigma_2 \geq 12$ implies $\widehat{\sigma}_2 \geq 16$.

2.2 A parameterized version of Inaba's proof

Betke, Henk, and Wills (1994) [3] introduced the parametric density, a form of packing density for finitely many disks which are allowed to overlap, during their work on a packing problem called the Sausage Conjecture [8]. See [3, 5, 9] for further details on these topics. Let

$$A_2 \coloneqq \left\{ \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \in \mathbb{R}^2 \;\middle|\; \begin{array}{l} x_1 = \sqrt{3}\mu, \\ x_2 = 2\lambda + \mu, \end{array} \; \lambda, \mu \in \mathbb{Z} \right\}$$

be the hexagonal lattice and $\mathcal{A}_2^\rho \coloneqq \left\{ \mathbf{c} + \rho B^2 \,\middle|\, \mathbf{c} \in A_2 \right\}$ be the collection of disks of radius $\rho \geq 1$ that are centered at the points of A_2. This radius ρ is called the *parameter*; the case $\rho = 1$ reduces to the usual hexagonal packing in Inaba's proof. We call the subset of $\mathbb{R}^2$ covered by exactly one disk $D_\mathbf{c} \in \mathcal{A}_2^\rho$ the "good" region of $\mathcal{A}_2^\rho$ and its complement the "bad" region. An exact cover of X requires each point in X to avoid the "bad" region. If $\rho > 1$, then neighboring disks of $\mathcal{A}_2^\rho$ overlap (Figure 3), so the "bad" region includes any part of the plane covered by multiple disks. The critical value for ρ minimizes the total area of the "bad" region and so maximizes the lower bound for $\widehat{\sigma}_2$ (over all coverings of the form $\mathcal{A}_2^\rho$).

We use the same argument as Inaba but with the parameterized family $\mathcal{A}_2^\rho$ and combine it with the Extension Argument for another proof of $\widehat{\sigma}_2 \geq 16$. However, $\mathcal{A}_2^\rho$ has another advantage over $\mathcal{A}_2$. Removing one disk from $\mathcal{A}_2$ strictly expands the "bad" region, so is never beneficial for exact covering. However, removing one disk $D_\mathbf{c}$ from $\mathcal{A}_2^\rho$ changes the subsets of $D_\mathbf{c}$ which are covered by another disk in $\mathcal{A}_2^\rho$ from "bad" to "good." In the next subsection, we use this feature to raise our lower bound for $\widehat{\sigma}_2$.

2.3 A redundant disk

Suppose that $X \in \mathcal{X}$ has a triangular or quadrilateral convex hull, $\mathbf{v}^1 \in X$ is a boundary point that is covered by two disks of $\mathcal{A}_2^\rho$, and $\mathcal{A}_2^\rho$ is an exact cover of $X \setminus \left\{\mathbf{v}^1\right\}$. Under certain conditions, we can remove one of the disks $D_\mathbf{c}$ that covers $\mathbf{v}^1$ without breaking the exact cover. In other words, although $\mathcal{A}_2^\rho$ is not an exact cover of X, we show that $\mathcal{A}_2^\rho \setminus \{D_\mathbf{c}\}$ is an exact cover of X. This "redundant disk" method offers a slight benefit:

▶ **Lemma 2.5.** *Let $X \in \mathcal{X}'$ with $|X| \leq 17$. If* $\operatorname{conv} X$ *is a triangle or a quadrilateral, then X can be exactly covered.*

We present the technical details and proofs in the full version of our paper. Note that unlike the Extension Argument and parameterized family, which do not depend on the underlying disk configuration, the redundant disk method uses specific properties of A_2.

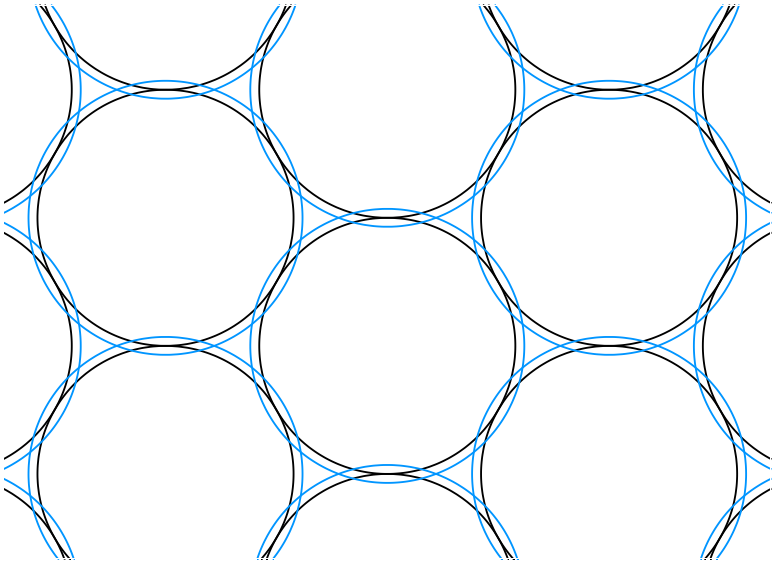


Figure 3 The disks of $\mathcal{A}_2^\rho$ for $\rho = 1$ (black) and $\rho = 1.07$ (blue).

Proof of Theorem 1.2. Let $X \in \mathcal{X}'$ with $|X| \leq 17$. If $\operatorname{conv} X$ has three or four sides, then X can be exactly covered by Lemma 2.5. If $\operatorname{conv} X$ has five or more sides, then X has at least five generalized boundary points, so X can be exactly covered by the Generalized Extension Argument 2.4 with Aloupis, Hearn, Iwasawa, and Uehara's [1] lower bound of $\widehat{\sigma}_2 \geq 12$. ◀

3 An upper bound

▶ **Definition 3.1.** Let $X \subset \mathbb{R}^2$ be a non-empty set. The *distance of a point* $\mathbf{y} \in \mathbb{R}^2$ *to* X is defined by

$$\operatorname{dist}(\mathbf{y}, X) \coloneqq \inf\{\|\mathbf{y} - \mathbf{x}\| \mid \mathbf{x} \in X\} \tag{1}$$

and the *ε-extension* of X (also called the *ε-thickening* of X) is given by

$$X_\varepsilon \coloneqq \left\{\mathbf{y} \in \mathbb{R}^2 \mid \operatorname{dist}(\mathbf{y}, X) \leq \varepsilon\right\}. \tag{2}$$

We say that X is an *ε-net* of $M \subset \mathbb{R}^2$ if $M \subset X_\varepsilon$.

▶ **Definition 3.2.** Let $M \subset \mathbb{R}^2$ and $\varepsilon > 0$. We say that M is an *ε-blocker* if every ε-net $X \in \mathcal{X}$ of M does not have an exact cover.

We recall that the covering number $N(M, \varepsilon)$ of a set $M \subset \mathbb{R}^2$ is the minimal cardinality of an ε-net of M. The following statement is a direct consequence of Definition 3.2.

▶ Proposition 3.3. Let $M \subset \mathbb{R}^2$ be an ε-blocker. Then $\widehat{\sigma}_2 < N(M, \varepsilon)$. ◀

Our upper bound on $\widehat{\sigma}_2$ follows from the following result, which asserts that every open disk of radius $R > 1$ is an ε-blocker for a suitably chosen $\varepsilon > 0$.

▶ Proposition 3.4. Let $\varepsilon \in \left(0, 7 - \sqrt{48} \approx 0.0718\right]$ and

$$R \geq \frac{3}{2}(1 + \varepsilon) - \frac{1}{2}\sqrt{1 - 14\varepsilon + \varepsilon^2}. \tag{3}$$

Then $D_{\mathbf{0},R} = \mathbf{0} + R\,B^2$ is an ε-blocker.

We now obtain Theorem 1.3 as a corollary to Proposition 3.3 by setting $\varepsilon \coloneqq 7 - \sqrt{48}$ and $R \coloneqq \frac{3}{2}(1 + \varepsilon) \approx 1.608$.

4 Conclusion

Our main result (Theorems 1.2 and 1.3) is $17 \leq \widehat{\sigma}_2 \leq 656$. An approach for improving the upper bound could be to search for small ε-nets of ε-blockers and to use Proposition 3.3.

The problem of finding $\widehat{\sigma}_2$ admits generalizations to all dimensions $d \geq 1$ and convex bodies $K \subset \mathbb{R}^d$. Let $\sigma(K)$ and $\widehat{\sigma}(K)$ be the largest n such that any n-point set in $\mathbb{R}^d$ can be covered by disjoint translates of K or exactly covered by translates of K, respectively (and write σ_d and $\widehat{\sigma}_d$ if $K = B^d$). Some of our methods, such as the Extension Argument and the parameterized family, have counterparts for other bodies K, but our other methods do not necessarily generalize.

Sphere packings are mostly empty space in high dimensions. Blichfeldt's upper bound of $\frac{d+2}{2} \cdot 2^{-\frac{1}{2}d}$ for the maximum packing density ([4], or see Section 6.1 of [18]) drops to less than or equal to 0.5 for $d \geq 6$. The density of the densest known packing in $d = 5$ is also below 0.5 (see Table 1.2 in Chapter 1 of [6], or [15]). Therefore, we cannot hope to cover many points by translating a dense packing of unit balls as in Inaba's proof [11, 17]. One possible strategy for "medium" dimensions around 5–10 is to choose one of several packings based on the arrangement of X.

With regard to lines of further research, we mention the computational complexity of disk covering. Considering the algorithmic issues that were discussed in Subsection 1.2, it is natural to ask the following question: Given $X \in \mathcal{X}$, is it NP-hard to decide whether X has an exact cover? Ashok, Basu Roy, and Govindarajan (2020) [2] showed that it is NP-hard to decide the following problem: Given a finite set $\mathcal{R}$ of unit squares and given an $X \in \mathcal{X}$, is there a subset $\mathcal{R}' \subset \mathcal{R}$ that exactly covers X? Their proof can be easily adopted for a given family $\mathcal{R}$ of unit disks. It might also be interesting to study the computational complexity if the number of disks in the exact cover is specified.

Acknowledgments

Thanks to László Kozma, Michaela Borzechowski, and Günter Rote for their feedback and helpful discussions.

This work was partially supported by the Deutsche Forschungsgemeinschaft (DFG), Graduiertenkolleg "Facets of Complexity" (GRK 2434).

References

1 Greg Aloupis, Robert A. Hearn, Hirokazu Iwasawa, and Ryuhei Uehara. Covering points with disjoint unit disks. In *Canadian Conference on Computational Geometry*, 2012. URL: https://api.semanticscholar.org/CorpusID:16280099.

2 Pradeesha Ashok, Aniket Basu Roy, and Sathish Govindarajan. Local search strikes again: PTAS for variants of geometric covering and packing. *J. Comb. Optim.*, 39(2):618–635, 2020. doi:10.1007/s10878-019-00432-y.

3 Ulrich Betke, Martin Henk, and Jörg Wills. Finite and infinite packings. *Journal für die reine und angewandte Mathematik*, 1994:165–192, 01 1994. doi:10.1515/crll.1994.453.165.

4 Hans Frederick Blichfeldt. The minimum value of quadratic forms, and the closest packing of spheres. *Mathematische Annalen*, 101(1):605–608, 1929. doi:10.1007/BF01454863.

5 Károly Böröczky, Jr. *Finite Packing and Covering*. Cambridge Tracts in Mathematics. Cambridge University Press, 2004.

6 John H. Conway and Neil J. A. Sloane. *Sphere Packings, Lattices and Groups*, volume 290 of *Grundlehren der mathematischen Wissenschaften*. Springer-Verlag New York, Inc., 3rd edition, 1999.

7 Veit Elser. Packing-constrained point coverings, 2011. arXiv:1101.3468.

8 László Fejes Tóth. Research problem 13. *Periodica Mathematica Hungarica*, 6(2):197–199, 1975.

9 Martin Henk and Jörg M. Wills. Packings, sausages and catastrophes. *Beiträge zur Algebra und Geometrie / Contributions to Algebra and Geometry*, 2020. doi:10.1007/s13366-020-00502-x.

10 Naoki Inaba, 2008. URL: http://inabapuzzle.com/hirameki/suuri_4.html.

11 Naoki Inaba, 2008. URL: http://inabapuzzle.com/hirameki/suuri_ans4.html.

12 Tommi Junttila and Petteri Kaski. Exact cover via satisfiability: An empirical study. In David Cohen, editor, *Principles and Practice of Constraint Programming – CP 2010*, pages 297–304. Springer Berlin Heidelberg, 2010.

13 Donald E. Knuth. Dancing links, 2000. arXiv:cs/0011047.

14 Donald E. Knuth. *The Art of Computer Programming*, volume 4B. Pearson Education, Inc., 2023.

15 Gabriele Nebe and Neil J. A. Sloane. Table of densest packings presently known, Feb 2012. URL: math.rwth-aachen.de/~Gabriele.Nebe/LATTICES/density.html [cited 2024-01-28].

16 Yosuke Okayama, Masashi Kiyomi, and Ryuhei Uehara. On covering of any point configuration by disjoint unit disks. *Geombinatorics*, 22(1):14–23, 2012.

17 Peter Winkler. Puzzled: Solutions and sources. *Commun. ACM*, 53(9):110, Sep 2010. doi:10.1145/1810891.1810917.

18 Chuanming Zong. *Sphere Packings*. Universitext. Springer-Verlag New York, Inc., 1999.

Fast Approximations and Coresets for (k, ℓ)-Median under Dynamic Time Warping

Jacobus Conradi[1], Benedikt Kolbe[2], Ioannis Psarros[3], and Dennis Rohde[1]

1 **Department of Computer Science, University of Bonn, Germany**
2 **Hausdorff Center for Mathematics, University of Bonn, Germany**
3 **Archimedes, Athena Research Center, Greece**

Abstract

We present algorithms for the computation of ε-coresets for k-median clustering of point sequences in $\mathbb{R}^d$ under the p-dynamic time warping (DTW) distance. Coresets under DTW have not been investigated before, and the analysis is not directly compatible with existing methods as DTW is not a metric. We achieve our results by investigating approximations of DTW that provide a trade-off between the provided accuracy and amenability to known techniques. In particular, we observe that given n curves under DTW, one can directly construct a metric that approximates DTW on this set, permitting the use of the wealth of results on metric spaces for clustering purposes. The resulting approximations are the first with polynomial running time and achieve a very similar approximation factor compared to state-of-the-art techniques.

1 Introduction

One of the core challenges of contemporary data analysis is the handling of massive data sets. A powerful approach to clustering problems involving such sets is data reduction, and ε-coresets offer a popular approach that has received substantial attention [4, 5, 14]. An ε-coreset is a problem-specific condensate of the given input set of reduced size which captures its core properties towards the problem at hand and can be used as a proxy to run an algorithm on, producing a solution with a relative error of $(1 \pm \varepsilon)$.

Clustering and especially k-median represent fundamental tasks in classification problems, where they have been extensively studied for various spaces. With the growing availability of e.g. geospatial tracking data, clustering problems for time series or curves have received growing attention both from a theoretical and applied perspective. In practice, time series classification largely relies on the dynamic time warping (DTW) distance and is widely used in the area of data mining. Simple nearest neighbor classifiers under DTW are considered hard to beat [17, 24] and much effort has been put into making classification using DTW computationally efficient [16, 19, 20, 21].

For time series and curves, k-median takes the shape of the (k, ℓ)-median problem, where the sought-for center curves are restricted to have a complexity (number of vertices) of at most ℓ, with a two-fold motivation. First, the otherwise NP-hard problem becomes tractable, and second, it suppresses overfitting.

The construction of ε-coresets for the (k, ℓ)-median problem for DTW is precisely what this paper will address. To this end, we adapt the framework of *sensitivity sampling* by Feldman and Landberg [13] to our setting. We rely on approximations of nearly all objects involved in our inquiry, thereby improving the bounds we obtain for the VC dimension of the range spaces in question and broadening the scope of our approach.

All presently known approaches to the approximation of the (k, ℓ)-median problem are based on an approximation scheme [6, 10, 12, 1, 7, 18]. For DTW, the best algorithm [8]

40th European Workshop on Computational Geometry, Ioannina, Greece, March 13–15, 2024.
This is an extended abstract of a presentation given at EuroCG'24. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

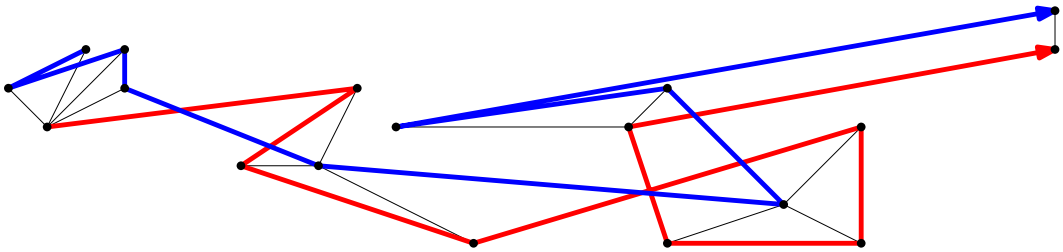


Figure 1 Example of a traversal between the red and blue curve realizing the dynamic time warping distance. The sum of the black distances is minimized.

has running time exponential in k, roughly with a dependency of $\widetilde{O}((32k^2\varepsilon^{-1})^{k+2}n)$.

Our results and methods We derive a bound on the VC dimension of a range space that approximates that of closed balls under DTW, obtained from a distance function approximating DTW. We modify and apply the sensitivity sampling framework by Feldman and Langberg [13], which relies on the bounds of the VC dimension and requires a first rough approximation of the (k, ℓ)-median problem under DTW, to construct coresets for (k, ℓ)-median under DTW. To adapt the sensitivity sampling framework to our (non-metric) setting, we investigate weaker versions of the triangle inequality for DTW and find a generalized iterated triangle inequality (Lemma 4.1). This novel inequality allows the approximation of DTW with a metric and thus the application of metric clustering algorithms.

2 Preliminaries

We think of a sequence $(p_1, \ldots, p_m) \in \left(\mathbb{R}^d\right)^m$ of points in $\mathbb{R}^d$ as a (polygonal) *curve*, with complexity m. We denote by $\mathbb{X}^d_{=m}$ the space of curves in $\mathbb{R}^d$ with complexity exactly m and by $\mathbb{X}^d_m$ the space of curves with complexity at most m.

▶ **Definition 2.1** (p-Dynamic Time Warping)**.** For given $m, \ell > 0$ we define the space $\mathcal{T}_{m,\ell}$ of (m, ℓ)-*traversals* as the set of sequences $((a_1, b_1), (a_2, b_2), \ldots, (a_l, b_l))$, such that

- $a_1 = 1$ and $b_1 = 1$; and $a_l = m$ and $b_l = \ell$,
- for all $i \in [l-1] := \{1, \ldots, l-1\}$ it holds that $(a_{i+1}, b_{i+1}) - (a_i, b_i) \in \{(1, 0), (0, 1), (1, 1)\}$.

For $p \in [1, \infty)$ and two curves $\sigma = (\sigma_1, \ldots, \sigma_m) \in \mathbb{X}^d_{=m}, \tau = (\tau_1, \ldots, \tau_\ell) \in \mathbb{X}^d_{=\ell}$ the (p-)Dynamic Time Warping distance (p-DTW) is defined as

$$\mathrm{dtw}_p(\sigma, \tau) = \min_{T \in \mathcal{T}_{m,\ell}} \left(\sum_{(i,j) \in T} \|\sigma_i - \tau_j\|_2^p \right)^{1/p},$$

where $\|\cdot\|_2^p$ is the Euclidean norm raised to the p-th power.

The central focus of the paper is the following clustering problem.

▶ **Definition 2.2** (Problem definition)**.** The (k, ℓ)-median problem for $\mathbb{X}^d_m$ and $k \in \mathbb{N}$ is the following: Given a set of $n \in \mathbb{N}$ input curves $T = \{\tau_1, \ldots, \tau_n\} \subset \mathbb{X}^d_m$, identify k center curves $C = \{c_1, \ldots, c_k\} \subset \mathbb{X}^d_\ell$ that minimize $\mathrm{cost}(T, C) = \sum_{\tau \in T} \min_{c \in C} \mathrm{dtw}(\tau, c)$.

An influential approach to solving k-median problems is to construct a point set that acts as proxy on which to run computationally more expensive algorithms that yield solutions with approximation guarantees. The condensed input set is known as a coreset.

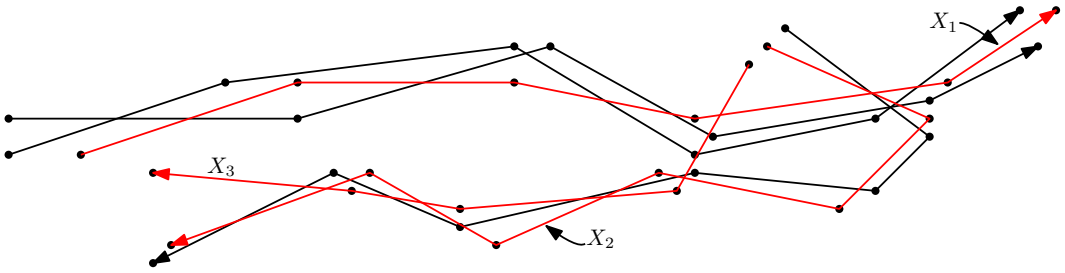


Figure 2 Illustration of a coreset (red), i.e. a weighted sparse representation of the original set of curves (in red and black). The weights in this case are $w(X_1) = 3$, $w(X_2) = 2$ and $w(X_3) = 1$.

▶ **Definition 2.3** (ε-coreset)**.** Let $T \subset \mathbb{X}_m^d$ be a finite set and $\varepsilon \in (0,1)$. Then a weighted multiset $S \subset \mathbb{X}_m^d$ with weight function $w : S \to \mathbb{R}_{>0}$ is a weighted ε-coreset for (k,ℓ)-median clustering of T under dtw_p if for all $C \subset \mathbb{X}_\ell^d$ with $|C| = k$

$$(1-\varepsilon)\,\mathrm{cost}(T,C) \leq \sum_{s\in S} w(s) \min_{c\in C} \mathrm{dtw}_p(s,c) \leq (1+\varepsilon)\,\mathrm{cost}(T,C).$$

▶ **Definition 2.4** ((α,β)-approximation)**.** Let a set of $n \in \mathbb{N}$ input curves $T = \{\tau_1,\ldots,\tau_n\} \subset \mathbb{X}_m^d$ be given. A set $\hat{C} \subset \mathbb{X}_\ell^d$ is called an (α,β)-approximation of (k,ℓ)-median, if $|\hat{C}| \leq \beta k$ and $\sum_{\tau\in T} \min_{c\in\hat{C}} \mathrm{dtw}_p(\tau,c) \leq \alpha \sum_{\tau\in T} \min_{c\in C} \mathrm{dtw}_p(\tau,c)$ for any $C \subset \mathbb{X}_\ell^d$ of size k.

Focusing on approximations allows us to pass through simplifications of the input curves.

▶ **Definition 2.5** ($(1+\varepsilon)$-approximate ℓ-simplifications)**.** Let $\sigma \in \mathbb{X}_m^d$, $\ell \in \mathbb{N}$ and $\varepsilon > 0$. We call $\sigma^* \in \mathbb{X}_\ell^d$ an $(1+\varepsilon)$-*approximate* ℓ-*simplification* if

$$\inf_{\sigma_\ell\in\mathbb{X}_\ell^d} \mathrm{dtw}_p(\sigma_\ell,\sigma) \leq \mathrm{dtw}_p(\sigma^*,\sigma) \leq (1+\varepsilon) \inf_{\sigma_\ell\in\mathbb{X}_\ell^d} \mathrm{dtw}_p(\sigma_\ell,\sigma).$$

A *range space* is defined as a pair of sets $(X,\mathcal{R})$, where X is the *ground set* and $\mathcal{R}$ is the *range set* which is a subset of the power set $\mathcal{P}(X) = \{X' | X' \subset X\}$. Let $(X,\mathcal{R})$ be a range space. For $Y \subseteq X$, we denote: $\mathcal{R}_{|Y} = \{R \cap Y \mid R \in \mathcal{R}\}$. If $\mathcal{R}_{|Y} = \mathcal{P}(Y)$, then Y is *shattered* by $\mathcal{R}$. A key property of range spaces is the so called Vapnik-Chernovenkis dimension [22, 23, 25] (VC dimension) which for a range space $(X,\mathcal{R})$ is the maximum cardinality of a shattered subset of X.

3 VC Dimension bounds and coresets for DTW

We now derive bounds on the VC dimension of a range space that approximates the range space induced by all closed balls in $\mathbb{X}_m^d$ centered at curves in $\mathbb{X}_\ell^d$ under p-DTW. The following lemma shows that one can determine (approximately) the p-DTW between two sequences, based solely on the signs of certain polynomials, that are designed to provide an approximation of all point-wise distances and forms the basis for the results in this section. Missing proofs and statements can be found in the full version [11].

▶ **Lemma 3.1.** *Let $\tau \in \mathbb{X}_{=\ell}^d$, $\sigma \in \mathbb{X}_{=m}^d$, $r > 0$ and $\varepsilon \in (0,1]$. For each $i \in [\ell]$, $j \in [m]$ and $z \in [\lfloor \varepsilon^{-1}+1 \rfloor]$ define $f_{i,j,z}(\tau,r,\sigma) = \|\tau_i - \sigma_j\|^2 - (z\cdot\varepsilon r)^2$. There is an algorithm that, given as input the values of $\mathrm{sign}(f_{i,j,z}(\tau,r,\sigma))$, for all $i \in [\ell], j \in [m]$ and $z \in [\lfloor \varepsilon^{-1}+1 \rfloor]$, outputs a value in $\{0,1\}$ such that if $\mathrm{dtw}_p(\tau,\sigma) \leq r$ then it outputs 1 and if $\mathrm{dtw}_p(\tau,\sigma) > (1+(m+\ell)^{1/p}\varepsilon)r$ then it outputs 0.*

The algorithm of Lemma 3.1 essentially implements approximate p-DTW balls membership and satisfies the requirements set by previous results that upper bound the VC dimension by decomposing the underlying predicate to sign evaluations of polynomials (Theorem 8.3 [2]). However, it is only defined on curves in $\mathbb{X}^d_{=\ell}$ and $\mathbb{X}^d_{=m}$. We extend the approach to all curves in $\mathbb{X}^d_m$, which provides the basis for a distance function $\widetilde{\mathrm{dtw}_p}$ between elements of $\mathbb{X}^d_m$ and $\mathbb{X}^d_\ell$ that approximates dtw_p with a relative error of $(1+\varepsilon)$. The properties of $\widetilde{\mathrm{dtw}_p}$ culminate in the following theorem giving a bound on the VC dimension on the approximate range space of p-DTW induced by $\widetilde{\mathrm{dtw}_p}$.

▶ **Theorem 3.2.** *Let $\varepsilon \in (0,1]$ and $\widetilde{\mathcal{R}}^p_{m,\ell} = \{\{x \in \mathbb{X}^d_m \mid \widetilde{\mathrm{dtw}_p}(x,\tau) \le r\} \subset \mathbb{X}^d_m \mid \tau \in \mathbb{X}^d_\ell, r > 0\}$ be the range set consisting of all balls centered at elements of $\mathbb{X}^d_\ell$ under $\widetilde{\mathrm{dtw}_p}$ in $\mathbb{X}^d_m$. The VC dimension of $(\mathbb{X}^d_m, \widetilde{\mathcal{R}}^p_{m,\ell})$ is in $O(d\ell \log(\ell m \varepsilon^{-1}))$.*

Sensitivity bounds and coresets for DTW To make use of the sensitivity sampling framework for coresets by Feldman and Langberg [13], we recast the input set $T \subset \mathbb{X}^d_m$ as a set of functions. Consider for any $y \in \mathbb{X}^d_m$ and $\varepsilon > 0$ the real-valued function $\widetilde{f}_y$ defined on (finite) subsets of $\mathbb{X}^d_\ell$ by $\widetilde{f}_y : \mathcal{P}(\mathbb{X}^d_\ell) \setminus \{\emptyset\} \to \mathbb{R}$ with $\widetilde{f}_y(C) = \min_{c \in C} \widetilde{\mathrm{dtw}_p}(y,c)$, transforming T into $\widetilde{F}_T = \{\widetilde{f}_\tau \mid \tau \in T\}$. To construct a coreset, one draws elements from T according to a fixed probability distribution over T, and reweighs each drawn element. Both the weight and sampling probability are expressed in terms of the *sensitivity* of the drawn element t, which describes the maximum possible relative contribution of t to the cost of any query evaluation. We bound the sensitivity of each $\widetilde{f}_\tau \in \widetilde{F}_T$ by a function $\gamma(\widetilde{f}_\tau)$, which solely depends on a (α,β)-approximation, m, ℓ and p. The sensitivity sampling framework and Theorem 3.2 then yield Theorem 3.3.

▶ **Theorem 3.3.** *For $\widetilde{f} \in \widetilde{F}$, let $\lambda(\widetilde{f}) = 2^{\lceil \log_2(\gamma(\widetilde{f}))\rceil}$, with $\gamma(\widetilde{f})$ the aforementioned sensitivity bound, associated to an (α,β)-approximation consisting of $\hat{k} \le \beta k$ curves, for (k,ℓ)-median for curves in $\mathbb{X}^d_m$ under dtw_p, $\Lambda = \sum_{\widetilde{f} \in \widetilde{F}} \lambda(\widetilde{f})$, $\psi(\widetilde{f}) = \frac{\lambda(\widetilde{f})}{\Lambda}$ and $\delta, \varepsilon \in (0,1)$. A sample S of*

$$\Theta\Big(\varepsilon^{-2}\alpha\hat{k}(m\ell)^{1/p}\Big((d\ell\log(\ell m\varepsilon^{-1}))k\log(k)\log(\alpha n)\log(\alpha\hat{k}(m\ell)^{1/p}) + \log(1/\delta)\Big)\Big)$$

elements $\tau_i \in T$, drawn independently with replacement with probability $\psi(\widetilde{f}_i)$ and weighted by $w(\widetilde{f}_i) = \frac{\Lambda}{|S|\lambda(\widetilde{f}_i)}$ is a weighted ε-coreset for (k,ℓ)-median clustering of T under dtw_p with probability at least $1-\delta$.

4 Linear time approximation algorithm for (k,ℓ)-median

As Theorem 3.3 requires an initial (α,β)-approximate solution of the (k,ℓ)-median problem to compute the bounds $\gamma(\cdot)$ of the sensitivities, we turn to developing approximation algorithms for (k,ℓ)-median for a set $T \subset \mathbb{X}^d_m$ of n curves. For this, we approximate dtw_p on T by a metric using a new inequality for dtw_p (Lemma 4.1). This allows the use of any approximation algorithm for metric k-median, leading to an initial approximation algorithm of the original problem. Combined with a k-median algorithm in metric spaces [15], we obtain a linear time $(O((m\ell)^{1/p}),1)$-approximation algorithm, which in turn allows us to compute a coreset in linear (in n) time.

Metrification of p-DTW We begin with the following more general triangle inequality for dtw_p, which motivates analysing the metric closure of the input set. While dtw_p does not

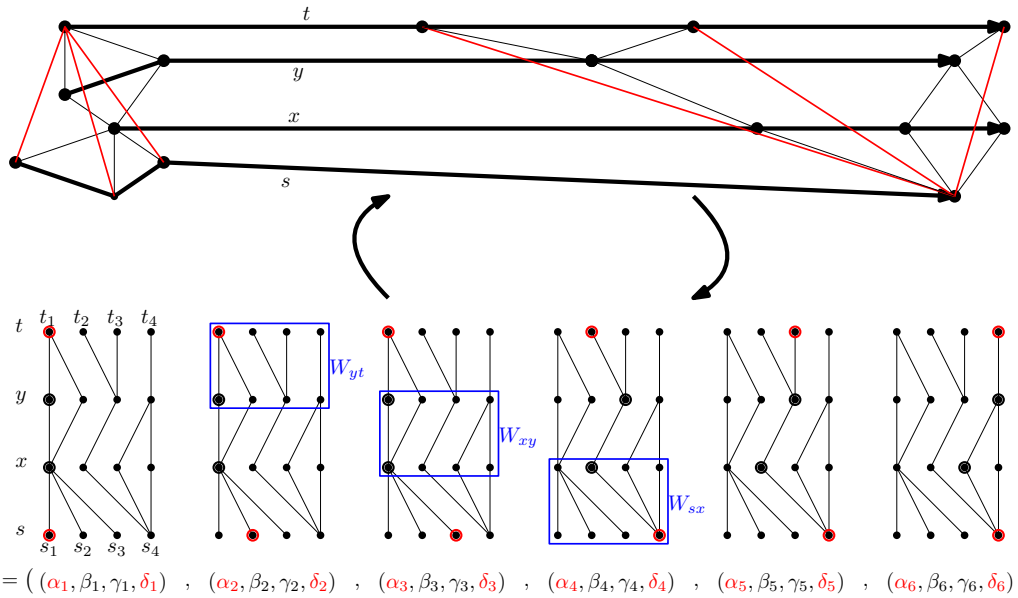


Figure 3 Illustration of how the optimal traversals W_{sx}, W_{xy} and W_{yt} of visited curves can be 'composed' to yield a set W that induces a traversal $\widetilde{W}$ (in red) of s and t. Any single matched pair of vertices in W_{sx}, W_{xy} or W_{yt} is at most $|W| \leq \ell + \ell'$ times a part of W.

satisfy the triangle inequality, the inequality shows it is never 'too far off'. Remarkably, the inequality does not depend on the complexity of the curves 'visited'. Missing proofs of this section can be found in the full version [11]. Figure 3 illustrates Lemma 4.1.

▶ **Lemma 4.1** (Iterated triangle inequality). *Let $s \in \mathbb{X}^d_\ell$, $t \in \mathbb{X}^d_{\ell'}$ and $X = (x_1, \ldots, x_r)$ be an arbitrary ordered set of curves in $\mathbb{X}^d_m$. Then*

$$\mathrm{dtw}_p(s,t) \leq (\ell+\ell')^{1/p} \left(\mathrm{dtw}_p(s,x_1) + \sum_{i<r} \mathrm{dtw}_p(x_i,x_{i+1}) + \mathrm{dtw}_p(x_r,t) \right).$$

▶ **Definition 4.2** (metric closure). Let (X,ϕ) be a finite set endowed with a distance function $\phi : X \times X \to \mathbb{R}$. The metric closure $\overline{\phi}$ of ϕ is the function

$$\overline{\phi} : X \times X \to \mathbb{R}, (s,t) \mapsto \min_{\substack{r \geq 2, \{\tau_1,\ldots,\tau_r\} \subset X \\ s=\tau_1, t=\tau_r}} \sum_{i<r} \phi(\tau_i, \tau_{i+1}).$$

▶ **Lemma 4.3.** *For any set of curves X and two curves $\sigma, \tau \in X$ of complexity at most m it holds that the metric closure $\overline{\mathrm{dtw}_p|_X}$ of the restriction of dtw_p onto the set X is bounded by $\mathrm{dtw}_p(\sigma,\tau) \leq (2m)^{1/p}\overline{\mathrm{dtw}_p|_X}(\sigma,\tau) \leq (2m)^{1/p}\,\mathrm{dtw}_p(\sigma,\tau)$.*

Linear time algorithm Naïvely, we would like to apply linear time algorithms [9] to the metric closure of dtw_p. However, constructing the metric closure usually takes cubic time resulting in cubic time algorithms. We circumvent this by applying Indyk's sampling technique for bicriteria k-median approximation [15], which reduces a k-median instance with n points to two k-median instances with $O(\sqrt{n})$ points, simply by sampling. We apply this technique twice, so that we only compute the metric closure on four sampled subsets of size $O(n^{1/4})$, resulting in the following theorem.

▶ **Theorem 4.4.** *For any $\varepsilon > 0$ there is an algorithm which computes a $(O(1+\varepsilon)(m\ell^3)^{(1/p)}, 4)$-approximation for (k,ℓ)-median for an input set X of n curves of complexity m under dtw_p in time*

$$O(nm^3d + nk\log(k)\ell^2 d + nk^2\log^2(k)\varepsilon^{-4}\log^2(\varepsilon^{-1}) + k^7\varepsilon^{-5}\log^5(n)).$$

5 Coreset Application

The theoretical derivations of the previous sections culminate in an approximation algorithm (Theorem 5.1) to (k,ℓ)-median that is particularly useful in the big data setting, where $n \gg m$. Our strategy is to first compute an efficient but not very accurate approximation(Theorem 4.4) of (k,ℓ)-median, which we use to construct a coreset. By virtue of the size reduction we greatly reduce the running time of slower more accurate algorithms, yielding a better approximation. Missing proofs can be found in the full version [11].

Algorithm 1 $((32+\varepsilon)(4m\ell)^{1/p})$-approximate (k,ℓ)-median

procedure (k,ℓ)-MEDIAN$(X \subset \mathbb{X}_m^d, p, \varepsilon)$
  $\varepsilon' \leftarrow \varepsilon/46$
  Compute $(O((16m\ell^3)^{1/p}), 4)$-approximation C' (Theorem 4.4)
  Compute bound $\gamma(\widetilde{f}_x)$ of sensitivity for each curve $x \in X$ from C'
  Compute sample size $s \leftarrow O(\varepsilon^{-2}d\ell k^2(m^2\ell^4)^{1/p}\log^3(m\ell)\log^2(k)\log(\varepsilon^{-1})\log(n))$
  Sample and weigh ε'-coreset S of X of size s (Theorem 3.3)
  Compute a 2-simplification for every curve in S resulting in the set S^* of curves
  Compute metric closure $\overline{\phi}(x,y) = \overline{\mathrm{dtw}_p\,|_{S^*}}(x,y)$ for every $x, y \in S^*$
  Return $(5+\varepsilon', 1)$-approximation of weighted metric k-median in $(S^*, \overline{\phi})$ (c.f. [3, 9])
end procedure

▶ **Theorem 5.1.** *Let $0 < \varepsilon \leq 1$. There is an $((32+\varepsilon)(4m\ell)^{1/p}, 1)$-approximate algorithm with constant success probability ((k,ℓ)-MEDIAN in Algorithm 1) for (k,ℓ)-median on curves under dtw_p with a running time of $\widetilde{O}\Big(n(m^3d + k^2 + k\ell^2 d) + \varepsilon^{-6}d^3\ell^3k^7\sqrt[p]{m^6\ell^{12}}\Big)$, where $\widetilde{O}$ hides polylogarithmic factors in n, m, ℓ, k and ε^{-1}.*

▶ **Corollary 5.2.** *There is an algorithm that computes an ε-coreset for (k,ℓ)-median in time $\widetilde{O}\Big(n(m^3d + k^2 + k\ell^2 d) + \varepsilon^{-6}d^3\ell^3k^7\sqrt[p]{m^6\ell^{12}}\Big)$ with constant success probability of size*

$$O(\varepsilon^{-2}d\ell k^2(m^2\ell^2)^{1/p}\log^3(m\ell)\log^2(k)\log(\varepsilon^{-1})\log(n)).$$

Acknowledgments. Jacobus Conradi was supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) - 313421352 (FOR 2535 Anticipating Human Behavior) and the iBehave Network: Sponsored by the Ministry of Culture and Science of the State of North Rhine-Westphalia. Benedikt Kolbe was partially supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – 459420781. Ioannis Psarros has been partially supported by project MIS 5154714 of the National Recovery and Resilience Plan Greece 2.0 funded by the European Union under the NextGenerationEU Program. Dennis Rohde was partially supported by the Hausdorff Center for Mathematics and the iBehave Network: Sponsored by the Ministry of Culture and Science of the State of North Rhine-Westphalia.

References

1 Marcel R. Ackermann, Johannes Blömer, and Christian Sohler. Clustering for metric and nonmetric distance measures. *ACM Transactions on Algorithms*, 6(4):59:1–59:26, 2010.

2 Martin Anthony and Peter L. Bartlett. *Neural Network Learning: Theoretical Foundations*. Cambridge University Press, 1999. doi:10.1017/CBO9780511624216.

3 Vijay Arya, Naveen Garg, Rohit Khandekar, Adam Meyerson, Kamesh Munagala, and Vinayaka Pandit. Local Search Heuristics for k-Median and Facility Location Problems. *SIAM Journal on Computing*, 33(3):544–562, 2004.

4 Olivier Bachem, Mario Lucic, and Andreas Krause. Scalable k-Means Clustering via Lightweight Coresets. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, (KDD)*, pages 1119–1127, 2018.

5 Vladimir Braverman, Vincent Cohen-Addad, Shaofeng H.-C. Jiang, Robert Krauthgamer, Chris Schwiegelshohn, Mads Bech Toftrup, and Xuan Wu. The power of uniform sampling for coresets. In *63rd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2022, Denver, CO, USA, October 31 - November 3, 2022*, pages 462–473. IEEE, 2022. doi:10.1109/FOCS54457.2022.00051.

6 Maike Buchin, Anne Driemel, and Dennis Rohde. Approximating (k, ℓ)-Median Clustering for Polygonal Curves. In Dániel Marx, editor, *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 2697–2717, Virtual Conference, January 2021. SIAM.

7 Maike Buchin, Anne Driemel, and Dennis Rohde. Approximating (k, ℓ)-median clustering for polygonal curves. *ACM Trans. Algorithms*, 19(1):4:1–4:32, 2023.

8 Maike Buchin, Anne Driemel, Koen van Greevenbroek, Ioannis Psarros, and Dennis Rohde. Approximating length-restricted means under dynamic time warping. In Parinya Chalermsook and Bundit Laekhanukit, editors, *Approximation and Online Algorithms - 20th International Workshop, WAOA 2022, Potsdam, Germany, September 8-9, 2022, Proceedings*, volume 13538 of *Lecture Notes in Computer Science*, pages 225–253. Springer, 2022.

9 Ke Chen. On Coresets for k-Median and k-Means Clustering in Metric and Euclidean Spaces and Their Applications. *SIAM Journal on Computing*, 39(3):923–947, 2009.

10 Siu-Wing Cheng and Haoqiang Huang. Curve simplification and clustering under fréchet distance. In Nikhil Bansal and Viswanath Nagarajan, editors, *Proceedings of the 2023 ACM-SIAM Symposium on Discrete Algorithms, SODA 2023, Florence, Italy, January 22-25, 2023*, pages 1414–1432. SIAM, 2023.

11 Jacobus Conradi, Benedikt Kolbe, Ioannis Psarros, and Dennis Rohde. Fast approximations and coresets for (k, l)-median under dynamic time warping. *CoRR*, abs/2312.09838, 2023. URL: https://doi.org/10.48550/arXiv.2312.09838, arXiv:2312.09838, doi:10.48550/ARXIV.2312.09838.

12 Anne Driemel, Amer Krivosija, and Christian Sohler. Clustering time series under the Fréchet distance. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 766–785, 2016.

13 Dan Feldman and Michael Langberg. A unified framework for approximating and clustering data. In Lance Fortnow and Salil P. Vadhan, editors, *Proceedings of the 43rd ACM Symposium on Theory of Computing*, pages 569–578. ACM, 2011.

14 Lingxiao Huang, Shaofeng H.-C. Jiang, Jian Li, and Xuan Wu. Epsilon-coresets for clustering (with outliers) in doubling metrics. In *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018*, pages 814–825, 2018.

15 Piotr Indyk. Sublinear time algorithms for metric space problems. In Jeffrey Scott Vitter, Lawrence L. Larmore, and Frank Thomson Leighton, editors, *Proceedings of the Thirty-First Annual ACM Symposium on Theory of Computing, May 1-4, 1999, Atlanta, Georgia, USA*, pages 428–434. ACM, 1999. doi:10.1145/301250.301366.

16 Youngseon Jeong, Myong Kee Jeong, and Olufemi A. Omitaomu. Weighted dynamic time warping for time series classification. *Pattern Recognit.*, 44(9):2231–2240, 2011.

17 Rohit J. Kate. Using dynamic time warping distances as features for improved time series classification. *Data Min. Knowl. Discov.*, 30(2):283–312, 2016.

18 Amit Kumar, Yogish Sabharwal, and Sandeep Sen. A Simple Linear Time $(1 + \varepsilon)$-Approximation Algorithm for k-Means Clustering in Any Dimensions. In *45th Symposium on Foundations of Computer Science (FOCS), 17-19 October, Rome, Italy, Proceedings*, pages 454–462. IEEE Computer Society, 2004.

19 François Petitjean, Germain Forestier, Geoffrey I. Webb, Ann E. Nicholson, Yanping Chen, and Eamonn J. Keogh. Dynamic time warping averaging of time series allows faster and more accurate classification. In Ravi Kumar, Hannu Toivonen, Jian Pei, Joshua Zhexue Huang, and Xindong Wu, editors, *2014 IEEE International Conference on Data Mining, ICDM 2014, Shenzhen, China, December 14-17, 2014*, pages 470–479. IEEE Computer Society, 2014.

20 François Petitjean, Germain Forestier, Geoffrey I. Webb, Ann E. Nicholson, Yanping Chen, and Eamonn J. Keogh. Faster and more accurate classification of time series by exploiting a novel dynamic time warping averaging algorithm. *Knowl. Inf. Syst.*, 47(1):1–26, 2016.

21 Thanawin Rakthanmanon, Bilson J. L. Campana, Abdullah Mueen, Gustavo E. A. P. A. Batista, M. Brandon Westover, Qiang Zhu, Jesin Zakaria, and Eamonn J. Keogh. Addressing big data time series: Mining trillions of time series subsequences under dynamic time warping. *ACM Trans. Knowl. Discov. Data*, 7(3):10:1–10:31, 2013.

22 Norbert Sauer. On the density of families of sets. *Journal of Combinatorial Theory Series A*, 13:145–147, 1972.

23 Saharon Shelah. A combinatorial problem; stability and order for models and theories in infinitary languages. *Pacific Journal of Mathematics*, 41(1), 1972.

24 Tuan Minh Tran, Xuan-May Thi Le, Hien T. Nguyen, and Van-Nam Huynh. A novel non-parametric method for time series classification based on k-nearest neighbors and dynamic time warping barycenter averaging. *Eng. Appl. Artif. Intell.*, 78:173–185, 2019.

25 Vladimir Vapnik and Alexey Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications*, 16:264–280, 1971.

On Maximal 3-Planar Graphs

Michael Hoffmann, Meghana M. Reddy, and Shengzhe Wang

Department of Computer Science, ETH Zürich, Switzerland
{hoffmann, meghana.mreddy}@inf.ethz.ch, wangshe@ethz.ch

Abstract

A graph is 3-*planar* if it admits a drawing in the plane such that every edge is crossed at most three times. A 3-planar graph is *maximal* 3-planar if addition of any edge results in a graph that is not 3-planar. A 3-planar graph on n vertices has at most $5.5n - 11$ edges, and a 3-planar graph that has exactly $5.5n - 11$ edges is an *optimal* 3-planar graph. In contrast to planar graphs where maximal and optimal graphs coincide, a maximal 3-planar graph may have fewer edges than an optimal 3-planar graph. In this paper, we study properties of maximal 3-planar graphs. First, we characterize the graphs on nine vertices that are (maximal) 3-planar. Second, we show that—in contrast to maximal 1– and 2-planar graphs—maximal 3-planar graphs may contain cut vertices. Third, we give a first upper bound on the minimal edge density by constructing maximal 3-planar graphs on n vertices with only $2.375n + O(1)$ edges.

1 Introduction

Planar graphs are graphs that can be drawn on the plane without crossings. If the addition of any edge to a planar graph makes it impossible for the resulting graph to admit a plane drawing, the planar graph is said to be a *maximal* planar graph. Maximal planar graphs are well-studied and have many interesting properties. For example, the number of edges in maximal planar graphs is solely dependent on the number of vertices. Specifically, every maximal planar graph on $n \geq 3$ vertices has $3n - 6$ edges. It is natural to further explore the edge density for various families of beyond-planar graphs, which have been extensively studied over the past decade [5, 8].

In this paper, we focus on 3-planar graphs. A graph is k-planar if it admits a drawing where each edge has at most k crossings. Pach and Tóth [11] gave upper bounds on the number of edges in a k-planar graph, which they used to improve the Crossing Lemma. Unlike planar graphs, the class of k-planar graphs is not closed under edge contractions. Thus, many useful properties of minor closed classes do not apply to k-planar graphs.

As k gets larger, the density of a k-planar graph on n vertices clearly increases, but the exact correlation is still unknown. Pach and Tóth [11] showed a general upper bound of $4.108\sqrt{k}n$ edges for k-planar graphs. For small values of k, i.e., $k = 1$ and $k = 2$, they also gave tight upper bounds. The class of 1-planar graphs was introduced by Ringel [12] in the context of planar graph colorings. A 1-planar graph has at most $4n - 8$ edges, and there are infinitely many *optimal* 1-planar graphs that achieve the bound [13]. A 2-planar graph on n vertices has at most $5n - 10$ edges, and there are infinitely many *optimal* 2-planar graphs that achieve this bound [11].

A lot is left to be explored for *maximal* k-planar graphs, where a graph is *maximal* k-*planar* if there is no edge that can be added such that the resulting graph is still k-planar. In contrast to planar graphs, maximal k-planar graphs are not necessarily optimal k-planar graphs. Indeed the gap between maximal and optimal can be very large for k-planar graphs. Hudák et al. [9] showed an infinite family of maximal 1-planar graphs with $\frac{8}{3}n+O(1) \approx 2.667n$ edges, and a sparser construction by Brandenburg et al. [4] only has $\frac{45}{17}n + O(1) \approx 2.647n$ edges. Brandenburg et al. [4] also proved that every maximal 1-planar graph has at least

40th European Workshop on Computational Geometry, Ioannina, Greece, March 13–15, 2024.
This is an extended abstract of a presentation given at EuroCG'24. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

$\frac{28}{13}n - O(1) \approx 2.153n$ edges. This lower bound was further improved by Barát and Tóth [3] to $\frac{20}{9}n \approx 2.22n$.

Interestingly, the density even decreases for maximal k-planar graphs when k increases from 1 to 2. Hoffmann and M. Reddy [6] showed that every maximal 2-planar graph on $n \geq 5$ vertices has at least $2n$ edges, and they also described an infinite family of maximal 2-planar graphs on n vertices with only $2n + O(1)$ edges. So, even though they allow more crossings, some maximal 2-planar graphs have a lower edge density than any maximal 1-planar graph.

Results. First, in Section 3 we characterize the graphs on nine vertices that are (maximal) 3-planar. Next, in Section 4 we exhibit maximal 3-planar graphs that contain vertices for which all incident edges are crossed, in every simple 3-plane drawing of the graph. As a consequence, maximal 3-planar graphs are not necessarily 2-connected and may contain vertices of degree one. In contrast, all maximal 1-and 2-planar graphs are 2-connected. Finally, in Section 5, we construct maximal 3-planar graphs on n vertices with only $2.375n + O(1)$ edges.

2 Preliminaries

A drawing is *simple* if any pair of edges has at most one common point, including endpoints. To analyze k-plane drawings of a graph, one typical restriction is to consider a drawing that minimizes the total number of crossings among all k-plane drawings of the graph, which is called a *crossing-minimal* k-plane drawing. The benefit of such a restriction is that for $k \leq 3$, a crossing-minimal k-plane drawing is always simple [10]. Consequently, 3-planar graphs always admit a simple 3-plane drawing.

▶ **Lemma 1.** *If a 3-planar graph G is not maximal 3-planar, then there exists a simple 3-plane drawing of G that is not maximal 3-plane.*

Proof. If G is not maximal 3-planar, then there exists a pair u, v of nonadjacent vertices such that $G' = G \cup e$ is 3-planar where $e = (u, v)$. Take any simple 3-plane drawing of G' and remove e to obtain a simple 3-plane drawing of G that is not maximal. ◀

In all figures of this paper, the edges are colored to indicate their number of crossings. Uncrossed edges are shown green, singly crossed edges are shown purple, doubly crossed edges are shown orange, and triply crossed edges are shown blue. Edges with an undetermined number of crossings are shown black.

3 Characterization of (Maximal) 3-planar Graphs on 9 Vertices

Angelini et al. [2] showed that K_8 is 3-planar, while K_9 is 4-planar (but not 3-planar). This motivated us to study the set of 3-planar graphs on nine vertices. With the help of a computer program, we can enumerate all possible simple 3-plane drawings of a given graph; see Section 6.

The basic idea is to check all graph structures on nine vertices in decreasing order based on the number of edges. Specifically, starting from $K_9 \setminus K_2$, which is generated by removing a single edge from a K_9, we check if the given graph admits a 3-plane drawing. Further, if we want to remove two edges from a clique, we can either remove two independent edges or remove a path P_3 of length two. If we restrict to a graph with nine vertices, that is equivalent to saying that any graph with nine vertices and thirty-four edges will be isomorphic to either $K_9 \setminus (K_2 + K_2)$ or $K_9 \setminus P_3$. We have a similar argument if we remove three edges.

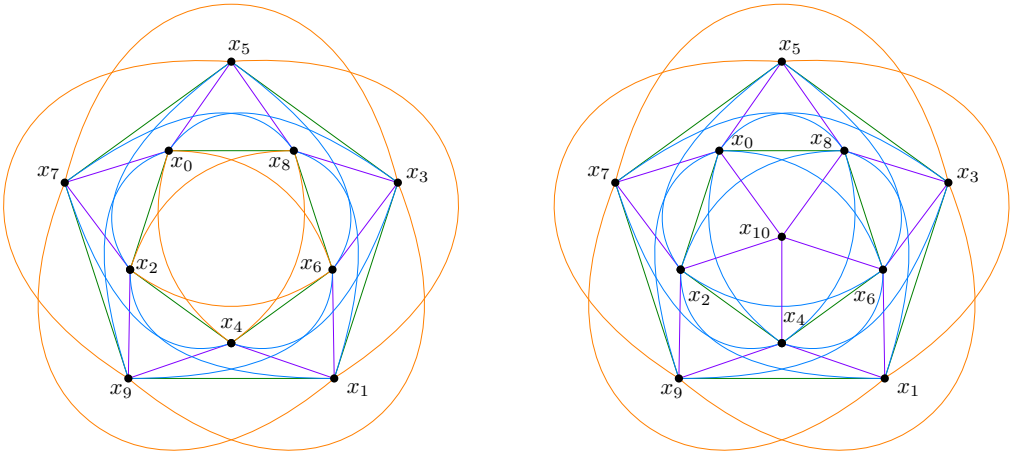


Figure 1 (Left) Drawing D_1 of graph G_1 by removing five independent edges from K_{10}. (Right) Drawing D_1' of graph G_1' by inserting a new vertex x_{10} to G_1 and adding five edges incident to x_{10}.

Using our computer program, we were able to verify that all graphs on nine vertices that have at least 34 edges do not admit a simple 3-plane drawing, while all the remaining graphs on nine vertices are 3-planar. The result can be verified with the code in our repository [7].

▶ **Theorem 2.** *A graph on nine vertices is* 3*-planar if and only if it has at most* 33 *edges, and it is maximal* 3*-planar if and only if it has exactly* 33 *edges.*

We therefore notice that maximal 3-planar graphs and maximum 3-planar graphs on n vertices coincide for $n \leq 9$.

4 Uncrossed Edge in Maximal 3-planar Graphs

Though we allow crossings in beyond-planar graphs, it does not necessarily mean that every edge will have a crossing. In a more general sense, crossings are not equally distributed over the edges. In maximal k-planar graphs where $k \leq 2$, it has been shown that every vertex must be incident to an uncrossed edge in every crossing-minimal k-plane drawing [6]. But when $k = 3$, the situation is different and we have the following

▶ **Lemma 3.** *There exist infinitely many maximal* 3*-planar graphs that each contains a vertex v such that all edges incident to v are crossed in every simple* 3*-plane drawing of the graph.*

Proof. The proof of Lemma 3 is based on a graph G_1' that has the required properties. Then we can use G_1' to create larger graphs. To construct a graph G_1', we start from a base graph G_1 that is isomorphic to K_{10} minus five independent edges. Specifically, we take the vertex set as $\{x_0, x_1, ..., x_9\}$, and include all edges of the induced complete graph except the five edges $\{x_0, x_1\}$, $\{x_2, x_3\}$, $\{x_4, x_5\}$, $\{x_6, x_7\}$ and $\{x_8, x_9\}$. We enumerated all simple 3-plane drawings of G_1 using our program and found that this graph has exactly one simple 3-plane drawing up to automorphism. This drawing is illustrated in Fig. 1, and let this drawing be D_1. We can observe that it is impossible to add an edge to D_1 while maintaining 3-planarity, thus it proves G_1 is a maximal 3-planar graph from Lemma 1.

We can further obtain a new graph G_1' by adding a new vertex x_{10} to G_1 and connecting it to the five vertices x_0, x_2, x_4, x_6, x_8. We claim that the drawing D_1' illustrated in Fig. 1(right)

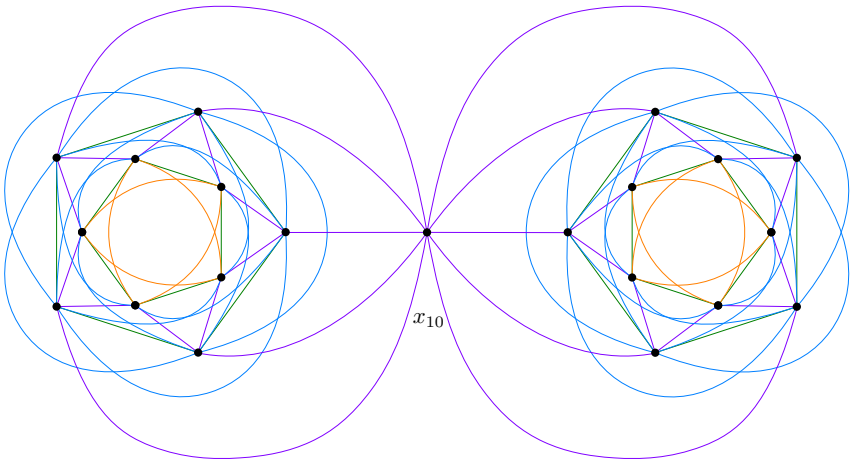


Figure 2 Graph G_2 by gluing two copies of G_1 on a merged vertex.

is the unique simple 3-plane drawing of G_1' (up to automorphisms). To see this, consider every face of D_1. It turns out that there exists exactly one face where the vertex x_{10} can be placed such that the five edges to vertices x_0, x_2, x_4, x_6, x_8 can be added while maintaining 3-planarity, and the resulting drawing is D_1'. This concludes that the graph G_1' is maximal 3-planar. We further note that x_{10} is not incident to any uncrossed edge in any 3-plane drawing of G_1'.

Further we can take two copies of the graph G_1' and merge the two vertices with degree five from each of the copies into a single vertex with degree ten to obtain a new graph G_2. It can be interpreted as gluing two copies of G_1' together at the vertex x_{10}. Refer to Fig. 2 for one possible 3-plane drawing of G_2.

We claim that G_2 is a maximal 3-planar graph. Consider an arbitrary simple 3-plane drawing of G_2. The subdrawings corresponding to the two copies of G_1 that are still simple drawings should be the same as D_1. It can also be viewed as adding a copy of D_1 into an existing drawing D_1. Again, considering all the faces of D_1 we can observe that the drawing shown in Fig. 2 is the only possible 3-plane drawing of G_2 up to automorphism.

This replication can be repeated infinitely many times to obtain larger graphs, and in each such graph the vertex x_{10} is not incident to any uncrossed edge in every simple 3-plane drawing, concluding the proof. ◀

Clearly, x_{10} is a cut vertex in G_2 and every graph obtained by following the replicating procedure. Thus, we have the following

▶ **Theorem 4.** *There exist infinitely many maximal* 3*-planar graphs that are not* 2*-connected.*

5 Number of Edges in Sparse Maximal 3-planar Graphs

We describe a construction of sparse maximal 3-planar graphs based on G_1 in this section.

▶ **Theorem 5.** *There exist an infinite family of maximal* 3*-planar graphs on* n *vertices with at most* $2.375n + O(1)$ *edges.*

Proof. We construct a nested graph with arbitrarily many layers where each layer is a variation of G_1 as shown in Fig. 3. Specifically, in each layer, we add a *hermit* vertex

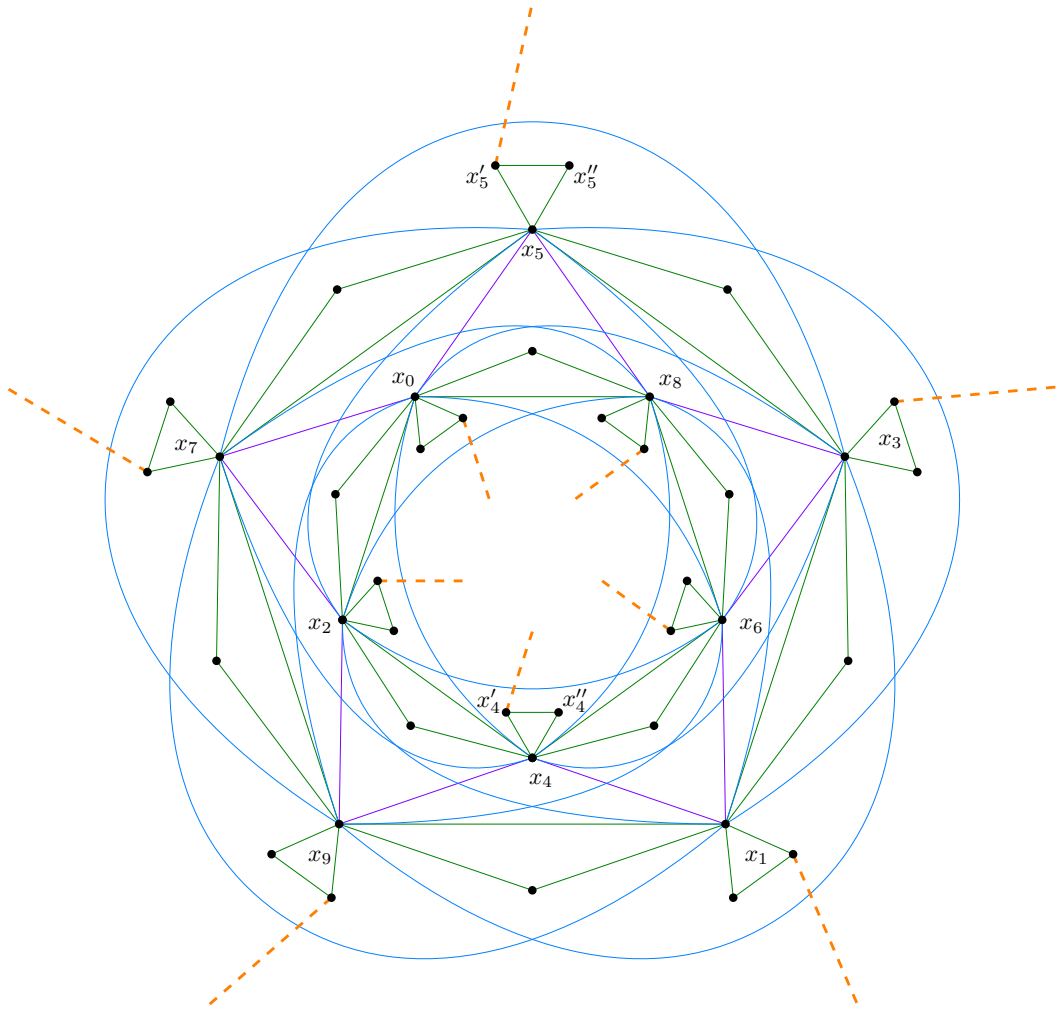


Figure 3 A single layer in the nested graph structure. Some labels are omitted for simplicity.

connecting to two endpoints of each planar edge, and a triangle to each original vertex, where a hermit is a degree-two vertex.

Further, as shown in Fig. 3, consecutive layers are connected with each other with orange dashed lines, which represent half edges. Specifically, suppose vertices from layer i are indexed as x_j^i for $0 \leq j \leq 9$, and corresponding triangle vertices are labelled as $x_j^i{}'$ and $x_j^i{}''$. To connect layers i and $i+1$, an edge is added between $x_j^i{}'$ and $x_{j+1}^{i+1}{}'$ for even j. To close the innermost and outermost layers, we simply use two 5-stars respectively to complete the graph.

We argue that the constructed graph still admits few simple 3-plane drawings. We start from each layer based on the unique simple drawing D_1. For those inserted triangles, since the drawing is symmetric considering the innermost face and the outermost face, it is enough to argue about the outer triangles. We note that outer vertices x_j^i of layer i for odd j should be reachable from a single face because all vertices from other layers have to be drawn in a single face of the drawing for layer i. Thus the only eligible face is the outermost face. Observe that $x_j^i{}''$ connects to x_j^i and $x_j^i{}'$, so the drawing of triangles has to be the same as shown in Fig. 3. We then note that each planar edge is enclosed by blue curves. Thus

every hermit vertex can only be drawn in faces on two sides of the planar edge. For the closing 5-star, the argument is similar as stated for graph G'_1. Then consider all possible simple 3-plane drawing of such a nested graph, it is impossible to add any edge in any of the drawing, and this concludes that the graph is maximal 3-planar.

For each layer, we have 40 vertices and 90 edges. And between two layers, there are 5 edges connecting them. We can charge these 5 edges to one layer, and extend to arbitrarily many layers. In total we will have $40k + 2$ vertices and $95k + 5$ edges for a graph with k layers. And it gives an edge density of $\frac{95k+5}{40k+2} \cdot n = 2.375n + O(1)$ where n is the number of vertices. This concludes the proof. ◀

It is possible that the construction for 2-planar graphs with roughly $2n$ edges [6] can be extended to the 3-planar case; see Fig. 4. But new ideas are needed to prove maximality.

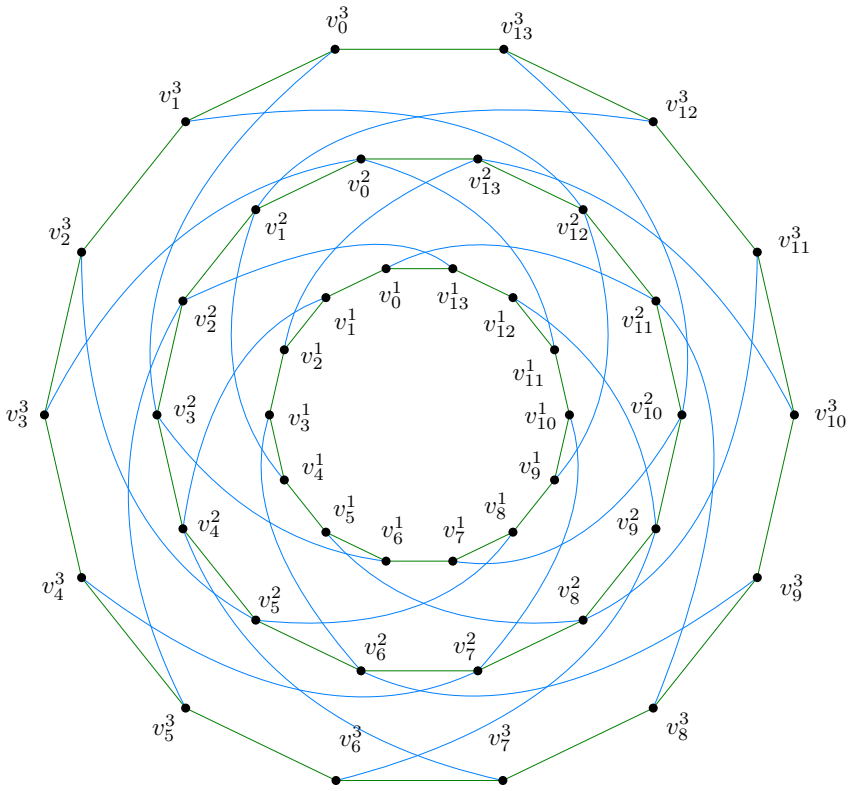


Figure 4 Nested C_{14} is a possible maximal 3-planar graph.

6 Enumeration of 3-plane drawings

In this section, we sketch the program we used to enumerate all possible simple 3-plane drawings of small graphs. This program was used to show certain properties of 3-planar graphs, like maximality. The basic idea is similar to previous work [1, 6]. We adapted the code for 2-planar graphs from [6] and extended it to 3-planar graphs. It is available in our repository [7].

To enumerate all simple k-plane drawings of a graph up to strong isomorphism (that is, up to a homeomorphism of the plane), we enumerate combinations of all possible drawings of each edge. We fix a labeling of the vertices and an ordering of the edges to then use

depth first search to explore all possible simple 3-plane drawings. The restriction to simple drawings is without loss of generality by Lemma 1.

We add edges one by one to the current drawing, and try to complete the given graph. Whenever we add an edge, we take every valid drawing of the edge into consideration. After we run out of different ways to draw the current edge, we backtrack and try to draw the previous edge differently. Whenever we successfully added all edges into the drawing, we found a simple 3-plane drawing of the given graph. We record the drawing and continue.

The time complexity of such a search is exponential in the number of edges, and thus it can be used for small graphs only. In practice, it takes roughly one hour to enumerate drawings for graphs on 9 vertices, and 30 hours for graphs on 10 vertices. For larger graphs it seems challenging to enumerate all simple 3-plane drawings within reasonable time limits.

References

1 Patrizio Angelini, Michael A. Bekos, Michael Kaufmann, and Thomas Schneck. Efficient generation of different topological representations of graphs beyond-planarity. In *Proc. Graph Drawing and Network Visualization (GD'19)*, pages 253–267, 2019.

2 Patrizio Angelini, Michael A. Bekos, Michael Kaufmann, and Thomas Schneck. Efficient generation of different topological representations of graphs beyond-planarity. *J. Graph Algorithms Appl.*, 24(4):573–601, 2020. URL: https://doi.org/10.7155/jgaa.00531, doi:10.7155/JGAA.00531.

3 János Barát and Géza Tóth. Improvements on the density of maximal 1-planar graphs, 2015. arXiv:1509.05548.

4 Franz J. Brandenburg, David Eppstein, Andreas Gleißner, Michael T. Goodrich, Kathrin Hanauer, and Josef Reislhuber. On the density of maximal 1-planar graphs. In *Proc. Graph Drawing (GD'13)*, pages 327–338, 2013. doi:10.1007/978-3-642-36763-2_29.

5 Walter Didimo, Giuseppe Liotta, and Fabrizio Montecchiani. A survey on graph drawing beyond planarity. *ACM Comput. Surv.*, 52(1), 2019. doi:10.1145/3301281.

6 Michael Hoffmann and Meghana M. Reddy. The number of edges in maximal 2-planar graphs. In *Proc. 39th Internat. Sympos. Comput. Geom. (SoCG'23)*, pages 39:1–39:15, 2023. URL: https://doi.org/10.4230/LIPIcs.SoCG.2023.39.

7 Michael Hoffmann, Meghana M. Reddy, and Shengzhe Wang. The number of edges in maximal 3-planar graphs—code repository. https://drive.google.com/drive/folders/1HUylhDtQh98f6TBmhYWL5lAY8YbAJ4NP?usp=sharing.

8 Seok-Hee Hong. Introduction. In *Beyond Planar Graphs: Communications of NII Shonan Meetings*, pages 1–9. Springer Singapore, 2020. doi:10.1007/978-981-15-6533-5_1.

9 Dávid Hudák, Tomáš Madaras, and Yusuke Suzuki. On properties of maximal 1-planar graphs. *Discussiones Mathematicae Graph Theory*, 32(4):737–747, 2012. URL: http://eudml.org/doc/271067.

10 Janos Pach, Rados Radoicic, Gabor Tardos, and Geza Toth. Improving the crossing lemma by finding more crossings in sparse graphs. *Discrete & Computational Geometry*, 36(4):527–552, 2006. doi:10.1007/s00454-006-1264-9.

11 János Pach and Géza Tóth. Graphs drawn with few crossings per edge. *Combinatorica*, 17(3):427–439, 1997. doi:10.1007/BF01215922.

12 Gerhard Ringel. Ein Sechsfarbenproblem auf der Kugel. *Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg*, 29(1):107–117, 1965. doi:10.1007/BF02996313.

13 R. Von Bodendiek, H. Schumacher, and K. Wagner. Bemerkungen zu einem Sechsfarbenproblem von G. Ringel. *Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg*, 53(1):41–52, 1983. doi:10.1007/BF02941309.

Star-Forest Decompositions of Certain Geometric Graphs

Todor Antić [*1], **Jelena Glišić** [*1], **and Milan Milivojčević** [1,2]

1 **Faculty of Mathematics and Physics, Department of Applied Mathematics, Charles University**
todor@kam.mff.cuni.cz, jglisic@matfyz.cz, milivojcevic@proton.me

2 **Faculty of Mathematics, Natural Sciences and Information Technologies, University of Primorska**

Abstract

We deal with the problem of decomposing a complete geometric graph into plane star-forests. In particular, we disprove a recent conjecture by Pach, Saghafian and Schnider by constructing an infinite family of complete geometric graphs on n vertices which can be decomposed into $\frac{2n}{3}$ plane star-forests. We also describe a method which can be potentially used to construct such infinite families of geometric graphs decomposable into cn plane star-forests given only a single such graph, for any given $c \in (\frac{1}{2}, 1)$.

1 Introduction

A classic question asked in graph theory is the following: "Given a graph G, what is the minimal number of subgraphs with property P that G can be partitioned into?" Historically, this question was asked for abstract graphs and property P was replaced with forests, trees, complete bipartite graphs and many more [2, 7, 10]. Similar questions can be asked about graphs drawn in the plane or on any other surface. Here we want to decompose our complete graph into plane/k-planar/k-quasiplanar subgraphs with a given property. Answering such questions is a similar, but separate research direction that has been pursued by many authors in discrete geometry and graph drawing communities.

A *geometric* graph is a graph drawn in the plane, with vertices represented by points in general position and edges as straight line segments between them.

Recently, there has been a lot of work done on decomposing geometric graphs into planar subgraphs of a special kind, such as trees, stars, double stars etc. [11, 6]. This paper will be concerned with plane star-forests. A *star* is a connected graph on k vertices with one vertex of degree $k-1$, which we call the center of the star, and $k-1$ vertices of degree one. This definition allows for a single edge to be a star, in this case we decide arbitrarily which of its endpoints is the center. A *star-forest* is a forest whose every connected component is a star. A star-forest is plane if it is drawn in the plane without crossings. It is easy to observe that a complete graph K_n can be decomposed into $n-1$ stars. A fact that is not obvious is that K_n cannot be decomposed into less than $n-1$ stars [3]. In the same paper, Akiyama and Kano proved that K_n can be decomposed into $\lceil \frac{n}{2} \rceil + 1$ star-forests.

The story is different for complete geometric graphs. Recently, Pach, Saghafian and Schnider [8] showed that a complete geometric graph whose vertices form a convex polygon

* The research was conducted during a scholarship provided by Višegrad Fund.

40th European Workshop on Computational Geometry, Ioannina, Greece, March 13–15, 2024.
This is an extended abstract of a presentation given at EuroCG'24. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

cannot be decomposed into fewer than $n-1$ plane star-forests. In the same paper, the authors posed the following question:

▶ Question 1.1. What is the minimal number of plane star-forests that a complete geometric graph can be decomposed into?

Based on their findings they made the following conjecture:

▶ **Conjecture 1.2** ([8]). *Let $n \geq 1$. There is no complete geometric graph with n vertices that can be decomposed into fewer than $\lceil 3n/4 \rceil$ plane star-forests.*

The authors give a special configuration of $n = 4k$ points and construct a simple decomposition into $3n/4$ star-forests. Motivated by this example, we find configurations of n points that define a complete geometric graph which can be decomposed into $\lceil 2n/3 \rceil$ plane star-forests, disproving the conjecture.

Note on new results After submission of the paper to EuroCG we managed to obtain some better results. Among other things, we answered Conejcture 4.1 positively, thus proving that the bound $\lceil \frac{n}{2} \rceil + 1$ is indeed tight. The current version of the paper is available on arXiv [5].

2 The Construction

Firstly, we will give the most general possible construction and then present the concrete counterexample. We will write GP for a complete geometric graph whose underlying pointset is $P \subset \mathbb{R}^2$.

▶ **Theorem 2.1.** *Let $c \in (1/2, 1)$ be a constant. If there is a complete geometric graph on n_0 points which can be partitioned into cn_0 star-forests, in such a way that each vertex is a center of at least one star, then for each integer $k \geq 1$, there exists a complete geometric graph on kn_0 points that can be partitioned into ckn_0 star-forests.*

Proof. Let S be the underlying point set of the original complete geometric graph and let $k > 1$ be an integer. Label the points in S by $a_1, \ldots, a_{n_0}$. Now, replace each a_i by a set $A_i = \{a_i^1, ..., a_i^k\}$ of k points in general position in such a way that if we choose $b_1, \ldots, b_{n_0}$ where $b_i \in A_i$, we obtain a point set of the same order type as S. Call the new point set S^k. Now if $F_1, \ldots, F_{cn_0}$ is the decomposition of GS into star-forests, from this, we will obtain the decomposition of GS^k into $c(kn_0)$ star-forests. Let a_j be the center of a star in F_i. We will construct k new stars with centers in $a_j^1, \ldots, a_j^k$. Start with a_j^1, add to it all of the edges of the form $\{a_j^1, a_j^l\}$ that were not already used (in the case of a_j^1, none were used). Now for each edge of the form $\{a_j, a_{j'}\}$ in F_i, add all of the edges from a_j^1 to the vertices in $A_{j'}$. Continue doing this for each vertex a_j^l, where $l \in \{1, 2, \ldots, k\}$. We do this for each star in F_i and for each forest in the original decomposition. The result of this process is cn_0 families of star-forests, each of size k. And the planarity of the star-forests follows from the definition of the point set S^k. To see this, assume that a tree in the new decomposition has an intersection. Then the intersection is between edges whose 4 vertices are in different A_i's. But if this was the case, then a choice of transversal that includes this 4 vertices would induce a crossing inside the original decomposition of GS. ◀

We note that the assumption that each point is a center of at least one forest is crucial as otherwise the star-forests constructed in the proof do not cover all of the edges. For example see Figure 1. The vertex v is not a center of any star and thus none of the edges between vertices in A_v are covered by the star-forests on the right.

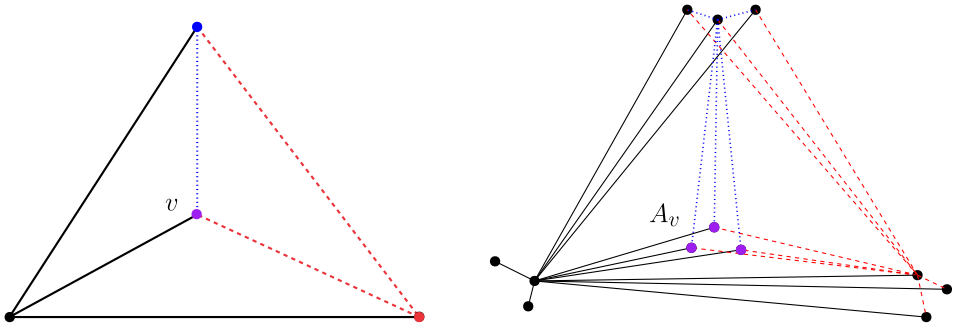


Figure 1 A complete geometric graph on 4 vertices decomposed into three star-forests and the corresponding graph on 12 vertices with the wrong "decomposition" into 9 star-forests. (only 4 are drawn for readability).

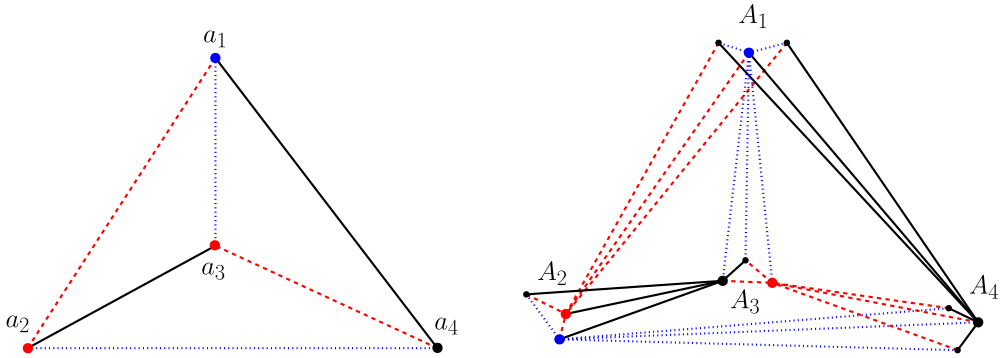


Figure 2 A complete geometric graph on 4 vertices decomposed into three plane star-forests and the corresponding graph on 12 vertices with the decomposition into 9 star-forests (only 4 are drawn for readability). Each vertex of the point set has been used as a center of some star and colored accordingly.

While Theorem 2.1 gives us a nice way of constructing infinitely many complete geometric graphs that can be partitioned into few plane star-forests, we still need concrete small examples to be able to produce the infinitudes. One example was given by the authors in [8] and can be found in Figure 2. This example motivated Conjecture 1.2. We proceed in a similar fashion.

▶ **Lemma 2.2.** *There exists a configuration of* 6 *points in the plane which can be partitioned into* 4 *plane star-forests in such a way that each point is a center of at least one star.*

Proof. We consider a configuration of 6 points which is crossing-minimal according to [9]. We decompose the graph into 4 star-forests as in the Figure 3. The graph has thus been decomposed into three 2-component star-forests colored in blue, red and black and one 3-component forest colored in purple. ◀

Now, using the pointset on $n_0 = 6$ elements from the above lemma, which can be decomposed into $2n_0/3 = 4$ star-forests, we obtain as an easy corollary a family of pointsets on $n = 6k$ points which can be decomposed into $2n/3$ star-forests, thus disproving Conjecture 1.2. We state this formally below.

▶ **Corollary 2.3.** *For every $k \in \mathbf{N}$, there exists a geometric graph on $n = 6k$ vertices which can be decomposed into $2n/3$ plane star-forests.*

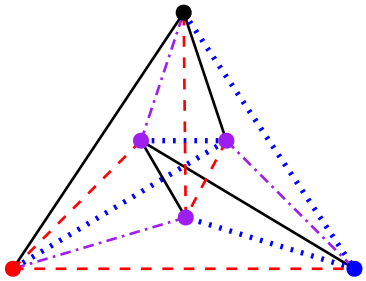


Figure 3 A complete geometric graph on 6 vertices decomposed into four star-forests, vertices are colored same as trees whose centers they are.

3 Computing Plane Star-Forest Decompositions on Pointsets with 6 Points

Using a simple computer search, we managed to find all pointsets on 6 points that can be decomposed into 4 plane star-forests. Out of the 16 order types which can be found on [1], we have found decompositions which satisfy the requirements from Theorem 2.1 for 6 of them. Those pointsets and corresponding partitions can be seen in Figure 4. The code is available at [4]. We plan to continue improving the code to be able to perform the search on bigger pointsets. Currently, the generation of appropriate decompositions is very slow, and since Stirling numbers grow very fast, we are not able to do the checks for bigger pointsets.

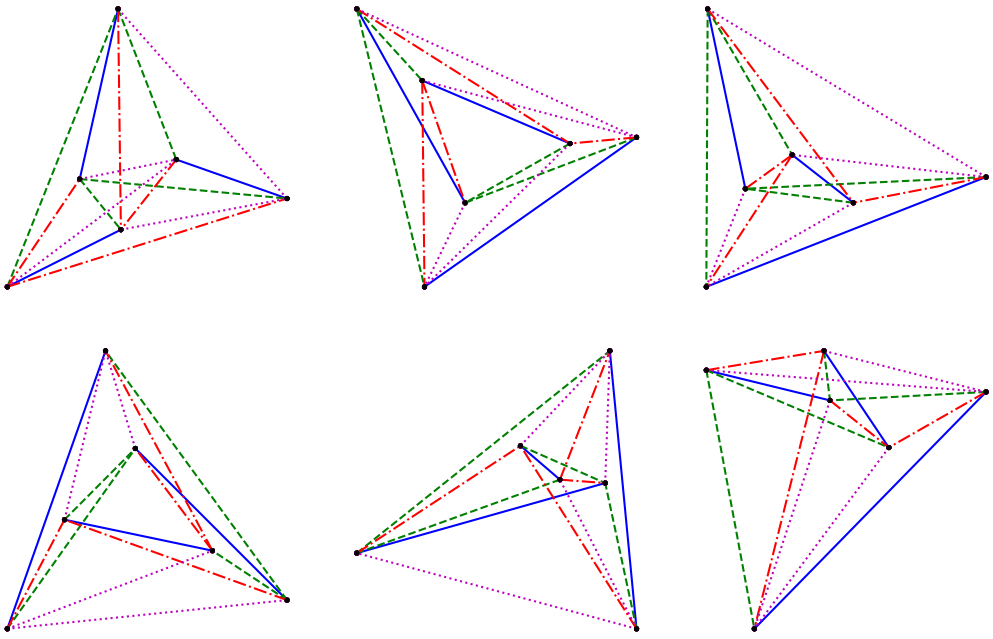


Figure 4 Star-forest decompostions of the pointsets that admit them.

4 Further Research and Open Questions

It is still unclear to us whether the number $2n/3$ is optimal, and we would be very surprised if it is. Thus we make the following conjecture:

▶ **Conjecture 4.1.** *For each $c \in (1/2, 1)$, there exists an $n \in \mathbb{N}$ and a complete geometric graph on n vertices which can be decomposed into $\lceil cn \rceil$ plane star-forests.*

If our conjecture is true, that would mean that the bound of $\lfloor n/2 \rfloor + 1$ is almost tight.

We also note that there is an interesting variation of this problem that we have not explored yet but where our approach can also be used. We define a *k-star-forest* to be a star-forest with at most k components. Authors in [8] proposed the following conjecture:

▶ **Conjecture 4.2.** *The number of plane k-star-forests needed to decompose a complete geometric graph is at least $\frac{(k+1)n}{2k}$.*

Our example does not show anything regarding Conjecture 4.2. But, it is not hard to see that the construction from Theorem 2.1 preserves the maximal number of components among all forests. Thus, we believe a similar approach could be used to attack this conjecture.

Acknowledgments This work is supported by project 23-04949X of the Czech Science Foundation (GAČR). We thank Pavel Valtr and Jan Kynčl who proposed the problem to us and gave a lot of useful comments and suggestions during the combinatorial problems seminar at Charles University.

References

1 Oswin Aichholzer. Enumerating order types for small point sets with applications. http://www.ist.tugraz.at/staff/aichholzer/research/rp/triangulations/ordertypes/.

2 Saieed Akbari, Tommy R. Jensen, and Mark H. Siggers. Decompositions of graphs into trees, forests, and regular subgraphs. *Discret. Math.*, 338(8):1322–1327, 2015.

3 Jin Akiyama and Mikio Kano. Path factors of a graph. In *Graphs and applications (Boulder, Colo., 1982)*, Wiley-Intersci. Publ., pages 1–21. Wiley, New York, 1985.

4 Todor Antić, Jelena Glišić, and Milan Milivojčević. https://github.com/milivojcevic6/Star-Forest-Decompositions.

5 Todor Antić, Jelena Glišić, and Milan Milivojčević. Star-forest decompositions of complete graphs. 2024. URL: `https://arxiv.org/abs/2402.11044`.

6 Prosenjit Bose, Ferran Hurtado, Eduardo Rivera-Campo, and David R. Wood. Partitions of complete geometric graphs into plane trees. *Comput. Geom.*, 34(2):116–125, 2006. URL: `https://doi.org/10.1016/j.comgeo.2005.08.006`, `doi:10.1016/J.COMGEO.2005.08.006`.

7 Zbigniew Lonc. Decompositions of graphs into trees. *J. Graph Theory*, 13(4):393–403, 1989.

8 János Pach, Morteza Saghafian, and Patrick Schnider. Decomposition of geometric graphs into star-forests. In *GD (1)*, volume 14465 of *Lecture Notes in Computer Science*, pages 339–346. Springer, 2023.

9 Alexander Pilz and Emo Welzl. Order on order types. *Discret. Comput. Geom.*, 59(4):886–922, 2018. URL: `https://doi.org/10.1007/s00454-017-9912-9`, `doi:10.1007/S00454-017-9912-9`.

10 Tay-Woei Shyu. Decomposition of complete graphs into paths and stars. *Discret. Math.*, 310(15-16):2164–2169, 2010.

11 Hazim Michman Trao, Gek L. Chia, Niran Abbas Ali, and Adem Kilicman. On edge-partitioning of complete geometric graphs into plane trees. 2019. URL: https://arxiv.org/abs/1906.05598, doi:10.48550/ARXIV.1906.05598.

Revisiting the Fréchet distance between piecewise smooth curves

Jacobus Conradi[1], Anne Driemel[2], and Benedikt Kolbe[3]

1 **Department of Computer Science, University of Bonn, Germany**
2 **Hausdorff Center for Mathematics, University of Bonn, Germany**
3 **Hausdorff Center for Mathematics, University of Bonn, Germany**

Abstract

With the notable exception of an algorithm for the decision problem for planar piecewise smooth curves due to Rote (2007), research into algorithms for computing the Fréchet distance has concentrated on comparing polygonal curves. We present an algorithm for the decision problem for piecewise smooth curves that is both conceptually simple and naturally extends to the first algorithm for the problem for piecewise smooth curves in $\mathbb{R}^d$. To this end, we introduce a decomposition of the free space diagram into a controlled number of pieces that can be used to solve the decision problem using techniques similar to the polygonal case. Assuming the algorithm is given two continuous curves, each consisting of a sequence of m, resp. n, smooth pieces, where each piece belongs to a sufficiently well-behaved class of curves, such as the set of algebraic curves of bounded degree, we solve the decision problem in $O(mn)$ time. Furthermore, we study approximation algorithms for piecewise smooth curves that are also c-packed. We adapt the existing framework for $(1+\varepsilon)$-approximations and show that an approximate decision can be computed in $O(cn/\varepsilon)$ time for any $\varepsilon > 0$.

1 Introduction and motivation

The Fréchet distance is a well-studied distance measure between curves, with a long history in both applications and algorithmic research. The wealth of work surrounding the analysis of algorithms for computing the Fréchet distance is centered primarily on polygonal curves. However, more complicated curves and especially splines are natural objects that have become commonplace in industrial applications for, e.g., computer graphics, robotics and to represent motion tracking or planning data. A crucial prerequisite to using smooth curves similarly to polygonal curves in such contexts is the ability to effectively answer elementary algorithmic questions for such curves. A natural and fundamental task in computational geometry is the computation of the Fréchet distance between smooth curves such as splines. Despite this, as far as we know, there is no known approach to realizing such a computation for curves in $\mathbb{R}^d$. To tackle the case of smooth curves in the plane ($d = 2$), Rote [3] introduced an approach based on analyzing the turning angle and planar curvature of the planar curves. However, this approach does not easily generalize to higher dimensions. We revisit this problem and present a novel, simpler approach, with the additional benefit that it works for higher dimensions, with the same time complexity. Our methods are conceptually simple, but rely on a number of key technical ingredients.

Problem definition Throughout the paper, γ_1 and γ_2 will be used to denote two piecewise smooth curves in $\mathbb{R}^d$ with d fixed, that is, continuous maps $\gamma_1, \gamma_2 : [0,1] \to \mathbb{R}^d$ that are comprised of m and n smooth pieces, each of class C^2. Let $A_{[0,1]}$ be the set of continuous and bijective maps $\alpha : [0,1] \to [0,1]$ that are increasing. The **Fréchet distance** between γ_1 and γ_2 is defined as $\mathrm{d}_{\mathcal{F}}(\gamma_1, \gamma_2) := \inf_{\alpha,\beta \in A_{[0,1]}} \max_{t \in [0,1]} \|\gamma_1(\alpha(t)) - \gamma_2(\beta(t))\|$. Our methods

40th European Workshop on Computational Geometry, Ioannina, Greece, March 13–15, 2024.
This is an extended abstract of a presentation given at EuroCG'24. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

naturally allow any fixed ℓ_p norm with $1 < p < \infty$ for the norm $\|\cdot\|$ (the cases $p = 1, \infty$, while possible, would add a level of technicality to our treatment that distracts from its relative simplicity). We focus primarily on the **decision problem** of deciding whether the Fréchet distance between two piecewise smooth curves is at most a given $\delta > 0$.

Results Our first main contribution is that we establish an algorithm to solve the decision problem for the Fréchet distance between piecewise smooth curves. Assuming that the curves are **algebraically bounded curves**, i.e., piecewise smooth algebraic curves where the degree of the curves is bounded by a constant, we obtain a bound of $O(mn)$ for the time complexity of the decision problem, which matches the polygonal case. The running time is independent of the ambient dimension but the algebraic complexity of the operations involved in the algorithm depends on the dimension and the nature of the curves. Our algorithm for the decision problem results in an algorithm for the computation of the Fréchet distance for algebraically bounded curves in $O(mn \log(mn))$ time using parametric search, similarly to the polygonal case.

It is known [1] that the decision problem cannot be solved in strongly subquadratic time, so research has focused on investigating algorithms for restricted classes of curves. Our second contribution is that we show that we can adapt the framework from [2] for an efficient $(1 + \epsilon)$-approximation algorithm for the Fréchet distance between two c-packed, polyognal curves to the setting of c-packed piecewise smooth curves in $\mathbb{R}^d$. To this end, we introduce a simplification procedure for piecewise smooth curves and distill the necessary ingredients to obtain a linear time decision algorithm for algebraically bounded c-packed curves.

Comparison to previous work To arrive at an algorithm for the decision problem for smooth planar curves for the ℓ_2-norm for a given δ in general position, Rote uses a partitioning of the smooth curves, induced by condition on the turning angle and planar curvature, to obtain pieces for which the associated **free space diagram** FSD_δ (Section 2) is well-behaved. In contrast to this, our approach is to analyze the free space diagram directly, by studying the boundary of the **free space** $\mathcal{D}_\delta$ in FSD_δ, leading to a conceptually simpler algorithm. The free space is defined as the set of parameter value pairs at which the curves are at most a distance of δ apart. We propose a refined decomposition of each cell of FSD_δ into a controlled number (depending on the degree of the curves) of subcells, for which determining the existence of a monotone path connecting two intervals on the boundary of a subcell is easy. Here, the role of convexity of the free space in a cell for polygonal curves is replaced by monotonicity of the boundary curves of $\mathcal{D}_\delta$ within each subcell of the refined decomposition. We emphasize that our construction of the refined decomposition exclusively accesses the same values that are also required in Rote's work to process each subcell of FSD_δ.

Unlike the polygonal case, the free space within a cell of FSD_δ can be very complicated, as illustrated by a contour plot of the distance function in parameter space for two degree 3 splines in $\mathbb{R}^3$ in Figure 1 for different values of δ. Figure 3 shows another example of the kind of behavior of the free space one can expect within a cell. We note that both Rote's decision algorithm as well as ours assume values of δ for which the boundary of $\mathcal{D}_\delta$ has no singularities. We show that singularities of the boundary of $\mathcal{D}_\delta$ are confined to a small number of critical values of δ and are thus not necessary for the computation of $\mathrm{d}_\mathcal{F}$.

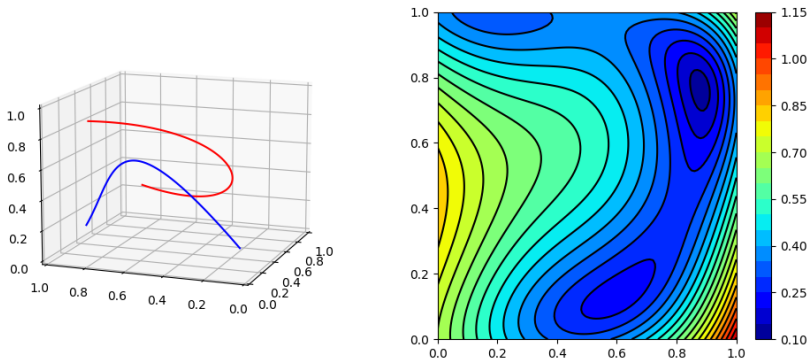


Figure 1 Two smooth curves in $\mathbb{R}^3$ and a contour plot of the associated distance function in the joint parametric space of the curves.

Computational assumptions We assume that we can compute the intersection of a curve with a sphere of a given radius centered at a point of another curve and find the parameter values in $[0, 1]$ that correspond to the intersections.

2 A combinatorial description of the free space diagram

For two piecewise smooth curves $\gamma_1, \gamma_2 : [0, 1] \to \mathbb{R}^d$ consisting of m and n pieces, respectively, and $\delta > 0$, the **free space** $\mathcal{D}_\delta = \mathcal{D}_\delta(\gamma_1, \gamma_2)$ is defined as

$$\mathcal{D}_\delta(\gamma_1, \gamma_2) = \left\{(x, y) \in [0, 1]^2 | \|\gamma_1(x) - \gamma_2(y)\| \leq \delta\right\}.$$

The complement of $\mathcal{D}_\delta$ in $[0, 1]^2$ is referred to as the **forbidden region**. There is a natural partition of the joint parameter space $[0, 1]^2$ of both curves into $m \cdot n$ rectangular cells such that γ_1 and γ_2 are smooth when restricting to the interior of each rectangle. The resulting decomposition of $[0, 1]^2$ together with the partitioning into the free space and forbidden region is known as the **free space diagram** FSD_δ. A key motivation behind the definition is the observation that $\mathrm{d}_\mathcal{F}(\gamma_1, \gamma_2) \leq \delta$ iff there is a path from $(0, 0)$ to $(1, 1)$ through the free space in $[0, 1]^2$ that is monotone in both coordinates.

Overview of the algorithm Similarly to the classical polygonal case, to solve the decision problem, we investigate the existence of a monotone (in both coordinates) path from $(0, 0)$ to $(1, 1)$ in the free space $\mathcal{D}_\delta$. To this end, we refine the free space diagram using the boundary B_δ of the free space. Our decision algorithm has the following high-level description.

1. Mark the minima and maxima of the boundary B_δ of the free space in FSD_δ in the x (horizontal) and y (vertical) direction.
2. Cut each cell of FSD_δ into subcells, horizontally (vertically) through each marked point if it has a vertical (horizontal) tangent. Mark each point of intersection of a cut with B_δ.
3. For each resulting subcell, pair the marked points on the boundary according to how they are connected by B_δ through monotone arcs, so that adjacent points are paired.
4. Solve the decision problem for FSD_δ using only the marked points and pairings by computing reachable intervals on the boundaries of cells, in particular
   a. process all cells in lexicographical order of their indices (row by row, from the left);
   b. for each cell, process all subcells within the cell in lexicographical order.

2.1 Refining the free space diagram

We consider the boundary B_δ of FSD_δ as a set of curves, as opposed to the boundary of a region. Let I_{sing} be the set of singularities of B_δ in the interior of the cells of FSD_δ, consisting of points where B_δ has a cusp or intersects itself. Like Rote, we assume that $I_{\mathrm{sing}} = \emptyset$. It turns out that for almost all δ, there are no singular points of B_δ in the interior of each cell in FSD_δ associated to the smooth pieces of the curves, so that $I_{\mathrm{sing}} = \emptyset$, after possibly applying a small perturbation to δ, as illustrated in Figure 2. Intuitively, the scarcity of critical values for δ can be explained by noting that each critical value corresponds to a value of δ for which there are points $(t_1, t_2) \in B_\delta$ such that

$$\frac{\mathrm{d}}{\mathrm{d}t_1}\|\gamma_1(t_1) - \gamma_2(t_2)\|_p = 0 = \frac{\mathrm{d}}{\mathrm{d}t_2}\|\gamma_1(t_1) - \gamma_2(t_2)\|_p, \tag{1}$$

equations which themselves do not depend on δ. In contrast, we note that for the norms ℓ_1 and ℓ_∞ in the definition of the Fréchet distance, B_δ may contain cusp singularities for all values of δ in an open interval.

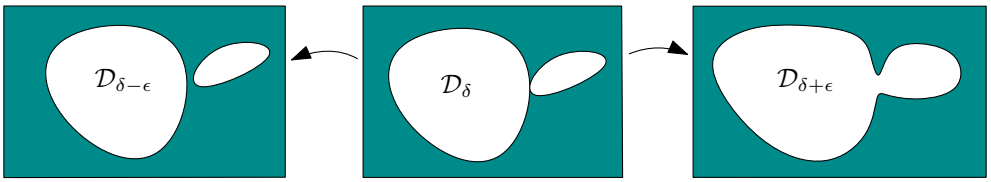


Figure 2 Illustration of singular points and changes of the boundary B_δ as δ changes.

Let $E_h \subset B_\delta$ ($E_v \subset B_\delta$) be the set of extrema of the free space in the y direction, with horizontal tangent (in the x direction, with vertical tangent). For simplicity of exposition, we assume that each of E_h and E_v is a collection of isolated points. In particular, B_δ does not have a vertical or horizontal segment, which means that there is no arc of one curve that lies at a constant distance δ from a point on the other.

For a point $z \in E_h$ (E_v), we fix the cell in FSD_δ containing z, and trace the vertical (horizontal) line incident to z inside this cell. The result is a refinement of each cell of FSD_δ into a collection of **subcells** $\{S\}$, illustrated in Figure 3 for one cell of FSD_δ.

▶ **Lemma 2.1.** *In the interior of each subcell in $\{S\}$, B_δ is a union of smooth arcs that are monotone in both coordinates of $\mathbb{R}^2$ and disjoint except possibly at the boundary of a subcell.*

We record each intersection $\mathcal{I}_\mathcal{S}$ of B_δ with the boundary of each subcell $\mathcal{S}$, which together form the set $\mathcal{I} = \bigcup_{\mathcal{S} \text{ is subcell}} \mathcal{I}_\mathcal{S}$ of all intersections of subcell walls with B_δ. Notice that B_δ can be naturally interpreted as a graph G_δ with vertex set $\mathcal{I}$, and each edge a monotone arc contained in a subcell. We partition the two sets E_h and E_v into the sets E_h^+ and E_h^-, and E_v^+ and E_v^-, respectively, according to whether the forbidden region lies locally to the right of or above the point $(-)$, or to the left of or below the point $(+)$. For the bottom and left edge of each subcell $\mathcal{S}$, we refer to the information of whether the boundary B_δ at each point in $\mathcal{I}_\mathcal{S}$ is increasing or decreasing as a function of the horizontal x-coordinate as the **slope information** of these points. In other words, the slope information at a point $z \in \mathcal{I}_\mathcal{S}$ can be thought of as an extra bit associated to z that encodes whether B_δ curves to the left or to the right at z, illustrated in Figure 3 by arrows.

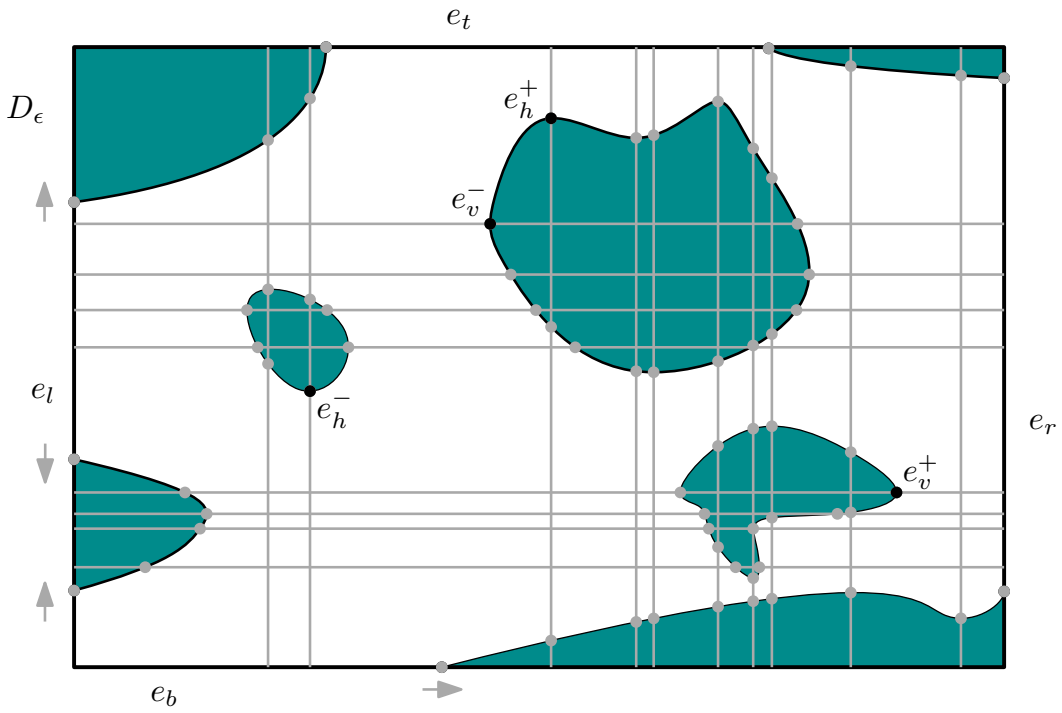


Figure 3 The decomposition of a cell of the free space diagram into subcells arising from the horizontal and vertical lines at extremities of the forbidden region in the coordinate directions.

The slope information on the bottommost and leftmost edges of the original cells of FSD_δ leads to a construction recipe for the combinatorial structure of G_δ from its vertex set.

▶ **Lemma 2.2.** *Assume δ is such that $I_{sing} = \emptyset$. There is an algorithm that reproduces the combinatorial structure of G_δ, using the sets $E_h^+, E_h^-, E_v^+, E_v^-$, and $\mathcal{I}$ along with the slope information on the bottommost and leftmost edges of* FSD_δ*, in time $O(|\mathcal{I}|)$.*

▶ Remark. Figure 4 illustrates the necessity of some knowledge of the slope information on edges of a subcell for the accurate reconstruction of B_δ inside a cell.

As illustrated in Figure 5, the slope information for points in a subcell can be gleaned by evaluating the distance between the two curve segments at certain test points.

The parts of the cell walls that are reachable by monotone paths in the free space can be computed in a structurally similar way to the polygonal case, leading to an algorithm for the

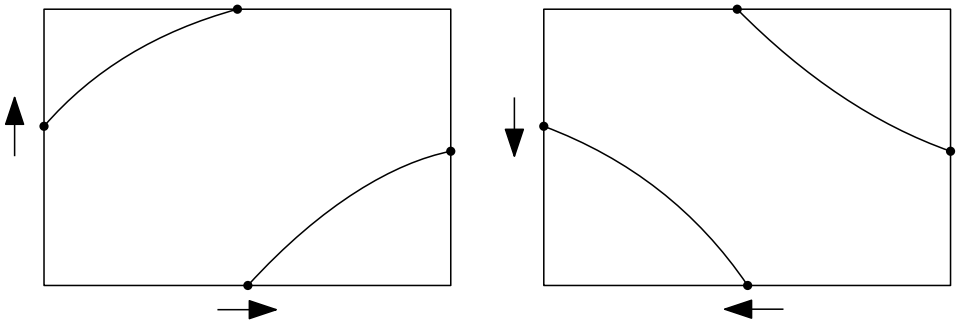


Figure 4 Two different sets of slope information and their combinatorial structures in a subcell.

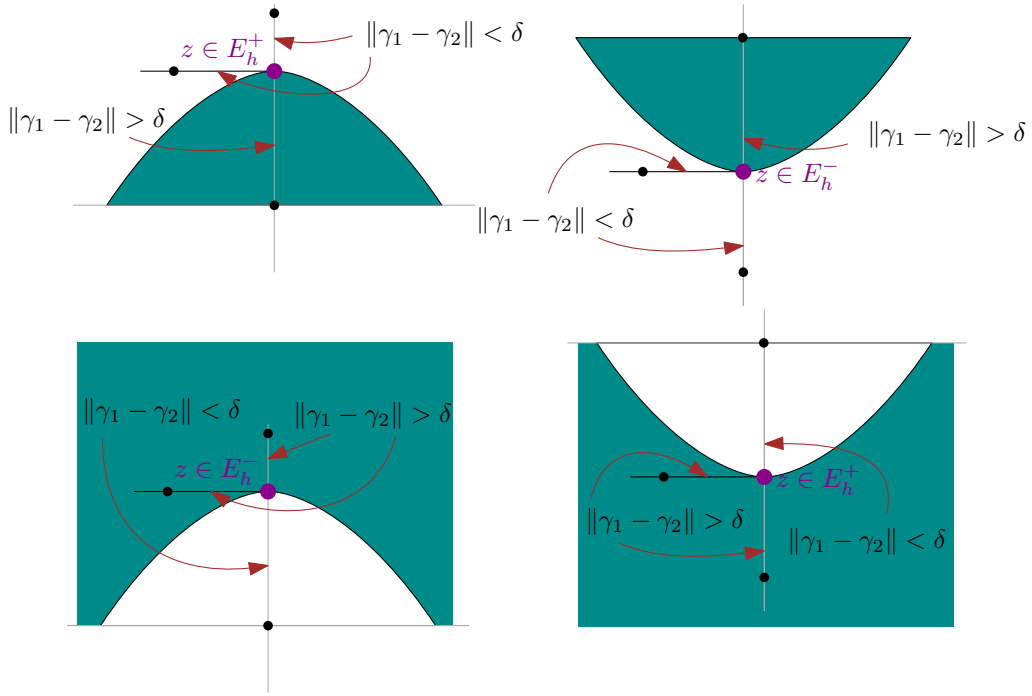


Figure 5 Finding slope information by evaluating the distance at points.

decision problem. The crucial insight is that each arc of the boundary of the free space inside each subcell is a monotone arc, which allows for to transfer the reachable intervals on the bottom and left subcell walls to neighboring cell walls in constant time. The following result is due to there only being a constant number of subcells in each original cell for algebraically bounded curves, with constant depending only on the allowed degree of the curves.

▶ Proposition 2.3. Given two algebraically bounded piecewise smooth curves γ_1, γ_2 in $\mathbb{R}^d$ comprised of m and n pieces, respectively, and a value of δ such that B_δ has no singularities, one can decide if $\mathrm{d}_{\mathcal{F}}(\gamma_1, \gamma_2) \leq \delta$. The running time is bounded by $O(mn)$.

The solution to the decision problem can be used to compute the Fréchet distance in the same way as in the case of polygonal curves, using parametric search. For this, the first step is to identify the $O(mn)$ critical values where marked points appear or disappear, components merge, appear, or start touching the boundary of cells. We then apply a binary search among these $O(mn)$ critical values to narrow down the range of values for the Fréchet distance to be a critical value corresponding to a change of the order in the x- and y-direction of the marked points in the free space diagram. Inbetween these $O(mn)$ critical values, Cole's variant of parametric search with a parallel sorting algorithm for both the x- and y-coordinates of all the marked points of B_δ yields an overall running time of $O(mn \log(mn))$ for the computation of the Fréchet distance.

▶ **Theorem 2.4.** *Let γ_1 and γ_2 be two algebraically bounded curves in $\mathbb{R}^d$ consisting of m and n pieces, respectively. Then the Fréchet distance between γ_1 and γ_2 can be computed in $O(mn)$ space and in $O(mn \log(mn))$ operations (of bounded algebraic complexity).*

3 The decision problem in linear time for c-packed curves

A curve γ is c-**packed** if the total arc length of γ inside any ball of radius r is at most cr. By utilizing a simplification procedure for piecewise smooth curves that transforms c-packed curves into c'-packed curves and guarantees a minimum arclength of each piece of the simplification, one can show that the number of grid cells that are reachable and contain free space, of simplified c-packed curves, depends linearly on n. This ultimately leads to our main result concerning approximate decision algorithms.

▶ **Corollary 3.1.** *Let γ_1 and γ_2 be two piecewise smooth algebraically bounded c-packed curves, $1 \geq \epsilon > 0$ and $\delta > 0$. There is an algorithm that correctly outputs, in $O(cn/\epsilon)$ time, either (i) a $(1+\epsilon)$-approximation to $\mathrm{d}_{\mathcal{F}}(\gamma_1, \gamma_2)$, (ii) $\mathrm{d}_{\mathcal{F}}(\gamma_1, \gamma_2) < \delta$, or (iii) $\mathrm{d}_{\mathcal{F}}(\gamma_1, \gamma_2) > \delta$.*

References

1 Karl Bringmann. Why walking the dog takes time: Frechet distance has no strongly subquadratic algorithms unless SETH fails. In Proceedings - Annual IEEE Symposium on Foundations of Computer Science, FOCS, pages 661–670, 2014. arXiv:1404.1448, doi:10.1109/FOCS.2014.76.

2 Anne Driemel, Sariel Har-Peled, and Carola Wenk. Approximating the Fréchet distance for realistic curves in near linear time. Discrete and Computational Geometry, 48(1):94–127, 2012. arXiv:1003.0460, doi:10.1007/s00454-012-9402-z.

3 Günter Rote. Computing the Fréchet distance between piecewise smooth curves. Computational Geometry: Theory and Applications, 37(3 SPEC. ISS.):162–174, aug 2007. doi:10.1016/J.COMGEO.2005.01.004.

Computing Enclosing Depth*

Bernd Gärtner[1], Fatime Rasiti[2], and Patrick Schnider[3]

1 **Department of Computer Science, ETH Zürich**
`gaertner@inf.ethz.ch`
2 **Department of Mathematics, ETH Zürich**
`frasiti@student.ethz.ch`
3 **Department of Computer Science, ETH Zürich**
`patrick.schnider@inf.ethz.ch`

Abstract

Enclosing depth is a recently introduced depth measure which gives a lower bound to many depth measures studied in the literature. So far, enclosing depth has only been studied from a combinatorial perspective. In this work, we give the first algorithms to compute the enclosing depth of a query point with respect to a data point set in any dimension. In the plane we are able to optimize the algorithm to get a runtime of $O(n \log n)$. In constant dimension, our algorithms still run in polynomial time.

Related Version arXiv:2402.12371

1 Introduction

Medians play an important role in statistics. In contrast to the mean value of some given data, the median depends only on the order of the data points and not on their exact positions. Hence, it is robust against outliers. As data sets are multidimensional in many cases, we are interested in an extension of the term 'median' to higher dimensions. Since there is no clear order of the data points, there are various generalizations of the median to higher dimensions [4, 12, 13]. In order to define the median of some data, the notion of depth of a query point has been introduced. A median is then a query point with the highest depth. Many depth measures only depend on the relative positions of the data points, just like the median, making them again robust against outliers.

After the first depth measure was introduced by Tukey [22] (and is therefore known as Tukey depth), Donoho and Gasko [9] established the idea of a multidimensional median as a deepest point relative to the data points. Various depth measures with different properties have since been introduced, such as simplicial depth [11] and convex hull peeling depth [4].

Depth measures are an important tool in Computer Science for example in geometric matching, pattern matching, clustering [8, 10, 19] and shape fitting applications [2]. Since depth measures give a way to compute medians of data points they also find applications in Statistics such as data visualization [22] and regression analysis [16, 21].

▶ **Definition 1.1** (Depth measure). Let $d \in \mathbb{N}$ and $(\mathbb{R}^d)^S$ be the family of all finite point sets in $\mathbb{R}^d$. A depth measure is a function $D : (\mathbb{R}^d)^S \times \mathbb{R}^d \to \mathbb{R}_{\geq 0}$, $(S, q) \mapsto D(S, q)$. In particular, the function D assigns to a given finite point set S and a query point q a value, which describes how deep the query point q lies within the data set S.

Assume we are given a data set S. Consider all hyperplanes spanned by the points of S. This arrangement A of hyperplanes divides $\mathbb{R}^d$ into connected components of $\mathbb{R}^d \backslash A$. We call

* This work is based on the Master thesis of the second author.

40th European Workshop on Computational Geometry, Ioannina, Greece, March 13–15, 2024.
This is an extended abstract of a presentation given at EuroCG'24. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

these connected components cells. A depth measure where all points in a cell have the same depth is called *combinatorial*.

Aloupis et al. [4] used the fact that simplicial depth is a combinatorial depth measure to compute the simplicial median for $d = 2$ in $O(n^4)$ time. Sachini [15] modified this algorithm to compute the simplicial depth for the whole plane in $O(n^4)$ time. For the case $d = 3$ there are various algorithms that compute the simplicial depth of a single query point in $O(n^2)$ [7]. Cheng and Ouyang [7] discussed an extension of this algorithm to compute simplicial depth in $\mathbb{R}^4$ in $O(n^4)$ time. Afshani et al. [1] later introduced methods to compute simplicial depth in $O(n^d \log n)$ time for $d > 4$. This bound was improved by Pilz et al. [14] to $O(n^{d-1})$.

Another well studied combinatorial depth measure is Tukey depth, also known as halfspace depth. For the case $d = 2$, Aloupis et al. [3] gave a worst case lower bound of $\Omega(n \log n)$ for computing the Tukey depth of an arbitrary query point q with respect to a given point set S of size n. In fact, the Tukey depth of a query point relative to a point set of size n can be computed in $O(n \log n)$ time [17]. There are different approaches to compute the Tukey depth in different dimensions [6, 5, 18]. The algorithm of Rousseeuw et al. [18] to compute the Tukey depth of a query point in $\mathbb{R}^d$ for $d > 2$ has a run time of $O(n^{d-1} \log n)$.

Studying more general families of combinatorial depth measures, Schnider introduced the notion of *enclosing depth*, which turns out to be a natural lower bounds for many combinatorial depth measures [20]. In this work, we will focus on enclosing depth and provide algorithms to compute it.

▶ **Definition 1.2** (k-enclosing)**.** Let S be a point set of size $(d+1)k$ in $\mathbb{R}^d$ and q a query point. If S can be partitioned into $d+1$ pairwise disjoint subsets $S_1, ..., S_{d+1}$, each of size k, such that for any transversal $p_1 \in S_1, ..., p_{d+1} \in S_{d+1}$ the point q lies in the convex hull of $p_1, ..., p_{d+1}$, then we say that $S_1, ..., S_{d+1}$ k-encloses the point q. (Figure 1)

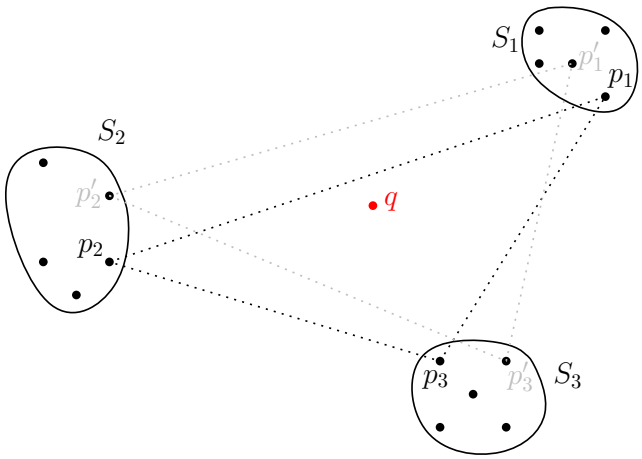


Figure 1 The set $S = S_1 \cup S_2 \cup S_3$ 5-encloses the query point q in $\mathbb{R}^2$.

▶ **Definition 1.3** (Enclosing Depth)**.** Let S be a finite point set in $\mathbb{R}^d$ and q be a query point. The enclosing depth of q with respect to S is the maximum k such that there exist subsets $S_1, ..., S_{d+1}$ of S which k-enclose q. We denote it by $ED(S, q)$. (Figure 2)

In this work, we present algorithms to compute the enclosing depth of a query point q with respect to a data set S in $\mathbb{R}^2$ (Section 2) as well as in general dimension (Section 3).

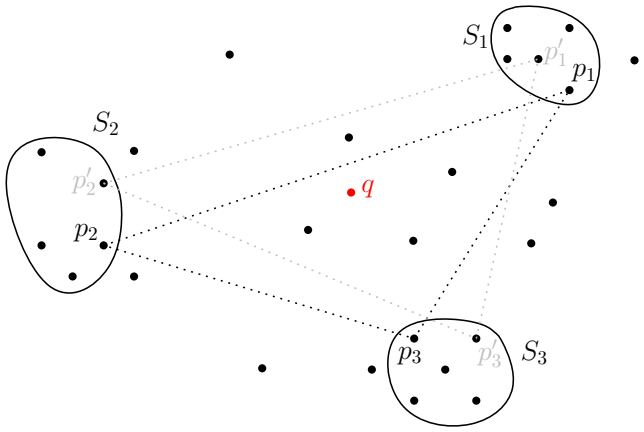


Figure 2 The enclosing depth of the query point q is at least 5.

2 The planar case

Before describing our algorithm, we introduce some combinatorial lemmas that will be helpful in proving the correctness of our algorithm. For these lemmas, we will assume that the query point q is the origin and that the data point set S lies on the unit circle. Combinatorially, this is not a restriction, as the following lemma shows.

▶ **Lemma 2.1.** *Let $S = \{s_1, \dots, s_n\} \subset \mathbb{R}^2$ be a data point set and $q \in \mathbb{R}^2$ a query point such that $S \cup \{q\}$ is in general position. Denote by $S' = \{s'_1, \dots, s'_n\}$ the point set defined by centrally projecting each point in S to a circle of unit radius with center q. Then q is in the convex hull of s_i, s_j, s_k if and only if it is in the convex hull of s'_i, s'_j, s'_k.*

Proof. Assume that q is not in the convex hull of a, b, c. Then there is a line ℓ through q having all of a, b, c on the same side. This is invariant under central projection from q. ◀

Let now $S = \{s_1, \dots, s_n\}$ on the unit circle be ordered in counter-clockwise direction. By an *interval* $[s_a, s_b]$ we denote all the points in S that lie between s_a and s_b, that is,

$$[s_a, s_b] = \begin{cases} \{s \in S \mid s_a \le s \le s_b\} & s_a \le s_b \\ \{s \in S \mid s \ge s_a \text{ or } s \le s_b\} & s_a > s_b \end{cases}$$

In the following, we write indices modulo n, that is, $s_i = s_{i-n}$ for $i \ge n$. We show that in order to find k-enclosing sets we can restrict our attention to intervals and their endpoints.

▶ **Lemma 2.2.** *Let $a_1, a_2, b_1, b_2, c_1, c_2 \in S$ such that the intervals $[a_1, a_2]$, $[b_1, b_2]$ and $[c_1, c_2]$ are pairwise disjoint and for every choice of $a \in \{a_1, a_2\}$, $b \in \{b_1, b_2\}$, $c \in \{c_1, c_2\}$ the origin lies in the convex hull of a, b, c. Then for every choice $a \in [a_1, a_2]$, $b \in [b_1, b_2]$, $c \in [c_1, c_2]$ the origin lies in the convex hull of a, b, c.*

Proof. Assume for the sake of contradiction that the convex hull of a, b, c does not contain the origin and let ℓ be a line through the origin that has all of a, b, c on the same side ℓ^+. As the intervals are pairwise disjoint, one of a_1 or a_2 must also be in ℓ^+. Thus the convex hull of a_i, b, c does not contain the origin for some $i \in \{1, 2\}$. Repeating this argument two more times, we find that one of the 8 triangles spanned by the endpoints does not contain the origin, which is a contradiction to the assumptions of the lemma. ◀

We want to restrict the considered intervals even further. To this end, we define for each point $s \in S$ its *opposite neighbors* $s^{(r)}$ and $s^{(\ell)}$ as the last point in S before $-s$ and the first point in S after $-s$, respectively, see Figure 3 for an illustration.

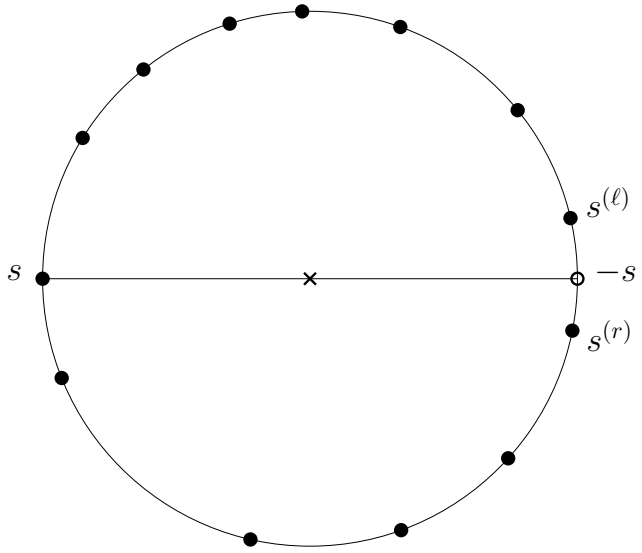


Figure 3 The opposite neighbors $s^{(r)}$ and $s^{(\ell)}$ of a point $s \in S$.

▶ **Lemma 2.3.** *Let $S_1 = [s_i, s_{i+k}]$, S_2 and S_3 be subsets of S that $(k+1)$-enclose the origin. Denote $s_j = s_i^{(r)}$ and $s_m = s_{i+k}^{(\ell)}$. Then S_1, $S_2' := [s_{j-k}, s_j]$ and $S_3' := [s_m, s_{m+k}]$ also $(k+1)$-enclose the origin.*

Proof. See Figure 4 for an illustration of the proof. We first note that (up to relabeling) we must have that $S_2 \subset [s_{i+k+1}, s_j]$ and $S_3 \subset [s_m, s_{i-1}]$. Indeed, both lines through s_i and the origin, as well as through s_{i+k} and the origin must separate S_2 and S_3, as otherwise there would be a choice $a \in S_1$, $b \in S_2$, $c \in S_3$ whose convex hull does not contain the origin. Thus, we can write $S_2 = [s_b, s_{b+k}]$ for $i + k + 1 \leq b \leq j - k$ and similarly $S_2 = [s_{c-k}, s_c]$ for $m + k \leq c \leq i - 1$. In particular both s_i, s_b, s_c and s_{i+k}, s_b, s_c contain the origin.

We claim that s_i, s_b, s_m also contains the origin. Assume it does not. Then either the line ℓ_i through s_i and the origin or the line ℓ_b through s_b and the origin must separate s_c and s_m. By construction, ℓ_i has both s_c and s_m on the same side, so it must be ℓ_b. But this would imply that $m > c$, which is a contradiction. A symmetric argument also shows that s_{i+1}, s_b, s_m contains the origin. Analogously it can be shown that s_i, s_j, s_c as well as s_{i+1}, s_j, s_c contains the origin. Finally, by construction s_i, s_j, s_m as well as s_{i+k}, s_j, s_m contains the origin. The statement now follows from Lemma 2.2 and the fact that $b \leq j - k$ and $m + k \leq c$. ◀

Description of the algorithm: We are given a set S of n data points in the plane and a query point q. In a pre-processing step we first sort the points radially around q, giving a counter-clockwise order $s_1, \ldots, s_n$ on S and then for each $s \in S$ we compute $s^{(r)}$ and $s^{(\ell)}$ using binary search. For the main part of the algorithm we run the following subroutine, which for a given integer k checks whether there are three sets that $(k+1)$-enclose q: for each $s_i \in S$, with $s_j = s_i^{(r)}$ and $s_m = s_{i+k}^{(\ell)}$ the subroutine checks whether all 8 triangles a, b, c for $a \in [s_i, s_{i+k}]$, $b \in [s_{j-k}, s_j]$, $c \in [s_m, s_{m+k}]$ contain q and the intervals are pairwise disjoint, returning TRUE if this holds for some s_i, and FALSE otherwise. By doing a binary

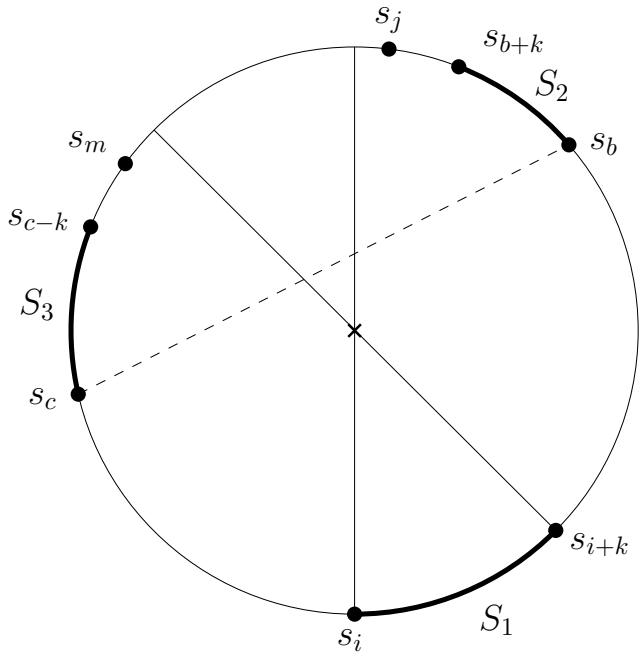


Figure 4 An illustration of the proof of Lemma 4.

search over the values of $k \in \{0, \dots, n\}$ we find the largest value k for which the subroutine returns true and return $(k+1)$.

▶ **Theorem 2.4.** *The above algorithm computes the enclosing depth of q with respect to $S \subset \mathbb{R}^2$ in time $O(n \log n)$.*

Proof. We first show the correctness of the algorithm. It follows from Lemma 2.2 that if the subroutine returns TRUE then the considered intervals are indeed $(k+1)$-enclosing. On the other hand, if there are $(k+1)$-enclosing sets, by Lemma 2.3 the subroutine will find them.

As for the runtime, we can sort in time $O(n \log n)$. After this, we perform $2n$ binary searches, each taking $O(\log n)$ time, thus the total runtime of the pre-processing step is $O(n \log n)$. For the runtime of the subroutine, we notice that for each choice of s_i the required checks can be done in time $O(1)$, so the runtime of the subroutine is $O(n)$. As we call it for $O(\log n)$ many values of k we get the desired runtime. ◀

3 Higher dimensions

Our algorithm in general dimension is based on the following observation:

▶ **Lemma 3.1** (Lemma 20 in [20]). *Let $S_1, \dots, S_{d+1} \subset S \subset \mathbb{R}^d$ be point sets which enclose a point q, where $S \cup \{q\}$ is in general position. Then there are $d+1$ closed halfspaces $H_1^-, \dots, H_{d+1}^-$ such that each H_i^- contains q on its boundary, $H_i^- \cap S = S_i$ for each i and $H_1^- \cup \dots \cup H_{d+1}^- = \mathbb{R}^d$.*

Denoting by H_i^+ the complement of H_i^- we show in the full version that we get $S_i \subset \bigcap_{j \neq i} H_j^+$ and $\bigcap_i H_i^+ = \{q\}$. As we show in the full version, given enclosing sets $S_1, \dots, S_{d+1}$ we can rotate the halfspaces H_i^+ to get closed halfspaces H_i' whose boundaries contain q and $d-1$ points of S and for which $H_i' \cap S = H_i^+ \cap S$ and $\bigcap_i H_i' = \{q\}$. On the other

hand, given halfspaces $H'_1, \ldots, H'_{d+1}$, each boundary containing q and $d-1$ points of S with $\bigcap_i H'_i = \{q\}$, defining $S_i := \bigcap_{j \neq i}(H'_j \cap S)$ we show in the full version that for every transversal $p_1 \in S_1, ..., p_{d+1} \in S_{d+1}$ the point q lies in the convex hull of $p_1, ..., p_{d+1}$. Combining these facts, we get the following strengthening of Lemma 3.1.

▶ **Lemma 3.2.** *Let $S_1, \ldots, S_{d+1} \subset S \subset \mathbb{R}^d$ and $q \in \mathbb{R}^d$ such that $S \cup \{q\}$ is in general position. Then $S_1, \ldots, S_{d+1}$ enclose q if and only if there are $d+1$ halfspaces $H'_1, \ldots H'_{d+1}$ whose boundaries contain q and $d-1$ points of S, and for which $S_i \subset \bigcap_{j \neq i} H_j^+$ and $\bigcap_i H_i^+ = \{q\}$.*

Description of the algorithm: For each choice of $d-1$ points in S, consider the two halfspaces defined by the hyperplane through q and these $d-1$ points. This defines a set $\mathcal{H}$ of $2\binom{n}{d-1}$ halfspaces. For any $d+1$ halfspaces $H_1, \ldots, H_{d+1} \in \mathcal{H}$ first check if $\bigcap_i H_i = \{q\}$. If not, continue with the next choice, otherwise count for each i the number of points of S in $\bigcap_{j \neq i} H_j$ and denote by k the smallest of the $d+1$ numbers. In the end, return the largest such k encountered over all choices of $d+1$ halfspaces in $\mathcal{H}$.

▶ **Theorem 3.3.** *The above algorithm computes the enclosing depth of q with respect to $S \subset \mathbb{R}^d$ in time $O(n^{d^2})$.*

Proof. The correctness follows from Lemma 3.2. As for the runtime, we have $|\mathcal{H}| \in O(n^{d-1})$ out of which we choose sets of $d+1$ elements, so there are $O(n^{(d-1)(d+1)})$ sets of halfspaces that we consider. For each set the check and the counting can be done in time $O(n)$ so the total runtime is $O(n^{(d-1)(d+1)+1}) = O(n^{d^2})$. ◀

4 Conclusion

We have given two algorithms to compute the enclosing depth of a query point q with respect to a data point set S, one for the plane and one in general dimension. The planar algorithm matches the runtimes of computing many other depth measures. For some measures, such as Tukey depth, matching lower bounds for the computation time have been shown [3]. It would be interesting to adapt these lower bounds to enclosing depth.

In higher dimension, many depth measures can be computed in time $O(n^{d-1})$, which is significantly faster than the runtime of our algorithm. We believe that enclosing depth can be computed more efficiently as well, but some additional ideas are likely required for this.

Finally, there are other natural algorithmic problems for depth measures even in the plane, such as computing the depth of the entire plane or finding a deepest query point. Using our algorithm and the fact that the arrangement spanned by the lines through all pairs of data points has $O(n^4)$ cells we can solve both those problems in $O(n^5 \log n)$, but we again believe that this is not optimal.

References

1 Peyman Afshani, Donald R. Sheehy, and Yannik Stein. Approximating the simplicial depth in high dimensions. *In Proceedings of 32nd European workshop on Computational Geometry*, pages 39–42, 2001.

2 Dror Aiger, Haim Kaplan, and Micha Sharir. Duality-based approximation algorithms for depth queries and maximum depth. *arXiv preprint*, page arXiv:2006.12318, 2020.

3 Greg Aloupis, Carmen Cortés, Francisco Gómez, Michael Soss, and Godfried Toussaint. Lower bounds for computing statistical depth. *Computational Statistics & Data Analysis*, 40(2):223–229, 2002.

4 Greg Aloupis, Stefan Langerman, Michael Soss, and Godfried Toussaint. Algorithms for bivariate medians and a fermat-torricelli problem for lines. *Computational Geometry*, 2003.

5 David Bremner, Dan Chen, John Iacono, Stefan Langerman, and Pat Morin. Output-sensitive algorithms for tukey depth and related problems. *Statistics and Computing*, 18(3): 259–266, 2008.

6 Dan Chen, Pat Morin, and Uli Wagner. Absolute approximation of tukey depth: Theory and experiments. *Computational Geometry*, 46(5): 566–573, 2013.

7 Andrew Y. Cheng and Ming Ouyang. On algorithms for simplicial depth. *In Proceedings of the 13th Canadian Conference of Computation Geometry (CCCG)*, pages 53–56, 2001.

8 Andreas Christmann, Paul Fischer, and Thorsten Joachims. Classification based on the support vector machine, regression depth, and discriminant analysis. page 225–230, 2002.

9 David L. Donoho and Miriam Gasko. Breakdown properties of location estimates based on halfspace depth and projected outlyingness. *The Annals of Statistics*, 20(4): 1803-1827 ,1992.

10 Rebecka Jörnsten. Clustering and classification based on the l1 data depth. *Journal of Multivariate Analysis*, 90(1): 67-89, 2004.

11 Regina Y. Liu. On a notion of data depth based on random simplices. *The Annals of Statistics*, 18(1): 405–414, 1990.

12 Regina Y. Liu, Jesse M. Parelius, and Kesar Singh. Multivariate analysis by data depth: descriptive statistics, graphics and inference. *Ann. Statist.*, 27(3): 783–858, 06.1999.

13 Karl Mosler. Springer Berlin Heidelberg, Berlin, Heidelberg. *Depth Statistics.*, pages 17–34, 06.1999.

14 Alexander Pilz, Emo Welzl, and Manuel Wettstein. From crossing-free graphs on wheel sets to embracing simplices and polytopes with few vertices. *Discrete & Computational Geometry*, 64(3): 1067–1097, 2020.

15 Sachini Kanchana Rajapakse. Computing Batched Depth Queries and the Depth of a Set of Points. *Faculty of Graduate Studies of The University of Manitoba Winnipeg*, pages 20–26, 2022.

16 Peter J. Rousseeuw and Mia Hubert. Regression depth. *Journal of the American Statistical Association*, 94(446): 388-402, 1999.

17 Peter J. Rousseeuw and Ida Ruts. Bivariate location depth. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 45(4): 516–526, 1996.

18 Peter J. Rousseeuw and Anja Struyf. Computing location depth and regression depth in higher dimensions. *Statistics and Computing*, 8(3): 193–203, 1998.

19 Ida Ruts and Peter J. Rousseeuw. Computing depth contours of bivariate point clouds. *Computational Statistics and Data Analysis*, 23(1): 153–168, 1996.

20 Patrick Schnider. Enclosing depth and other depth measures. *Combinatorica*, pages 1–23, 2023.

21 Patrick Schnider and Pablo Soberón. Combinatorial depth measures for hyperplane arrangements. In *39th International Symposium on Computational Geometry (SoCG 2023)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2023.

22 John W. Tukey. Mathematics and the picturing of data. 1975.

Polylogarithmic Time Algorithms for Shortest Path Forests in Programmable Matter*

Andreas Padalkin[1] and Christian Scheideler[2]

1 **Paderborn University, Germany**
andreas.padalkin@upb.de
2 **Paderborn University, Germany**
scheideler@upb.de

Abstract

In this paper, we study the computation of shortest paths within the *geometric amoebot model*, a commonly used model for programmable matter. Shortest paths are essential for various tasks and therefore have been heavily investigated in many different contexts. For example, for the geometric amoebot model, Kostitsyna et al. have utilized shortest path trees to transform one amoebot structure into another [DISC, 2023]. We consider the *reconfigurable circuit extension* of the model where the amoebot structure is able to interconnect amoebots by so-called circuits. These circuits permit the instantaneous transmission of simple signals between connected amoebots.

We propose two distributed algorithms for the *shortest path forest problem* where, given a set of k sources and a set of ℓ destinations, the amoebot structure has to compute a forest that connects each destination to its closest source on a shortest path. For hole-free structures, the first algorithm constructs a shortest path tree for a single source within $O(\log \ell)$ rounds, and the second algorithm a shortest path forest for an arbitrary number of sources within $O(\log n \log^2 k)$ rounds. The former algorithm also provides an $O(1)$ rounds solution for the *single pair shortest path problem* (SPSP) and an $O(\log n)$ rounds solution for the *single source shortest path problem* (SSSP) since these problems are special cases of the considered problem.

Related Version *Full Version*: https://arxiv.org/abs/2402.12123

1 Introduction

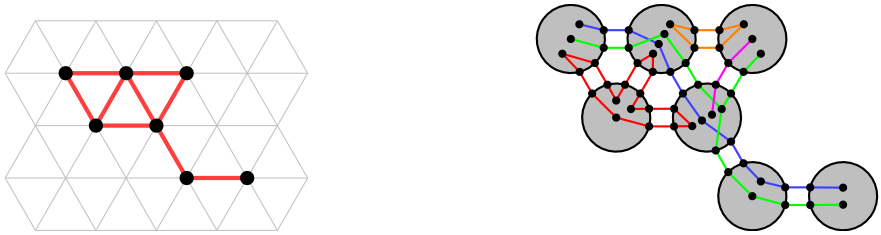
Programmable matter is matter that has the ability to change its physical properties in a programmable fashion [15]. Many exciting applications have already been envisioned for programmable matter such as self-healing structures and minimal invasive surgery, and shape-changing robots have already been prominent examples of the potentials of programmable matter in many blockbuster movies.

In the *amoebot model*, the matter consists of simple particles (called *amoebots*). In the geometric variant of the model, the amoebots form a connected amoebot structure on the infinite triangular grid, on which they move by *expansions* and *contractions*. However, since information can only travel amoebot by amoebot, many problems come with a natural lower bound of $\Omega(d)$ where d is the diameter of the structure.

For that reason, we consider the *reconfigurable circuit extension* to the amoebot model where the amoebot structure is able to interconnect amoebots by so-called *circuits*. These circuits permit the instantaneous transmission of simple signals between connected amoebots. The extension allows polylogarithmic solutions for various fundamental problems, e.g., leader election [8], and spanning tree construction [12].

* This work was supported by the DFG Project SCHE 1592/10-1. The research was initialized at the Dagstuhl Seminar 23091 "Algorithmic Foundations of Programmable Matter". We thank Shantanu Das, Yuval Emek, Maria Kokkou, Irina Kostitsyna, Tom Peters, and Andrea Richa for helpful discussions.

40th European Workshop on Computational Geometry, Ioannina, Greece, March 13–15, 2024.
This is an extended abstract of a presentation given at EuroCG'24. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.



(a) Amoebot structure. **(b)** Reconfigurable circuit extension.

Figure 1 Amoebot model. The left figure shows an amoebot structure. The nodes indicate X. The red edges indicate E_X. The right figure shows an amoebot structure with $c = 2$ external links between adjacent amoebots. The nodes on the boundary are the pins. The nodes within the amoebots indicate the partition sets. An edge between a partition set PS and a pin p implies $p \in PS$. Each color indicates another circuit. The figures were taken from [8].

In this paper, we consider the *shortest path forest problem* where, given a set of sources and a set of destinations, the amoebots have to find a shortest path from each destination to the closest source. Shortest paths are a fundamental problem in both centralized and distributed systems and have also a number of important applications in the amoebot model.

For example, consider shape formation. Many algorithms for the amoebot model utilize a canonical shape, e.g., a line, as an intermediate structure [6, 11]. However, this is rather inefficient if the target structure is already close to the initial structure. For such cases, Kostitsyna et al. proposed an algorithm that utilizes shortest paths to move amoebots through the structure to their target positions [10]. Another application for shortest paths is energy distribution [4, 16]. The amoebots require energy to perform their movements that can be provided by other amoebots, e.g., amoebots located at external energy sources. In order to minimize energy loss, it is more efficient to transfer the energy via a shortest path.

2 Geometric Amoebot Model

The *(geometric) amoebot model* was proposed by Derakhshandeh et al. [5]. The model places a set of n anonymous finite state machines (called *amoebots*) on some graph $G = (V, E)$. Each amoebot occupies one node and every node is occupied by at most one amoebot. Let the *amoebot structure* $X \subseteq V$ be the set of nodes occupied by the amoebots. We assume that $G_X = (X, E_X)$ is connected, where $G_X = G_\Delta|_X$ is the graph induced by X. In the geometric variant of the model, G is the infinite regular triangular grid graph $G_\Delta = (V_\Delta, E_\Delta)$ (see Figure 1a).

3 Reconfigurable Circuit Extension

In the *reconfigurable circuit extension* [8], each edge between two neighboring amoebots u and v is replaced by c edges called *external links* with endpoints called *pins*, for some constant $c \geq 1$ that is the same for all amoebots. For each of these links, one pin is owned by u while the other pin is owned by v. In this paper, we assume that neighboring amoebots have a common labeling of their incident external links.

Each amoebot u *partitions* its *pin set* $PS(u)$ into a collection $\mathcal{C}(u)$ of pairwise disjoint subsets such that the union equals the pin set, i.e., $PS(u) = \bigcup_{C \in \mathcal{C}(u)} C$. We call $\mathcal{C}(u)$ the *pin configuration* of u and $C \in \mathcal{C}(u)$ a *partition set* of u. Let $\mathcal{C} = \bigcup_{u \in S} \mathcal{C}(u)$ be the collection of all partition sets in the system. Two partition sets are *connected* iff there is at least one

external link between those sets. Let L be the set of all connections between the partition sets in the system. Then, we call $H = (\mathcal{C}, L)$ the *pin configuration* of the system and any connected component C of H a *circuit* (see Figure 1b). Note that if each partition set of $\mathcal{C}$ is a *singleton*, i.e., a set with exactly one element, then every circuit of H just connects two neighboring amoebots. An amoebot is part of a circuit iff the circuit contains at least one of its partition sets. A priori, an amoebot u may not know whether two of its partition sets belong to the same circuit or not since initially it only knows $\mathcal{C}(u)$.

Each amoebot u can send a primitive signal (a *beep*) via any of its partition sets $C \in \mathcal{C}(u)$ that is received by all partition sets of the circuit containing C at the beginning of the next round. The amoebots are able to distinguish between beeps arriving at different partition sets. More specifically, an amoebot receives a beep at partition set C if at least one amoebot sends a beep on the circuit belonging to C, but the amoebots neither know the origin of the signal nor the number of origins.

We assume the fully synchronous activation model, i.e., the time is divided into synchronous rounds, and every amoebot is active in each round. On activation, each amoebot may update its state, reconfigure its pin configuration, and activate an arbitrary number of its partition sets. The beeps are propagated on the updated pin configurations. The time complexity of an algorithm is measured by the number of synchronized rounds required by it.

4 Problem Statement and Our Contribution

Let $S, D \subseteq X$ be two non-empty subsets. We call each amoebot in S a *source*, and each amoebot in D a *destination*. A (S, D)*-shortest path forest* is a set of rooted trees that satisfies the following properties.

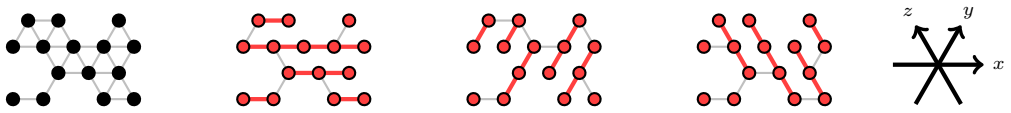
1. For each $s \in S$, the set contains a tree $T_s = (V_s, E_s)$ rooted at s with $V_s \subseteq X$ and $E_s \subseteq E_X$.
2. For each $s \in S$, each leaf of T_s is in $S \cup D$.
3. For each $s_1, s_2 \in S$, V_{s_1} and V_{s_2} are disjoint.
4. For each $u \in D$, there is a tree T_s such that $u \in V_s$, i.e., $D \subseteq \bigcup_{s \in S} V_s$.
5. For each $s \in S$ and $u \in V_s$, the unique path from s to u in T_s is a shortest path from s to u in G_X, and s has the smallest distance to u among all amoebots in S.

We call an (S, X)-shortest path forest also an S*-shortest path forest.*

We consider the (k, ℓ)*-shortest path forest problem* ((k, ℓ)*-SPF*) for $k, \ell \geq 1$. Let two sets $S, D \subseteq X$ of amoebots be given such that $|S| = k$ and $|D| = \ell$, i.e., each amoebot $u \in X$ knows whether $u \in S$ and whether $u \in D$. We say that X computes a (S, D)-shortest path forest if each amoebot in $\bigcup_{s \in S} V_s \setminus S$ knows its parent within the (S, D)-shortest path forest. The goal of the amoebot structure is to compute a (S, D)-shortest path forest.

Note that we obtain the classical *single pair shortest path problem (SPSP)* for $k = \ell = 1$, and the classical *single source shortest path problem (SSSP)* for $k = 1$ and $\ell = n$.

We will present two deterministic algorithms for (k, ℓ)-SPF. For hole-free amoebot structures, our *shortest path tree algorithm* solves the problem within $O(\log \ell)$ rounds for $k = 1$, and our *shortest path forest algorithm* within $O(\log n \log^2 k)$ for $k \geq 1$. Note that the former result implies that we can solve SPSP within $O(1)$ rounds, and SSSP within $O(\log n)$ rounds.



(a) Initial structure. **(b)** x-portal graph. **(c)** y-portal graph. **(d)** z-portal graph. **(e)** Axes.

Figure 2 Portal graphs. Each red connected component indicates a portal. We obtain the portal graphs by fusing the amoebots of each portal to a single node.

5 Related Work

The reconfigurable circuit extension was introduced by Feldmann et al. [8]. They proposed solutions for the leader election, compass alignment, and chirality agreement problems. Each of these solutions requires $O(\log n)$ rounds w.h.p. Afterwards, they considered the recognition of various classes of shapes. An amoebot structure is able to detect parallelograms with linear or polynomial side ratio within $O(\log n)$ rounds w.h.p. Further, an amoebot structure is able to detect shapes composed of triangles within $O(1)$ rounds if the amoebots agree on a chiracilty. Feldmann et al. proposed the PASC algorithm which allows the amoebot structure to compute distances along a path [8, 12]. With the help of it, Padalkin et al. were able to solve the global maxima, spanning tree, and symmetry detection problems in polylogarithmic time [12].

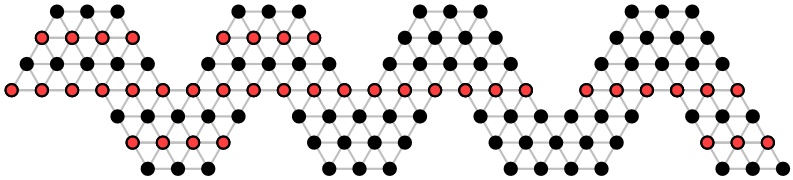
Shortest path problems are broadly studied both in the sequential and distributed setting. In distributed setting, adjacent nodes are able to communicate via messages. The CONGEST model limits the size of each message to a logarithmic number of bits (in n). For weighted SSSP, Chechik and Mukhtar proposed a randomized algorithm that takes $\tilde{O}(\sqrt{n}d^{1/4} + d)$ rounds [1]. The best known lower bound is $\Omega(\sqrt{n} + d)$ [7, 13].

In the amoebot model, Kostitsyna et al. were the first to consider SSSP [9, 10]. By applying a breadth-first search approach, they compute a shortest path tree within $O(n^2)$ rounds. For simple amoebot structures without holes, they introduced feather trees – a special type of shortest path trees. These can be computed within $O(d)$ rounds where d is the diameter of the structure. To our knowledge, there is no further work on shortest path problems in the amoebot model or its reconfigurable circuit extension.

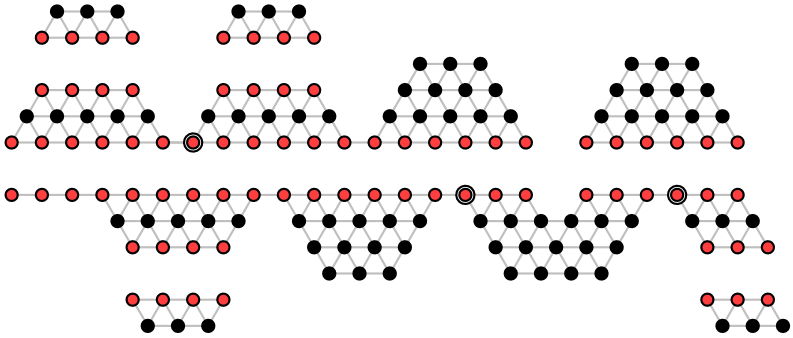
6 Results

Coy et al. [3] have solved the shortest path problem for hybrid communication networks that can be modelled as grid graphs without holes. Our shortest path tree algorithm for $k = 1$ adapts their approach as follows. The idea is to utilize portal graphs (see Figure 2). The vertices (called *d-portals*) of the *d-portal graph* are the connected components if we remove all edges that are not in parallel with the d-axis. Two d-portals are adjacent in the d-portal graph iff there exists an edge between them. We can show that for triangular grid graphs without holes, the distance between two nodes is half the sum of the distances of their portals in the portal graphs. Note that this equation does not hold anymore if the amoebot structure has holes. Further, it can be shown that the portal graph is a tree. This allows us to utilize the Euler tour technique [14] in combination with the PASC algorithm to compute distances in the portal graphs. We obtain the following result.

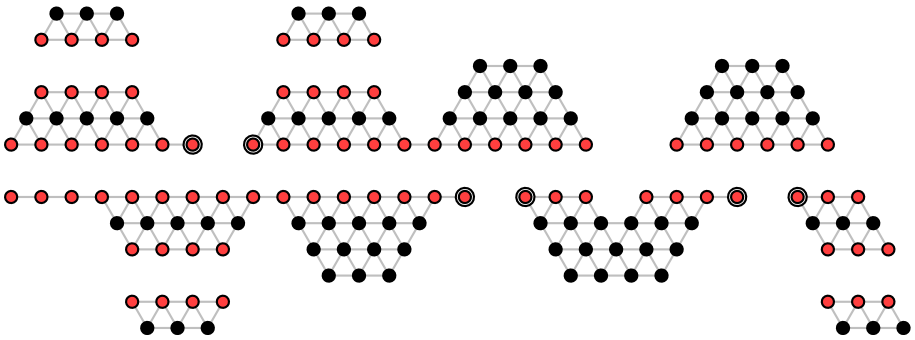
▶ **Theorem 6.1.** *The shortest path tree algorithm computes an $(\{s\}, D)$-shortest path forest within $O(\log \ell)$ rounds.*



(a) Initial amoebot structure.



(b) Regions after the first phase.



(c) Regions after the second phase.

Figure 3 Regions. The red amoebots indicate the splitting portals. The encircled amoebots indicate the splitting amoebots.

Our shortest path forest algorithm for $k \geq 1$ takes a divide and conquer approach. The idea is to split the amoebot structure into two regions, to recursively compute a shortest path forest for both regions, and to finally merge them. In the following, we will elaborate on these steps.

In the first step, we split the amoebot structure into smaller regions. For that, we make use of so-called *x-portals* [2], i.e., connected components of the intersections of the amoebot structure with an x-axis (see red amoebots in Figure 3a). Our goal is to split the amoebot structure until each region intersects at most two x-portals with at least one source. We split the amoebot structure in two phases. In the first phase, we split the amoebot structure at all portals with at least one source and at most Ok further portals (see Figure 3b). The splitting portal is part of both regions. In the second step, we split the amoebot structure further at amoebots at bottlenecks (see Figure 3c). The splitting amoebot is part of both regions. We can easily compute a shortest path forest for each resulting region. We omit the details for these computations. Each region will maintain a shortest path forest.

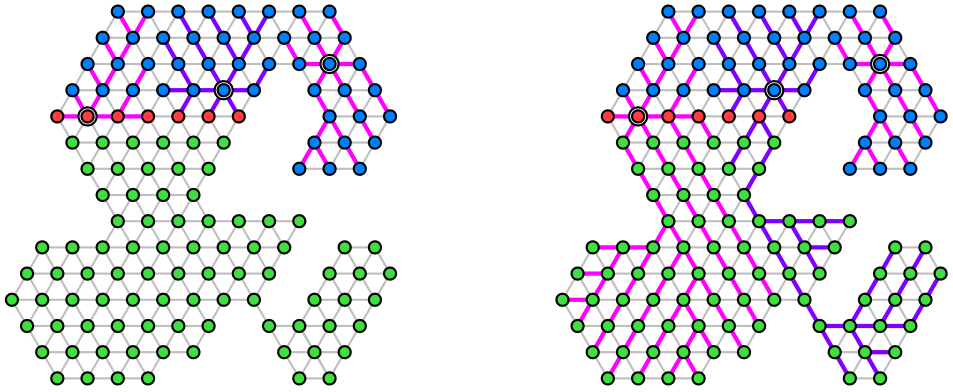


Figure 4 Propagation of the shortest path forest of the blue region into the green region. The left figure shows the initial situation, and the right figure the result. The red amoebots indicate the portal. The encircled amoebots indicate the sources. The pink and purple edges indicate the shortest path trees.

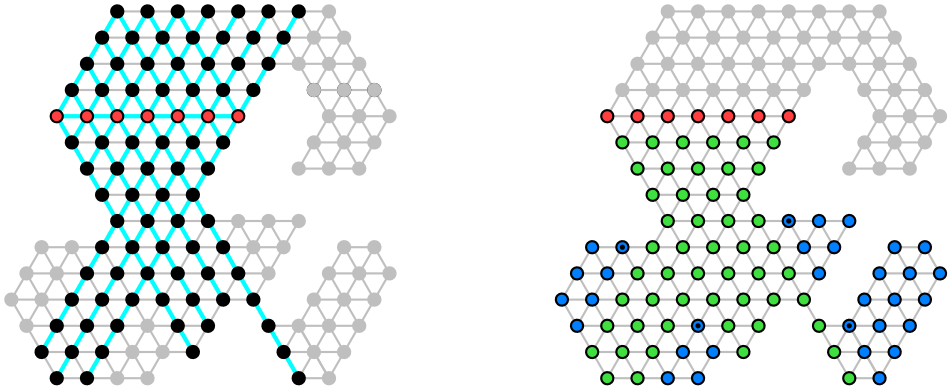


Figure 5 Phases of the propagation procedure. The red amoebots indicate the portal. The left figure indicates the amoebots visible by the portal. The right figure shows the phases. In the first phase, we propagate the shortest path forest to the green amoebots. In the second phase, we propagate the shortest path forest to the blue amoebots. We propagate the shortest path forests through the amoebots with a dot.

In order to merge two adjacent regions and with that their shortest path forests, we proceed as follows. We first propagate the shortest path forests of both regions into the other region, respectively (see Figure 4). The propagation happens in two phases. In the first phase, we propagate the shortest path forest to all amoebots *visible* by the splitting portal (see green amoebots in Figure 5). We can show that each amoebot can determine its parent by comparing the distances of two amoebots on the portal to their closest sources. In the second phase, we propagate the shortest path forest to the remaining amoebots, which may form several connected components. We propagate the shortest path forest into each of those independently of each other. For each connected component, we can propagate the shortest path forest through one of the its amoebots (see amoebots with a dot in Figure 5). This allows us to apply our shortest path tree algorithm for $k = 1$ with that amoebot as the source to finish the propagation (see Theorem 6.1).

Finally, we have to merge the resulting shortest path forests of both regions. By applying the PASC algorithm on each path from the source to a leaf in both shortest path forests,

each amoebots computes its distance to the closest source of both regions. This allows each amoebot to determine the closest source of both regions and with that to choose its parent in the merged shortest path forest.

Note that all steps (splitting, propagation, merging, etc.) make extensive use of the geometric properties of the amoebot structure, e.g., that it is free of holes. By utilizing $O(k)$ portals to split the amoebot structure into regions and by merging the regions with respect to a centroid decomposition tree, we obtain the following result.

▶ **Theorem 6.2.** *The shortest path forest algorithm computes an (S, D)-shortest path forest within $O(\log n \log^2 k)$ rounds.*

References

1 Shiri Chechik and Doron Mukhtar. Single-source shortest paths in the CONGEST model with improved bounds. *Distributed Comput.*, 35(4):357–374, 2022.

2 Sam Coy, Artur Czumaj, Michael Feldmann, Kristian Hinnenthal, Fabian Kuhn, Christian Scheideler, Philipp Schneider, and Martijn Struijs. Near-shortest path routing in hybrid communication networks. In *OPODIS*, volume 217 of *LIPIcs*, pages 11:1–11:23. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

3 Sam Coy, Artur Czumaj, Christian Scheideler, Philipp Schneider, and Julian Werthmann. Routing schemes for hybrid communication networks. In *SIROCCO*, volume 13892 of *Lecture Notes in Computer Science*, pages 317–338. Springer, 2023.

4 Joshua J. Daymude, Andréa W. Richa, and Jamison W. Weber. Bio-inspired energy distribution for programmable matter. In *ICDCN*, pages 86–95. ACM, 2021.

5 Zahra Derakhshandeh, Shlomi Dolev, Robert Gmyr, Andréa W. Richa, Christian Scheideler, and Thim Strothmann. Brief announcement: amoebot - a new model for programmable matter. In *SPAA*, pages 220–222. ACM, 2014.

6 Zahra Derakhshandeh, Robert Gmyr, Andréa W. Richa, Christian Scheideler, and Thim Strothmann. Universal shape formation for programmable matter. In *SPAA*, pages 289–299. ACM, 2016.

7 Michael Elkin. An unconditional lower bound on the time-approximation trade-off for the distributed minimum spanning tree problem. *SIAM J. Comput.*, 36(2):433–456, 2006.

8 Michael Feldmann, Andreas Padalkin, Christian Scheideler, and Shlomi Dolev. Coordinating amoebots via reconfigurable circuits. *J. Comput. Biol.*, 29(4):317–343, 2022.

9 Irina Kostitsyna, Tom Peters, and Bettina Speckmann. Brief announcement: An effective geometric communication structure for programmable matter. In *DISC*, volume 246 of *LIPIcs*, pages 47:1–47:3. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.

10 Irina Kostitsyna, Tom Peters, and Bettina Speckmann. Fast reconfiguration for programmable matter. In *DISC*, volume 281 of *LIPIcs*, pages 27:1–27:21. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023.

11 Giuseppe Antonio Di Luna, Paola Flocchini, Nicola Santoro, Giovanni Viglietta, and Yukiko Yamauchi. Shape formation by programmable particles. *Distributed Comput.*, 33(1):69–101, 2020.

12 Andreas Padalkin, Christian Scheideler, and Daniel Warner. The structural power of reconfigurable circuits in the amoebot model. In *DNA*, volume 238 of *LIPIcs*, pages 8:1–8:22. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.

13 Atish Das Sarma, Stephan Holzer, Liah Kor, Amos Korman, Danupon Nanongkai, Gopal Pandurangan, David Peleg, and Roger Wattenhofer. Distributed verification and hardness of distributed approximation. *SIAM J. Comput.*, 41(5):1235–1265, 2012.

14 Robert Endre Tarjan and Uzi Vishkin. An efficient parallel biconnectivity algorithm. *SIAM J. Comput.*, 14(4):862–874, 1985.

15 Tommaso Toffoli and Norman Margolus. Programmable matter: Concepts and realization. *Int. J. High Speed Comput.*, 5(2):155–170, 1993.

16 Jamison W. Weber, Tishya Chhabra, Andréa W. Richa, and Joshua J. Daymude. Energy-constrained programmable matter under unfair adversaries. *CoRR*, abs/2309.04898, 2023.

Reconfiguration and Locomotion with Joint Movements in the Amoebot Model*

Andreas Padalkin[1], Manish Kumar[2], and Christian Scheideler[3]

1 **Paderborn University, Germany**
andreas.padalkin@upb.de
2 **Bar-Ilan University, Israel**
manish.kumar@biu.ac.il
3 **Paderborn University, Germany**
scheideler@upb.de

Abstract

We are considering the geometric amoebot model where a set of n *amoebots* is placed on the triangular grid. An amoebot is able to send information to its neighbors, and to move via expansions and contractions. Since amoebots and information can only travel node by node, most problems relevant for programmable matter have a natural lower bound of $\Omega(D)$ where D denotes the diameter of the structure. Inspired by the nervous and muscular system, Feldmann et al. have proposed the *reconfigurable circuit extension* and the *joint movement extension* of the amoebot model with the goal of breaking this lower bound.

In the joint movement extension, the way amoebots move is altered. Amoebots become able to push and pull other amoebots. Feldmann et al. demonstrated the power of joint movements by transforming a line of amoebots into a rhombus within $O(\log n)$ rounds. However, they left the details of the extension open. The goal of this paper is therefore to formalize the joint movement extension. In order to provide a proof of concept for the extension, we consider two fundamental problems of modular robot systems: *reconfiguration* and *locomotion*.

We approach these problems by defining meta-modules of rhombical and hexagonal shape, respectively. The meta-modules are capable of movement primitives like sliding, rotating, and tunneling. This allows us to simulate reconfiguration algorithms of various modular robot systems. Finally, we construct three amoebot structures capable of locomotion by rolling, crawling, and walking, respectively.

Related Version *Full Version*: https://arxiv.org/abs/2305.06146

1 Introduction

Programmable matter consists of homogeneous nano-robots that are able to change the properties of the matter in a programmable fashion, e.g., the shape, the color, or the density [18]. We are considering the geometric amoebot model [3, 4, 5] where a set of n nano-robots called *amoebots* is placed on the triangular grid. An amoebot is able to send information to its neighbors, and to move by first expanding into an unoccupied adjacent node, and then contracting into that node. Since amoebots and information can only travel node by node, most problems relevant for programmable matter have a natural lower bound of $\Omega(D)$ where D denotes the diameter of the structure. Inspired by the *nervous* and *muscular system*, Feldmann et al. [9] proposed the *reconfigurable circuit extension* and the *joint movement extension* with the goal of breaking this lower bound.

* This work has been supported by the DFG Project SCHE 1592/10-1 and Israel Science Foundation under Grants 867/19 and 554/23.

40th European Workshop on Computational Geometry, Ioannina, Greece, March 13–15, 2024.
This is an extended abstract of a presentation given at EuroCG'24. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

In the reconfigurable circuit extension, the amoebot structure is able to interconnect amoebots by *circuits*. Each amoebot can send a primitive signal on circuits it is connected to. The signal is received by all amoebots connected to the same circuit. Among others, Feldmann et al. [9] solved the leader election problem, compass alignment problem, and chirality agreement problem within $O(\log n)$ rounds with high probability (w.h.p.). These problems will be important to coordinate the joint movements. Afterwards, Padalkin et al. [16] explored the structural power of the circuits by considering the spanning tree problem and symmetry detection problem. Both problems can be solved within polylogarithmic time w.h.p.

In the joint movement extension, the way amoebots move is altered. In a nutshell, an expanding amoebot is capable of pushing other amoebots away from it, and a contracting amoebot is capable of pulling other amoebots towards it. Feldmann et al. [9] demonstrated the power of joint movements by transforming a line of amoebots into a rhombus within $O(\log n)$ rounds. However, they left the details of the extension open. The goal of this paper is therefore to formalize the joint movement extension. In order to provide a proof of concept for the extension, we consider two fundamental problems of modular robot systems (MRS): reconfiguration and locomotion. We study these problems from a centralized view to explore the limits of the extension.

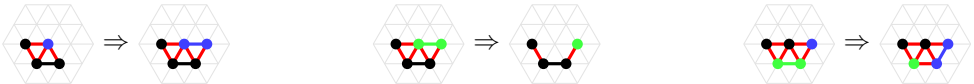
In the reconfiguration problem, an MRS has to reconfigure its structure into a given shape. Examples for reconfiguration algorithms in the original amoebot model can be found in [6, 7, 14, 15]. However, all of these are subject of the aforementioned natural lower bound. To our knowledge, polylogarithmic time solutions were found for two types of MRSs: in the nubot model [20] and crystalline atom model [2].

In the locomotion problem, an MRS has to move along an even surface as fast as possible. In the original amoebot model, one would use the spanning tree primitive to move along the surface [3]. However, we only obtain a constant velocity with that. In terrestrial environments, there are three basic types of locomotion: rolling, crawling, and walking [11, 13]. For each of these locomotion types, we will present an amoebot structure.

2 Geometric Amoebot Model

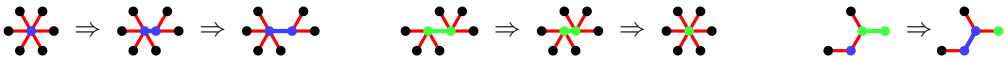
In this section, we introduce the *geometric amoebot model* [4]. We slightly deviate from the original model to make it more suitable to our extension. A set of n amoebots is placed on the infinite regular triangular grid graph $G_\Delta = (V, E)$ (see Figure 1). An amoebot is an anonymous, randomized finite state machine in form of a line segment. The endpoints may either occupy the same node or two adjacent nodes. If the endpoints occupy the same node, the amoebot has length 0 and is called *contracted* and otherwise, it has length 1 and is called *expanded*. Every node of G_Δ is occupied by at most one amoebot. Two endpoints of different amoebots that occupy adjacent nodes in G_Δ are connected by *bonds* (red edges). An amoebot can move through *contractions* and *expansions*. We refer to [4] for more details.

Let the *amoebot structure* $S \subseteq V$ be the set of nodes occupied by the amoebots. We say



(a) Expansion. **(b)** Contraction. **(c)** Handover.

Figure 1 Movement in the geometric amoebot model. Red lines indicate bonds. Blue amoebots are expanding. Green amoebots are contracting.



(a) Expansion. **(b)** Contraction. **(c)** Handover.

Figure 2 Movements in the extension. Red lines indicate bonds. Blue amoebots are expanding. Green amoebots are contracting. The first two figures show a movement in 0.5 time steps.

that S is *connected* if and only if G_S is connected, where $G_S = G_\Delta|_S$ is the graph induced by S. Initially, S is connected. Also, we assume the fully synchronous activation model, i.e., the time is divided into synchronous rounds, and every amoebot is active in each round. We justify this assumption with the reconfigurable circuit extension. On activation, each amoebot may perform a movement and update its state as a function of its previous state. However, if an amoebot fails to perform its movement, it remains in its previous state. The time complexity of an algorithm is measured by the number of synchronized rounds required by it.

3 Joint Movement Extension

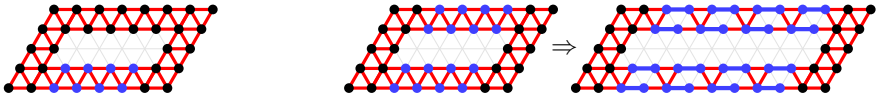
In the *joint movement extension* [9], the way the amoebots move is altered. The idea behind the extension is to allow amoebots to push and pull other amoebots. The necessary coordination of such movements can be provided by the reconfigurable circuit extension [9, 16]. In the following, we formalize the joint movement extension. Joint movements are performed in two steps.

In the first step, the amoebots remove bonds from G_S as follows. Each amoebot can decide to release an arbitrary subset of its currently incident bonds in G_S. A bond is removed if and only if either of the amoebots at the endpoints releases the bond. Let $E_L \subseteq E_S$ denote the set of all edges occupied by amoebots and $E_R \subseteq E_S$ the set of the remaining bonds, and $G_R = (S, E_L \cup E_R)$ the resulting graph. We require that G_R is connected since otherwise, disconnected parts might float apart. We say that a *connectivity conflict* occurs if and only if G_R is not connected. Whenever a connectivity conflict occurs, the amoebot structure transitions into an undefined state such that we become unable to make any statements about the structure.

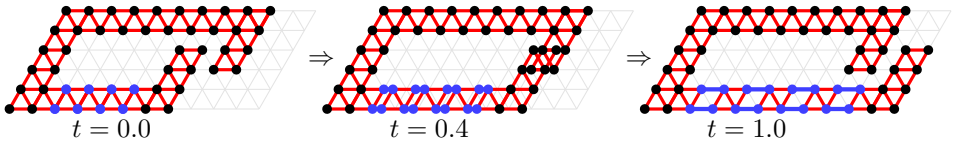
In the second step, each amoebot may perform one of the following movements within the time period $[0, 1]$. A contracted amoebot may expand on one of the axes as follows (see Figure 2a). At $t = 0$, the amoebot can reorientate itself and reassign each of its incident bonds to one of its endpoints. At $t \in [0, 1]$, the amoebots has a length of t. In the process, the incident bonds do not change their orientations or lengths. As a result, the expanding amoebot pushes all connected amoebots. An expanded amoebot may contract analogously by reversing the contraction (see Figure 2b). Thereby, it pulls all connected amoebots.

Furthermore, a contracted amoebot x occupying node u and an expanded amoebot y occupying nodes v and w may perform a handover if there is a bond b between u and v, as follows (see Figure 2c). At an arbitrary $t \in [0, 1]$, we flip $b = \{u, v\}$ and $\{v, w\}$ such that x becomes an expanded amoebot with endpoints occupying nodes u and v, y becomes a contracted amoebot with both endpoints occupying w, and b becomes $\{v, w\}$. We have to include the handover to ensure universality of the model since otherwise, it would not be possible to move through a narrow tunnel.

In certain situations, the amoebots may not be able to perform their movements. We distinguish between two cases. First, the amoebots may not be able to perform their movements while maintaining their relative positions (see Figure 3a). We call that a *structural*



(a) Structural conflicts and mappings. In the left figure, the expansions cause a structural conflict. In the right figure, the amoebot structure can be mapped onto the triangular grid after the movement.



(b) Collision. Initially, we have a valid amoebot structure given ($t = 0$). The resulting amoebot structure can be mapped on G_Δ ($t = 1$). However, for $t \in [0.25, 0.75]$, parts of the structure collide.

Figure 3 Joint movements. Red lines indicate bonds. Blue amoebots are expanding horizontally.

conflict. Second, parts of the structure may collide into each other. More precisely, a *collision* occurs if there is a $t \in [0, 1]$ such that two non-adjacent bonds intersect at some point (see Figure 3b). Whenever either a structural conflict or a collision occurs, the amoebot structure transitions into an undefined state such that we become unable to make any statements about the structure. The detection of structural conflicts and collisions is not within the scope of this paper simply because we only consider movements where structural conflicts and collisions cannot occur. We refer to [10] for more details.

Otherwise, at $t = 1$, we obtain a graph $G_M = (S, E_M)$ that can be mapped on the triangular grid G_Δ (see Figure 3a). In compliance with the orientations of all bonds and line segments, the mapping of G_M is unique except for translations since G_R is connected. We choose any mapping as our next amoebot structure. Afterwards, the amoebots reestablish all possible bonds.

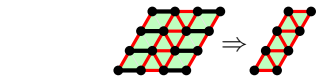
We assume that the joint movements are performed within *look-compute-move* cycles. In the *look phase*, each amoebot observes its neighborhood and receives signals (beeps) from other amoebots, according to the reconfigurable circuit structure of the system. In the *compute phase*, each amoebot may perform computations, change its state, and decide the actions to perform in the next phase (i.e., which bonds to release, and which movement to perform). In the *move phase*, each amoebot may release an arbitrary subset of its incident bonds, and perform a movement.

4 Proof of Concept

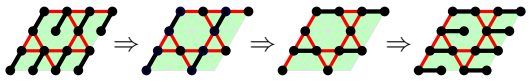
We now provide a proof of concept for the joint movement extension by considering two fundamental problems: reconfiguration and locomotion.

In a first step, we combine multiple amoebots to meta-modules. In other models for programmable matter and modular robots, meta-modules have proven to be very useful. For example, they allow us to bypass restrictions on the reconfigurability [8, 19] and to simulate (reconfiguration) algorithms for other models [1, 17]. We present two types of meta-modules: meta-modules of rhombical and hexagonal shape. For these, we can show various movement primitives (see Figures 4 and 5).

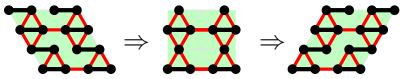
The meta-modules allow us to simulate reconfiguration algorithms for lattice-type MRSs of similar shape if we can implement the same movement primitives. By simulating the reconfiguration algorithm of Aloupis et al. [2] for rhombical robots (crystalline atoms) and the reconfiguration algorithm of Hurtado et al. [12] for hexagonal robots, we obtain the



(a) Contraction primitive.

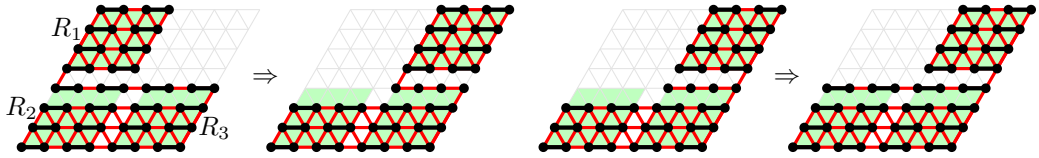


(b) Reorientation primitive.

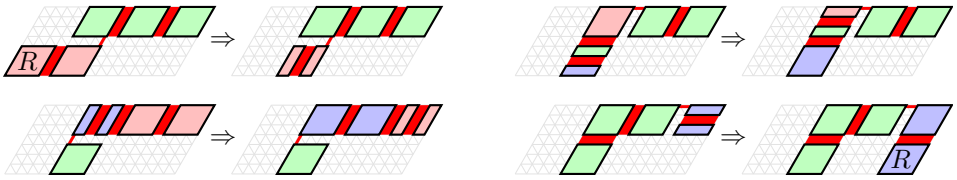


(c) Realignment primitive.

(d) Rotation primitive.



(e) Slide primitive.



(f) k-tunnel primitive.

Figure 4 Movement primitives for rhombical meta-modules. Red meta-modules perform a pull operation, and blue meta-modules a push operation.

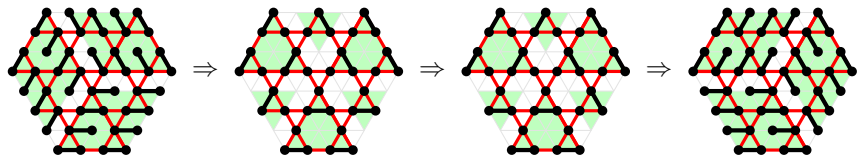
following results.

▶ **Theorem 4.1.** *There is a centralized reconfiguration algorithm for m hexagonal meta-modules that requires $O(m)$ rounds. Each module has to perform at most $O(m)$ moves.*

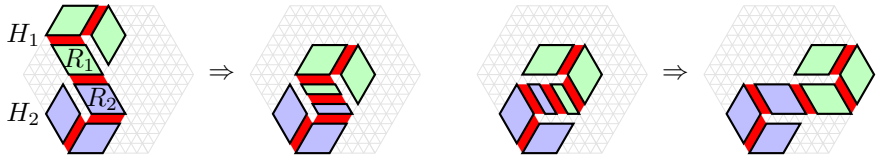
▶ **Theorem 4.2.** *There is a centralized reconfiguration algorithm for m rhombical meta-modules that requires $O(\log m)$ rounds and performs $\Theta(m \log m)$ moves overall.*

Finally, we consider amoebot structures capable of locomotion along an even surface. There are three basic types of terrestrial locomotion: rolling, crawling, and walking [11, 13]. In the following, we will construct an amoebot structure for each type.

Our rolling amoebot structure imitates a continuous track that rotates around a set of wheels. We build it from hexagonal meta-modules of alternating side lengths ℓ and $\ell - 1$



(a) Switching primitive.



(b) Rotation primitive.

Figure 5 Movement primitives for hexagonal meta-modules.

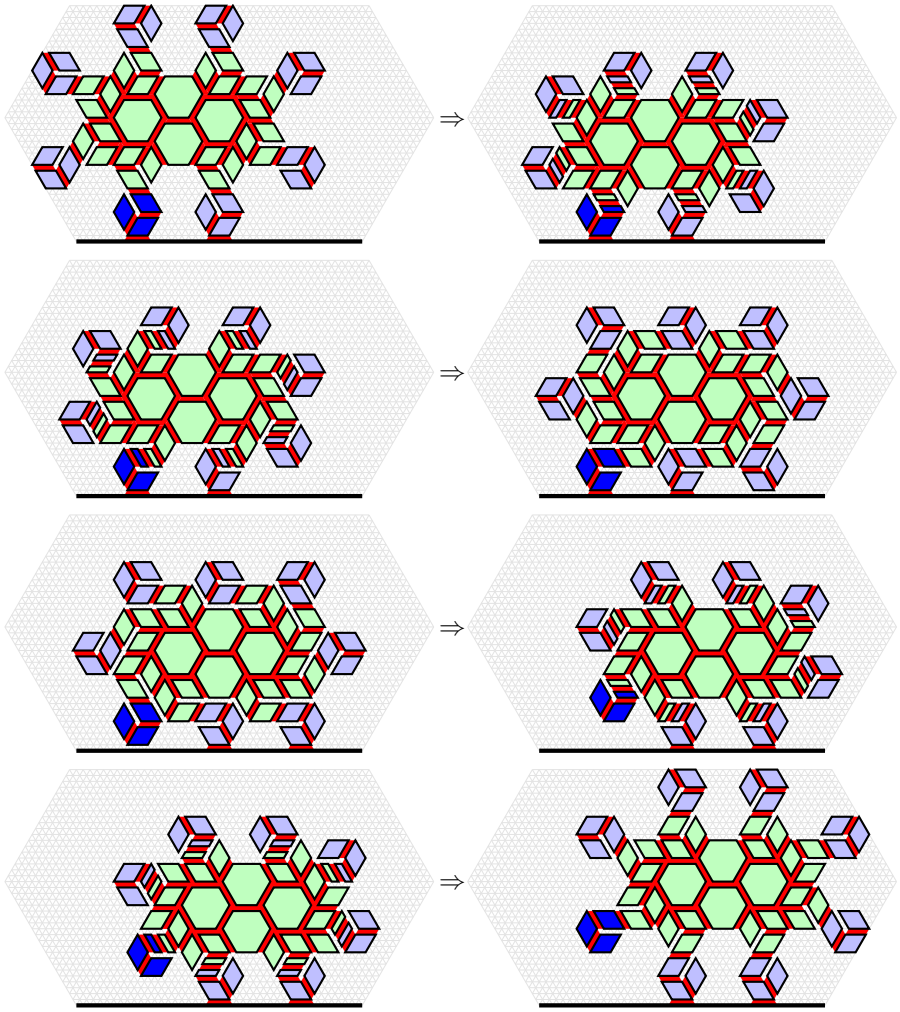


Figure 6 Rolling structure. The blue meta-modules rotate clockwise around the green meta-modules. We highlight one of the rotating meta-modules in a darker blue.

(see Figure 6). The structure consists of two parts: a connected substrate structure (green meta-modules), and a closed chain of meta-modules rotating along the outer boundary of the substrate (blue meta-modules). The amoebot structure moves by rotating the blue meta-modules around the substrate (compare to Figure 5b). We obtain the initial structure after two rotations. In doing so, the structure has moved a distance of $2 \cdot \ell$. By performing the movements periodically, we obtain the following theorem.

▶ **Theorem 4.3.** *Our rolling structure composed of hexagonal meta-modules of alternating side lengths ℓ and $\ell - 1$ moves a distance of $2 \cdot \ell$ within each period of constant length.*

Our crawling amoebot structure imitates earthworms. It consists of r rhombical meta-modules of side length $\ell - 1$ (see Figure 7). The amoebot structure moves by alternately contacting and expanding its body (compare to Figure 4a) while utilizing the meta-module at the front and at the end as an anchor, respectively. As a result, the contraction (expansion) pulls (pushes) the structure to the front. By performing the movements periodically, we obtain the following theorem.

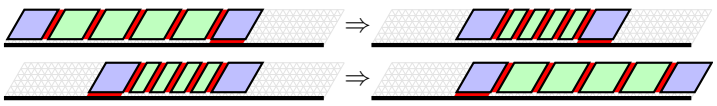


Figure 7 Crawling structure.

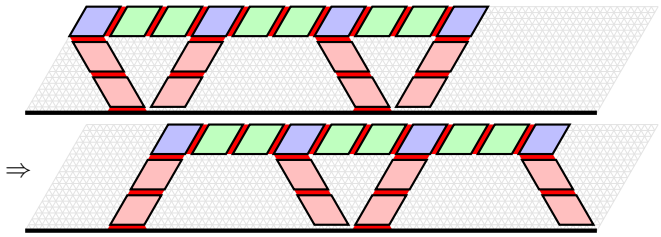


Figure 8 Walking structure.

▶ **Theorem 4.4.** *A line of r rhombical meta-modules of side length $\ell - 1$ moves a distance of $\frac{r-2}{2} \cdot \ell$ every 2 rounds.*

Our walking amoebot structure imitates millipedes. It consists of rhombical meta-modules of side length $\ell - 1$ (compare to Figure 8). Let p denote the number of legs. The body and each leg consists of a line of q rhombical meta-modules. The structure moves by moving the legs back and forth. For that, we simply apply the realignment primitive (see Figure 4c) on all meta-modules within the legs. Note that we reach the initial amoebot structure after two leg movements. Hence, we obtain the following theorem.

▶ **Theorem 4.5.** *Our walking structure composed of rhombical meta-modules of side length $\ell - 1$ with p legs composed of q rhombical meta-modules moves a distance of $2 \cdot q \cdot \ell$ within each period of constant length.*

References

1 Greg Aloupis, Nadia M. Benbernou, Mirela Damian, Erik D. Demaine, Robin Y. Flatland, John Iacono, and Stefanie Wuhrer. Efficient reconfiguration of lattice-based modular robots. *Comput. Geom.*, 46(8):917–928, 2013.

2 Greg Aloupis, Sébastien Collette, Erik D. Demaine, Stefan Langerman, Vera Sacristán Adinolfi, and Stefanie Wuhrer. Reconfiguration of cube-style modular robots using O(log n) parallel moves. In *ISAAC*, volume 5369 of *Lecture Notes in Computer Science*, pages 342–353. Springer, 2008.

3 Joshua J. Daymude, Kristian Hinnenthal, Andréa W. Richa, and Christian Scheideler. Computing by programmable particles. In *Distributed Computing by Mobile Entities*, volume 11340 of *Lecture Notes in Computer Science*, pages 615–681. Springer, 2019.

4 Joshua J. Daymude, Andréa W. Richa, and Christian Scheideler. The canonical amoebot model: Algorithms and concurrency control. In *DISC*, volume 209 of *LIPIcs*, pages 20:1–20:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

5 Zahra Derakhshandeh, Shlomi Dolev, Robert Gmyr, Andréa W. Richa, Christian Scheideler, and Thim Strothmann. Brief announcement: amoebot - a new model for programmable matter. In *SPAA*, pages 220–222. ACM, 2014.

6 Zahra Derakhshandeh, Robert Gmyr, Andréa W. Richa, Christian Scheideler, and Thim Strothmann. An algorithmic framework for shape formation problems in self-organizing particle systems. In *NANOCOM*, pages 21:1–21:2. ACM, 2015.

7 Zahra Derakhshandeh, Robert Gmyr, Andréa W. Richa, Christian Scheideler, and Thim Strothmann. Universal shape formation for programmable matter. In *SPAA*, pages 289–299. ACM, 2016.

8 Daniel J. Dewey, Michael P. Ashley-Rollman, Michael DeRosa, Seth Copen Goldstein, Todd C. Mowry, Siddhartha S. Srinivasa, Padmanabhan Pillai, and Jason Campbell. Generalizing metamodules to simplify planning in modular robotic systems. In *IROS*, pages 1338–1345. IEEE, 2008.

9 Michael Feldmann, Andreas Padalkin, Christian Scheideler, and Shlomi Dolev. Coordinating amoebots via reconfigurable circuits. *J. Comput. Biol.*, 29(4):317–343, 2022.

10 Siddharth Gupta, Marc J. van Kreveld, Othon Michail, and Andreas Padalkin. Collision detection for modular robots - it is easy to cause collisions and hard to avoid them. *CoRR*, abs/2305.01015, 2023.

11 Shigeo Hirose. Three basic types of locomotion in mobile robots. In *Fifth International Conference on Advanced Robotics' Robots in Unstructured Environments*, pages 12–17. IEEE, 1991.

12 Ferran Hurtado, Enrique Molina, Suneeta Ramaswami, and Vera Sacristán Adinolfi. Distributed reconfiguration of 2d lattice-based modular robotic systems. *Auton. Robots*, 38(4):383–413, 2015.

13 Matthew Kehoe and Davide Piovesan. Taxonomy of two dimensional bio-inspired locomotion systems. In *EMBC*, pages 3703–3706. IEEE, 2019.

14 Irina Kostitsyna, Christian Scheideler, and Daniel Warner. Fault-tolerant shape formation in the amoebot model. In *DNA*, volume 238 of *LIPIcs*, pages 9:1–9:22. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.

15 Giuseppe Antonio Di Luna, Paola Flocchini, Nicola Santoro, Giovanni Viglietta, and Yukiko Yamauchi. Shape formation by programmable particles. *Distributed Comput.*, 33(1):69–101, 2020.

16 Andreas Padalkin, Christian Scheideler, and Daniel Warner. The structural power of reconfigurable circuits in the amoebot model. In *DNA*, volume 238 of *LIPIcs*, pages 8:1–8:22. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.

17 Irene Parada, Vera Sacristán, and Rodrigo I. Silveira. A new meta-module for efficient reconfiguration of hinged-units modular robots. In *ICRA*, pages 5197–5202. IEEE, 2016.

18 Tommaso Toffoli and Norman Margolus. Programmable matter: Concepts and realization. *Int. J. High Speed Comput.*, 5(2):155–170, 1993.

19 Sergei Vassilvitskii, Jeremy Kubica, Eleanor Gilbert Rieffel, John W. Suh, and Mark Yim. On the general reconfiguration problem for expanding cube style modular robots. In *ICRA*, pages 801–808. IEEE, 2002.

20 Damien Woods, Ho-Lin Chen, Scott Goodfriend, Nadine Dabby, Erik Winfree, and Peng Yin. Active self-assembly of algorithmic shapes and patterns in polylogarithmic time. In *ITCS*, pages 353–354. ACM, 2013.

Range Reporting for Time Series via Rectangle Stabbing*

Lotte Blank[1] and Anne Driemel[1]

1 **Department of Computer Science, University of Bonn, Germany**

Abstract

We study the Fréchet queries problem. It is a data structure problem, where we are given a set S of n polygonal curves and a distance threshold ρ. The data structure should support queries with a polygonal curve q for the elements of S, for which the continuous Fréchet distance to q is at most ρ. We study the case that the ambient space of the curves is 1-dimensional and show an intimate connection to the well-studied rectangle stabbing problem. Using known data structures for rectangle stabbing or orthogonal range searching this directly leads to a data structure with size in $\mathcal{O}(n \log^{t-1} n)$ and query time in $\mathcal{O}(\log^{t-1} n + k)$, where k denotes the output size and t can be chosen as the maximum number of vertices of either (a) the stored curves or (b) the query curves. Note that we omit factors depending on the complexity of the curves that do not depend on n.

Related Version arXiv:2401.03762

1 Introduction

The *Fréchet distance* is a popular measure of similarity of two curves q and s. We focus on a data structuring problem which we refer to as the *Fréchet queries problem*. Here, in the preprocessing phase, we are given a set S of n polygonal curves of complexity at most t_s, a distance threshold ρ, and the complexity t_q of the query time series. The task is to store this set in a data structure that can answer the following type of queries efficiently: For a polygonal curve q of complexity t_q, output all curves in S that have Fréchet distance at most ρ to q. We denote with the *complexity* of a curve the number of vertices that defines it. Afshani and Driemel [2] studied this problem in 2018 for 2-dimensional curves providing non-trivial upper and lower bounds for the exact case. Their data structure is based on multi-level partition trees using semi-algebraic range searching and has size in $\mathcal{O}\left(n(\log\log n)^{\mathcal{O}({t_s}^2)}\right)$ and uses query time in $\mathcal{O}\left(\sqrt{n} \cdot \log^{\mathcal{O}({t_s}^2)} n + k\right)$, where k is the output size and t_s and t_q are assumed to be constant. Recently, Cheng and Huang [6] have generalized their approach for higher dimensions. Other works on variants of this problem have focused on the approximate setting [4, 7, 8, 9, 10]. We study the exact setting and—following previous work by Bringmann, Driemel, Nusser and Psarros [4] and Driemel and Psarros [8]—we restrict the ambient space of the curves to be 1-dimensional, that is, they are time series.

Preliminaries For any two points $p_1, p_2 \in \mathbb{R}^d$, $\overline{p_1 p_2}$ is the directed line segment from p_1 to p_2. The linear interpolation of each pair of consecutive vertices of a sequence of vertices $s_1, \ldots, s_{t_s} \in \mathbb{R}^d$ is called a polygonal curve. This curve is also denoted as $\langle s_1, \ldots, s_{t_s} \rangle$. We can represent polygonal curves as functions $s : [1, t_s] \to \mathbb{R}^d$, where $s(i+\alpha) = (1-\alpha)s_i + \alpha s_{i+1}$

* This work was funded by 390685813 (Germany's Excellence Strategy – EXC-2047/1: Hausdorff Center for Mathematics); 416767905; and the Deutsche Forschungsgemeinschaft (DFG, German Research-Foundation) – 459420781 (FOR AlgoForGe)

40th European Workshop on Computational Geometry, Ioannina, Greece, March 13–15, 2024.
This is an extended abstract of a presentation given at EuroCG'24. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

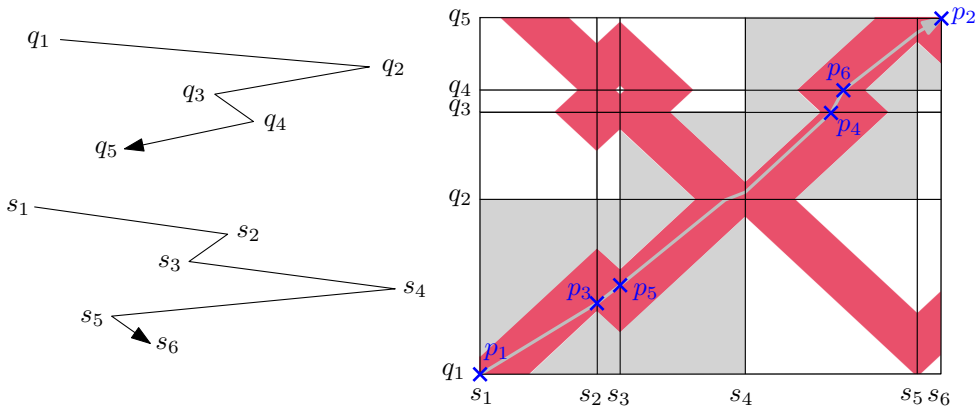


Figure 1 The free space diagram $F_\rho(q, s)$ of two time series with a feasible path trough a feasible sequence of cells $\mathcal{C} = ((1,1),(1,2),(1,3),(2,3),(2,4),(3,4),(4,4),(4,5))$, which is drawn in grey. Predicates $(P_1),(P_2),(P_3(1,2)),(P_4(3,4)),(P_5(1,2,3))$ and $(P_6(3,4,4))$ are true, because the points p_i are contained in the free space.

for $i \in \{1,\ldots,t_s-1\}$ and $\alpha \in [0,1]$. The *(continuous) Fréchet distance* between polygonal curves $q : [1,t_q] \to \mathbb{R}^d$ and $s : [1,t_s] \to \mathbb{R}^d$ is defined as

$$d_{\mathrm{F}}(q,s) = \inf_{h_q \in \mathcal{F}_q, h_s \in \mathcal{F}_s} \max_{p \in [0,1]} \|q(h_q(p)) - s(h_s(p))\|_2,$$

where $\mathcal{F}_q$ is the set of all continuous, non-decreasing functions $h_q : [0,1] \to [1,t_q]$ with $h_q(0) = 1$ and $h_q(1) = t_q$, respectively $\mathcal{F}_s$ for s.

We show an intimate connection of the Fréchet queries problem to the following classical problems studied in computational geometry. For *rectangle stabbing*, a set S of n axis-aligned d-dimensional rectangles in $\mathbb{R}^d$ needs to be preprocessed into a data structure so that all rectangles in S containing a query point q can be reported efficiently, ensuring that each such rectangle is reported exactly once. *Orthogonal range searching* is its dual. Here, a set S of n points in $\mathbb{R}^d$ is preprocessed into a data structure so that for a d-dimensional axis-aligned query rectangle R all points contained in S can be reported efficiently, ensuring that each such point is reported exactly once.

2 Predicates for Evaluating the Fréchet distance

In this section, we review the predicates used by Afshani and Driemel and how they enable the evaluation of the Fréchet distance in a data structure context. For this, we first recall the definition of the free space diagram from Alt and Godau [3]. For time series $q : [1,t_q] \to \mathbb{R}$ and $s : [1,t_s] \to \mathbb{R}$, the set $F_\rho(q,s) := \{(x,y) \in [1,t_q] \times [1,t_s] \mid |q(x) - s(y)| \leq \rho\}$ is called *free space diagram*. Refer to Figure 1 for an example. They showed that there exists a path in $F_\rho(q,s)$ from $(1,1)$ to (t_q,t_s) which is monotone in both coordinates if and only if $d_F(q,s) \leq \rho$. For such a path, we say it is *feasible*.

We can decompose the rectangle $[1,t_q] \times [1,t_s]$ into $(t_q-1)\cdot(t_s-1)$ cells such that the cell $C_{ij} = [i,i+1] \times [j,j+1]$ corresponds to the part in the free space diagram defined by the edges $\overline{q_i q_{i+1}}$ and $\overline{s_j s_{j+1}}$. By definition of the free space diagram, it follows that $C_{ij} \cap F_\rho(q,s)$ lies between two parallel lines. Therefore, we focus on the boundary of the cells C_{ij}.

Our query algorithm will iterate over all possibilities of sequences of cells that a feasible path could traverse in the free space diagram. In light of this, we call a sequence of cells

$\mathcal{C} = ((i_1, j_1), \dots, (i_t, j_t))$ *valid*, if $(i_1, j_1) = (1, 1)$, $(i_t, j_t) = (t_q - 1, t_s - 1)$, and $(i_{m+1}, j_{m+1}) \in \{(i_m, j_m + 1), (i_m + 1, j_m)\}$ for all $m < t$. The tuple (i, j) represents the cell C_{ij}. Further, a valid sequence of cells is called *feasible* in $F_\rho(q, s)$, if there exists a feasible path in $F_\rho(q, s)$ traversing exactly the cells in $\mathcal{C}$. The following predicates due to Afshani and Driemel [2] can be used to decide whether a valid sequence of cells is feasible in $F_\rho(q, s)$. See Figure 1 for an example.

$(\boldsymbol{P_1})$ *(Endpoint (start))* This predicate is true iff $|s_1 - q_1| \leq \rho$.
$(\boldsymbol{P_2})$ *(Endpoint (end))* This predicate is true iff $|s_{t_s} - q_{t_q}| \leq \rho$.
$(\boldsymbol{P_3(i,j)})$ *(Vertex of s - edge of q)* This predicate is true iff $\exists\ p_3 \in \overline{q_i q_{i+1}}$ s.t. $|p_3 - s_j| \leq \rho$.
$(\boldsymbol{P_4(i,j)})$ *(Vertex of q - edge of s)* This predicate is true iff $\exists\ p_4 \in \overline{s_j s_{j+1}}$ s.t. $|p_4 - q_i| \leq \rho$.
$(\boldsymbol{P_5(i,j,k)})$ *(Monotone in q)* This predicate is true iff $\exists\ p_3, p_5 \in \overline{q_i q_{i+1}}$ s.t. p_3 lies not after p_5 on the time series q and $|p_3 - s_j| \leq \rho$ and $|p_5 - s_k| \leq \rho$.
$(\boldsymbol{P_6(i,l,j)})$ *(Monotone in s)* This predicate is true iff $\exists\ p_4, p_6 \in \overline{s_j s_{j+1}}$ s.t. p_4 lies not after p_6 on the time series s and $|p_4 - q_i| \leq \rho$ and $|p_6 - q_l| \leq \rho$.

The following lemma verifies that the predicates can be used to test if the Fréchet distance between two curves is at most a given value.

▶ **Lemma 2.1** (Afshani and Driemel [2]). *Let $\mathcal{C} = ((i_1, j_1), (i_2, j_2), \dots, (i_t, j_t))$ be a valid sequence of cells. Then $\mathcal{C}$ is feasible in $F_\rho(q, s)$ if and only if the following predicates defined by q, s and ρ are true: (P_1) and (P_2) and $(P_3(i, j))$ if $(i, j-1), (i, j) \in \mathcal{C}$ and $(P_4(i, j))$ if $(i-1, j), (i, j) \in \mathcal{C}$ and $(P_5(i, j, k))$ if $(i, j-1), (i, k) \in \mathcal{C}$ for $j < k$ and $(P_6(i, l, j))$ if $(i-1, j), (l, j) \in \mathcal{C}$ for $i < l$.*

3 Simplification of the Predicates

Given a sequence of cells $\mathcal{C}$ and a time series s, we want to find intervals $I_1, \dots, I_{t_q}$ such that $\mathcal{C}$ is feasible in $F_\rho(q, s)$ if and only if $q_i \in I_i$ for all i, where $q = \langle q_1, \dots, q_{t_q} \rangle$ is a time series with some additional properties. The intervals will be defined using the predicates. Lemma 2.1 shows which predicates need to be true such that $\mathcal{C}$ is feasible in $F_\rho(q, s)$.

▶ **Lemma 3.1.** *Let $q = \langle q_1, \dots, q_{t_q} \rangle$ and $s = \langle s_1, \dots, s_{t_s} \rangle$ be two time series. Then the following holds for the predicates in the free space diagram $F_\rho(q, s)$:*

(i) (P_1) *is true* $\Leftrightarrow q_1 \in [s_1 - \rho, s_1 + \rho]$,
(ii) (P_2) *is true* $\Leftrightarrow q_{t_q} \in [s_{t_s} - \rho, s_{t_s} + \rho]$,
(iii) $(P_3(i, j))$ *is true* $\Leftrightarrow$ *if* $q_i \leq q_{i+1}$: $q_i \leq s_j + \rho$ *and* $q_{i+1} \geq s_j - \rho$ *and*
if $q_i \geq q_{i+1}$: $q_i \geq s_j - \rho$ *and* $q_{i+1} \leq s_j + \rho$,
(iv) $(P_4(i, j))$ *is true* $\Leftrightarrow q_i \in [\min\{s_j - \rho, s_{j+1} - \rho\}, \max\{s_j + \rho, s_{j+1} + \rho\}]$,
(v) $(P_5(i, j, k)$ *is true* $\Leftrightarrow (P_3(i, j))$ *and* $(P_3(i, k))$ *are true and one of the following holds:*
- $|s_j - s_k| \leq 2\rho$, *or*
- $|s_j - s_k| > 2\rho$ *and* $s_j \leq s_k$ *and* $q_i \leq s_j + \rho$ *and* $q_{i+1} \geq s_k - \rho$, *or*
- $|s_j - s_k| > 2\rho$ *and* $s_j > s_k$ *and* $q_i \geq s_j - \rho$ *and* $q_{i+1} \leq s_k + \rho$.

To determine the truth value of the monotone in s predicates (P_6), we introduce the new concept of *forward* and *backward numbers* $f_i(q)$ and $b_i(q)$. Here, we take advantage of the fact that the direction of each edge of a time series can only be orientated forward or backward with respect to the x-axis. Refer to Figure 2 as an example.

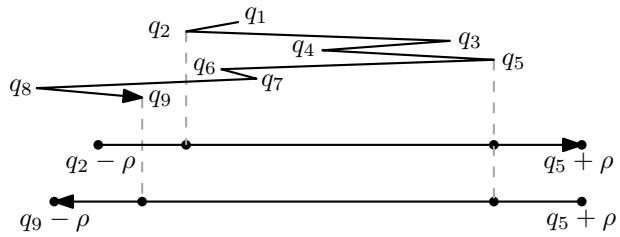


Figure 2 Illustration of the values $f_2(s) = 5$ and $b_5(s) = 9$ for a time series s.

▶ **Definition 3.2** (forward and backward numbers)**.** For a time series $q = \langle q_1, \ldots, q_{t_q} \rangle$ and $i \in \{1, \ldots, t_q\}$, we denote by the *forward number* $f_i(q) \leq t_q$ (resp. *backward number* $b_i(q) \leq t_q$) the highest number such that $\langle q_i - \rho, q_{f_i(q)} + \rho \rangle$ (resp. $\langle q_i + \rho, q_{b_i(q)} - \rho \rangle$) is oriented forward (resp. backward) and its Fréchet distance to the time series $\langle q_i, \ldots, q_{f_i(q)} \rangle$ (resp. $\langle q_i, \ldots, q_{b_i(q)} \rangle$) is at most ρ, i.e.,

$$f_i(q) := \max\{k \in \{i, \ldots, t_q\} \mid d_{\mathrm{F}}(\langle q_i, \ldots, q_k \rangle, \langle q_i - \rho, q_k + \rho \rangle) \leq \rho \text{ and } q_i - \rho \leq q_k + \rho\},$$
$$b_i(q) := \max\{k \in \{i, \ldots, t_q\} \mid d_{\mathrm{F}}(\langle q_i, \ldots, q_k \rangle, \langle q_i + \rho, q_k - \rho \rangle) \leq \rho \text{ and } q_i + \rho \geq q_k - \rho\}.$$

Note, that for all $i \leq x \leq f_i(q)$, it holds that $d_{\mathrm{F}}(\langle q_i, \ldots, q_x \rangle, \langle q_i - \rho, q_x + \rho \rangle) \leq \rho$ and $q_i - \rho \leq q_x + \rho$. Respectively, for $b_i(q)$. The next lemma shows how the forward and backward numbers can be used to determine values of the predicates (P_6). To decide whether a valid sequence of cells is feasible or not in $F_\rho(q, s)$, predicate $(P_6(i, l, j))$ needs to be true only if all predicates $(P_6(x, y, j))$ need to be true with $i \leq x < y \leq l$ by Lemma 2.1.

▶ **Lemma 3.3.** *Let $q = \langle q_1, \ldots, q_{t_q} \rangle$ and $s = \langle s_1, \ldots, s_{t_s} \rangle$ be time series, $i, l \in \{1, \ldots, t_q\}$ with $i < l$ and $j \in \{1, \ldots, t_s - 1\}$. If $s_j \leq s_{j+1}$ (resp. $s_j \geq s_{j+1}$), then $(P_6(x, y, j))$ is true $\forall i \leq x < y \leq l$ if and only if $f_i(q) \geq l$ (resp. $b_i(q) \geq l$) and $(P_4(x, j))$is true $\forall i \leq x \leq l$.*

4 Data Structure

In this section, we present two data structures solving the Fréchet queries problem. We start with some assumptions, that can be made for the time series. Let $s = \langle s_1, \ldots, s_t \rangle$ be a time series. Then, we can assume that either $s_{2j-1} \leq s_{2j} \geq s_{2j+1}$ for all j (*M-shaped*), or $s_{2j-1} \geq s_{2j} \leq s_{2j+1}$ for all j (*W-shaped*). Moreover, we can assume that the complexity of all time series in S is exactly t_s by simply adding dummy vertices in the end otherwise.

By Lemma 2.1, a sequence of cells $\mathcal{C}$ is feasible in the free space diagram $F_\rho(q, s)$ if and only if the predicates in Lemma 2.1 defined by $\mathcal{C}$ are true. The truth assignment of all predicates $(P_1), (P_2), (P_3), (P_4)$ and (P_5) can be determined using intervals defined by s and ρ. Furthermore, $\mathcal{C}$ can only be feasible in $F_\rho(q, s)$ if for all $(i-1, j), (l, j) \in \mathcal{C}$ with $i \leq l$, the monotone in s predicate $(P_6(i, l, j))$ is true. By Lemma 3.3, we can use the forward number $f_i(q)$ in the case that $s_j \leq s_{j+1}$ (i.e., j is odd if s is M-shaped) to determine whether $(P_6(i, l, j))$ is true. We define the *forward number* $f_i(\mathcal{C})$ as the highest such number l that is needed for $\mathcal{C}$ to be feasible in $F_\rho(q, s)$. Respectively, if $s_j \geq s_{j+1}$ (i.e., j is even if s is M-shaped) for $b_i(q)$ and we define the *backward number* $b_i(\mathcal{C})$. Formally, we get

$$f_i(\mathcal{C}) = \begin{cases} l \geq i, & \text{if } \exists\ (i-1, j), (l, j) \in \mathcal{C} \text{ s.t. } j \text{ is odd and } (l+1, j) \notin \mathcal{C}, \\ i, & \text{otherwise;} \end{cases}$$

$$b_i(\mathcal{C}) = \begin{cases} l \geq i, & \text{if } \exists\ (i-1, j), (l, j) \in \mathcal{C} \text{ s.t. } j \text{ is even and } (l+1, j) \notin \mathcal{C}, \\ i, & \text{otherwise.} \end{cases}$$

As $\mathcal{C}$ is valid there exists a unique j such that $(i-1,j),(i,j),\dots,(l,j) \in \mathcal{C}$. Hence, the numbers $f_i(\mathcal{C})$ and $b_i(\mathcal{C})$ are well-defined.

The Data structure. Let S_M be the set of stored time series that are M-shaped and S_W the set of those that are W-shaped. We will describe how S_M is stored. The time series in S_W are stored in the same way after they were mirrored at the origin. Consequently, for those the query algorithm mirrors the query time series q at the origin and is then the same as for the time series in S_M. For all valid sequences of cells $\mathcal{C}$, we build two associated rectangle stabbing data structures storing the time series in S_M as t_q-dimensional axis-aligned rectangles. One for the case that the query time series q is M-shaped and the other one for the case that q is W-shaped. Knowing the shape of q, Lemma 2.1 and 3.1 define for every $s \in S_M$ an interval for every vertex q_i of the query time series in which it must lie such that $\mathcal{C}$ can be feasible in $F_\rho(q,s)$. For a time series s, we store the Cartesian product of those t_q intervals in the associated rectangle stabbing data structure. Note that even if the complexity of the stored time series is greater than t_q, we store only a t_q-dimensional rectangle for it.

The Query Algorithm. Let q be a query time series of complexity t_q. The query algorithm starts with computing the numbers $f_1(q),\dots,f_{t_q}(q),b_1(q),\dots,b_{t_q}(q)$. For all valid sequences of cells $\mathcal{C}$, we check whether $f_i(\mathcal{C}) \le f_i(q)$ and $b_i(\mathcal{C}) \le b_i(q)$ for all i. If so, we do a query search in the rectangle stabbing data structure depending on $\mathcal{C}$ and the shape of q with the point $(q_1,q_2,\dots,q_{t_q})$ and output all time series associated with a rectangle containing this point.

▶ **Theorem 4.1.** *The Fréchet queries problem for constant parameters $t_q \ge 2$ and t_s can be solved with a data structure of size in $\mathcal{O}(n \log^{t_q-2} n)$ and query time in $\mathcal{O}(\log^{t_q-1} n + k)$, where k is the size of the output (without duplicates).*

Proof. The number of valid sequences of cells is constant and the forward and backwards numbers can be computed in constant time because t_s and t_q are constant. So, the size and the query time of the data structure follow by using the rectangle stabbing data structure by Afshani, Arge and Larsen [1]. The correctness follows by the discussion above and the fact that there exists a feasible valid sequences of cells in $F_\rho(q,s)$ if and only if $d_F(q,s) \le \rho$. ◀

Using an orthogonal range searching data structure, it is possible to store the time series in S as t_s-dimensional points and the query time series defines t_s-dimensional axis-aligned rectangles depending on $\mathcal{C}$. The data structure by Afshani, Arge and Larsen [1] leads to the following.

▶ **Corollary 4.2.** *The Fréchet queries problem for constant parameters t_q and $t_s > 2$ can be solved with a data structure of size in $\mathcal{O}\left(n(\log n/\log\log n)^{t_s-1}\right)$ that uses query time in $\mathcal{O}(\log n(\log n/\log\log n)^{t_s-3}+k)$, where k is the size of the output (without duplicates).*

Known lower bounds for rectangle stabbing and orthogonal range searching by Afshani, Arge and Larsen [1] and by Chazelle [5] can be applied to the Fréchet queries problem, because those problems can be transformed to it. Consider a data structure that solves the Fréchet queries problem and operates on a pointer machine. If it uses nh space, it must use query time in $\Omega(\log n(\log n/\log h)^{\lfloor t/2\rfloor-2}+k)$. And, if it uses query time in $\mathcal{O}(\log^c n + k)$, where c is a constant, it must use space in $\Omega(n(\log n/\log\log n)^{\lfloor t/2\rfloor-1})$. In both cases, k denotes the size of the output (without duplicates) and $t = \min\{t_q,t_s\}$.

References

1. P. Afshani, L. Arge, and K.G. Larsen. Higher-dimensional orthogonal range reporting and rectangle stabbing in the pointer machine model. In *Proceedings of the 2012 Symposuim on Computational Geometry*, pages 323–338, 2012. doi:10.1145/2261250.2261299.
2. P. Afshani and A. Driemel. On the complexity of range searching among curves. In *Proceedings of the 2018 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 898–917, 2018. doi:10.1137/1.9781611975031.58.
3. H. Alt and M. Godau. Computing the Fréchet distance between two polygonal curves. *International Journal of Computational Geometry and Applications*, 5(01& 02):75–91, 1995. doi:10.1142/S0218195995000064.
4. K. Bringmann, A. Driemel, A. Nusser, and I. Psarros. Tight bounds for approximate near neighbor searching for time series under the Fréchet distance. In *Proceedings of the 2022 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 517–550, 2022. doi:10.1137/1.9781611977073.25.
5. B. Chazelle. Lower bounds for orthogonal range searching: I. the reporting case. *Journal of the ACM*, 37(02):200–212, 1990. doi:10.1145/77600.77614.
6. Siu-Wing Cheng and Haoqiang Huang. Solving Fréchet distance problems by algebraic geometric methods. *ArXiv*, abs/2308.14569, 2023. doi:10.48550/arXiv.2308.14569.
7. Mark de Berg, Atlas F. Cook, and Joachim Gudmundsson. Fast Fréchet queries. *Computational Geometry*, 46(6):747–755, 2013. doi:10.1016/j.comgeo.2012.11.006.
8. A. Driemel and I. Psarros. ANN for time series under the Fréchet distance. In *Algorithms and Data Structures*, pages 315–328, 2021. doi:10.1007/978-3-030-83508-8_23.
9. A. Driemel and F. Silvestri. Locality-sensitive hashing of curves. In *33rd International Symposium on Computational Geometry*, volume 77, pages 37:1–37:16, 2017. doi:10.4230/LIPIcs.SoCG.2017.37.
10. A. Filtser, O. Filtser, and M.J. Katz. Approximate nearest neighbor for curves: simple, efficient, and deterministic. *Algorithmica*, 2022. doi:10.1007/s00453-022-01080-1.

Capturing the Shape of a Point Set with a Line Segment

Nathan van Beusekom[1], Marc van Kreveld[2], Max van Mulken[1], Marcel Roeloffzen[1], Bettina Speckmann[1], and Jules Wulms[1]

1 **Department of Mathematics and Computer Science, TU Eindhoven**
[n.a.c.v.beusekom | m.j.m.v.mulken | m.j.m.roeloffzen | b.speckmann | j.j.h.m.wulms]@tue.nl
2 **Department of Information and Computing Sciences, Utrecht University**
m.j.vankreveld@uu.nl

Abstract

Detecting groups or clusters in point sets is an important task in a wide variety of application areas. In addition to detecting such groups, the group's shape carries meaning. In this paper, we aim to represent a group's shape using a simple geometric object: a line segment. Specifically, given a radius r, we say a line segment represents the shape of a point set P if it is within Hausdorff distance r from each point $p \in P$. Finding the shortest such line segment is equivalent to stabbing a set of circles of radius r using the shortest line segment. We describe an algorithm for this task that runs in $O(n \log h + h^2)$ time, where n is the size of the point set and h is the size of its convex hull.

Related Version A full version of the paper is available at arxiv.org/abs/2402.12285

1 Introduction

Studying groups or clusters in point sets is an important task in a wide variety of application areas. There are many algorithms and approaches to find such groups; examples include the well-known *k-means clustering* [11] or *DBSCAN* [9]. In addition to the mere existence of such groups, the group's characteristics can carry important information as well. In wildlife ecology, for example, the perceived shape of herds of prey animals contains information about the behavioral state of animals within the herd [15]. Since shape is an abstract concept that can get arbitrarily complex, it is often useful to have a simplified representation of group shape that can efficiently be computed.

In this paper, we use a simple geometric object, a line segment, as a shape descriptor of a group of entities in a point-location data set. Specifically, our input is a set P of n points in $\mathbb{R}^2$, and a radius r. Our goal is to find an (oriented) line segment q_1q_2 that lies within Hausdorff distance r from each point $p \in P$. We call such a line segment a *shape-representing* line segment of P. We propose an algorithm that finds the shortest shape-representing line segment in $O(n \log h + h^2)$ time, where h is the size of the convex hull of P.

For a line segment q_1q_2 to be within Hausdorff distance r from a point p, it must intersect the circle of radius r centered at p. Thus, we reformulate the problem: given a set of circles C_P of radius r centered at points in P, we must find the shortest line segment q_1q_2 that intersects all circles in C_P (see Figure 1). We assume that the set P is in general position; no three points of P lie on a line, and that at most two circles of C_P intersect in a point.

Due to space constraints, most proofs are omitted; they can be found in the full version.

40th European Workshop on Computational Geometry, Ioannina, Greece, March 13–15, 2024.
This is an extended abstract of a presentation given at EuroCG'24. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

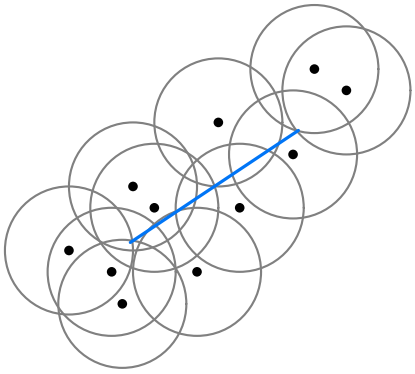


Figure 1 The line segment (blue) must hit every circle of radius r, centered at the points in P.

Related work. A number of shape descriptors have been proposed over the years. A few popular ones are the *alpha shape* of a point set [7] or the *characteristic shape* [6], both of which generate shape-representing polygons. Another way to generate the shape of a point set is to fit a function to the point set [3, 10, 17]. The set of problems of finding one or more geometric objects that intersect a different set of geometric objects is known as the set of *stabbing* problems [8], and several variants have been studied [2, 5, 14]. To our knowledge, stabbing a set of circles with the shortest line segment has not been studied. However, inverse variants that stab line segments with one or more circles have been studied [4, 13].

2 Computing the Shortest Shape-Representing Line Segment

Our algorithm is similar to the rotating calipers algorithm [16]. We start by finding the shortest shape-representing line segment for fixed orientation α, after which we rotate by π while maintaining the line segment, and return the shortest one. Note that, even though a shape-representing line segment does not exist for every orientation, we can easily find an initial orientation α for which it does exist using rotating calipers; these are the orientations at which the rotating calipers have width $\leq 2r$. Although, our input point set P can be of any shape, the following lemma shows that it suffices to consider only its convex hull $\mathrm{CH}(P)$.

▶ **Lemma 2.1.** *If a line segment q_1q_2 intersects all circles defined by the vertices of the convex hull $CH(P)$, then q_1q_2 also intersects all circles defined by the points in P.*

Proof. Since q_1q_2 crosses each circle defined by $\mathrm{CH}(P)$, each vertex of $\mathrm{CH}(P)$ has a distance of at most r to q_1q_2. Any point on the edges of the convex hull are also at most r to q_1q_2, by definition. All other points in P are inside the convex hull and thus each point in P must have a distance of at most r to q_1q_2. ◀

We can compute $\mathrm{CH}(P)$ in $O(n \log h)$ time, where h is the size of the convex hull [1, 12]. Observe that, if rotating calipers initially finds no orientation with width $> 2r$, then point set P can be enclosed by a circle of radius at most r, and the shortest shape-representing line segment is the center point of this enclosing circle. Hence, in the rest of this paper we assume that the shortest shape-representing line segment has non-zero length.

Fixed orientation. We describe how to find the shortest shape-representing line segment with fixed orientation α. Using rotating calipers [16], we can find all orientations in which a shape-representing line segment exists. We pick α such that such a solution exists; for ease

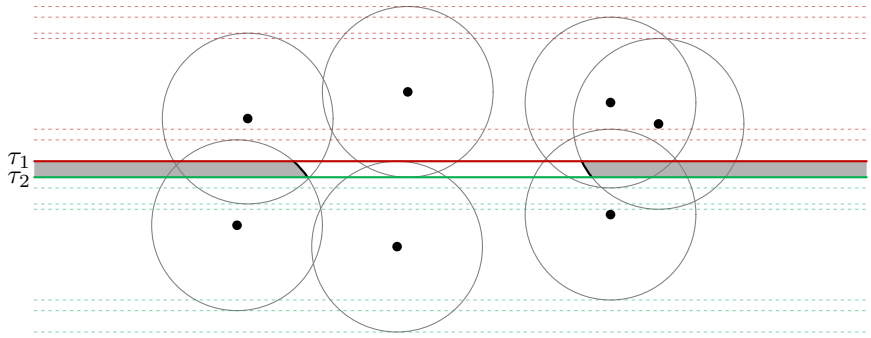


Figure 2 Two extremal tangents τ_1 and τ_2 for horizontal orientation α. The shortest line segment of orientation α that intersects all circles, ends at the boundary of the gray regions.

of exposition and without loss of generality, we assume α to be horizontal. Let the *left/right half-circle* of a circle c be the half-circle between $\pi/2$ and $3\pi/2$ and between $3\pi/2$ and $5\pi/2$, respectively. Lemma 2.1 permits us to consider only points of P on the convex hull, thus for the remainder of this paper we use C_P to indicate the set of circles of radius r centered at the vertices of $\mathrm{CH}(P)$. We use $\mathcal{A}(C_P)$ to denote the circle arrangement of C_P.

Observe that every horizontal line that lies below the bottom-most top horizontal tangent τ_1 and above the top-most bottom horizontal tangent τ_2 of all circles crosses all circles (see Figure 2). If τ_1 lies below τ_2, then there exists no horizontal line that crosses all circles.

To place q_1q_2 in the strip between τ_1 and τ_2, we can define two regions R_1, R_2 in which endpoints q_1 and q_2 must be placed such that the line segment between q_1 and q_2 intersects all circles (see Figure 2). The boundaries of R_1 and R_2 are defined by a *convex sequence* S_1 and S_2 of (in horizontal orientation) the right-most left arcs and left-most right arcs, delimited by the two tangents τ_1, τ_2, as well as these tangents themselves. If R_1 and R_2 intersect, then we can place a single point in their intersection at distance at most r from all points in P. Note that q_1 and q_2 must be on the convex sequences S_1 and S_2, respectively; otherwise, we can move the endpoint onto the convex sequence, shortening q_1q_2 and still intersecting all circles.

▶ **Lemma 2.2.** *For fixed orientation α, we can compute S_1 and S_2 in $O(h^2)$.*

Next, we must place q_1 and q_2 on S_1 and S_2, respectively, such that q_1q_2 is shortest. We show that q_1q_2 is the shortest line segment of orientation α when the tangents of S_1 at q_1 and S_2 at q_2 have equal slope. Vertices on S_1 and S_2 have a range of tangents (see Figure 3).

▶ **Lemma 2.3.** *Let S_1 and S_2 be two convex sequences of circular arcs, and let q_1 and q_2 be points on S_1 and S_2, respectively, such that line segment q_1q_2 has orientation α. If the tangent on S_1 at q_1 and the tangent on S_2 at q_2 have equal slope, then q_1q_2 is minimal.*

Observe that the length of q_1q_2 is unimodal between τ_1 and τ_2. We can hence binary search in $O(\log h)$ time for the optimal placement of q_1 and q_2. By Lemmata 2.2 and 2.3 we can compute the shortest shape-representing line segment of orientation α in $O(h^2)$ time.

Rotation. After finding the shortest line segment for a fixed orientation α, as described in the previous section, we sweep through all orientations α while maintaining τ_1, τ_2, S_1, S_2, and the shortest shape-representing line segment q_1q_2 of orientation α. We allow all of these maintained structures to change continuously as the orientation changes, and store the shortest shape-representing line segment found. Any time a discontinuous change would

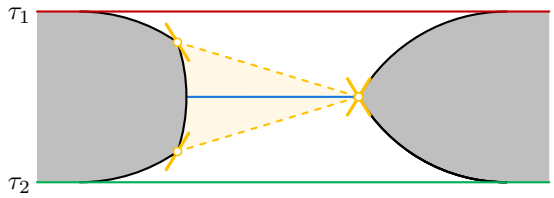


Figure 3 Two convex sequences between τ_1 and τ_2. There are multiple points on the left convex sequence that have the same tangent as the right yellow vertex. Still, there is only one line segment in horizontal orientation for which the tangents of its endpoints are equal (blue).

happen, we trigger an *event* to reflect these changes. We pre-compute and maintain a number of *certificates* in an event queue, which indicate at which orientation the next event occurs. This way we can perform the continuous motion until the first certificate is violated, recompute the maintained structures, repair the event queue, and continue rotation. We distinguish four types of events:

1. q_1 or q_2 moves onto/off of a vertex of S_1 or S_2;
2. τ_1 or τ_2 is a bi-tangent with the next circle on the convex hull;
3. τ_1 or τ_2 hits a (prospective) vertex of S_1 or S_2;
4. τ_1 and τ_2 are the same line.

Since the shortest line segment q_1q_2 in orientation α is completely determined by τ_1, τ_2, S_1, and S_2, the above list forms a complete description of all possible events. Thus, we maintain at most two certificates for events of type 1, 2, and 3, and only a single certificate for type-4 events, which are stored in a constant-size event queue Q, ordered by appearance orientation. Insert, remove, and search operations on Q can hence be performed in $O(1)$ time.

Event handling. In the following descriptions, we assume that an event happens at orientation α, and that ε is picked such that no other events occur between $\alpha - \varepsilon$ and $\alpha + \varepsilon$. Some event may occur in two symmetric cases; one of each is omitted here.

(1) q_1/q_2 moves onto/off of a vertex of S_1/S_2. We describe, without loss of generality, how to handle the event involving q_1 and S_1; the case for q_2 and S_2 are analogous. See Figure 4 for an example of this event. Observe that, since vertices of S_1 cover a range of tangents, there are intervals of orientations at which q_1 remains at a vertex of S_1. As such, we describe two different cases for this event: q_1 moves *onto* or *off* a vertex of S_1.

If q_1 was moving over an arc of S_1 at $\alpha - \varepsilon$ and encounters a vertex at α, then the movement path of q_1 is then updated to remain on the encountered vertex. Additionally, we

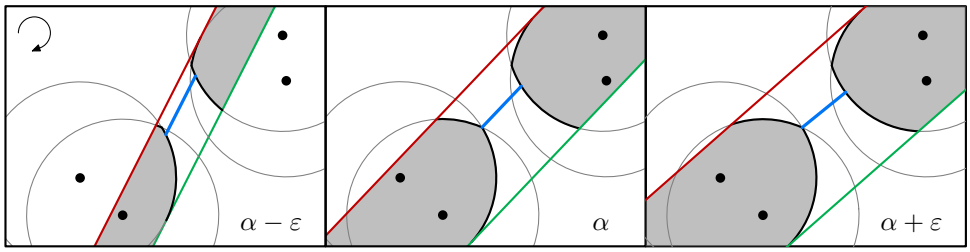


Figure 4 When q_1/q_2 is at a vertex of S_1/S_2, it stops moving.

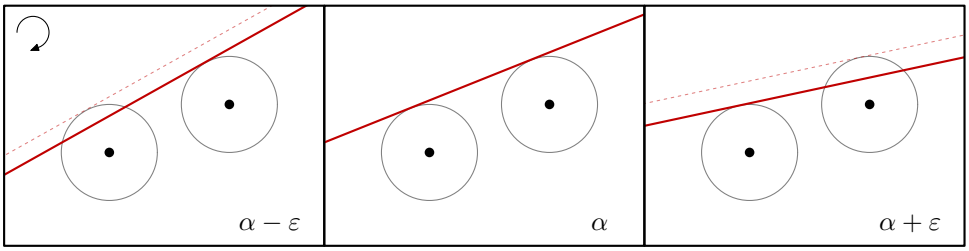


Figure 5 When the defining circle of τ_1/τ_2 changes, τ_1/τ_2 is parallel to a convex hull edge.

place a new type-1 certificate into the event queue that is violated when q_1 should move off of the vertex, e.g. when the final orientation covered by the vertex is reached.

▶ **Lemma 2.4.** *Throughout the full π rotation, q_1/q_2 moves onto/off of a vertex of S_1/S_2 at most $O(h^2)$ times, and we can resolve each occurrence of such an event in $O(1)$ time.*

(2) τ_1 or τ_2 is bi-tangent with the next circle on the convex hull. We describe, without loss of generality, how to handle the event involving τ_1; the case for τ_2 is analogous. See Figure 5 for an example of this event. When τ_1 is a bi-tangent of two circles defined by their centers $u, v \in P$ then, by definition of τ_1, u and v must both be the extremal points in the direction θ perpendicular to α. Therefore, (u, v) must be an edge on the convex hull. Suppose that, without loss of generality, u was the previous extremal vertex in direction $\theta - \varepsilon$, then v is extremal in direction $\theta + \varepsilon$. As such, τ_1 belongs to u at $\alpha - \varepsilon$, and to v at $\alpha + \varepsilon$. When this happens, we insert a new type-2 certificate into the event queue that is violated at the orientation of the next convex hull edge. Additionally, we recompute the certificates of type 3 and 4 that are currently in the event queue to reflect the updated τ_1.

▶ **Lemma 2.5.** *Throughout the full π rotation, τ_1 or τ_2 are bi-tangent with another circle at most $O(h)$ times. The two circles that define such a bi-tangent are adjacent in the convex hull, and we can resolve each occurrence of such an event in $O(1)$ time.*

(3) τ_1 or τ_2 hits a (prospective) vertex of S_1 or S_2. We describe, without loss of generality, how to handle the event involving τ_1 and S_1; the case for τ_2 and S_2 is analogous. Additionally, we can distinguish between the case where τ_1 does not belong to an arc on S (Figure 6, event 3.1) and the case where it does (Figure 7, event 3.2). We will describe the prior case here. The latter is similar, but has different certificates; the differences are described below. Let vertex v be a vertex on convex sequence S_1 that is intersected by τ_1 at orientation α. Then either vertex v is on S_1 at orientation $\alpha - \varepsilon$ but no longer on S_1 at $\alpha + \varepsilon$, or vice versa.

If the arc of S_1 intersecting τ_1 is shrinking, then at orientation α, that arc is completely removed from S_1; vertex v becomes the endpoint of S_1 and starts moving along the next arc of S_1 with the intersection point between τ_1 and S_1. If the affected arc or vertex appeared in a type-1 certificate in the event queue, it is updated to reflect the removal of the arc and the new movement of the vertex. Additionally, we place a new type-3 certificate into the event queue that is violated when τ_1 intersects the next vertex on S_1 (event 3.1), or when τ_1 hits an intersection point between S_1 and the defining circle of τ_1 (event 3.2).

▶ **Lemma 2.6.** *Throughout the full π rotation, τ_1 or τ_2 hits a vertex of S_1 or S_2 at most $O(h^2)$ times, and we can resolve each occurrence of such an event in $O(1)$ time.*

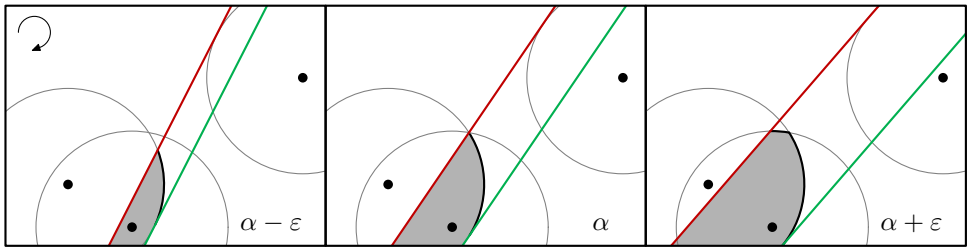


Figure 6 When τ_1/τ_2 hits a vertex of $\mathcal{A}(C_P)$, an arc may need to be added to S_1/S_2.

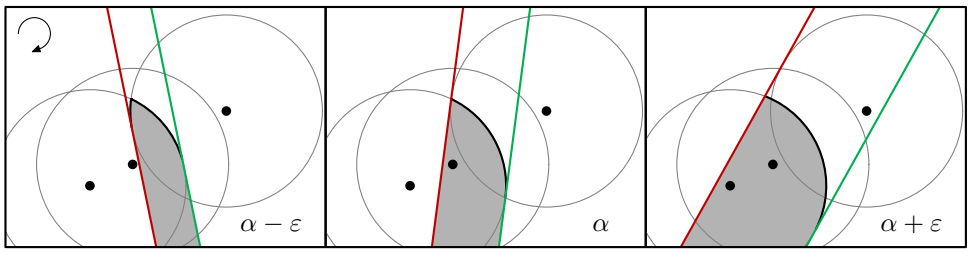


Figure 7 When τ_1/τ_2 hits an intersection of its defining circle, the composition of S_1/S_2 changes.

(4) τ_1 and τ_2 are the same line. When this event takes place, then τ_1 and τ_2 are the inner bi-tangent of their two respective defining circles. See Figure 8 for an example. We distinguish two different cases for this event: either there is a solution at $\alpha - \varepsilon$ and no solution at $\alpha + \varepsilon$, or vice versa.

If there was a solution at $\alpha - \varepsilon$ and there is none at $\alpha + \varepsilon$, we simply stop maintaining q_1q_2, S_1 and S_2 until there exists a solution again. As such, we remove all type-1 and type-3 certificates from the event queue and place a new type-4 certificate into the event queue that is violated at the next orientation where τ_1 and τ_2 are the same line.

▶ **Lemma 2.7.** *Throughout the full π rotation, τ_1 and τ_2 become the same line at most $O(h)$ times, and we can resolve each occurrence of such an event in $O(h)$ time.*

▶ **Theorem 2.8.** *Given a point set P consisting of n points and a radius r, we can find the shortest shape-representing line segment in $O(n \log h + h^2)$ time.*

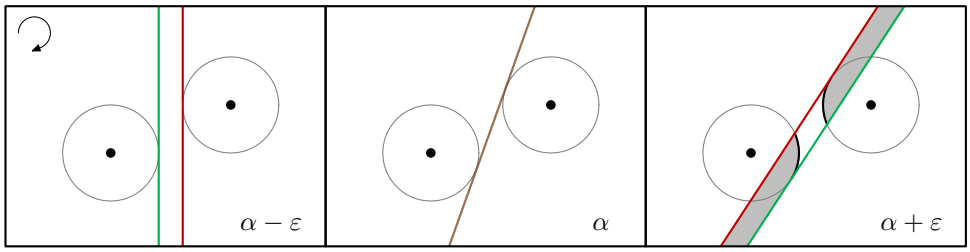


Figure 8 When τ_1 and τ_2 are the same line, they are an inner bi-tangent of their two defining circles.

3 Discussion

An obvious open question is whether the shortest shape-representing line segment can be computed in $O(n \log h)$. In the full version of this abstract, we show that we can actually compute the convex sequence in linear time, given the convex hull. The question is then whether the convex sequence can be traversed efficiently, without using the full circle arrangement $\mathcal{A}(C_P)$. We expect that this may be possible, yet, even this is not sufficient: Observe that in a regular k-gon with a diameter $2r + \varepsilon$, a solution appears/disappears $O(n)$ times. Then, a linear-time convex sequence construction is not sufficient. Furthermore, the circles contributing to a convex sequence may be as far as $n/2$ vertices apart on the convex hull, making amortization difficult. In the full version of this abstract we show that we can compute a $(1 + \varepsilon)$-approximation in $O(n \log h + h/\varepsilon)$ time by sampling orientations and applying the fixed orientation algorithm.

References

1 Timothy M. Chan. Optimal output-sensitive convex hull algorithms in two and three dimensions. *Discret. Comput. Geom.*, 16(4):361–368, 1996.

2 Timothy M. Chan, Thomas C. van Dijk, Krzysztof Fleszar, Joachim Spoerhase, and Alexander Wolff. Stabbing rectangles by line segments–how decomposition reduces the shallow-cell complexity. In *29th International Symposium on Algorithms and Computation*, 2018.

3 Danny Z. Chen and Haitao Wang. Approximating points by a piecewise linear function. *Algorithmica*, 66(3):682–713, 2013.

4 Merce Claverol, Elena Khramtcova, Evanthia Papadopoulou, Maria Saumell, and Carlos Seara. Stabbing circles for sets of segments in the plane. *Algorithmica*, 80:849–884, 2018.

5 José Miguel Díaz-Báñez, Matias Korman, Pablo Pérez-Lantero, Alexander Pilz, Carlos Seara, and Rodrigo I Silveira. New results on stabbing segments with a polygon. *Computational Geometry*, 48(1):14–29, 2015.

6 Matt Duckham, Lars Kulik, Mike Worboys, and Antony Galton. Efficient generation of simple polygons for characterizing the shape of a set of points in the plane. *Pattern recognition*, 41(10):3224–3236, 2008.

7 Herbert Edelsbrunner, David Kirkpatrick, and Raimund Seidel. On the shape of a set of points in the plane. *IEEE Transactions on information theory*, 29(4):551–559, 1983.

8 Herbert Edelsbrunner, Hermann A. Maurer, Franco P. Preparata, Arnold L. Rosenberg, Emo Welzl, and Derick Wood. Stabbing line segments. *BIT Numerical Mathematics*, 22:274–281, 1982.

9 Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Second International Conference on Knowledge Discovery and Data Mining (KDD-96)*, pages 226–231. AAAI Press, 1996.

10 S. Louis Hakimi and Edward F. Schmeichel. Fitting polygonal functions to a set of points in the plane. *CVGIP: Graphical Models and Image Processing*, 53(2):132–136, 1991.

11 John A. Hartigan and Manchek A. Wong. Algorithm as 136: A k-means clustering algorithm. *Journal of the royal statistical society.*, 28(1):100–108, 1979.

12 David G Kirkpatrick and Raimund Seidel. The ultimate planar convex hull algorithm? *SIAM journal on computing*, 15(1):287–299, 1986.

13 Konstantin Kobylkin. Stabbing line segments with disks: complexity and approximation algorithms. In *International Conference on Analysis of Images, Social Networks and Texts*, pages 356–367. Springer, 2017.

14 Lena Marie Schlipf. *Stabbing and Covering Geometric Objects in the Plane*. PhD thesis, FU Berlin, 2014.

15 Ariana Strandburg-Peshkin, Damien R. Farine, Margaret C. Crofoot, and Iain D. Couzin. Habitat and social factors shape individual decisions and emergent group structure during baboon collective movement. *eLife*, 6:e19505, 2016.

16 Godfried T. Toussaint. Solving geometric problems with the rotating calipers. In *IEEE Melecon*, volume 83, page A10, 1983.

17 Der Perng Wang. A new algorithm for fitting a rectilinear x-monotone curve to a set of points in the plane. *Pattern Recognition Letters*, 23(1-3):329–334, 2002.

Representing Hypergraphs by Point-Line Incidences

Alexander Dobler[1], Stephen Kobourov[2], William J. Lenhart[3], Tamara Mchedlidze[4], Martin Nöllenburg[1], and Antonios Symvonis[5]

1 **Algorithms and Complexity Group, TU Wien, Austria**
adobler@ac.tuwien.ac.at,noellenburg@ac.tuwien.ac.at
2 **University of Arizona, Tucson, AZ, USA**
kobourov@cs.arizona.edu
3 **Williams College, Williamstown, USA**
wlenhart@williams.edu
4 **Utrecht University, Utrecht, The Netherlands**
t.mtsentlintze@uu.nl
5 **NTUA, Athens, Greece**
symvonis@math.ntua.gr

Abstract

We consider hypergraph visualization that represent vertices as points and hyperedges as lines with few bends passing through points of their incident vertices. Guided by point-line incidence theory we show several theoretical results: if every vertex is part of at most two hyperedges, then we can find such a visualization without bends. There exist hypergraphs with three vertices per hyperedge and three hyperedges incident to each vertex requiring an arbitrary number of bends. It is $\exists\mathbb{R}$-hard to decide whether an arbitrary hypergraph can be visualized without bends. This only answers some interesting questions for such visualizations and we conclude with many open research questions.

1 Introduction

Hypergraphs arise in many domains and visualizing them is a non-trivial challenge. Classical approaches, such as Venn and Euler diagrams [1, 12, 15], do not scale to large instances. Recent experimental work [20] has shown that representing hypergraphs as collections of polylines (for the hyperedges) and common intersection points (for the vertices) allows for faster and more accurate performance of hypergraph-related tasks. In particular, the MetroSets approach [11] uses the metro map metaphor by representing each hyperedge with a metro line and each vertex as an interchange station. It attempts to visualize the result in the traditional octolinear fashion; see Figure 1. The visual complexity of the result depends on the total number of bends along the metro lines in the embedding. Minimizing the visual complexity makes the representations simpler to understand and work with. The natural question is to ask which hypergraphs can be represented with just one bendless line per hyperedge. Naturally, only hypergraphs such that hyperedges share at most one vertex – called *linear hypergraphs* – can be represented in such a way. Otherwise, lines of hyperedges could coincide and could not be distinguished. Further, the *rank* of a hypergraph is the maximum cardinality of a hyperedge; the *degree* is defined equivalently as for classic graphs. With this in mind we show that:

- Maximum degree two linear hypergraphs can be represented with one line per hyperedge.
- Not all rank-three maximum degree-three linear hypergraphs can be represented with one line per hyperedge. In fact, there is a family of such hypergraphs requiring an arbitrary number of bends.
- Determining whether an arbitrary-rank linear hypergraph can be represented with one line per hyperedge is $\exists\mathbb{R}$-hard.

40th European Workshop on Computational Geometry, Ioannina, Greece, March 13–15, 2024.
This is an extended abstract of a presentation given at EuroCG'24. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

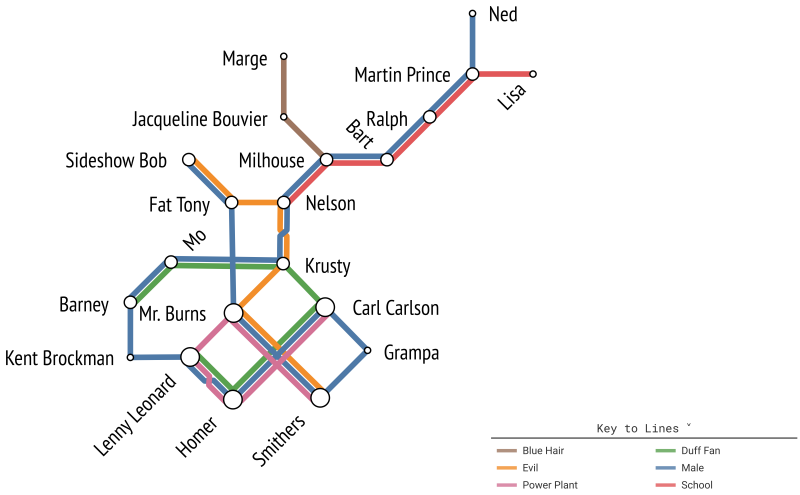


Figure 1 A visualization of a Simpsons hypergraph dataset using the MetroSets metaphor [11], the visualization is taken from `https://metrosets.ac.tuwien.ac.at/`. Hyperedges are represented by metro lines and elements are represented by stations.

Here, lines are infinite lines and not line segments.

2 Related Work

Representing hypergraphs with one line per hyperedge is related to classical geometric problems going as far back as the 19th century. In particular, it is related to the study of *configurations*, a set of points and an arrangement of lines, such that each point is incident to the same number of lines and each line is incident to the same number of points [9,10,16]. This was the main topic of the PhD thesis of Steinitz [18] and was of interest to many with notable examples including the configurations of Desargues, Pappus, and Möbius–Kantor [10]. If the representation of a hypergraph should however form a non-crossing straight-line drawing of a tree, then it can be decided in polynomial time whether such a representation exists [19]. Graph-based techniques for drawing hypergraphs via support graphs [2–4] have a different focus and do not take into account geometric straightness or bends of hyperedges.

If we ask for crossing-free representations of hypergraphs with line segments instead of lines, there is only one line of research that we are aware of [6, 8]. Namely, Gonçalves [8] has shown that there are planar linear hypergraphs which cannot be represented with straight line segments (in contract to planar graphs which can always be drawn with straight lines).

The problem seems to be related to stretchability of pseudolines [17], but is different because the order of the vertices along each hyperedge is not specified. A similar $\exists\mathbb{R}$-hard problem, called matroid representability [13] will be used to show $\exists\mathbb{R}$-hardness of one of the problems studied in this paper. We will formally explain matroid representability in the corresponding Section 6.

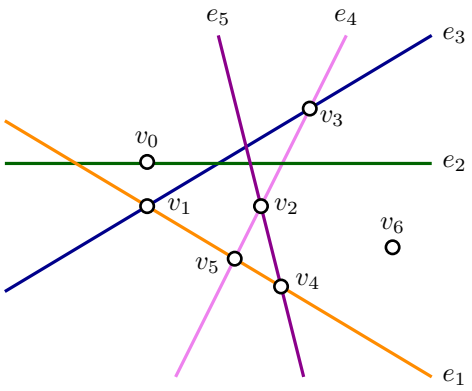


Figure 2 Illustration of a max-degree-2 hypergraph $H = (V, E)$ with $V = \{v_0, \ldots, v_6\}$ and $E = \{e_1, \ldots, e_5\}$, where $e_1 = \{v_1, v_4, v_5\}$, $e_2 = \{v_0\}$, $e_3 = \{v_1, v_3\}$, $e_4 = \{v_2, v_3, v_5\}$, $e_5 = \{v_2, v_4\}$.

3 Preliminaries

A hypergraph $H = (V, E)$ is defined by a vertex set V and a hyperedge set E, where each $e \in E$ is a subset of V. The *degree* of a vertex v is the number of hyperedges containing v. The *rank* of a hypergraph is the maximum cardinality of a hyperedge $|e|$ taken over all e in E; hence a rank-2 hypergraph is an ordinary graph. A hypergraph is *k-uniform* if very hyperedge has cardinality exactly k and it has degree k if every vertex has degree k. A hypergraph is *linear* if $|e \cap e'| \leq 1$ for every pair of distinct hyperedges $e, e' \in E$. A representation of a hypergraph consists of an injective mapping α of vertices to points in $\mathbb{R}^2$ and an injective mapping β of hyperedges to lines in $\mathbb{R}^2$ such that $v \in e$ if and only if $\alpha(v) \in \beta(e)$ for $v \in V, e \in E$.

4 Max-Degree-2 Hypergraphs

▶ **Theorem 4.1.** *There exists a representation for any max-degree-2 linear hypergraph.*

Proof. Let H be a hypergraph with n vertices $v_1, \ldots, v_n$ and m hyperedges $e_1, \ldots, e_m$. Consider m lines $\ell_1, \ldots, \ell_m$ on $\mathbb{R}^2$ in general position, such that any two of these lines cross; see Figure 2. Let v be a vertex of H. If the degree of v is exactly 2, then we place v at the intersection of ℓ_i and ℓ_j, where ℓ_i and ℓ_j are the lines corresponding to the hyperedges e_i and e_j containing v. If the degree of v is 1, then we place v at any point of the line corresponding to the hyperedge containing v that is not an intersection point of the lines $\ell_1, \ldots, \ell_m$. If the degree of v is zero, then we place v at any point of $\mathbb{R}^2$ that is not on any of the lines $\ell_1, \ldots, \ell_m$. This yields a representation of H with one line per hyperedge. ◀

5 Rank-3 Degree-3 Hypergraphs

The PhD thesis of Ernst Steinitz [18] claims that every 3-uniform degree-3 hypergraph can be represented with one line per hyperedge, except maybe of one hyperedge (which could be represented as a polyline with one bend). However, more careful consideration shows that this is indeed not true as already pointed out by Grünbaum [10]. Similar results exist, but none show the claim of Steinitz [7, 14]. We show a construction that has at least two hyperedges that must have a bend and this construction can be generalized. For this, we

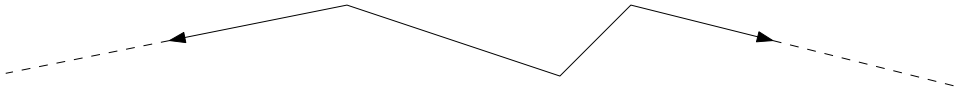


Figure 3 Illustration of an infinite polygonal chain with 3 bends that extends infinitely in the left and right direction.

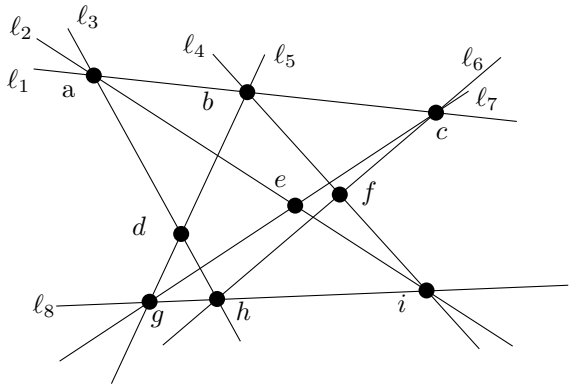


Figure 4 The Pappus configuration without the line through d, e, f.

define for $t \in \mathbb{N}_0$, t-bend representations for hypergraphs H as follows. In the original definition of representations, we replace lines by what we call *infinite polygonal chains*. An infinite polygonal chain consists of a (possibly empty) polygonal chain and two rays, one ending at the first point of the chain, one ending at the last point (see Figure 3). Essentially, an infinite polygonal chain is a polyline whose first and last segment is extended infinitely. Further, two distinct infinite polygonal chains $\beta(e), \beta(e')$ for $e, e' \in E$ must not share a line segment nor a bend point. Lastly, we require that the total number of all bends in $\beta(E)$ is exactly t. A representation in the original sense is hence a 0-bend representation.

Consider the 3-uniform degree-3 hypergraph H defined as follows. Let H_1 be the hypergraph depicted in Figure 4 with points $a, b, \ldots, i$ and hyperedges defined by the lines $\ell_1, \ell_2, \ldots, \ell_8$. Pappus's theorem [5, Chapter 3.5] says that in any representation of H_1, the points d, e, f must be collinear[1]. Now let H_2 be a copy of H_1 with the points $a', b', \ldots, i'$ and hyperedges defined equivalently. The hypergraph H is the union of H_1, H_2, and the two hyperedges $\{d, e, f'\}$ and $\{d', e', f\}$.

▶ **Lemma 5.1.** *There is no t-bend representation for H with $t < 2$.*

Proof. If every hyperedge in the subhypergraph H_1 is represented without a bend, then $\beta(\{d, e, f'\})$ must pass through f. Thus, at least one hyperedge of H_1 must be represented with at least one bend. Applying the same argument to H_2 we see that we require at least two bends. ◀

The above construction can be generalized so that there is no t-bend representation for any $t < x$ for an arbitrary $x \in \mathbb{N}$. Instead of one copy of H_1, we add $x - 1$ copies $H_2, H_3, \ldots, H_x$. Further we add x hyperedges $e_1, e_2, \ldots, e_x$ such that e_i contains d, e of H_i and f of $H_{(i \mod x)+1}$.

[1] Formally, Pappus's theorem is as follows: let points a, b, c be on one line, and points g, h, i be on another line. Let the three points d, e, f be defined by the intersections of line ah with bg, ai with cg, and bi with ch, respectively. Then d, e, f are collinear. It is clear that our formulation is equivalent.

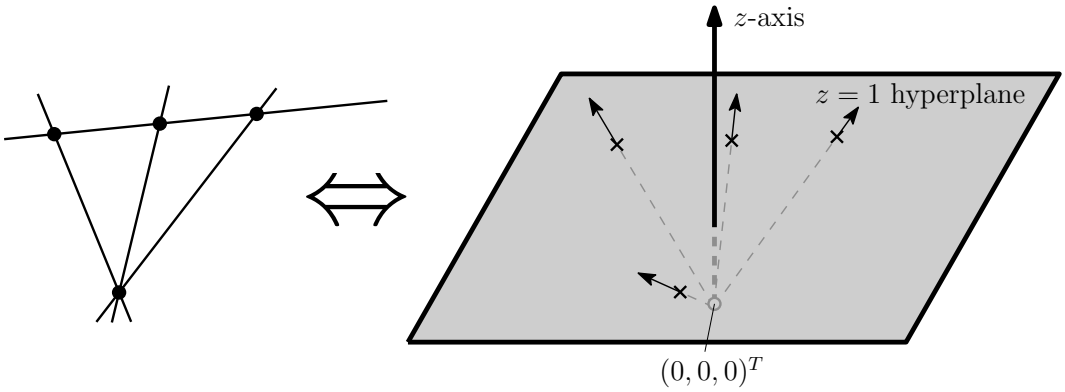


Figure 5 Illustration of the equivalence between representations of H and representations of M. Lines on the left side correspond to linear subpaces of dimension two that are common to at least two vectors on the right side.

▶ **Theorem 5.2.** *For any $x \in \mathbb{N}$ there exists a rank-3 hypergraph H such that there exists no t-bend representation for H for any $t < x$.*

Similar constructions exist (at least for $x = 2$) [10], the generalization has not been stated explicitly.

6 $\exists\mathbb{R}$-hardness

In this section we want to show that it is $\exists\mathbb{R}$-hard to decide whether there exists a representation for a given hypergraph H. We reduce from a problem that lends itself for a nice reduction, called Matroid Representability [13]. For the purposes of a simple description, we give here a simplified description of a variant of that problem that is still $\exists\mathbb{R}$-hard [13]. We start with definitions. A matroid M is given as $M = (X, \mathcal{I})$ where X is the finite *ground set* and $\mathcal{I} \subseteq 2^X$ is the set of independent sets with

1. $\emptyset \in \mathcal{I}$,
2. $I' \subset I \in \mathcal{I}$ implies $I' \in \mathcal{I}$, and
3. $I_1, I_2 \in \mathcal{I}$ with $|I_1| < |I_2|$ implies that there is an $x \in I_2 \setminus I_1$ with $I_1 \cup \{x\} \in \mathcal{I}$.

A representation of M is an injective mapping $f(X) : X \to \mathbb{R}^3$ such that for any $Y \subseteq X$ we have $Y \in \mathcal{I}$ if and only if $f(Y)$ forms a set of linearly independent vectors in $\mathbb{R}^3$. The $\exists\mathbb{R}$-hard problem Matroid Representability is given as input a matroid and the question is whether there is a representation f of M. For the vectors $v \in \mathbb{R}^3$ we call the first, second, and third coordinate the x, y, and z-coordinates, respectively.

We start by making some normalizations to the instance M. First, we can assume that every independent set $I \in \mathcal{I}$ has cardinality of at most 3, as there is otherwise no representation. Second, we can assume that each pair $x, x' \in X$ of distinct elements forms an independent set, i.e. $\{x, x'\} \in \mathcal{I}$. Otherwise, $f(x) = cf(x')$ for some $c \in \mathbb{R}$ must hold for any representation. We can remove x' from X and replace any occurence of x' in $\mathcal{I}$ by x, and obtain an equivalent instance w.r.t. representability.

We are ready to state our reduction. The main idea is to identify the vectors $f(X)$ by points in the plane due to a projective transformation; see Figure 5.

▶ **Theorem 6.1.** *It is $\exists\mathbb{R}$-hard to decide whether a linear hypergraph can be represented.*

Proof. We reduce from Matroid Representability. We are given a matroid $M = (X, \mathcal{I})$ (we assume both above normalizations were applied already) and transform it to a hypergraph $H = (V, E)$ as follows. First, we set $V = X$. The hyperedges E are defined as follows. Let $Y \subseteq X$ be a maximal subset of X that must lie in the same linear subspace U of $\mathbb{R}^3$ (which by definition contains the origin) with $\dim U = 2$ for any representation f of M. Add Y to E. All such Y can be determined in polynomial time: first, we can determine in polynomial time all triples $x, x', x'' \in X$ that do not form an independent set. If two triples share two elements, the union of both triples forms a four-tuple that has to be part of a common subspace in any representation. Continuing this process greedily, we can find all such Y. This process terminates with the desired sets Y, as the matroid M specifies for each triple exactly if the vectors of their representation are dependent or not. Notice also that there can only exist polynomially many such Y as each pair of $x, x' \in X$ can be part of at most one such Y. Lastly, we have to add pairs $\{x, x'\}$, $x, x' \in X$, to E, that are not part of some Y defined above. The vectors $f(x), f(x')$ certainly have to span a subspace U of dimension two not containing any other vectors $f(x'')$.

We claim that M has a representation f if and only if H has a representation (α, β).

"$\Rightarrow$": let f be a representation of M. First, if any $f(x')$, $x' \in X$ has z-coordinate 0 we multiply all $f(x)$, $x \in X$, by the same rotation matrix $R \in SO(3) \subseteq \mathbb{R}^{3\times 3}$ such that no $f(x)$ has z-coordinate 0. This is possible as X is finite. Second, we scale each $f(x)$ by 1 divided by its z-coordinate so that the z-coordinates of all vectors in $f(X)$ are 1. The new f still is a representation. For $x \in X$, we set $\alpha(x)$ equal to the point defined by the first two coordinates of $f(x)$. Essentially, we applied a projective transformation. Lastly, for each hyperedge $e \in E$ we set $\beta(e)$ to the line through any two distinct points $\alpha(x)$ and $\alpha(x')$ with $x, x' \in e$ (E does not contain edges of size 1). It is now easy to verify that (α, β) is a representation of H: let $e \in E$. For each $x \in e$ the point $\alpha(x)$ must lie on the line $\beta(e)$ because $f(x)$ must be in the same linear subspace of dimension two as all $f(x'), x' \in e$. No other point can lie on that line, as e would not have been maximal for our construction of Y above.

"$\Leftarrow$": let (α, β) be a representation of H. For $x \in X$, let $\alpha(x) = (r, s)^T$. We set $f(x) = (r, s, 1)^T$ and claim that f is a representation of M. Indeed, if $Y \subseteq X$ does not form an independent set in M, then $Y \subseteq e$ for some $e \in E$. Thus, points $\alpha(Y)$ lie on the same line and vectors $f(Y)$ are in the same linear subspace of dimension two and are thus dependent as $|Y| \geq 3$. If $Y \in \mathcal{I}$ there are two cases.

- If $|Y| \leq 2$, $f(Y)$ is clearly independent as α is injective.
- If $|Y| = 3$, let $Y = \{x_1, x_2, x_3\}$. Because of the construction of E, there exists $e \in E$ such that $x_1, x_2 \in e$ and $x_3 \notin e$. Thus $\alpha(Y)$ forms a triangle and $f(Y)$ is independent. ◀

7 Conclusions and Open Problems

Motivated by a hypergraph visualization using polylines for hyperedges [11], we initiated the investigation of visualizations with hyperedges drawn with as few bends as possible. We provide results for special classes of hypergraphs. If the maximum vertex degree is 2, any linear hypergraph can always be drawn without a bend. For rank-3 linear hypergraphs arbitrarily many bends may be required. Lastly, it is even $\exists\mathbb{R}$-hard to decide whether an arbitrary linear hypergraph can be drawn without bends.

Our results are highly inconclusive and many open research directions are possible.

- We have considered representations with lines and infinite polygonal chains as we could use results from point-line incidence theory. If we replace these with line segments and polygonal chains, respectively, our results from Sections 5 and 6 do not extend. It may as

well be that for such definitions each connected 3-uniform degree-3 hypergraph (besides the Fano and Möbius-Kantor configurations [10]) has a representation with line segments without bends, and it might be that it is "easy" to decide whether such a line segment representation exists.
- Our $\exists\mathbb{R}$-hardness result from Section 6 works for arbitrary rank hypergraphs. Can we state a similar hardness result for bounded-rank linear hypergraphs, e.g. rank 3?
- Are there polynomial-time algorithms to represent a 3-uniform degree-3 linear hypergraph in such a way that the number of bends in the representation is a constant factor away from optimum (minimum number of bends)? Are there families of 3-uniform degree-3 hypergraphs requiring more than a linear number of bends?

Acknowledgments.

This work started at the Bertinoro Workshop on Graph Drawing 2023. We thank Michael A. Bekos for the discussions.

References

1 Bilal Alsallakh, Luana Micallef, Wolfgang Aigner, Helwig Hauser, Silvia Miksch, and Peter J. Rodgers. The state-of-the-art of set visualization. *Comput. Graph. Forum*, 35(1):234–260, 2016. URL: https://doi.org/10.1111/cgf.12722, doi:10.1111/CGF.12722.

2 Ulrik Brandes, Sabine Cornelsen, Barbara Pampel, and Arnaud Sallaberry. Path-based supports for hypergraphs. *J. Discrete Algorithms*, 14:248–261, 2012. URL: https://doi.org/10.1016/j.jda.2011.12.009, doi:10.1016/J.JDA.2011.12.009.

3 Kevin Buchin, Marc van Kreveld, Henk Meijer, Bettina Speckmann, and Kevin Verbeek. On planar supports for hypergraphs. *J. Graph Algorithms Appl.*, 15(4):533–549, 2011. doi:10.7155/jgaa.00237.

4 Thom Castermans, Mereke van Garderen, Wouter Meulemans, Martin Nöllenburg, and Xiaoru Yuan. Short plane supports for spatial hypergraphs. *J. Graph Algorithms Appl.*, 23(3):463–498, 2019. doi:10.7155/jgaa.00499.

5 Harold Scott Macdonald Coxeter and Samuel L Greitzer. *Geometry revisited*, volume 19. Mathematical Association of America, 1967.

6 Hubert de Fraysseix and Patrice Ossona de Mendez. Representations by contact and intersection of segments. *Algorithmica*, 47(4):453–463, 2007. doi:10.1007/S00453-006-0157-X.

7 David G. Glynn. On the representation of configurations in projective spaces. *Journal of Statistical Planning and Inference*, 86(2):443–456, May 2000. doi:10.1016/S0378-3758(99)00124-X.

8 Daniel Gonçalves. A planar linear hypergraph whose edges cannot be represented as straight line segments. *Eur. J. Comb.*, 30(1):280–282, 2009. URL: https://doi.org/10.1016/j.ejc.2007.12.004, doi:10.1016/J.EJC.2007.12.004.

9 Harald Gropp. Configurations and their realization. *Discret. Math.*, 174(1-3):137–151, 1997. doi:10.1016/S0012-365X(96)00327-5.

10 Branko Grünbaum. *Configurations of Points and Lines*. American Mathematical Society, 2009.

11 Ben Jacobsen, Markus Wallinger, Stephen G. Kobourov, and Martin Nöllenburg. Metrosets: Visualizing sets as metro maps. *IEEE Trans. Vis. Comput. Graph.*, 27(2):1257–1267, 2021. doi:10.1109/TVCG.2020.3030475.

12 David S. Johnson and H. O. Pollak. Hypergraph planarity and the complexity of drawing Venn diagrams. *J. Graph Theory*, 11(3):309–325, 1987. doi:10.1002/jgt.3190110306.

13 Eunjung Kim, Arnaud de Mesmay, and Tillmann Miltzow. Representing matroids over the reals is ∃R-complete. *CoRR*, abs/2301.03221, 2023. doi:10.48550/arXiv.2301.03221.

14 William Kocay and Ryan Szypowski. The application of determining sets to projective configurations. *Ars Combinatoria*, 53:193–208, 1999.

15 Erkki Mäkinen. How to draw a hypergraph. *Int. J. Computer Math.*, 34(3–4):177–185, 1990. doi:10.1080/00207169008803875.

16 Tomaž Pisanski and Brigitte Servatius. Configurations from a graphical viewpoint. *Ars Math. Contemp.*, 5(2), 2012. URL: https://amc-journal.eu/index.php/amc/article/view/342/193.

17 Peter W. Shor. Stretchability of pseudolines is NP-hard. In Peter Gritzmann and Bernd Sturmfels, editors, *Proc. Applied Geometry And Discrete Mathematics (DIMACS 1990)*, volume 4 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 531–554. DIMACS/AMS, 1990. doi:10.1090/DIMACS/004/41.

18 Ernst Steinitz. *Über die Construction der Configurationen n_3*. PhD thesis, Breslau, 1894.

19 R. Swaminathan and Donald K. Wagner. On the consecutive-retrieval problem. *SIAM J. Comput.*, 23(2):398–414, 1994. doi:10.1137/S0097539792235487.

20 Markus Wallinger, Ben Jacobsen, Stephen G. Kobourov, and Martin Nöllenburg. On the readability of abstract set visualizations. *IEEE Trans. Vis. Comput. Graph.*, 27(6):2821–2832, 2021. doi:10.1109/TVCG.2021.3074615.

Recognition Complexity of Subgraphs of 2- and 3-Connected Planar Cubic Graphs

Miriam Goetze[1], Paul Jungeblut[2], and Torsten Ueckerdt[3]

1 **Karlsruhe Institute of Technology, Germany**
`miriam.goetze@kit.edu`
2 **Karlsruhe Institute of Technology, Germany**
`paul.jungeblut@kit.edu`
3 **Karlsruhe Institute of Technology, Germany**
`torsten.ueckerdt@kit.edu`

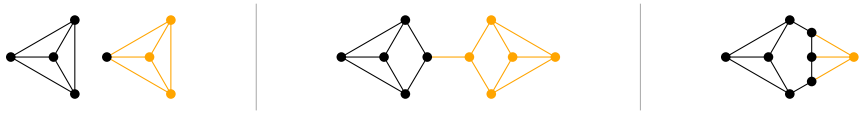
Abstract

We study the recognition complexity of subgraphs of 2- and 3-connected planar cubic graphs. Recently, we presented [ESA 2022] a quadratic-time algorithm to recognize subgraphs of planar cubic bridgeless (but not necessarily connected) graphs, both in the variable and fixed embedding setting (the latter only for 2-connected inputs). Here, we extend our results in two directions: First, we present a quartic-time algorithm to recognize subgraphs of 2-connected planar cubic graphs in the fixed embedding setting, even for disconnected inputs. Second, we prove NP-hardness of recognizing subgraphs of 3-connected planar cubic graphs in the variable embedding setting.

1 Introduction

Given a planar graph G of maximum degree at most 3 (i.e., a *subcubic* planar graph), we want to augment G by adding further vertices and edges to obtain a 3-regular (i.e., *cubic*) planar graph H, which is then called a 3-*augmentation* of G. An embedding[1] of H induces an embedding $\mathcal{E}$ of G, and each face f of $\mathcal{E}$ may contain edges of $E(H) - E(G)$, called *new edges*, or even vertices of $V(H) - V(G)$, called *new vertices*, or none of these. It is easy to see that 3-augmentations always exist. However, this becomes non-trivial if we require H to be k-connected for some $k \in \{1, 2, 3\}$. See Fig. 1 for some problematic cases. Here, we study whether a subcubic planar graph G admits some k-connected 3-augmentation H, or equivalently, the recognition of subgraphs of k-connected cubic planar graphs. We consider several variants where the input graph G is given with a fixed embedding $\mathcal{E}$ and the desired 3-augmentation H must extend $\mathcal{E}$, and/or where the input graph G is already k'-connected for some $k' \in \{0, 1, 2\}$. (If G is 3-connected, then $H = G$ is the only connected 3-augmentation.)

[1] We consider combinatorial (crossing-free) embeddings with no specific choice of an outer face.



$G_1 \subset H_1$ $G_2 \subset H_2$ $G_3 \subset H_3$

Figure 1 For $k = 1, 2, 3$, the planar subcubic graph G_k (in black) admits a $(k-1)$-connected 3-augmentation H_k (new vertices and edges in orange), but no k-connected 3-augmentation.

40th European Workshop on Computational Geometry, Ioannina, Greece, March 13–15, 2024.
This is an extended abstract of a presentation given at EuroCG'24. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

Previous Results. Motivated by the problem whether a given planar graph G is 3-edge-colorable (whose complexity is still open), we recently considered the sufficient (although not necessary) condition of whether G admits a bridgeless 3-augmentation. In fact, this can be tested in quadratic-time in the variable embedding setting, and also in the fixed embedding setting, provided that G is 2-connected [3].

Hartmann, Rollin and Rutter [5] studied, for each $k, r \in \mathbb{N}$, whether a planar graph G can be augmented by adding edges (but no vertices!), to a k-connected r-regular planar graph H. In particular, for $r = 3$, they show that the problem is NP-complete in the variable embedding setting for all $k \in \{0, 1, 2, 3\}$, as well as in the fixed embedding setting when $k = 3$. For the remaining cases of fixed embedding and $k \in \{0, 1, 2\}$ they present a polynomial-time algorithm.

Let us also refer to [3] for more related work and other augmentation problems.

Our Results. We resolve the complexity of finding a k-connected 3-augmentation for a given subcubic planar graph G in two new cases. See Fig. 2 and the following theorem.

▶ **Theorem 1.1.** *Let G be an n-vertex planar graph with an embedding $\mathcal{E}$ and $\Delta(G) \leq 3$.*

1. *We can compute, in time $\mathcal{O}(n^4)$, a 2-connected 3-augmentation H extending $\mathcal{E}$, or conclude that none exists. If G is connected, then $\mathcal{O}(n^2)$ time suffices.*
2. *It is NP-complete to decide whether G admits a 3-connected 3-augmentation.*

Note that Statements 1 and 2 concern the fixed and variable embedding setting, respectively.

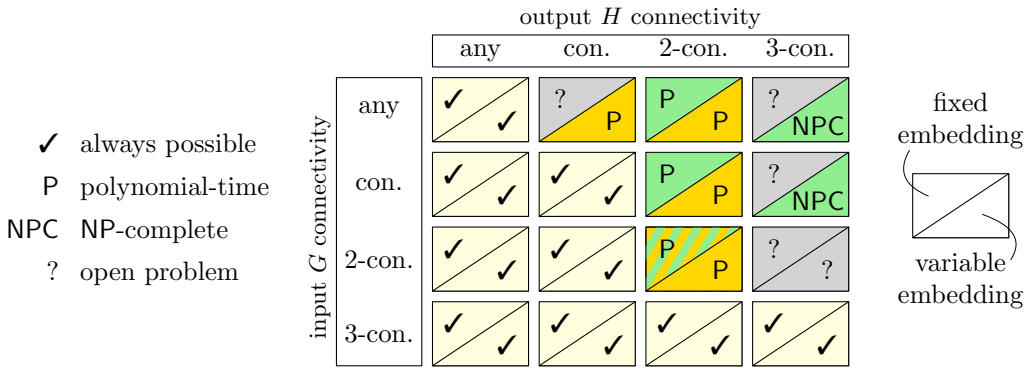


Figure 2 Complexity of finding 3-augmentations. Colors indicate results in this paper and [3].

▶ Remark. For our polynomial-time algorithms in the fixed embedding setting, we shall reduce the problem to a particular version of the Generalized Factor problem (definitions in Section 2). This approach is similar to the treatment of 2-connected input graphs in [3]. But here, for graphs containing bridges or consisting of several connected components, additional tools and a more refined analysis are needed, which is also reflected in the increased runtime.

Statements marked with (⋆) are proven in the full version [4].

2 Preliminaries

A graph $G = (V, E)$ is *k-connected* if $G - S$ is connected for every set $S \subseteq V$ of at most $k - 1$ vertices in G. Similarly, G is *k-edge-connected* if $G - S$ is connected for every set $S \subseteq E$ of

at most $k-1$ edges in G. We denote by $\theta(G)$ the largest k for which G is k-edge-connected. If G has maximum degree at most 3, then G is k-connected if and only if $\theta(G) \geq k$.

For an integer $\ell \geq 3$, let W_ℓ be the graph obtained from $C_\ell \square P_2$ by subdividing each edge in one cycle C_ℓ exactly once. See Fig. 3 (left) for an illustration. Consider a planar graph G with an embedding $\mathcal{E}$, and a vertex $v \in V(G)$ with $\deg_G(v) = \ell \geq 3$. A *wheel-extension at* v is the graph and embedding obtained by replacing v with W_ℓ, and by attaching v's incident edges to the subdivision vertices of W_ℓ in a one-to-one non-crossing way. See Fig. 3 (right).

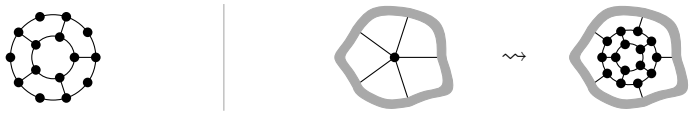


Figure 3 Left: $C_5 \square P_2$ with subdivision vertices. Right: Wheel-extension.

▶ **Observation 2.1 (⋆).** *Let G be a graph (possibly containing multi-edges, but no loops), let $v \in V(G)$ be a vertex with $\deg_G(v) \geq 3$, and let G' be obtained from G by a wheel-extension at v. Then $\theta(G') \geq \min\{\theta(G), 3\}$.*

Generalized Factors. Let H be a graph with a set $B(v) \subseteq \{0, \dots, \deg_H(v)\}$ assigned to each vertex $v \in V(H)$. Following Lovász, a spanning subgraph $G \subseteq H$ is called a *B-factor* of H if and only if $\deg_G(v) \in B(v)$ for every vertex $v \in V(H)$ [6]. Deciding whether a graph H admits a B-factor is known as the Generalized Factor problem. In general, the Generalized Factor problem is NP-complete [6]. Still, for certain well-behaved sets $B(\cdot)$, the problem becomes polynomial-time solvable. A set $B(v)$ is said to have a *gap of length* $\ell \geq 1$ if there is an integer $i \in B(v)$ such that $i+1, \dots, i+\ell \notin B(v)$, and $i+\ell+1 \in B(v)$. If all gaps of each $B(v)$ have length 1, then an algorithm by Cornuéjols can compute a B-factor in time $\mathcal{O}\big(|V(H)|^4\big)$ [1]. Moreover, if there are no two consecutive forbidden degrees $i, i+1 \in \{0, \dots, \deg_H(v)\}$ for any v, i.e., $i, i+1 \notin B(v)$, then a B-factor can be computed in time $\mathcal{O}\big(|V(H)| \cdot |E(H)|\big)$ by a result of Sebő [7]. (The latter condition is slightly stronger than requiring gaps of length at most 1, explaining the better runtime.)

3 2-Connected 3-Augmentations with a Fixed Embedding

We consider the 3-augmentation problem for arbitrary input graphs G and 2-connected output graphs H, corresponding to the third column of the table in Fig. 2. For the variable embedding setting, a quadratic-time algorithm is given in [3, Theorem 2]. For the fixed embedding setting here, we present a quartic-time algorithm. We start with a reduction to graphs G with $\delta(G) \geq 2$.

▶ **Lemma 3.1 (⋆).** *Let G be a planar graph with embedding $\mathcal{E}$. There is a planar supergraph $G' \supseteq G$ with $\delta(G') \geq 2$ whose embedding $\mathcal{E}'$ extends $\mathcal{E}$, such that G has a 2-connected 3-augmentation extending $\mathcal{E}$ if and only if G' has one extending $\mathcal{E}'$.*

The proof is simple and just requires the following preprocessing of G: Replace all vertices $v \in V(G)$ with $\deg_G(v) \leq 1$ by copies of K_3 as shown in Fig. 4 (left/middle).

▶ **Lemma 3.2.** *Let G be a planar n-vertex graph with an embedding $\mathcal{E}$, $\delta(G) \geq 2$, and $\Delta(G) \leq 3$. Then we can compute, in time $\mathcal{O}(n^4)$, a 2-connected 3-augmentation H of G extending $\mathcal{E}$, or conclude that none exists. If G is connected, then time $\mathcal{O}(n^2)$ suffices.*

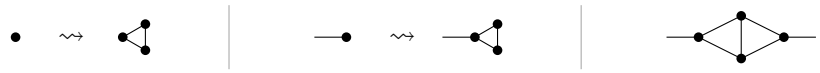


Figure 4 Left/Middle: Replacement rules. Right: Gadget to avoid parallel edges.

Proof. The proof is by a linear-time reduction to an equivalent instance A of the Generalized Factor problem, such that A fulfills the necessary condition to apply an $\mathcal{O}(n^4)$-time algorithm by Cornuéjols [1, Section 3], or even an $\mathcal{O}(n^2)$-time algorithm by Sebő [7, Section 3].

We construct the 2-connected 3-augmentation H of G by adding new edges and vertices into the faces of $\mathcal{E}$. Therefore, the obtained embedding of H extends $\mathcal{E}$.

Some faces of $\mathcal{E}$ stand out, as these *must* contain new edges (and possibly vertices) to reach 2-connectedness. We call these the *connecting faces* F_c. Obviously, all faces incident to at least two connected components are connecting faces. Further, for each bridge e of G, the unique face f incident to both sides of e is a connecting face because the only way to add new connections between the components separated by e is through f. Recall that a 3-regular graph is 2-connected if and only if it is connected and bridgeless, so these are the only two types of connecting faces. All other faces are considered to be *normal* faces, denoted by F_n.

For a connecting face $f \in F_c$, let G_f be the subgraph of G on the vertices and edges incident to f, let B_f be its blocks (i.e., maximal 2-connected components or bridges), and let T_f be its block-cut-forest. We partition B_f into $S_f \cup I_f \cup L_f$:

$$
\begin{aligned}
S_f &\coloneqq \{b \in B_f \mid b \text{ forms a trivial (i.e., single-vertex) tree in } T_f\} && (\textit{singleton} \text{ blocks})\\
I_f &\coloneqq \{b \in B_f \mid b \text{ is an inner vertex of a non-trivial tree in } T_r\} && (\textit{inner} \text{ blocks})\\
L_f &\coloneqq \{b \in B_f \mid b \text{ is a leaf in a non-trivial tree in } T_f\} && (\textit{leaf} \text{ blocks})
\end{aligned}
$$

The Generalized Factor instance A is a bipartite graph with bipartition classes $\mathcal{V}$ and $\mathcal{F}$. Here, $\mathcal{V} \coloneqq \{v \in V \mid \deg_G(v) = 2\}$ contains all vertices of G not yet having degree 3. Similarly, vertices in $\mathcal{F}$ represent the faces of $\mathcal{E}$. Edges of a B-factor of A will determine the faces of $\mathcal{E}$ containing the new edges. In particular, $\mathcal{F}$ contains one vertex corresponding to each normal face in F_n of $\mathcal{E}$. Additional vertices in $\mathcal{F}$ are needed to handle the connecting faces. For each connecting face $f \in F_c$, we add all blocks in B_f as vertices to $\mathcal{F}$. (If there are two faces f, g in $\mathcal{E}$ such that B_f and B_g contain blocks corresponding to the same subgraph of G, then $\mathcal{F}$ contains two such vertices: one corresponding to the block in B_f, and another to the block in B_g.)

In A, each $x \in \mathcal{F}$ is incident to exactly the following $v \in \mathcal{V}$: If x is a normal face $f_n \in F_n$, then x is connected to all $v \in \mathcal{V}$ that are incident to f_n in $\mathcal{E}$. Otherwise, if x is a block $b \in B_f$ for some connecting face $f_c \in F_c$, then x is connected to all $v \in \mathcal{V}$ that are contained in b. See Fig. 5 for an example.

Lastly, we need to assign a set $B(x) \subseteq \{0, 1, \dots, \deg_A(x)\}$ of possible degrees to each vertex $x \in (\mathcal{V} \cup \mathcal{F})$:

$$
B(x) \coloneqq \begin{cases}
\{1\}, & x \in \mathcal{V}\\
\{0, 2, 3, \dots, \deg_A(x)\}, & x \in F_n\\
\{0, 1, 2, \dots, \deg_A(x)\}, & x \in I_f \text{ for some connecting face } f \in F_c\\
\{1, 2, 3, \dots, \deg_A(x)\}, & x \in L_f \text{ for some connecting face } f \in F_c\\
\{2, 3, 4, \dots, \deg_A(x)\}, & x \in S_f \text{ for some connecting face } f \in F_c
\end{cases}
$$

The order and size of A are linear in n as every vertex $v \in V(G)$ is contained in at most

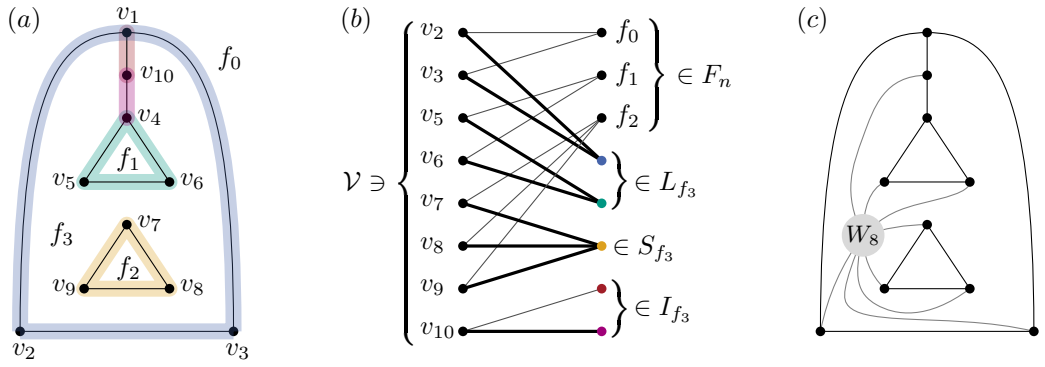


Figure 5 (a) A planar subcubic graph G. (b) Its corresponding Generalized Factor instance. Thick edges denote a possible solution. (c) A 2-connected 3-augmentation of G (with an indicated wheel-extension).

three distinct faces, and at most three blocks of G_f for any face f of $\mathcal{E}$. Moreover, A can be computed in linear time; see the full version [4, Claim 3.3] for a proof.

Also in the full version [4, Claims 3.4 and 3.5], we formally prove that A admits a B-factor if and only if G has a 2-connected 3-augmentation H. To convey an intuition, consider a B-factor A' of A. Every vertex $v \in \mathcal{V}$ is incident to exactly one edge $vx \in E(A')$. Here, x is either a face f, or there is a face f such that x is a block in B_f. In order to obtain H, we add a new half-edge incident to v into f. Now, for each face f of $\mathcal{E}$, we connect all half-edges within f to a new vertex v_f. Applying a wheel-extension (Obs. 2.1) to every vertex v_f of degree larger than 3, and replacing each vertex v_f of degree 2 by the gadget in Fig. 4 (right), yields a 2-connected 3-augmentation of G. For the other direction, observe that each degree-2-vertex of G is, in H, incident to a new edge inside a face f of $\mathcal{E}$. A B-factor A' of A is obtained by adding vx to $E(A')$, where $x \in \mathcal{F}$ is either f or a block in B_f (containing v).

It remains to argue that we can compute a B-factor of A efficiently. By inspecting the sets $B(x)$ for all $x \in (\mathcal{V} \cup \mathcal{F})$, we can see that none of them contains a gap of size 2 or greater. Therefore, we are in a special case of the Generalized Factor problem that can be solved, in $\mathcal{O}(n^4)$ time, by Cornuéjols' algorithm [1].

A closer inspection yields that only for $x \in S_f$ the sets $B(x)$ contain two forbidden degrees. (Note that $\deg_A(v) \leq 2$ for all $v \in \mathcal{V}$: If there is a face f such that v is contained in two blocks of G_f, then both edges incident to v are bridges; thus v is incident to no other face. Otherwise, this follows from $\deg_G(v) \leq 2$, i.e., v being incident to at most two faces.) Therefore, if $S_f = \emptyset$ for all connecting faces $f \in F_C$, then we can even apply the algorithm by Sebő, taking only $\mathcal{O}(n^2)$ time [7]. In particular, this is the case if G is connected. ◀

4 NP-Hardness for 3-Connected 3-Augmentations

In this section, we shall prove that deciding whether a given planar graph G admits a 3-connected 3-augmentation is NP-complete. In particular, we show that the problem remains NP-complete when restricted to connected graphs G. This implies the NP-completeness results represented in the fourth column of the table in Fig. 2.

Recall that an embedding of any 3-connected 3-augmentation H induces an embedding $\mathcal{E}$ of G, and for convenience, let us call the pair $(H, \mathcal{E})$ a *solution* for G. Let us also define a (≤ 2)-*subdivision* of a graph R to be the result of subdividing each edge in R with up to two vertices. Note that, if R is 2-connected, then so is every (≤ 2)-subdivision of R.

▶ **Lemma 4.1** (⋆). *Let G be a graph obtained from a (≤ 2)-subdivision R_2 of a 3-connected planar graph R by attaching a degree-1 vertex to each subdivision vertex. Then G admits a solution $(H, \mathcal{E})$ if and only if no face of $\mathcal{E}$ has exactly one or two incident degree-1 vertices.*

By Lem. 4.1, any graph G as described in the lemma admits a 3-connected 3-augmentation if and only if it admits an embedding $\mathcal{E}$ with no face incident to exactly one or two degree-1 vertices. Testing such graphs for such embeddings, however, turns out to be NP-complete.

▶ **Theorem 4.2** (⋆). *Deciding whether a given graph is a subgraph of a 3-regular 3-connected planar graph is* NP*-complete.*

Proof Idea. We reduce from the NP-complete problem Planar-Monotone-3SAT [2], where an instance is a monotone 3SAT-formula Ψ together with a planar embedding $\mathcal{E}_\Psi$ of its bipartite variable-clause incidence graph I_Ψ, such that **(1)** the x-axis contains each variable, **(2)** no edge crosses the x-axis, and **(3)** each clause above (respectively below) the x-axis is positive (respectively negative).

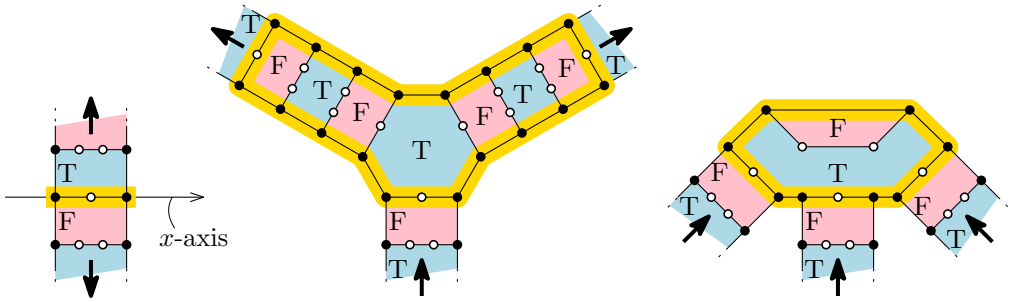


Figure 6 Gadgets used in the NP-hardness reduction.

We obtain a graph G_Ψ from the embedding $\mathcal{E}_\Psi$ using the gadgets in Fig. 6. Each variable gadget is the start of one upper and one lower corridor, each with alternating red (standing for FALSE) and blue (standing for TRUE) faces. Each positive splitter gadget splits one upper corridor into two, and each positive clause gadget is the end of one upper corridor of each appearing variable. Negative splitter and negative clause gadgets below the x-axis are symmetric, with red and blue swapped. See Fig. 7 for a full example.

With some extra vertices and edges, the result is a (≤ 2)-subdivision of a 3-connected 3-regular graph R. Attaching degree-1 vertices as in Lem. 4.1 gives G_Ψ, and we ask for an embedding $\mathcal{E}$ of G_Ψ with no face having one or two degree-1 vertices. The crux is, that except for the two highlighted faces in the clause gadgets, any pair of neighboring red and blue faces has in total at most five subdivision vertices. Thus, for each variable x, either only blue or only red faces in all splitters and corridors have degree-1 vertices. Setting x to TRUE in the former and to FALSE in the latter case, satisfies all clauses. See Fig. 8 for an example.

In fact, Ψ is satisfiable if and only if G_Ψ admits an embedding as required by Lem. 4.1. ◀

▶ Remark. The (≤ 2)-subdivision in the above reduction behaves quite similar to G_Ψ. The only problem is that a face f may have been chosen by exactly *two* incident degree-2 vertices to contain their third (new) edge without creating a 2-edge-cut; namely, with a direct edge. Thus, the above reduction also yields NP-completeness of recognizing *induced* subgraphs of 3-connected 3-regular planar graphs, even for 2-connected inputs with a unique embedding.

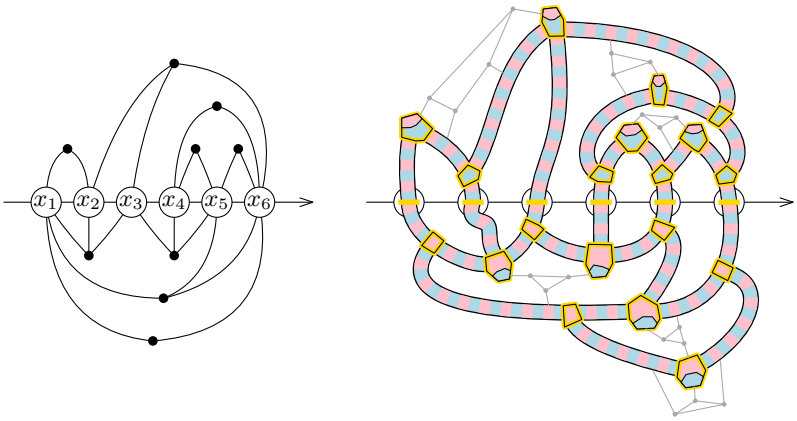


Figure 7 Illustration of a Planar-Monotone-3SAT embedding $\mathcal{E}_\Psi$ and a corresponding graph G_Ψ. Extra vertices and edges added for 3-connectivity of R are shown in gray.

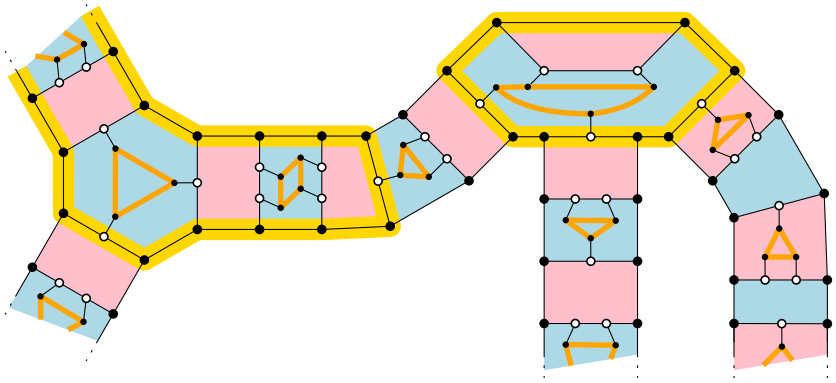


Figure 8 Part of the graph G_Ψ together with a 3-connected 3-augmentation in orange. This corresponds to a clause with two true variables (left with a splitter gadget, and middle) and one false variable (right).

References

1 Gérard Cornuéjols. General Factors of Graphs. *Journal of Combinatorial Theory, Series B*, 45(2):185–198, 1988. doi:10.1016/0095-8956(88)90068-8.

2 Mark de Berg and Amirali Khosravi. Optimal Binary Space Partitions in the Plane. In My T. Thai and Sartaj Sahni, editors, *Computing and Combinatorics (COCOON 2010)*, volume 6196 of *Lecture Notes in Computer Science*, pages 216–225, 2010. doi:10.1007/978-3-642-14031-0_25.

3 Miriam Goetze, Paul Jungeblut, and Torste Ueckerdt. Efficient Recognition of Subgraphs of Planar Cubic Bridgeless Graphs. In Shiri Chechik, Gonzalo Navarro, Eva Rotenberg, and Grzegorz Herman, editors, *30th Annual European Symposium on Algorithms (ESA 2022)*, volume 244 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 62:1–62:15, 2022. doi:10.4230/LIPIcs.ESA.2022.62.

4 Miriam Goetze, Paul Jungeblut, and Torsten Ueckerdt. Recognition Complexity of Subgraphs of 2- and 3-Connected Planar Cubic Graphs. arXiv preprint, 2024. URL: https://arxiv.org/abs/2401.05892.

5 Tanja Hartmann, Jonathan Rollin, and Ignaz Rutter. Regular augmentation of planar graphs. *Algorithmica*, 73(2):306–370, 2015. doi:10.1007/s00453-014-9922-4.

6 László Lovász. The Factorization of Graphs. II. *Acta Mathematica Academiae Scientiarum Hungarica*, 23(1–2):223–246, 1972. doi:10.1007/BF01889919.

7 András Sebő. General Antifactors of Graphs. *Journal of Combinatorial Theory, Series B*, 58(2):174–184, 1993. doi:10.1006/jctb.1993.1035.

Faster and Deterministic Subtrajectory Clustering

Ivor van der Hoog[1], Thijs van der Horst[2], and Tim Ophelders[2]

1 **Department of Applied Mathematics and Computer Science, Technical University of Denmark, Denmark**
vanderhoog@gmail.com

2 **Department of Information and Computing Sciences, Utrecht University, the Netherlands**
Department of Mathematics and Computer Science, TU Eindhoven, the Netherlands
{t.w.j.vanderhorst|t.a.e.ophelders}@uu.nl

Abstract

We study the subtrajectory clustering problem. Given a trajectory T, the goal is to identify a set of subtrajectories such that each point on T is included in at least one subtrajectory, and subsequently group these subtrajectories together based on similarity under the Fréchet distance. We wish to minimize the set of groups. This problem was shown to be NP-complete by Akitaya, Brüning, Chambers, and Driemel (2021), and the focus has mainly been on approximation algorithms. We study a restricted variant, where we may only pick subtrajectories that start and end at vertices of T, and give an approximation algorithm that significantly improves previous algorithms in both running time and space, whilst being deterministic.

Related Version Full version: arXiv:2402.13117

Acknowledgements

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 899987. Tim Ophelders is partially supported by the Dutch Research Council (NWO) under the project number VI.Veni.212.260.

1 Introduction

Buchin *et al.* [8] proposed the subtrajectory clustering problem. The goal is to partition an input trajectory T of n vertices into subtrajectories, and to group these subtrajectories into *clusters* such that all subtrajectories in a cluster have low Fréchet distance to one another. The clustering under the Fréchet distance is a natural application of Fréchet distance and a well-studied topic [9, 10, 11, 14, 15] that has applications in for example map reconstruction [6, 7]. Throughout recent years, several variants of the algorithmic problem have been proposed [1, 3, 5, 8]. Agarwal *et al.* [1] aim to give a general definition for subtrajectory clustering by defining a function f that evaluates the quality of a set of clusters C. Their definition, however, does not encompass the definition in [8] and has nuances with respect to [3, 5, 13]. Regardless of variant, the subtrajectory clustering problem has been shown to be NP-complete [1, 3, 8]. Agarwal *et al.* [1] therefore propose a bicriterial approximation scheme. They present a heuristic algorithm for the following. Suppose that we are given some $\Delta \geq 0$. Let k denote the smallest integer such that there exists a clustering C with k clusters with score $f(C) \leq \Delta$. The goal is to compute a clustering C' with $\mathcal{O}(k \operatorname{polylog} n)$ clusters and score $f(C) \in \Theta(\Delta)$.

40th European Workshop on Computational Geometry, Ioannina, Greece, March 13–15, 2024.
This is an extended abstract of a presentation given at EuroCG'24. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

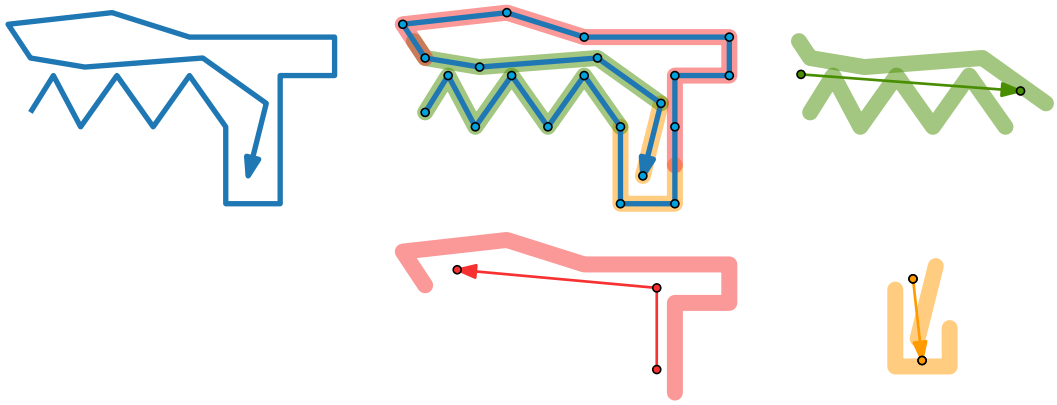


Figure 1 The trajectory T (blue, left) is covered by four pathlets. Each pathlet is given by a reference curve (green, red, yellow) and the subcurve(s) of T the curve covers.

Akitaya *et al.* [3] present a less general, but more computable, bicriterial approximation problem: suppose that we are given some $\Delta \geq 0$ and integer $\ell \geq 1$. An (ℓ, Δ)-clustering is a set C^* of clusters (sets of subtrajectories) where (Figure 1):

- C^* *covers* T: for all points t on T, there is a cluster $Z \in C^*$ and curve $S \in Z$ containing t, and
- every cluster $Z \in C^*$ contains a "reference curve" P_Z with at most ℓ vertices, and
- for every $Z \in C^*$, all subtrajectories $Q \in Z$ have $d_F(P_Z, Q) \leq \Delta$.

Under the discrete Fréchet distance, [3] compute an $(\ell, 19\Delta)$-clustering C of $\mathcal{O}(k\ell^2 \log k\ell)$ size, using $\mathcal{O}(n)$ space and $\tilde{\mathcal{O}}(kn^2)$ expected running time. Brüning *et al.* [5] compute, under the continuous Fréchet distance, an $(\ell, \Theta(\Delta))$-clustering of $\mathcal{O}(k\ell \log k\ell)$ size (where the constant in $\Theta(\Delta)$ is considerably large). Their algorithm uses $\tilde{\mathcal{O}}(n^3)$ space and has $\tilde{\mathcal{O}}(kn^3)$ expected running time. Recently, Conradi and Driemel [13] improve both the size and the distance of the clustering. Under the continuous Fréchet distance, they compute an $(\ell, 11\Delta)$-clustering of $\mathcal{O}(k \log n)$ size using $\tilde{\mathcal{O}}(n^4\ell)$ space and $\tilde{\mathcal{O}}(kn^4\ell + n^4\ell^2)$ time.

Contribution. In this abstract, we give significant improvements in both space and running time complexities, for the restricted case where the clustering needs to be *target-discrete*. A clustering is target-discrete whenever all subtrajectories in a cluster are restricted to have their endpoints lie on vertices of the input. Under this natural restriction, even under the continuous Fréchet distance, our algorithm is near-quadratic and uses near-quadratic space. In the full version we further lower the space used to $\mathcal{O}(n\ell \log n)$. Additionally, in the full version we investigate the unrestricted setting, as well as other variants.

2 Problem Statement

Curves and subcurves. A *curve* (or polyline) with n vertices is a piecewise-linear map $P\colon [1, n] \to \mathbb{R}^d$ whose breakpoints (called *vertices*) are at each integer parameter, and whose pieces are called *edges*. For $1 \leq a \leq b \leq n$, we define the *subcurve* $P[a, b]$ of P that starts at $P(a)$ and ends at $P(b)$. If a and b are integers, we call $P[a, b]$ a *vertex-subcurve* of P.

Fréchet distance. We define the Fréchet distance using the concept of the *free space diagram*. For $\Delta \geq 0$, the Δ-free space diagram is defined as follows.

▶ **Definition 2.1** (Free space diagram)**.** For two curves P and Q with n and m vertices, respectively, the Δ-*free space diagram* Δ-FSD(P,Q) of P and Q is the set of all points $(x,y) \in [1,n] \times [1,m]$ in their parameter space with Euclidean distance $d(P(x),Q(y)) \leq \Delta$.

The *grid cells* of the free space diagram are the squares $[i,i+1] \times [j,j+1]$ for integers i and j. The *obstacle space* of Δ-FSD(P,Q) is its complement $([1,n] \times [1,m]) \setminus \Delta$-FSD$(P,Q)$.

Alt and Godau [4] observed that two curves P and Q have Fréchet distance at most Δ between them precisely if there exists a bimonotone path from $(1,1)$ to (n,m) in Δ-FSD(P,Q).

Input and output. We consider clustering a curve T with n vertices that we call the *trajectory*. Given parameters $\ell \in \mathbb{N}$ and $\Delta \geq 0$, we consider constructing an (ℓ,Δ)-*clustering* of T, which is a set of (ℓ,Δ)-*pathlets*:

▶ **Definition 2.2** (Pathlet)**.** An (ℓ,Δ)-*pathlet* is a tuple $(P,\mathcal{I})$ where P is a curve with at most ℓ vertices and $\mathcal{I}$ is a set of intervals in $[1,n]$ where $d_F(P,T[a,b]) \leq \Delta$ for all $[a,b] \in \mathcal{I}$.

We call P the *reference curve* of $(P,\mathcal{I})$. We call the pathlet *target-discrete* if all intervals in $\mathcal{I}$ have integer endpoints. The *coverage* of a pathlet is $\text{Cov}(P,\mathcal{I}) = \bigcup \mathcal{I}$.

We can see a pathlet $(P,\mathcal{I})$ as a cluster, where the center is P and all subtrajectories induced by $\mathcal{I}$ get mapped to P. The coverage of a set of pathlets C is $\text{Cov}(C) := \bigcup_{(P,\mathcal{I}) \in C} \text{Cov}(P,\mathcal{I})$. An (ℓ,Δ)-*clustering* is a set C of (ℓ,Δ)-pathlets with $\text{Cov}(C) = [1,n]$.

3 Quality of greedy algorithm

Subtrajectory clustering is closely related to the *set cover* problem. In this problem, we have a discrete universe $\mathcal{U}$ and a family of sets $\mathcal{S}$ in this universe, and the goal is to pick a minimum number of sets in $\mathcal{S}$ such that their union is the whole universe. For subtrajectory clustering with target-discrete pathlets, we can set $\mathcal{U} = \{[i,i+1] \mid i \in [1,n-1] \cap \mathbb{N}\}$ and $\mathcal{S} = \{[i,i+1] \in \bigcup \mathcal{I} \mid \text{there exists an } (\ell,\Delta)\text{-pathlet } (P,\mathcal{I})\}$. That is, the universe is the set of intervals parameterizing the edges of T, and $\mathcal{S}$ is the family of sets of edge parameterizations of T that can be covered by a pathlet.

The decision variant of set cover is NP-complete [17]. However, the following greedy strategy gives an $\mathcal{O}(\log n)$ approximation of the minimal set cover size [12]. Suppose we have picked a set $\hat{\mathcal{S}} \subseteq \mathcal{S}$ that does not yet cover all of $\mathcal{U}$. The idea is then to add a set $S \in \mathcal{S}$ that maximizes $S \cap (\mathcal{U} \setminus \hat{\mathcal{S}})$, and repeat the procedure until $\mathcal{U}$ is fully covered.

We use this greedy strategy to focus on constructing a pathlet that covers the most uncovered edges of T. In other words, we greedily grow a set of pathlets C, each time adding a (ℓ,Δ)-pathlet $(P,\mathcal{I})$ that (approximately) maximizes the coverage $\|\text{Cov}(P,\mathcal{I}) \setminus \text{Cov}(C)\|$.

4 Pathlet-preserving simplification

Naively, the subtrajectory clustering problem searches over an infinite number of reference curves for the pathlets, namely, all curves in $\mathbb{R}^d$. In prior work, Brüning *et al.* [5] used a simplification of T that significantly reduces the search space at the cost of increasing Δ by a factor 11, with the assumption that reference curves should be line segments. We propose an alternative solution that increases Δ by only a factor 4, while also giving guarantees for reference curves of arbitrary length.

Our simplification algorithm combines the simplification algorithms of Guibas *et al.* [16] and Agarwal *et al.* [2]. The simplification is essentially a greedy procedure. Consider moving along T, starting at $T(1)$. We look for a locally maximal value t for which $d_F(T[1,t], \overline{T(1)T(t)}) \leq \Delta$ (see fig. 2), and we replace $T[1,t]$ by the directed line segment $\overline{T(1)T(t)}$. We then recursively apply this procedure to the subcurve $T[t,n]$, until $t = n$.

Let S be the resulting curve. In the full version we prove Lemma 4.1.

▶ **Lemma 4.1.** *The curve S has Fréchet distance at most Δ to T, and has the property that for any (ℓ, Δ)-pathlet $(P, \mathcal{I})$, there exists an $(\ell + 2, 4\Delta)$-pathlet $(P', \mathcal{I})$ with the same coverage, where P' is a subcurve of S. Moreover, we can construct S in $\mathcal{O}(n \log n)$ time.*

The consequence of simplification. The above result lets us focus on subcurves of S as reference curves only (see Figure 3). Still, the endpoints of reference curves may lie anywhere on S. However, any such reference curve $S[a,b]$ is either a segment (if $\lfloor a \rfloor = \lfloor b \rfloor$), or it can be decomposed into three subcurves: $S[a, \lceil a \rceil]$, $S[\lceil a \rceil, \lfloor b \rfloor]$ and $S[\lfloor b \rfloor, b]$, the second of which is a vertex-subcurve, and the other two are a prefix and suffix of an edge, respectively. This observation leads to the following algorithm (Algorithm 1) where we iteratively grow a set of (ℓ, Δ')-pathlets C (where $\Delta' \in \Theta(\Delta)$). In essence, we run the greedy algorithm of Section 3. Each iteration, let C be the current set of pathlets and let $(P, \mathcal{I})$ be the (ℓ, Δ)-pathlet that maximizes the coverage $\|\mathrm{Cov}(P, \mathcal{I}) \backslash \mathrm{Cov}(C)\|$. We compute:

- a vertex-subcurve with maximal coverage, and
- a prefix or suffix of an edge e with maximal coverage.

Via our above argument, the coverage $\|\mathrm{Cov}(P', \mathcal{I}') \backslash \mathrm{Cov}(C)\|$ of the pathlet $(P', \mathcal{I}')$ that we compute with maximal coverage is at least $\frac{1}{3}$'rd of $\|\mathrm{Cov}(P, \mathcal{I}) \backslash \mathrm{Cov}(C)\|$.

In this abstract, we restrict ourselves to vertex-subcurves of S, constructing a *vertex-to-vertex* (ℓ, Δ')-pathlet $(P, \mathcal{I})$, where P is a vertex-subcurve of S and $\Delta' = 4\Delta$. In the full version, we show how to construct an optimal $(2, \Delta')$-pathlet $(P, \mathcal{I})$, where P is a prefix or suffix of an edge of S.

5 Constructing pathlets with a given reference curve

Suppose first that we have been given the reference trajectory $S[i, i']$ and need to construct an optimal (ℓ, Δ')-pathlet $(S[i, i'], \mathcal{I})$ using it (for $|S[i, i']| \leq \ell$ and some given Δ'). We then proceed as follows. For every integer $j' \in [1, n]$, we consider the minimum integer j for which $d_F(S[i, i'], T[j, j']) \leq \Delta'$, and if it exists, we create a unique interval $I_{j'} = [j, j']$ and add it to $\mathcal{I}$. This trivially maximizes the coverage $\|\mathrm{Cov}(S[i, i'], \mathcal{I}) \setminus \mathrm{Cov}(C)\|$ for any set of pathlets C, and thus gives an optimal pathlet given $S[i, i']$.

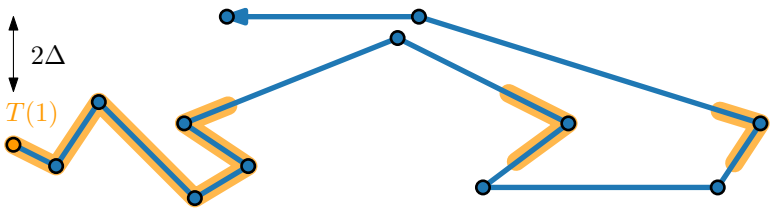


Figure 2 A trajectory T with all points $T(t)$ where $d_F(T[1,t], \overline{T(1)T(t)}) \leq \Delta$ indicated in orange. The last point of each orange connected component may be used for the simplification.

Construct a pathlet-preserving simplification (S, M^{ST}, M^{TS}) of T
Set $\Delta' \leftarrow 4\Delta$
Set $C \leftarrow \emptyset$
while $\mathrm{Cov}(C) \neq [1, n]$ **do**
  Construct a vertex-to-vertex (ℓ, Δ')-pathlet $(P_{\mathrm{ver}}, \mathcal{I}_{\mathrm{ver}})$
  Construct a prefix $(2, \Delta')$-pathlet $(P_{\mathrm{pre}}, \mathcal{I}_{\mathrm{pre}})$
  Construct a suffix $(2, \Delta')$-pathlet $(P_{\mathrm{suf}}, \mathcal{I}_{\mathrm{suf}})$
  Set $(P, \mathcal{I}) \in \{(P_{\mathrm{ver}}, \mathcal{I}_{\mathrm{ver}}), (P_{\mathrm{pre}}, \mathcal{I}_{\mathrm{pre}}), (P_{\mathrm{suf}}, \mathcal{I}_{\mathrm{suf}})\}$ to be the pathlet with maximum coverage over $[1, n] \setminus \mathrm{Cov}(C)$
  Add $(P, \mathcal{I})$ to C
return C

Algorithm 1: SubtrajectoryClustering(T, ℓ, Δ)

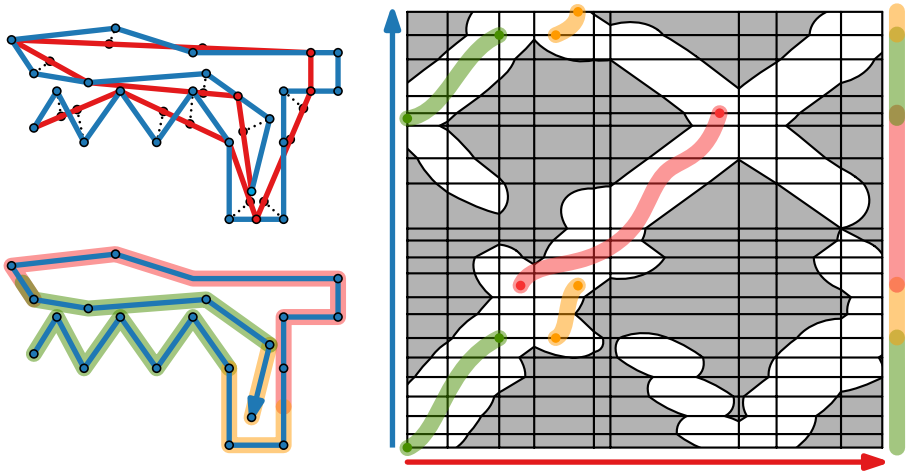


Figure 3 The simplification S of T (top-left, red). The bottom-left clustering of T may be achieved by the right set of paths in Δ'-FSD(S, T). Endpoints of paths with the same color must align vertically, and the horizontal projections of the paths must cover the whole vertical axis.

We compute the above intervals using a data structure that stores information about certain points in the free space diagram. The types of queries we use are:

1. given integers i, i', j and j', determine if $d_F(S[i, i'], T[j, j']) \leq \Delta'$, and
2. given integers i, i' and j', determine if $d_F(S[i, i'], T[j, j']) \leq \Delta'$ for some integer j.

▶ **Lemma 5.1.** *In $\mathcal{O}(n^2 \log n)$ time, we can preprocess S and T into a data structure of $\mathcal{O}(n^2 \log n)$ size, such that queries of type **1** can be answered in $\mathcal{O}(\log n)$ time, and queries of type **2** can be answered in $\mathcal{O}(1)$ time.*

With the above data structure, we compute the intervals $I_{j'}$ in $\mathcal{O}(n \log n)$ time altogether. First, compute the first integer j' for which (i', j') is reachable by some point (i, j) in column i. This takes j' queries of type **2**. All intervals $I_{k'}$ for $k' < j'$ are empty. Next, we compute the first integer j for which $d_F(S[i, i'], T[j, j']) \leq \Delta'$. This takes j queries of type **1**. We then set $I_{j'} = [j, j']$.

Observe that by planarity of the free space diagram, no non-empty interval $I_{k'} = [k, k']$ with $k' \geq j'$ has $k < j$. This is because if $I_{k'} = [k, k']$, then there exists a bimonotone path from (i, k) to (i', k'), which intersects the bimonotone path from (i, j) to (i', j'). Thus there

also exists a bimonotone path from (i, k) to (i', j'), and since $k < j$, we would have that $I_{j'} = [k, j']$.

With the above observation, we can ignore the interval $[1, j-1]$ for the first endpoints of the intervals $I_{k'}$, for $k' > j'$. It follows that we use only $\mathcal{O}(n)$ queries, of either type, to determine all intervals. This gives the following result:

▶ **Theorem 5.2.** *Let C be a set of target-discrete pathlets and $\Delta' \geq 0$. Given the data structure of Lemma 5.1, we can construct in $\mathcal{O}(n \log n)$ time a target-discrete (ℓ, Δ')-pathlet $(P, \mathcal{I})$ maximizing $\|\mathrm{Cov}(P, \mathcal{I}) \setminus \mathrm{Cov}(C)\|$.*

6 Subtrajectory clustering

As stated at the end of Section 4, in the full version we give an algorithm for constructing an optimal $(2, \Delta')$-pathlet $(P, \mathcal{I})$ where P is a subsegment of an edge of S. This algorithm reports an optimal pathlet in $\mathcal{O}(n^2 \log n)$ time, using $\mathcal{O}(n)$ space. Together with the algorithm of Section 5, we can construct an optimal $(\ell, 4\Delta)$-pathlet whose reference curve is either a vertex-subcurve of S or a subsegment of an edge of S. The construction takes $\mathcal{O}(n^2 \ell \log n)$ time and uses $\mathcal{O}(n^2 \log n)$ space. The greedy set cover argument of Section 3 yields:

▶ **Theorem 6.1.** *Let T be a trajectory with n vertices, and let $\ell \in \mathbb{N}$ and $\Delta \geq 0$ be parameters. In $\mathcal{O}(kn^2 \ell \log^2 n)$ time and $\mathcal{O}(n^2 \log n)$ space, we can construct a target-discrete $(\ell, 4\Delta)$-clustering of T with at most $3k \ln n + 1$ pathlets, where k is the minimum number of pathlets in an (ℓ, Δ)-clustering.*

References

1. Pankaj K. Agarwal, Kyle Fox, Kamesh Munagala, Abhinandan Nath, Jiangwei Pan, and Erin Taylor. Subtrajectory clustering: Models and algorithms. In *proc. 37th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems (PODS)*, pages 75–87, 2018. doi:10.1145/3196959.3196972.
2. Pankaj K. Agarwal, Sariel Har-Peled, Nabil H. Mustafa, and Yusu Wang. Near-linear time approximation algorithms for curve simplification. *Algorithmica*, 42(3):203–219, 2005. doi:10.1007/s00453-005-1165-y.
3. Hugo A. Akitaya, Frederik Brüning, Erin Chambers, and Anne Driemel. Subtrajectory clustering: Finding set covers for set systems of subcurves. *Computing in Geometry and Topology*, 2(1):1:1–1:48, 2023. doi:10.57717/cgt.v2i1.7.
4. Helmut Alt and Michael Godau. Computing the Fréchet distance between two polygonal curves. *International Journal of Computational Geometry & Applications*, 5:75–91, 1995. doi:10.1142/S0218195995000064.
5. Frederik Brüning, Jacobus Conradi, and Anne Driemel. Faster approximate covering of subcurves under the fréchet distance. In Shiri Chechik, Gonzalo Navarro, Eva Rotenberg, and Grzegorz Herman, editors, *proc. 30th Annual European Symposium on Algorithms (ESA)*, volume 244 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 28:1–28:16, Dagstuhl, Germany, 2022. doi:10.4230/LIPIcs.ESA.2022.28.
6. Kevin Buchin, Maike Buchin, David Duran, Brittany Terese Fasy, Roel Jacobs, Vera Sacristan, Rodrigo I. Silveira, Frank Staals, and Carola Wenk. Clustering trajectories for map construction. In *proc. 25th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 1–10, 2017. doi:10.1145/3139958.3139964.
7. Kevin Buchin, Maike Buchin, Joachim Gudmundsson, Jorren Hendriks, Erfan Hosseini Sereshgi, Vera Sacristán, Rodrigo I. Silveira, Jorrick Sleijster, Frank Staals, and Carola Wenk. Improved map construction using subtrajectory clustering. In *proc. 4th ACM*

SIGSPATIAL Workshop on Location-Based Recommendations, Geosocial Networks, and Geoadvertising, pages 1–4, 2020. doi:10.1145/3423334.3431451.

8 Kevin Buchin, Maike Buchin, Joachim Gudmundsson, Maarten Löffler, and Jun Luo. Detecting commuting patterns by clustering subtrajectories. *International Journal of Computational Geometry & Applications*, 21(03):253–282, 2011. doi:10.1142/S0218195911003652.

9 Kevin Buchin, Anne Driemel, Joachim Gudmundsson, Michael Horton, Irina Kostitsyna, Maarten Löffler, and Martijn Struijs. Approximating (k, ℓ)-center clustering for curves. In *proc. Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2922–2938, 2019.

10 Maike Buchin and Dennis Rohde. Coresets for (k, ℓ)-median clustering under the Fréchet distance. In *proc. Conference on Algorithms and Discrete Applied Mathematics*, pages 167–180, 2022.

11 Siu-Wing Cheng and Haoqiang Huang. Curve simplification and clustering under Fréchet distance. In *proc. 2023 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1414–1432, 2023.

12 Vasek Chvátal. A greedy heuristic for the set-covering problem. *Mathematics of Operations Research*, 4(3):233–235, 1979. doi:10.1287/MOOR.4.3.233.

13 Jacobus Conradi and Anne Driemel. Finding complex patterns in trajectory data via geometric set cover. *arXiv preprint arXiv:2308.14865*, 2023.

14 Anne Driemel, Amer Krivošija, and Christian Sohler. Clustering time series under the Fréchet distance. In *proc. twenty-seventh annual ACM-SIAM symposium on Discrete algorithms (SODA)*, pages 766–785, 2016.

15 Joachim Gudmundsson and Sampson Wong. Cubic upper and lower bounds for subtrajectory clustering under the continuous Fréchet distance. In *proc. 2022 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 173–189, 2022.

16 Leonidas J. Guibas, John Hershberger, Joseph S. B. Mitchell, and Jack Snoeyink. Approximating polygons and subdivisions with minimum link paths. *International Journal of Computational Geometry & Applications*, 3(4):383–415, 1993. doi:10.1142/S0218195993000257.

17 Richard M. Karp. Reducibility among combinatorial problems. In Raymond E. Miller and James W. Thatcher, editors, *proc. symposium on the Complexity of Computer Computations*, The IBM Research Symposia Series, pages 85–103, 1972. doi:10.1007/978-1-4684-2001-2_9.

Non-degenerate monochromatic triangles in the max-norm plane*

Alexander Natalchenko[1] and Arsenii Sagdeev[2]

1 **MIPT, Moscow, Russia**
`natalchenko.ae@gmail.com`
2 **Alfréd Rényi Institute of Mathematics, Budapest, Hungary**
`sagdeevarsenii@gmail.com`

Abstract

For all non-degenerate triangles T, we determine the minimum number of colors needed to color the plane such that no max-norm isometric copy of T is monochromatic.

Related Version arXiv:2302.09972

1 Introduction

Modern *combinatorial geometry* is both deep and wide with powerful tools galore. Nevertheless, some of its questions, that may look quite simple at first glance, repel all the attempts to answer them for several decades. Perhaps the most famous problem of this sort is due to Nelson who asked in 1950 to find the *chromatic number* $\chi(\mathbb{R}^2)$ of the Euclidean plane defined as the minimum number of colors needed to color the plane $\mathbb{R}^2$ such that no two points at unit Euclidean distance apart are of the same color. Despite the long history of research, it is only known that $5 \leq \chi(\mathbb{R}^2) \leq 7$, where the lower bound was obtained less than five years ago, see [3, 7]. For multidimensional versions of this problem, see [2].

In their celebrated trilogy [4, 5, 6], Erdős, Graham, Montgomery, Rothschild, Spencer, and Straus laid the foundation of *Euclidean Ramsey theory* which deals with questions of the similar flavor but with more complex configurations forbidden to be monochromatic, see [9]. After a pair of points, the second simplest configuration is a (vertex set of a) triangle. We denote by $\chi(\mathbb{R}^2, T)$ the minimum number of colors needed to color the plane such that no *isometric* (i.e., translated and rotated) copy of a triangle T is monochromatic. Erdős et al. conjectured in [6, Conjecture 3] that $\chi(\mathbb{R}^2, T) \geq 3$ for all triangles T except for an equilateral one[1], i.e., that two colors are never enough. Despite the efforts of various researchers, this conjecture was verified only for a few special families of triangles, see [6, 15, 16]. From the other direction, it is easy to see that $\chi(\mathbb{R}^2, T) \leq \chi(\mathbb{R}^2)$ and thus $\chi(\mathbb{R}^2, T) \leq 7$ for all triangles T. Perhaps surprisingly, no better general upper bound is known, though Graham conjectured, see [9, Conjecture 11.1.3] and [17], that $\chi(\mathbb{R}^2, T) \leq 3$ for all triangles T, which was confirmed for 'not very flat' triangles in [1]. Let us also mention that currently the best bounds for multidimensional variant of this problem were recently obtained in [14].

In this paper, we continue the line of research from [8, 11, 12, 13] and consider a max-norm counterpart of the aforementioned problem. To give a formal definition, let us recall some basic notions and facts first. The ℓ_∞-*distance* between $\mathbf{z}_1 = (x_1, y_1)$, $\mathbf{z}_2 = (x_2, y_2) \in \mathbb{R}^2$ is given by $\|\mathbf{z}_1 - \mathbf{z}_2\|_\infty = \max\{|x_1 - x_2|, |y_1 - y_2|\}$. In contrast to the Euclidean case, it is easy

* Supported by ERC Advanced Grant 'GeoScape' No. 882971.

[1] For an equilateral triangle $\triangle$, the same group of authors observed that $\chi(\mathbb{R}^2, \triangle) = 2$ and conjectured in [6, Conjecture 1] that the corresponding two-coloring is unique, which was later disproved in [10].

40th European Workshop on Computational Geometry, Ioannina, Greece, March 13–15, 2024.
This is an extended abstract of a presentation given at EuroCG'24. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

to find the exact value of $\chi(\mathbb{R}^2_\infty)$ defined as the minimum number of colors needed to color the plane such that no two points at unit ℓ_∞-distance apart are of the same color: see the folklore proof that $\chi(\mathbb{R}^2_\infty) = 4$ in the left-hand side of Figure 1 or e.g. in [12, Section 2.1] for more details. A subset $T' \subset \mathbb{R}^2$ is called an *ℓ_∞-isometric copy of* $T \subset \mathbb{R}^2$, if there exists a bijection $f : T \to T'$ such that $\|\mathbf{z}_1 - \mathbf{z}_2\|_\infty = \|f(\mathbf{z}_1) - f(\mathbf{z}_2)\|_\infty$ for all $\mathbf{z}_1, \mathbf{z}_2 \in T$. Finally, we denote by $\chi(\mathbb{R}^2_\infty, T)$ the minimum number of colors needed to color the plane such that no ℓ_∞-isometric copy of T is monochromatic. As earlier, it is easy to see that $\chi(\mathbb{R}^2_\infty, T) \le \chi(\mathbb{R}^2_\infty)$ and thus $\chi(\mathbb{R}^2_\infty, T)$ is equal to either 2, or 3, or 4 for every triangle T. We will show that all three of these options indeed take place.

Note that the value of $\chi(\mathbb{R}^2_\infty, T)$ depends only on the side lengths of T and is independent of the particular position of T in the plane. For all positive $a \le b \le c$ satisfying the triangle inequality $c \le a + b$, let $T(a, b, c)$ be an arbitrary triple of points in the plane with pairwise ℓ_∞-distances between them of a, b, c. This triangle is *degenerate* if $c = a + b$, otherwise it is *non-degenerate*. Observe that if both fractions $\frac{a}{c}$ and $\frac{b}{c}$ are rational, then after a proper scaling, we can assume without loss of generality that a, b, c are coprime integers. For all such non-degenerate triangles T, our next result gives the exact value of $\chi(\mathbb{R}^2_\infty, T)$.

▶ **Theorem 1.1.** *Let $a \le b \le c$ be positive integers such that $c < a + b$ and $\gcd(a, b, c) = 1$. Put $T = T(a, b, c)$. If (1) $a + b + c$ is odd, or (2) a and b are odd, $c \ge a + b - \gcd(a, b)$, then $\chi(\mathbb{R}^2_\infty, T) = 2$. Otherwise, $\chi(\mathbb{R}^2_\infty, T) = 3$.*

Our next result covers the remaining case of 'irrational' non-degenerate triangles.

▶ **Theorem 1.2.** *Let $a \le b \le c$ be positive reals such that $c < a + b$ and $\frac{a}{c}$ or $\frac{b}{c}$ is irrational. Put $T = T(a, b, c)$. If $a = q_1\xi$, $b = q_2\eta$, $c = p_1\xi + p_2\eta$ for some odd integers p_1, p_2, q_1, q_2 and reals ξ, η such that $\frac{\xi}{\eta}$ is irrational, then $\chi(\mathbb{R}^2_\infty, T) = 3$. Otherwise, $\chi(\mathbb{R}^2_\infty, T) = 2$.*

For a degenerate triangle $T = T(a, b, a + b)$, several results follow from [8], where much more general problems were studied. First, observe that every five-point subset of a nine-element set $\{0, a, a+b\}^2 \subset \mathbb{R}^2$ contains an ℓ_∞-isometric copy of T and thus $\chi(\mathbb{R}^2_\infty, T) \ge 3$. If $\frac{a}{b}$ is irrational, the axiom of choice allows one to construct the corresponding three-coloring of the plane showing that this bound is tight, see [8, Section 5]. Otherwise, after a proper scaling, we can assume without loss of generality that a and b are coprime integers. In case $a \equiv b \pmod 3$, it is again not hard to construct a coloring of the plane matching the aforementioned lower bound, see the right-hand side of Figure 1 for an illustration and [8, Section 4] for a formal proof. In the remaining case $a \not\equiv b \pmod 3$, we conjecture that three colors are not enough, which we verified for $a + b \le 7$ by computer search.

▶ **Conjecture 1.3.** *If $a, b \in \mathbb{N}$ are such that $a \not\equiv b \pmod 3$, then $\chi\big(\mathbb{R}^2_\infty, T(a, b, a + b)\big) = 4$.*

In what follows, we refer to *ℓ_∞-distances* and *ℓ_∞-isometric copies* simply as *distances* and *copies*, respectively. Whenever we consider a two-coloring of the plane, we call these colors red and blue or 0 and 1 for clarity. We also assume that side lengths a, b, c of a triangle $T = T(a, b, c)$ are reals satisfying $a \le b \le c$ and $c < a + b$, i.e., that T is non-degenerate.

We structure the reminder of the paper as follows. In Section 2, we give a necessary condition for a triple of points to form a copy of T. In Section 3, we find some properties satisfied by every red-blue coloring of the plane containing no monochromatic copies of T, for one of which, namely for Lemma 3.3, we do not provide a proof in this version of the article. Though this proof is similar to others and based on an absolutely elementary idea (if two vertices of T are red, then the third one must be blue, and vice versa), its exact implementation is not that simple, and we have to omit it in order to meet the condition

on manuscript length. Finally, in Section 4, we show that these properties are mutually exclusive if T satisfies neither (1) nor (2), which would complete the proof of Theorem 1.1. Note that Theorem 1.2 follows from these properties in a very similar manner, but we have to omit this proof too.

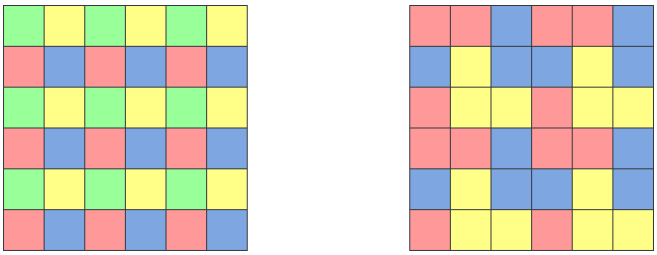


Figure 1 A 4-coloring of the plane with no monochromatic points at unit distance apart, and a 3-coloring of the plane with no monochromatic copies of degenerate triangles $T(a, b, a+b)$ for all coprime $a, b \in \mathbb{N}$ such that $a \equiv b \pmod 3$. All squares are unit. Each colored square includes only the bottom left vertex along with the left and the bottom sides from its boundary.

2 Copies of a non-degenerate triangle

▶ **Lemma 2.1.** *Let $\mathbf{z}_1 = (x_1, y_1), \mathbf{z}_2 = (x_2, y_2), \mathbf{z}_3 = (x_3, y_3) \in \mathbb{R}^2$ be a copy of T. Then at least one of the differences $|y_1 - y_2|, |y_2 - y_3|, |y_3 - y_1|$ equals either a, or b, or c. Moreover, at least one of the differences $|x_1 + y_1 - x_2 - y_2|, |x_2 + y_2 - x_3 - y_3|, |x_3 + y_3 - x_1 - y_1|$ equals either $a + b - c$, or $c + a - b$, or $b + c - a$.*

Proof. The distance between two points is determined by the absolute value of the difference of their either x- or y-coordinates. Therefore, one of the axes, say x, determines at least two of the distances $\|\mathbf{z}_1 - \mathbf{z}_2\|_\infty, \|\mathbf{z}_2 - \mathbf{z}_3\|_\infty, \|\mathbf{z}_3 - \mathbf{z}_1\|_\infty$ by the pigeonhole principle. Assume that $\|\mathbf{z}_2 - \mathbf{z}_3\|_\infty = |x_2 - x_3| = a$, $\|\mathbf{z}_3 - \mathbf{z}_1\|_\infty = |x_3 - x_1| = b$. Observe that x_3 cannot lie between x_1 and x_2, since in that case we would get that $c = \|\mathbf{z}_1 - \mathbf{z}_2\|_\infty \geq |x_1 - x_2| = a + b$, a contradiction. Hence, let us assume that $x_1 = x_3 + b$, $x_2 = x_3 + a$. This implies that $|x_1 - x_2| = b - a < c = \|\mathbf{z}_1 - \mathbf{z}_2\|_\infty$ and thus $|y_1 - y_2| = c$ as desired. To prove the second half of the statement, note that if $y_1 = y_2 + c$, then $|x_1 + y_1 - x_2 - y_2| = b + c - a$, while if $y_1 = y_2 - c$, then $|x_1 + y_1 - x_2 - y_2| = c + a - b$. The same reasoning works in all the remaining cases, corresponding to possible permutations of axes, indexes, and side lengths. ◀

This simple statement immediately gives the following sufficient condition for a *horizontal coloring*, which is constant on every horizontal line $y = y_0$, or for a *diagonal coloring*, which is constant on every diagonal line $x + y = y_0$, to contain no monochromatic copies of T.

▶ **Corollary 2.2.** *The following two statements are valid:*

1. *Let $\bar{C}(\cdot)$ be a coloring of the line such that no two points at distance a, b, or c apart are monochromatic. Then the corresponding horizontal coloring of the plane, defined by the equation $C(x, y) = \bar{C}(y)$, contains no monochromatic copies of T.*
2. *Let $C'(\cdot)$ be a coloring of the line such that no two points at distance $a + b - c$, $c + a - b$, or $b + c - a$ apart are monochromatic. Then the corresponding diagonal coloring of the plane, defined by the equation $C(x, y) = C'(x + y)$, contains no monochromatic copy of T.*

3 Patterns in an arbitrary two-coloring

For this section, let us fix a red-blue coloring of the plane such that no copy of T is monochromatic. We call a vector (x_0, y_0) its *period* (resp. *anti-period*) if for all $x, y \in \mathbb{R}$, the colors of two points (x, y) and $(x + x_0, y + y_0)$ are the same (resp. distinct). It is clear that the addition of these vectors resembles the multiplication of signs: the sum of two periods or two anti-periods is a period, while the sum of an anti-period and a period is an anti-period.

▶ **Lemma 3.1.** *If one of the four vectors $(\pm b, a - c)$, $(a - c, \pm b)$ is not an anti-period, then there exists a monochromatic axis-parallel segment of length $c + a - b$. Similar statements are valid for other permutations of the side length.*

Proof. If $(a - c, b)$ is not an anti-period, then there exist $x_1, y_1 \in \mathbb{R}$ such that the two points (x_1, y_1) and $(x_1 + c - a, y_1 - b)$ are of the same color, say, both are red. It is easy to check that this pair together with an arbitrary point from the segment $\{(x_1+c, y) : y_1-c \le y \le y_1+a-b\}$ form a copy of T. Since there are no red copies of T, we conclude that this vertical segment of length $c+a-b$ is entirely blue, as desired. Similar arguments work for the other cases. ◀

▶ **Corollary 3.2.** *If no axis-parallel segment of length $c + a - b$ is monochromatic, then all eight vectors $(\pm a, c - b)$, $(c - b, \pm a)$, $(\pm b, a - c)$, $(a - c, \pm b)$ are anti-periods, and all six vectors $(2a, 0)$, $(2b, 0)$, $(2c, 0)$, $(0, 2a)$, $(0, 2b)$, $(0, 2c)$ are periods. Moreover, if there also exist $n, m, k \in \mathbb{Z}$ such that $0 < 2an + 2bm + 2ck \le a + b - c$, then four vectors $(\pm c, b - a)$, $(b - a, \pm c)$ are anti-periods as well.*

▶ **Lemma 3.3.** *If $a < b$ and there exists a monochromatic axis-parallel segment of length $c + a - b$, then there also exists a monochromatic axis-parallel line.*

▶ **Lemma 3.4.** *If the horizontal line $y = 0$ is red, then both lines $y = a$ and $y = b$ are blue. Moreover, if there also exist $n, m \in \mathbb{Z}$ such that $n + m$ is even and $c - b \le an + bm \le a$, then the line $y = c$ is blue as well.*

Proof. Each point (x_0, a) on the line $y = a$ forms a copy of T together with two red points $(x_0 - c, 0)$ and $(x_0 + b - c, 0)$. Thus the line $y = a$ is entirely blue. Similarly, each point (x_0, b) on the line $y = b$ forms a copy of T together with two red points $(x_0 - c, 0)$ and $(x_0 + a - c, 0)$. Thus the line $y = a$ is also entirely blue.

To prove the second half of the statement, observe that the first half implies that the color of the horizontal line $y = an + bm$ is determined by the parity of $n + m$. In particular, if $n + m$ is even, then this line is red. Now it is easy to see that each point (x_0, c) on the line $y = c$ forms a copy of T together with two red points $(x_0 - b, an + bm)$ and $(x_0 + a - b, 0)$. Hence, the line $y = c$ is also entirely blue, as desired. ◀

4 Proof of Theorem 1.1

First of all, note that the upper bound $\chi(\mathbb{R}^2_\infty, T) \le 3$ is immediate from the first half of Corollary 2.2 and the following special case of [18, Corrolary 2.1] due to Zhu.

▶ **Theorem 4.1.** *Let $a \le b \le c$ be positive integers such that $c < a + b$ and $\gcd(a, b, c) = 1$. Then there exists a three-coloring of the line such that no two points at distance a, b, or c apart are monochromatic.*

Therefore, to complete the proof, we only need to show that there exists a two-coloring of the plane such that no copy of T is monochromatic if and only if either (1) or (2) holds. We begin by showing the sufficiency of these conditions using the following two explicit colorings.

▶ **Lemma 4.2.** *If* (1) *holds, then no copy of* T *is monochromatic under a diagonal two-coloring of the plane defined by* $C(x, y) = \lfloor x + y \rfloor \bmod 2$.

Proof. It is clear that $\lfloor x_1 \rfloor \not\equiv \lfloor x_2 \rfloor \pmod 2$ whenever $x_1, x_2 \in \mathbb{R}$ are at odd distance apart. Since all three values $a + b - c$, $c + a - b$, and $b + c - a$ are odd by (1), the second half of Corollary 2.2 completes the proof. ◀

▶ **Lemma 4.3.** *If* (2) *holds, then no copy of* T *is monochromatic under a horizontal two-coloring of the plane defined by* $C(x, y) = \lfloor y/d \rfloor \bmod 2$, *where* $d = \gcd(a, b)$.

Proof. Assume the contrary, namely that for some $\mathbf{z}_1 = (x_1, y_1), \mathbf{z}_2 = (x_2, y_2), \mathbf{z}_3 = (x_3, y_3) \in \mathbb{R}^2$ that form a copy of T, the values $\lfloor y_1/d \rfloor, \lfloor y_2/d \rfloor, \lfloor y_3/d \rfloor$ are of the same parity, say all three are even. By Lemma 2.1, we can assume without loss of generality that $y_1 - y_2$ equals either a, or b, or c. Note that the former two cases immediately yield a contradiction since both fractions a/d and b/d are odd by (2). So in what follows we suppose that $y_1 - y_2 = c$. In particular, this implies that $\|\mathbf{z}_1 - \mathbf{z}_2\|_\infty = c$, and thus one of the distances $\|\mathbf{z}_2 - \mathbf{z}_3\|_\infty$, $\|\mathbf{z}_3 - \mathbf{z}_1\|_\infty$ equals a, while the other one equals b.

It is easy to check that if $\|\mathbf{z}_2 - \mathbf{z}_3\|_\infty = a$, $\|\mathbf{z}_3 - \mathbf{z}_1\|_\infty = b$, then $y_1 - b \le y_3 \le y_2 + a$. Observe that both $\lfloor (y_1 - b)/d \rfloor = \lfloor y_1/d \rfloor - b/d$ and $\lfloor (y_2 + a)/d \rfloor = \lfloor y_2/d \rfloor + a/d$ are odd. Moreover, the length of this segment is equal to $a + b - c$ which does not exceed d by (2). Therefore, this segment is too short for the parity of $\lfloor y/d \rfloor$ to change from odd to even and back again as y ranges between the endpoints. Hence, $\lfloor y_3/d \rfloor$ is also odd, and we see the contradiction. In the remaining case when $\|\mathbf{z}_2 - \mathbf{z}_3\|_\infty = b$, $\|\mathbf{z}_3 - \mathbf{z}_1\|_\infty = a$, we have $y_1 - a \le y_3 \le y_2 + b$, and the similar argument completes the proof. ◀

To prove the second half of the theorem, observe that if neither (1) nor (2) holds, then either
(3) a and b are of different parity, c is odd, or
(4) a and b are odd, c is even, and $c < a + b - \gcd(a, b)$.
So it remains only to show that in each of these two cases, there are no two-colorings of the plane with no monochromatic copies of T. Let us assume the contrary and fix an arbitrary such red-blue coloring.

First, we suppose that no axis-parallel segment of length $c + a - b$ is monochromatic. On the one hand, the first half of Corollary 3.2 implies that all six vectors $(2a, 0)$, $(2b, 0)$, $(2c, 0)$, $(0, 2a)$, $(0, 2b)$, $(0, 2c)$ are periods, and so are all their linear combinations. Since $\gcd(2a, 2b, 2c) = 2$, we conclude that both $(2, 0)$ and $(0, 2)$ are periods. Hence, every vector such that both its coordinates are even integers is also a period. On the other hand, note that $a + b - c$ is a positive even integer, and thus $a + b - c \ge 2 = \gcd(2a, 2b, 2c)$. Therefore, we can also apply the second half of Corollary 3.2 in our case to find twelve anti-periods in total including $(a, c - b)$, $(b, a - c)$ and $(c, b - a)$. However, both coordinates of one of these three vectors are even integers, and so this vector should be a period instead, a contradiction.

Second, suppose that there exists a monochromatic axis-parallel segment of length $c+a-b$. Besides, note that each of the conditions (3) and (4) yields $a < b$. So we can apply Lemma 3.3 to find a monochromatic axis-parallel line. Without loss of generality, assume that this line is given by the equation $y = 0$ and that it is entirely red. Now it is easy to deduce from the first half of Lemma 3.4 that for all $i, j \in \mathbb{Z}$ such that $i + j$ is odd, the horizontal line $y = ai + bj$ is blue. If (3) holds, then we obtain a contradiction by taking $i = b$, $j = -a$.

If (4) holds, we use a slightly more complex argument. Observe that there exist $n, m \in \mathbb{Z}$ such that $an + bm = a - \gcd(a, b)$. Since both a and b are odd, we conclude that $n + m$ is even. Moreover, the inequality $c - b \le a - \gcd(a, b) = an + bm$ is immediate from (4). Therefore, we can aslo apply the second half of Lemma 3.4 in our case to deduce that the

horizontal line $y = ai + cj$ is blue for all $i, j \in \mathbb{Z}$ such that $i + j$ is odd. Finally, we obtain the desired contradiction by taking $i = c$, $j = -a$.

References

1 Oswin Aichholzer and Daniel Perz. Triangles in the colored euclidean plane. In *35th European Workshop on Computational Geometry (EuroCG 2019)*, 2019.

2 D. Cherkashin, A. Kulikov, and A. Raigorodskii. On the chromatic numbers of small-dimensional euclidean spaces. *Discrete Applied Mathematics*, 243:125–131, 2018.

3 Aubrey de Grey. Chromatic number of the plane is at least 5. *Geombinatorics*, 28(1):559–583, 2018.

4 P. Erdős, R.L. Graham, P. Montgomery, B.L. Rothschild, J. Spencer, and E.G. Straus. Euclidean ramsey theorems i. *J. Combin. Theory, Ser. A*, 14:341–363, 1973.

5 P. Erdős, R.L. Graham, P. Montgomery, B.L. Rothschild, J. Spencer, and E.G. Straus. Euclidean ramsey theorems ii. *Colloq. Math. Soc. J. Bolyai, Infinite and Finite Sets*, 10:529–557, 1975.

6 P. Erdős, R.L. Graham, P. Montgomery, B.L. Rothschild, J. Spencer, and E.G. Straus. Euclidean ramsey theorems iii. *Colloq. Math. Soc. J. Bolyai, Infinite and Finite Sets*, 10:559–583, 1975.

7 Geoffrey Exoo and Dan Ismailescu. The chromatic number of the plane is at least 5: a new proof. *Discrete & Computational Geometry*, 64(1):216–226, 2020.

8 Nóra Frankl, Andrey Kupavskii, and Arsenii Sagdeev. Max-norm ramsey theory. *arXiv:2111.08949*, 2021.

9 R.L. Graham. Euclidean ramsey theory. In *Handbook of discrete and computational geometry, third edition*, pages 281–297. Chapman and Hall/CRC, 2017.

10 Vít Jelínek, Jan Kynčl, Rudolf Stolař, and Tomáš Valla. Monochromatic triangles in two-colored plane. *Combinatorica*, 29(6):699–718, 2009.

11 Valeriya Kirova and Arsenii Sagdeev. Two-colorings of normed spaces without long monochromatic unit arithmetic progressions. *arXiv:2203.04555*, 2022.

12 Andrey Kupavskii and Arsenii Sagdeev. All finite sets are ramsey in the maximum norm. In *Forum of Mathematics, Sigma*, volume 9. Cambridge University Press, 2021.

13 Andrey Kupavskii, Arsenii Sagdeev, and Nóra Frankl. Infinite sets can be ramsey in the chebyshev metric. *Russian Mathematical Surveys*, 77(3):549–551, 2022.

14 Andrey Kupavskii, Arsenii Sagdeev, and Dmitrii Zakharov. Cutting corners. *arXiv:2211.17150*, 2022.

15 Leslie E Shader. All right triangles are ramsey in e^2! *Journal of Combinatorial Theory, Series A*, 20(3):385–389, 1976.

16 Ilya D Shkredov. On some problems of euclidean ramsey theory. *Analysis Mathematica*, 41(4):299–310, 2015.

17 A Soifer. Triangles in a three-colored plane. *Geombinatorics*, 1(2):11–12, 1991.

18 Xuding Zhu. Circular chromatic number of distance graphs with distance sets of cardinality 3. *Journal of Graph Theory*, 41:195 – 207, 2002.

Fully Dynamic Maximum Independent Sets of Disks in Polylogarithmic Update Time

Sujoy Bhore[1], Martin Nöllenburg[2], Csaba D. Tóth[3], and Jules Wulms[4]

1 **Indian Institute of Technology Bombay**
sujoy@cse.iitb.ac.in
2 **Algorithms and Complexity Group, TU Wien**
noellenburg@ac.tuwien.ac.at
3 **California State University Northridge; Tufts University**
csaba.toth@csun.edu
4 **TU Eindhoven**
j.j.h.m.wulms@tue.nl

Abstract

A fundamental question is whether one can maintain a maximum independent set (MIS) in polylogarithmic update time for a dynamic collection of geometric objects in Euclidean space. For a set of intervals, it is known that no dynamic algorithm can maintain an exact MIS in sublinear update time. Therefore, the typical objective is to explore the trade-off between update time and solution size. Substantial efforts have been made in recent years to understand this question for various families of geometric objects, such as intervals, hypercubes, hyperrectangles, and fat objects.

We present the first fully dynamic approximation algorithm for disks of arbitrary radii in the plane that maintains a constant-factor approximate MIS in polylogarithmic expected amortized update time. Moreover, for a fully dynamic set of n unit disks in the plane, we show that a 12-approximate MIS can be maintained with worst-case update time $O(\log n)$, and optimal output-sensitive reporting.

Related Version When details are missing, refer to the full version at arxiv.org/abs/2308.00979.

1 Introduction

The maximum independent set (MIS) problem is a fundamental problem in theoretical computer science, and it is one of Karp's 21 classical NP-complete problems [18]. In the MIS problem, we are given a graph $G = (V, E)$, and the objective is to choose a subset $S \subseteq V$ of maximum cardinality such that no two vertices in S are adjacent. The intractability of MIS carries even under strong algorithmic paradigms. For instance, it is known to be hard to approximate: no polynomial-time algorithm can achieve an approximation factor $n^{1-\varepsilon}$ (for $|V| = n$ and a constant $\varepsilon > 0$) unless P=ZPP [23]. In fact, even if the maximum degree of G is bounded by 3, no polynomial-time approximation scheme (PTAS) is possible [4].

Geometric Independent Set. In geometric settings, the input is a collection $\mathcal{L} = \{\ell_1, \ldots, \ell_n\}$ of geometric objects, e.g., intervals, disks, squares, rectangles, etc., and we wish to compute a maximum independent set in their intersection graph G: Each vertex in G corresponds to an object in $\mathcal{L}$, and each edge connects the vertices of two intersecting objects. Thus a MIS of G corresponds to a maximum cardinality subset $\mathcal{L}' \subseteq \mathcal{L}$ of pairwise disjoint objects. A large body of work has been devoted to geometric MIS problems, due to their wide applicability, for example in scheduling [2], VLSI design [16], map labeling [1], and data mining [19, 3].

40th European Workshop on Computational Geometry, Ioannina, Greece, March 13–15, 2024.
This is an extended abstract of a presentation given at EuroCG'24. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

Objects	Approx. Ratio	Update time	Reference
Intervals	$1+\varepsilon$	$O(\varepsilon^{-1}\log n)$	[12]
Squares	$O(1)$	$O(\log^5 n)$ amortized	[5]
Arbitrary radii disks	$O(1)$	$(\log n)^{O(1)}$ expec. amortized	Theorem 1.2
Unit disks	$O(1)$	$O(\log n)$ worst-case	Theorem 1.1
	$1+\varepsilon$	$n^{(1/\varepsilon)^{\Omega(1)}}$	Full version [8]
f-fat objects in $\mathbb{R}^d$	$O_f(1)$	$O_f(\log n)$ worst-case	Full version [8]
d-dimensional hypercubes	$(1+\varepsilon)\cdot 2^d$	$O_{d,\varepsilon}(\log^{2d+1} n \cdot \log^{2d+1} U)$	[15]

Table 1 Summary of results on dynamic independent sets for n geometric objects.

Dynamic Geometric Independent Set. In dynamic settings, objects are inserted into or deleted from the collection $\mathcal{L}$ over time. The typical objective is to achieve (almost) the same approximation ratio as in the offline (static) case while keeping the update time, i.e., the time to update the solution after insertion/deletion, as small as possible. We call this the *Dynamic Geometric Maximum Independent Set* problem (for short, DGMIS).

Henzinger et al. [15] studied DGMIS for various geometric objects, such as intervals, hypercubes, and hyperrectangles. Many of their results extend to the weighted version of DGMIS, as well. Based on a lower bound of Marx [22] for the offline problem, they showed that any dynamic $(1+\varepsilon)$-approximation for squares in the plane requires $\Omega(n^{1/\varepsilon})$ update time for any $\varepsilon > 0$, ruling out the possibility of sub-polynomial time dynamic approximation schemes. On the positive side, they obtained dynamic algorithms with update time polylogarithmic in both n and N, where the corners of the objects are in a $[0, N]^d$ integer grid, for any constant dimension d (therefore their aspect ratio is also bounded by N). Bhore et al. [5] presented the first fully dynamic algorithms with polylogarithmic update time for DGMIS, where the input objects are intervals and axis-aligned squares. For intervals, they presented a fully dynamic $(1+\varepsilon)$-approximation algorithm with logarithmic update time. Later, Compton et al. [12] achieved a faster update time for intervals, by using a new partitioning scheme. Recently, Bhore et al. [6] studied the MIS problem for intervals in the streaming settings, and obtained lower bounds.

Moreover, Bhore et al. [7] studied the DGMIS problem in the context of dynamic map labeling and presented dynamic algorithms for several subfamilies of rectangles that also perform well in practice. Cardinal et al. [9] designed dynamic algorithms for fat objects in fixed dimension d with sublinear worst-case update time. However, despite the remarkable progress on the DGMIS problem in recent years, the following question remained unanswered.

▶ Question 1. Does an algorithm exist that, for a given dynamic set of disks in the plane, maintains a constant-factor approximate MIS in polylogarithmic update time?

Our Contributions In this paper, we answer Question 1 in the affirmative (Theorems 1.1–1.2); see Table 1. As a first step, we address the case of unit disks in the plane.

▶ **Theorem 1.1.** *For a fully dynamic set of unit disks in the plane, a 12-approximate MIS can be maintained with worst-case update time $O(\log n)$, and optimal output-sensitive reporting.*

We prove Theorem 1.1 in the full version [8]. Similarly to classical approximation algorithms for the static problem [16], we lay out four shifted grids such that any unit disk lies in a grid cell for at least one of the grids, see Figure 1. For each grid, we maintain an independent set that contains at most one disk from each grid cell, thus we obtain four

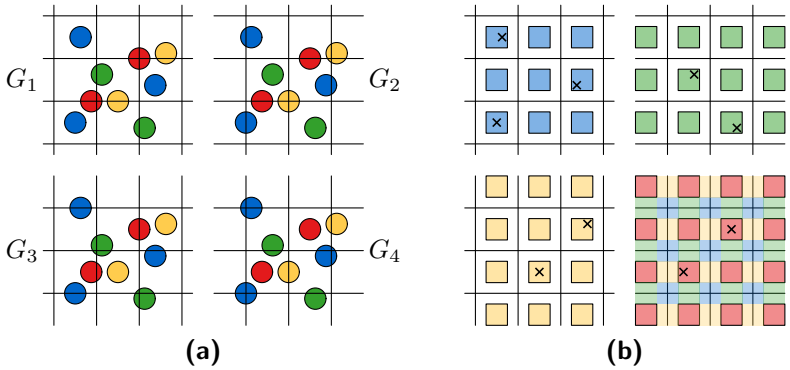


Figure 1 (a) The four shifted grids $G_1, \ldots, G_4$, which respectively do not intersect the blue, green, yellow, and red disks. **(b)** The radius-1 squares inside grid cells, along with the center points of the disks that lie completely inside grid cells, as crosses. In the bottom right, besides red squares for G_4, the squares of all other grids are added to show that the squares together partition the plane.

independent sets $S_1, \ldots, S_4$ at all times, where the largest is a constant-factor approximation of the MIS. Using the MIX algorithm [9], we can maintain an independent set $S \subset \bigcup_{i=1}^{4} S_i$ of size $\Omega(\max\{|S_1|, |S_2|, |S_3|, |S_4|\})$ at all times, which is a $O(1)$-approximation of the MIS.

Moreover, our dynamic data structure for unit disks easily generalizes to fat objects of comparable sizes in $\mathbb{R}^d$ for any constant dimension $d \in \mathbb{N}$ (see the full version [8]).

Our main result is a dynamic data structure for MIS of disks of arbitrary radii in $\mathbb{R}^2$.

▶ **Theorem 1.2.** *For a fully dynamic set of disks of arbitrary radii in the plane, an $O(1)$-approximate MIS can be maintained in polylogarithmic expected amortized update time.*

To prove Theorem 1.2 in Section 2, we extend the core ideas developed for unit disks.

Finally, we note that, even for a dynamic set of unit disks in the plane, it is impossible to maintain a $(1+\varepsilon)$-approximate MIS with amortized update time $n^{O((1/\varepsilon)^{1-\delta})}$ for any ε, $\delta > 0$, unless the Exponential Time Hypothesis (ETH) fails. This follows from a reduction to a result by Marx [22], resembling the same result for hypercubes by Henzinger et al. [15].

2 Disks of Arbitrary Radii in the Plane

Summary of our data structures and update algorithms. When considering disks of arbitrary radii, the general idea of our new data structure is to break the set of disks $\mathcal{D}$ into subsets of disks of comparable radius. We will use several instances of shifted grids $G_1^i, \ldots, G_4^i$, as we also used in the unit disk case, where the grid cells now have side length 3^i, are shifted by $\frac{3^i}{2}$, and store disks with radius r, where $\frac{3^{i-1}}{4} < r \leq \frac{3^i}{4}$, for $i \in \mathbb{Z}$. The resulting hierarchies of recursively 3×3 subdivided grid cells form so-called *nonatrees*.

The main algorithmic ideas in using these nonatrees revolve around a bottom-up traversal of the nonatree using a well-known greedy strategy [21, 13]. In the static case, greedily considering fat objects in ascending order of size allows us to find a constant-factor approximate MIS. In the dynamic case, we want to mimic this idea by traversing paths in a nonatree towards the root. However, the height of such a nonatree (even compressed) may be $\Theta(n)$ for n disks (see Figure 2). Thus, for dynamic updates we cannot afford to traverse ascending paths in their entirety with polylogarithmic update time.

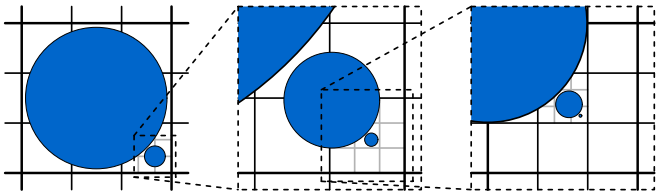


Figure 2 A nonatree with height linear in the number of stored disks, whose radii decay exponentially. A compressed nonatree (with compressed nodes) has linear height.

Approximating a Maximum Independent Set. Regardless of the above observation, we intend to traverse the nonatrees in bottom-up fashion, computing an $O(1)$-approximate MIS. In the dynamic setting, we then ensure that we only have to update the nonatrees locally.

We start by explaining how we compute an $O(1)$ approximation with our nonatrees. We refer to the data structures G_k^i associated with each value $i \in \mathbb{Z}$ as a *bucket*, and we will use only those buckets that store any disks, which we call *relevant* buckets. Within these buckets, we call grid cells that contain disks the *relevant* grid cells. Furthermore, to prevent computational overhead, our nonatrees are *compressed*, similar to compressed quadtrees [14, Chapter 2]. Figure 3 illustrates the concepts in the previous and upcoming paragraph.

Two crucial high-level steps can be distinguished in our approach:

1. For each cell $c \in G_k^i$ in our compressed nonatrees, we communicate upwards which disks have been included in our independent set. To do so, we use *obstacle disks*, and only input disks disjoint from obstacle disks can be chosen in the independent set. Once all cells are handled, we output the largest independent sets computed for the nonatrees $N_1, \ldots, N_4$, to get an $O(1)$-approximation (Lemmata 6-8 in the full version [8]).
2. We want an obstacle disk for a cell c to cover the disks in the independent set selected in the subtree rooted at c, to prevent overlaps. The obstacle disks for a cell c is hence defined as the smallest enclosing disk of c. Additionally, if the independent set of the children originates from more than one child, we do not add a disk from c, even if possible. We still obtain an $O(1)$ approximation under these constraints (Lemmata 9 and 10 in [8]).

▶ **Lemma 2.1.** *For a set of disks in the plane, one of our shifted nonatrees $N_1, \ldots, N_4$ maintains an independent set of size $\Omega(|\mathrm{OPT}|)$, where* OPT *is a MIS.*

Modifications to Support Dynamic Maintenance. We now highlight two changes in the above data structures, to support efficient updates, while maintaining an $O(1)$-approximation. Dynamic updates trigger bottom-up traversals through a nonatree, and these changes are aimed at keeping updates local, leading to expected amortized polylogarithmic update time.

Obstacle cells. To support dynamic updates, we use slightly enlarged obstacle disks to prevent cascading effects during updates. Obstacle disks are associated with so-called obstacle cells of the nonatree N_k. Those cells that contribute to independent set S_k, are called *true obstacles*. Cells of the nonatree with two or more children are also considered as obstacle cells and are *merge obstacles*. The obstacle cells decompose the nonatree into ascending paths in which each cell has relevant descendants in only one subtree (see Figure 4a). Inside an ascending path, disks either intersect the obstacle disk of the (closest) obstacle cell below them, or are part of S_k and therefore define a true obstacle cell (see Figures 4b and 4c).

▶ **Lemma 2.2.** *A disk d in cell $c \in N_k$ added to S_k can intersect only the disk $d_o \in S_k$ in the next obstacle cell c_o on the ascending path $P(d)$ from c towards the root, if d_o even exists.*

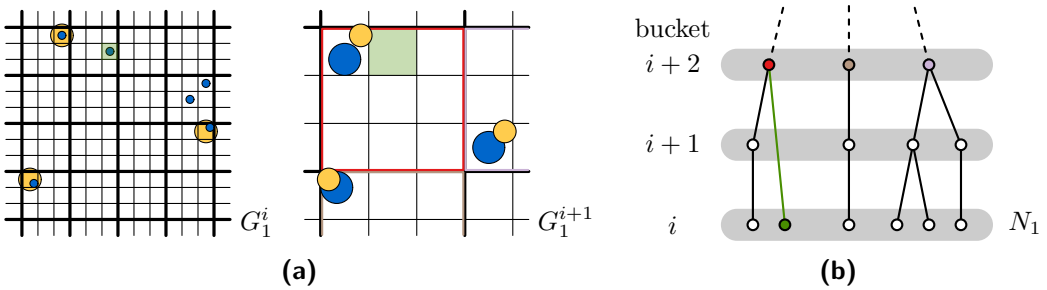


Figure 3 (a) Two compatible grids in buckets i and $i+1$, with (blue) disks of D in relevant cells. In particular, the green cell in G_1^i is relevant, but its (green) parent cell in G_1^{i+1} is not. Three (yellow) obstacle disks of G_1^i are drawn in both grids. Only one blue disk in G_1^{i+1} is disjoint from an obstacle, and can be chosen in the greedy bottom-up strategy. **(b)** Part of the compressed nonatree N_1 corresponding to **(a)**: The colored nodes of bucket $i+2$ correspond to colored squares in **(a)** of the same color. Because the green cell in G_1^{i+1} is not relevant, and does not have relevant children in two subtrees, it is not represented in N_1. Instead, the green node, corresponding to the green relevant cell in G_1^i, directly connects to an ancestor in bucket $i+2$ (by the green edge).

Barrier disks. The naïve approach for a dynamic update of the independent set S in a nonatree N would work as follows: When a new disk d is inserted or deleted, we find a nonatree N and a cell $c \in N$ associated with d; and then in an ascending path of N from c to the root, we re-compute the disks in S to repair the greedy bottom-up property. Unfortunately, we cannot afford this (recall Fig. 2). Instead, we run the greedy process only locally, on an ascending path of N between two cells $c_1 \prec c_2$ that contain disks $s_1, s_2 \in S$, respectively. Here, $\prec$ denotes that c_1 is a descendant of c_2. The greedy process guarantees that new disks added to S are disjoint from any smaller disk in S, including s_1, but they might intersect the larger disk $s_2 \in S$. In this case, we remove s_2 from S, keep it as a "placeholder" in a set B of *barrier disks*, and ensure that $S \cup B$ is a dominating set of $\mathcal{D}$. This is one of the invariants that we maintain to show that the described changes still result in a $O(1)$-approximate MIS (Lemma 16 in [8]). Furthermore, we maintain an assignment β between barrier disks and the closest obstacle cells below them. Each barrier disk $\beta(c_1)$ lies along an ascending path between two obstacle cells $c_1 \prec c_2$. Importantly, another of our invariants states that each ascending path contains at most one barrier disk.

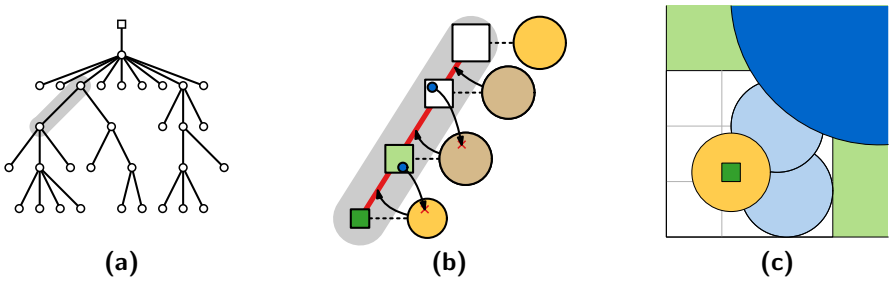


Figure 4 (a) Decomposition of a nonatree into ascending paths between merge obstacle cells. Only relevant leaves are drawn. One ascending path between merge nodes is highlighted in grey. This path is shown **(b)** abstractly and **(c)** geometrically: The merge obstacle cells at the top and bottom (with yellow obstacle disks) each have no disk of S_k associated with them. Every other obstacle cell on the path also defines a brown obstacle disk. Each such cell contains a (dark blue) disk of S_k, disjoint from the obstacle disk below it (indicated by red crosses). All (light blue) disks on the (red) ascending path intersect the obstacle below. Green colors identify cells in **(b)** and **(c)**.

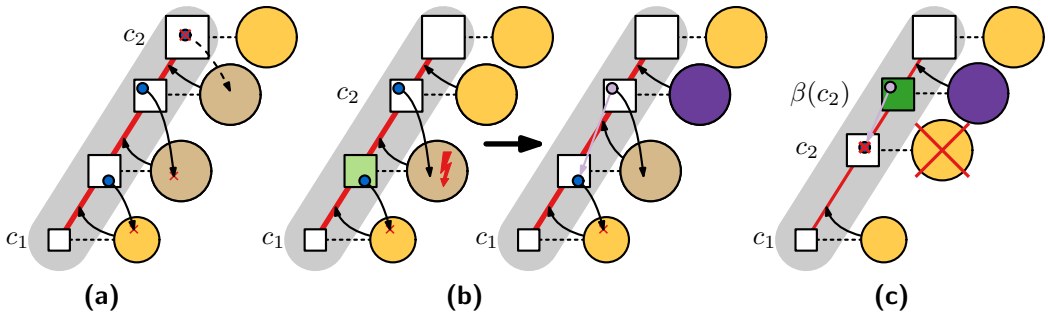


Figure 5 Greedy updates in an ascending path: **(a)** There is no disk $s_2 \in S_k$ in c_2 that can intersect the new (brown) obstacle disks in the gray ascending path. **(b)** The disk $s_2 \in S_k$ in c_2 is turned into a barrier if it overlaps the obstacle disk of the highest new disk in the light green cell. **(c)** If $\beta(c_2)$ exists, remove c_2 from S_k and run the greedy algorithm up to the dark green cell.

Dynamic maintenance using farthest neighbor data structures. We now sketch how to maintain our data structures with polylogarithmic update times. One key component is the use of the dynamic farthest neighbor (DFN) data structure by Kaplan et al. [17] (generalizing Chan's famous dynamic convex hull data structure [10, 11]). We adapt this data structure to efficiently find disks that are disjoint from obstacle disks in ascending paths of our nonatrees in polylogarithmic time, and with polylogarithmic expected amortized update time.

When a disk d associated with a cell $c \in N_k$ is inserted or deleted, then c lies in an ascending path $P(d)$ between two obstacle cells, say $c_1 \preceq c \prec c_2$. To update the independent set S_k and the barrier disks B_k, in general we run the greedy algorithm in this path. The greedy process queries the DFN data structure to find disks that are disjoint from any smaller disk in S_k. Now we distinguish between three cases (see Figure 5): (a) If c_2 is a merge obstacle cell, then we are done. (b) However, if c_2 is a true obstacle cell, then the last disk added to S_k may intersect the disk $s_2 \in S_k$ associated with c_2 (and only s_2, by Lemma 2.2). If so, we delete s_2 from S_k, insert it into B_k, and assign it to the highest disk in S_k in $P(d)$ below s_2.(c) Finally, if s_2 was already associated with a barrier disk, $\beta(c_2)$, then adding s_2 to B_k would result in two barrier disks between consecutive obstacle cells, which is not allowed. For this reason, if $\beta(c_2)$ exists, we remove s_2 from S_k, run the greedy algorithm up to the cell associated with $\beta(c_2)$, and then reassign $\beta(c_2)$ to the highest disk added to S_k.

3 Conclusions

One bottleneck in our framework is the nearest/farthest neighbor data structure [17, 20], which provides only *expected amortized* polylogarithmic update time. This is the only reason why our algorithm does not guarantee deterministic worst-case update time, and it does not extend to balls in $\mathbb{R}^d$ for $d \geq 3$, or to arbitrary fat objects in $\mathbb{R}^2$. It remains open whether there is a dynamic nearest/farthest neighbor data structure in constant dimensions $d \geq 2$ with a worst-case polylogarithmic update and query time: Such a result would immediately carry over to a fully dynamic algorithm for an approximate MIS for balls in higher dimensions.

References

1 Pankaj K Agarwal, Marc Van Kreveld, and Subhash Suri. Label placement by maximum independent set in rectangles. *Computational Geometry*, 11(3-4):209–218, 1998. doi:10.1016/S0925-7721(98)00028-5.

2 Reuven Bar-Yehuda, Magnús M Halldórsson, Joseph Naor, Hadas Shachnai, and Irina Shapira. Scheduling split intervals. *SIAM Journal on Computing*, 36(1):1–15, 2006. doi:10.1137/S0097539703437843.

3 Piotr Berman, Bhaskar DasGupta, S. Muthukrishnan, and Suneeta Ramaswami. Efficient approximation algorithms for tiling and packing problems with rectangles. *J. Algorithms*, 41(2):443–470, 2001. doi:10.1006/jagm.2001.1188.

4 Piotr Berman and Toshihiro Fujito. On approximation properties of the independent set problem for low degree graphs. *Theory of Computing Systems*, 32:115–132, 1999. doi:10.1007/s002240000113.

5 Sujoy Bhore, Jean Cardinal, John Iacono, and Grigorios Koumoutsos. Dynamic geometric independent set. In *Abstracts of 23rd Thailand-Japan Conference on Discrete and Computational Geometry, Graphs, and Games (TJDCG)*, 2021. arXiv:2007.08643.

6 Sujoy Bhore, Fabian Klute, and Jelle J. Oostveen. On streaming algorithms for geometric independent set and clique. In *Proc. 20th International Workshop on Approximation and Online Algorithms (WAOA)*, volume 13538 of *LNCS*, pages 211–224. Springer, 2022. doi:10.1007/978-3-031-18367-6_11.

7 Sujoy Bhore, Guangping Li, and Martin Nöllenburg. An algorithmic study of fully dynamic independent sets for map labeling. *ACM Journal of Experimental Algorithmics (JEA)*, 27(1):1–36, 2022. doi:10.1145/3514240.

8 Sujoy Bhore, Martin Nöllenburg, Csaba D. Tóth, and Jules Wulms. Fully dynamic maximum independent sets of disks in polylogarithmic update time. *CoRR*, abs/2308.00979, 2023. arXiv:2308.00979, doi:10.48550/ARXIV.2308.00979.

9 Jean Cardinal, John Iacono, and Grigorios Koumoutsos. Worst-case efficient dynamic geometric independent set. In *Proc. 29th European Symposium on Algorithms (ESA)*, volume 204 of *LIPIcs*, pages 25:1–25:15, 2021. See also arXiv:2108.08050. arXiv:arXiv:2108.08050.

10 Timothy M. Chan. A dynamic data structure for 3-D convex hulls and 2-D nearest neighbor queries. *J. ACM*, 57(3):16:1–16:15, 2010. doi:10.1145/1706591.1706596.

11 Timothy M. Chan. Dynamic geometric data structures via shallow cuttings. *Discret. Comput. Geom.*, 64(4):1235–1252, 2020. doi:10.1007/s00454-020-00229-5.

12 Spencer Compton, Slobodan Mitrovic, and Ronitt Rubinfeld. New partitioning techniques and faster algorithms for approximate interval scheduling. In *Proc. 50th International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 261 of *LIPIcs*, pages 45:1–45:16. Schloss Dagstuhl, 2023. doi:10.4230/LIPIcs.ICALP.2023.45.

13 Alon Efrat, Matthew J. Katz, Frank Nielsen, and Micha Sharir. Dynamic data structures for fat objects and their applications. *Comput. Geom.*, 15(4):215–227, 2000. doi:10.1016/S0925-7721(99)00059-0.

14 Sariel Har-Peled. *Geometric Approximation Algorithms*, volume 173 of *Mathematical Surveys and Monographs*. AMS, 2011. URL: https://bookstore.ams.org/surv-173/.

15 Monika Henzinger, Stefan Neumann, and Andreas Wiese. Dynamic approximate maximum independent set of intervals, hypercubes and hyperrectangles. In *Proc. 36th International Symposium on Computational Geometry (SoCG)*, volume 164 of *LIPIcs*, pages 51:1–51:14, 2020. doi:10.4230/LIPIcs.SoCG.2020.51.

16 Dorit S. Hochbaum and Wolfgang Maass. Approximation schemes for covering and packing problems in image processing and VLSI. *J. ACM*, 32(1):130–136, 1985. doi:10.1145/2455.214106.

17 Haim Kaplan, Wolfgang Mulzer, Liam Roditty, Paul Seiferth, and Micha Sharir. Dynamic planar Voronoi diagrams for general distance functions and their algorithmic applications. *Discret. Comput. Geom.*, 64(3):838–904, 2020. doi:10.1007/s00454-020-00243-7.

18 Richard M. Karp. Reducibility among combinatorial problems. In *Proceedings of a symposium on the Complexity of Computer Computations*, The IBM Research Symposia Series, pages 85–103. Plenum Press, New York, 1972. doi:10.1007/978-1-4684-2001-2_9.

19 Sanjeev Khanna, Shan Muthukrishnan, and Mike Paterson. On approximating rectangle tiling and packing. In *Proc. 9th ACM-SIAM Symposium on Discrete algorithms (SODA)*, volume 98, pages 384–393, 1998. URL: https://dl.acm.org/doi/10.5555/314613.314768.

20 Chih-Hung Liu. Nearly optimal planar k nearest neighbors queries under general distance functions. *SIAM J. Comput.*, 51(3):723–765, 2022. doi:10.1137/20m1388371.

21 Madhav V. Marathe, Heinz Breu, Harry B. Hunt III, Sekharipuram S. Ravi, and Daniel J. Rosenkrantz. Simple heuristics for unit disk graphs. *Networks*, 25(2):59–68, 1995. doi:10.1002/net.3230250205.

22 Dániel Marx. On the optimality of planar and geometric approximation schemes. In *Proc. 48th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 338–348, 2007. doi:10.1109/FOCS.2007.26.

23 David Zuckerman. Linear degree extractors and the inapproximability of max clique and chromatic number. *Theory Comput.*, 3(1):103–128, 2007. doi:10.4086/toc.2007.v003a006.

Robust Bichromatic Classification using Two Lines

Erwin Glazenburg[1], Thijs van der Horst[1,2], Tom Peters[2], Bettina Speckmann[2], and Frank Staals[1]

1 **Utrecht University**
[e.p.glazenburg, t.w.j.vanderhorst, f.staals]@uu.nl
2 **TU Eindhoven**
[t.peters1, b.speckmann]@tue.nl

Abstract

Given two sets R and B of at most n points in the plane, we present efficient algorithms to find a two-line linear classifier that best separates the "red" points in R from the "blue" points in B and is robust to outliers. More precisely, we find a region $\mathcal{W}_B$ bounded by two lines, so either a halfplane, strip, wedge, or double wedge, containing (most of) the blue points B, and few red points. Our running times vary between optimal $O(n \log n)$ and $O(n^4)$, depending on the type of region $\mathcal{W}_B$ and whether we wish to minimize only red outliers, only blue outliers, or both.

Related Version A full version can be found on arXiv [5]

1 Introduction

Let R and B be two sets of at most n points in the plane. Our goal is to best separate the "red" points R from the "blue" points B using at most two lines. That is, we wish to find a region $\mathcal{W}_B$ bounded by lines ℓ_1 and ℓ_2 containing (most of) the blue points B, so that the number k_R of points from R in the interior $\text{int}(\mathcal{W}_B)$ of $\mathcal{W}_B$ and/or the number k_B of points from B in the interior $\text{int}(\mathcal{W}_R)$ of the region $\mathcal{W}_R = \mathbb{R}^2 \setminus \mathcal{W}_B$ is minimized. We refer to these sets of red and blue outliers as $\mathcal{E}_R = R \cap \text{int}(\mathcal{W}_B)$ and $\mathcal{E}_B = B \cap \text{int}(\mathcal{W}_R)$, respectively, and define $\mathcal{E} = \mathcal{E}_R \cup \mathcal{E}_B$ and $k = k_R + k_B$.

Region $\mathcal{W}_B$ is either: (i) a halfplane, (ii) a *strip* bounded by two parallel lines ℓ_1 and ℓ_2, (iii) a *wedge*, i.e. one of the four regions induced by a pair of intersecting lines ℓ_1, ℓ_2, or (iv) a *double wedge*, i.e. two opposing regions induced by a pair of intersecting lines ℓ_1, ℓ_2. See Figure 1. We can reduce the case that $\mathcal{W}_B$ would consist of three regions to the single-wedge case, by recoloring the points. For each of these cases for the shape of $\mathcal{W}_B$ we consider three problems: allowing only red outliers ($k_B = 0$) and minimizing k_R, allowing only blue outliers ($k_R = 0$) and minimizing k_B, or allowing both outliers and minimizing k. We present efficient algorithms for each of these problems, see Table 1.

Related work. Binary classification is a key problem in computer science. Linear classifiers such as SVMs [3] compute a hyperplane separating R and B; when R and B are not linearly

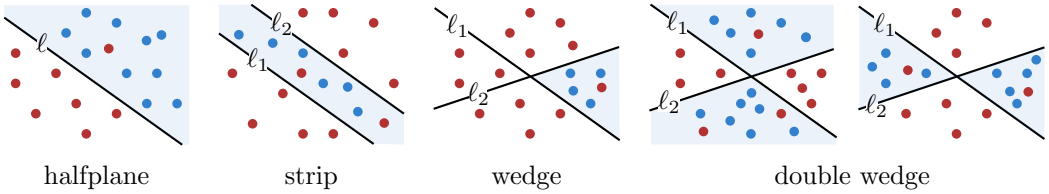


Figure 1 We consider separating R and B by at most two lines. This gives rise to four types of regions $\mathcal{W}_B$; halfplanes, strips, wedges, and two types of double wedges; hourglasses and bowties.

40th European Workshop on Computational Geometry, Ioannina, Greece, March 13–15, 2024.
This is an extended abstract of a presentation given at EuroCG'24. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

region $\mathcal{W}_B$	minimize k_R	minimize k_B	minimize k
halfplane	$O(n \log n)$ ⋆	$O(n \log n)$ ⋆	$O((n+k^2)\log n)$ [2]
strip	$\Theta(n \log n)$ [8], §3	$O(n^2 \log n)$ ⋆	$O(n^2 \log n)$ ⋆
wedge	$O(n^2)$ [8]	$O(n^2 k_B$	$O((n^2 k + nk^3)$
	$O(n \log n)$ §4	$\log n \log k_B)$ ⋆	$\log n \log k)$ ⋆
double (bowtie) wedge	$O(n^2)$ §5	$O(n^2 \log n)$ ⋆	$O(n^4)$ ⋆

Table 1 An overview of our results. A star ⋆ means this result is shown in the full version.

separable like in Figure 2 one could try using different (non-linear) separators, or allowing for outliers. Hurtato et al. [6, 7] give $O(n \log n)$ algorithms for perfectly separating R and B using two lines (i.e. a strip, wedge or double wedge) without outliers, which are optimal [1]. Alternatively, Chan [2] presented algorithms for linear programming in constant dimension that allow for up to k violations, and thus solve hyperplane separation with up to k outliers.

A combination of the above, i.e. using more general separators while giving guarantees on the number of outliers, seems to be less well studied. Seara [8] showed how to compute a strip containing all blue points and minimal red points in $O(n \log n)$ time, and a wedge with the same properties in $O(n^2)$ time. In this paper, we take some further steps toward the fundamental problem of computing robust non-linear separators with performance guarantees.

Results. We present efficient algorithms for computing a region $\mathcal{W}_B$ (strip, wedge, or double wedge) that minimizes red (k_R), blue (k_B), or both (k) outliers. Refer to Table 1 for an overview. In this extended abstract we focus on three entries of Table 1: minimizing k_R for strips (Section 3), wedges (Section 4), and double wedges (Section 5). The other results and omitted proofs can be found in the full version [5] on arXiv.

Most notably, our optimal $\Theta(n \log n)$ algorithm for computing a wedge minimizing k_R improves the earlier $O(n^2)$ time algorithm from Seara [8]. We also provide the first algorithms for minimizing k_B for strips, wedges, and double wedges, and surprisingly these problems seem more difficult than their counterpart of minimizing k_R.

2 Preliminaries

We assume $B \cup R$ contains at least three points and is in general position, i.e. no two points have the same x- or y-coordinate, and no three points are co-linear.

Notation. Let ℓ^- and ℓ^+ be the two halfplanes bounded by line ℓ, with ℓ^- below ℓ (or left of ℓ if ℓ is vertical). Any pair of lines ℓ_1 and ℓ_2, with the slope of ℓ_1 smaller than that of ℓ_2, subdivides the plane into at most four interior-disjoint regions $\text{North}(\ell_1, \ell_2) = \ell_1^+ \cap \ell_2^+$, $\text{East}(\ell_1, \ell_2) = \ell_1^+ \cap \ell_2^-$, $\text{South}(\ell_1, \ell_2) = \ell_1^- \cap \ell_2^-$ and $\text{West}(\ell_1, \ell_2) = \ell_1^- \cap \ell_2^+$. When ℓ_1 and ℓ_2

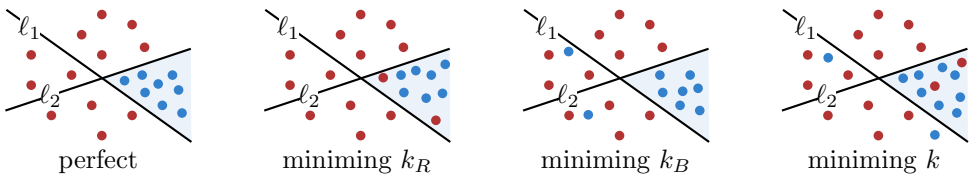


Figure 2 When considering outliers, we may allow only red outliers, only blue outliers, or both.

are clear from the context we may simply write North to mean North(ℓ_1, ℓ_2) etc. We assign each of these regions to either B or R, so that $\mathcal{W}_B$ $(= \mathcal{W}_B(\ell_1, \ell_2))$ and $\mathcal{W}_R$ $(= \mathcal{W}_R(\ell_1, \ell_2))$ are the union of some elements of $\{\text{North}, \text{East}, \text{South}, \text{West}\}$. In case ℓ_1 and ℓ_2 are parallel, we assume that ℓ_1 lies below ℓ_2, and thus $\mathcal{W}_B = \text{East}$.

Duality. We make frequent use of the standard point-line duality [4], where we map objects in *primal* space to objects in a *dual* space. In particular, a primal point $p = (a, b)$ is mapped to the dual line $p^* : y = ax - b$ and a primal line $\ell : y = ax + b$ is mapped to the dual point $\ell^* = (a, -b)$. If primal point p lies above line ℓ, then dual line p^* lies below point ℓ^*.

For a set of lines L, we are often interested in the *arrangement* $\mathcal{A}(L)$, i.e. the vertices, edges, and faces formed by the lines in L. Let $\mathcal{U}(L)$ be the upper envelope of L, i.e. the polygonal chain following the highest line in $\mathcal{A}(L)$, and $\mathcal{L}(L)$ the lower envelope.

Property of an optimal wedge. It can be shown that, for any (double) wedge classification problem, there exists an optimum where both lines go through a blue and a red point. Therefore there exists a somewhat simple $O(n^4)$ algorithm for finding (double) wedges minimizing either k_R, k_B, or k, which considers all pairs of lines through red and blue points.

3 Strip separation with red outliers

We first consider the case where W_B forms a strip, bounded by parallel lines ℓ_1 and ℓ_2, with ℓ_2 above ℓ_1. We want B to be inside the strip, and R outside, and here we show how to minimize red outliers k_R. We do this in the dual, where we want to find two points ℓ_1^* and ℓ_2^* with the same x-coordinate such that vertical segment $\overline{\ell_1^* \ell_2^*}$ intersects all blue lines and as few red lines as possible. Note that ℓ_1^* must be above $\mathcal{U}(B^*)$ and ℓ_2^* must be below $\mathcal{L}(B^*)$. Since shortening a segment can not make it intersect more red lines, we can even assume they lie exactly on the envelopes.

As $\mathcal{U}(B^*)$ and $\mathcal{L}(B^*)$ are x-monotone, there is only one degree of freedom for choosing our segment: its x-coordinate. We parameterize $\mathcal{U}(B^*)$ and $\mathcal{L}(B^*)$ over $\mathbb{R}$, our one-dimensional *parameter space*, such that each point $p \in \mathbb{R}$ corresponds to the vertical segment $\overline{\ell_1^* \ell_2^*}$ on the line $x = p$. We wish to find a point in this parameter space, i.e. an x-coordinate, whose corresponding segment minimizes the number of red misclassifications. Let the *forbidden regions* of a red line r be those intervals on the parameter space in which corresponding segments intersect r. We distinguish between four types of red lines, as in Figure 3:

- Line a intersects $\mathcal{U}(B^*)$ in points a_1 and a_2, with $a_1 \leq a_2$. Segments with ℓ_1^* left of a_1 or right of a_2 misclassify a, so a produces two forbidden intervals: $(-\infty, a_1)$ and (a_2, ∞).
- Line b intersects $\mathcal{L}(B^*)$ in points b_1 and b_2, with $b_1 \leq b_2$. Similar to line a this produces forbidden intervals $(-\infty, b_1)$ and (b_2, ∞).
- Line c intersects $\mathcal{L}(B^*)$ in c_1 and $\mathcal{U}(B^*)$ in c_2. Only segments between c_1 and c_2 misclassify c. This gives one forbidden interval: $(\min\{c_1, c2\}, \max\{c_1, c_2\})$.
- Line d intersects neither $\mathcal{U}(B^*)$ nor $\mathcal{L}(B^*)$. All segments misclassify d. This gives one trivial forbidden region, namely the entire space $\mathbb{R}$.

The above list is exhaustive. To see this, note that the two lines supporting the unbounded edges of $\mathcal{U}(B^*)$ also support the unbounded edges of $\mathcal{L}(B^*)$.

Our goal is to find a point that lies in as few of these forbidden regions as possible. We can compute such a point in $O(n \log n)$ time by sorting and scanning. Computing $\mathcal{U}(B^*)$ and $\mathcal{L}(B^*)$ takes $O(n \log n)$ time. Given a red line $r \in R^*$ we can compute its intersection points

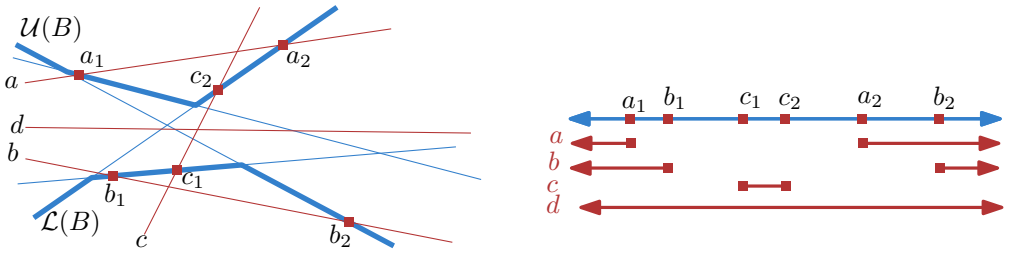


Figure 3 Four types of red lines for strip separation, with restrictions on their parameter space.

with $\mathcal{U}(B^*)$ and $\mathcal{L}(B^*)$ in $O(\log n)$ time using binary search (since $\mathcal{U}(B^*)$ and $\mathcal{L}(B^*)$ are convex). Computing the forbidden regions thus takes $O(n \log n)$ time in total. We conclude:

▶ **Theorem 3.1.** *Given two sets of n points $B, R \subset \mathbb{R}^2$, we can construct a strip $\mathcal{W}_B$ minimizing the number of red outliers k_R in $O(n \log n)$ time.*

4 Wedge separation with red outliers

We consider the case where the region $\mathcal{W}_B$ is a single wedge and $\mathcal{W}_R$ is the other three wedges. Here we show how to compute an optimal East or West wedge minimizing red outliers, i.e. we compute two lines ℓ_1 and ℓ_2 such that every blue point and as few red points as possible lie above ℓ_1 and below ℓ_2. In the dual this corresponds to two points ℓ_1^* and ℓ_2^* such that all blue lines and as few red lines as possible lie below ℓ_1^* and above ℓ_2^*, as in Figure 4. In the full version, we compute an optimal North or South wedge in a similar way.

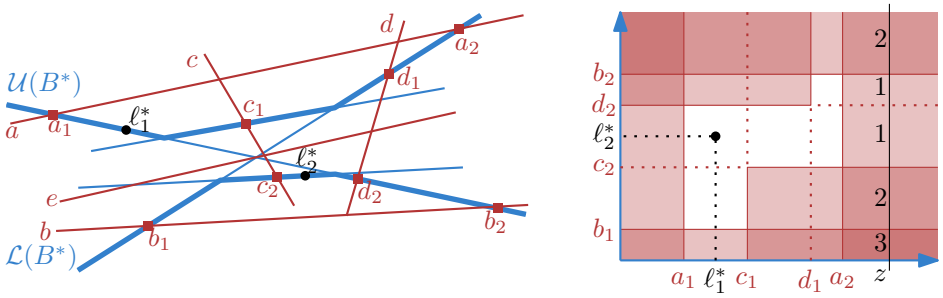


Figure 4 The arrangement of $B^* \cup R^*$ with its parameter space and forbidden regions.

Clearly ℓ_1^* must lie above $\mathcal{U}(B^*)$, and ℓ_2^* below $\mathcal{L}(B^*)$; as in the strip case, we can even assume they lie exactly on $\mathcal{U}(B^*)$ and $\mathcal{L}(B^*)$. Similar to the case of strips we parameterize $\mathcal{U}(B^*)$ and $\mathcal{L}(B^*)$ over $\mathbb{R}^2$, such that a point (p, q) in this two-dimensional parameter space corresponds to two dual points ℓ_1^* and ℓ_2^*, with ℓ_1^* on $\mathcal{U}(B^*)$ at $x = p$ and ℓ_2^* on $\mathcal{L}(B^*)$ at $x = q$. See Figure 4. We wish to find a value in our parameter space whose corresponding segment minimizes the number of red misclassifications. Let the forbidden regions of a red line r again be those regions in the parameter space in which corresponding segments misclassify r. We distinguish between five types of red lines, as in Figure 4 (left):

- Line a intersects $\mathcal{U}(B^*)$ in a_1 and a_2, with a_1 left of a_2. Only segments with ℓ_1^* left of a_1 or right of a_2 misclassify a. This produces two forbidden regions: $(-\infty, a_1) \times (-\infty, \infty)$ and $(a_2, \infty) \times (-\infty, \infty)$.
- Line b intersects $\mathcal{L}(B^*)$ in b_1 and b_2, with b_1 left of b_2. Symmetric to line a this produces forbidden regions $(-\infty, \infty) \times (-\infty, b_1)$ and $(-\infty, \infty) \times (b_2, \infty)$.

- Line c intersects $\mathcal{U}(B^*)$ in c_1 and $\mathcal{L}(B^*)$ in c_2, with c_1 left of c_2. Only segments with endpoints after c_1 and before c_2 misclassify c, producing the region $(c_1, \infty) \times (-\infty, c_2)$. (Segments with endpoints before c_1 and after c_2 do intersect c, but do not misclassify it)
- Line d intersects $\mathcal{U}(B^*)$ in d_1 and $\mathcal{L}(B^*)$ in d_2, with d_1 right of d_2. Symmetric to line c it produces the forbidden region $(-\infty, d_1) \times (d_2, \infty)$.
- Line e intersects neither $\mathcal{U}(B^*)$ nor $\mathcal{L}(B^*)$. All segments misclassify e. This produces one forbidden region; the entire plane $\mathbb{R}^2$.

Our goal is again to find a point that lies in as few of these forbidden regions as possible. Since all regions are axis-aligned rectangles, we can do so using a simple sweepline algorithm in $O(n \log n)$ time. Constructing $\mathcal{U}(B^*)$ and $\mathcal{L}(B^*)$, finding the intersections of every red line r with $\mathcal{U}(B^*)$ and $\mathcal{L}(B^*)$, determining the type of r $(a - e)$, and constructing its forbidden regions all take $O(n \log n)$ time as well.

▶ **Theorem 4.1.** *Given two sets of n points $B, R \subset \mathbb{R}^2$, we can construct an* East *or* West *wedge containing all points of B and the fewest points of R in $O(n \log n)$ time.*

5 Double wedge separation with red outliers

Although the wedge algorithm was a direct extension of the strip algorithm, the double wedge algorithm uses different techniques, which we briefly review; see the full version for details. We consider finding a *bowtie* wedge $\mathcal{W}_B$ while minimizing red outliers, i.e. all of B and as little of R as possible lies in the West and East wedge. In the dual this corresponds to a line segment intersecting all of B^*, and as little of R^* as possible.

Observe that a segment intersecting all lines of B^* must have endpoints in antipodal outer faces of $\mathcal{A}(B^*)$, i.e. two opposite outer faces sharing the same two infinite bounding lines. For all $O(n)$ pairs of antipodal faces, we could apply a very similar algorithm to the wedge algorithm in Section 4, resulting in $O(n \cdot n \log n) = O(n^2 \log n)$ time.

Alternatively, we construct the entire arrangement $\mathcal{A}(B^* \cup R^*)$ of all lines explicitly in $O(n^2)$ time (see e.g. [4]). Consider a pair of faces P and Q that are antipodal in $\mathcal{A}(B^*)$, and assume w.l.o.g. they are separated by the x-axis, with P above Q. There are two types of red lines: *splitting* lines that intersect both P and Q once, and *stabbing* lines that intersect at most one of P and Q, see Figure 5. A red line is a splitting line for exactly one pair of antipodal faces, while it can be a stabbing line for multiple pairs. Recall that we wish to find a segment from P to Q intersecting as few red lines as possible. The s splitting lines divide the boundary of P and Q into $s + 1$ chains $P_0..P_s$ $(Q_0..Q_s)$. Within one such chain P_i on P we only need to consider the point p_i with the most stabbing lines above it: a segment from p_i to Q will not intersect those lines, since Q is below P_i. Similarly, we only need to consider point q_j on chain Q_j with the most stabbing lines below it. Using dynamic programming we can then find the pair of chains P_i, Q_j such that $\overline{p_i q_j}$ intersects the fewest red lines in $O(n + s^2)$ time. Doing so for all pairs of antipodal faces yields a total running time of $O(n^2)$.

▶ **Theorem 5.1.** *Given two sets of n points $B, R \subset \mathbb{R}^2$, we can construct the bowtie double wedge $\mathcal{W}_B$ minimizing the number of red outliers k_R in $O(n^2)$ time.*

Consider the related problem of finding a bowtie wedge while minimizing k_B, which we solve in $O(n^2 \log n)$ time in the full version. Note that we can not just recolor the points and use the above $O(n^2)$ time algorithm: after recoloring, we would wish to find a blue hourglass wedge minimizing k_R, which is a different problem. Therefore, unfortunately, finding any double wedge (bowtie or hourglass) while minimizing k_R still takes $O(n^2 \log n)$ time.

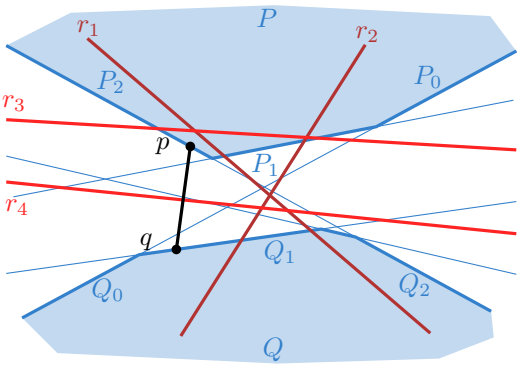


Figure 5 Two antipodal faces P and Q, with two splitting lines r_1, r_2 and two stabbing lines r_3, r_4, and an optimal segment $\overline{pq}$ from P to Q.

References

1 Esther M. Arkin, Ferran Hurtado, Joseph S. B. Mitchell, Carlos Seara, and Steven Skiena. Some lower bounds on geometric separability problems. *International Journal of Computational Geometry & Applications*, 16(1):1–26, 2006. doi:10.1142/S0218195906001902.

2 Timothy M. Chan. Low-dimensional linear programming with violations. *SIAM Journal on Computing*, 34(4):879–893, 2005. doi:10.1137/S0097539703439404.

3 Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, Sep 1995. doi:10.1007/BF00994018.

4 Mark de Berg, Otfried Cheong, Marc J. van Kreveld, and Mark H. Overmars. *Computational geometry: algorithms and applications, 3rd Edition*. Springer, 2008.

5 Erwin Glazenburg, Thijs van der Horst, Tom Peters, Bettina Speckmann, and Frank Staals. Robust bichromatic classification using two lines, 2024. arXiv:2401.02897.

6 Ferran Hurtado, Mercè Mora, Pedro A. Ramos, and Carlos Seara. Separability by two lines and by nearly straight polygonal chains. *Discrete Applied Mathematics*, 144(1-2):110–122, 2004. doi:10.1016/j.dam.2003.11.014.

7 Ferran Hurtado, Marc Noy, Pedro A. Ramos, and Carlos Seara. Separating objects in the plane by wedges and strips. *Discrete Applied Mathematics*, 109(1-2):109–138, 2001. doi:10.1016/S0166-218X(00)00230-4.

8 Carlos Seara. *On geometric separability*. PhD thesis, Univ. Politecnica de Catalunya, 2002.

Constrained One-Sided Boundary Labeling*

Thomas Depian[1], Martin Nöllenburg[1], Soeren Terziadis[2], and Markus Wallinger[1]

1 **Algorithms and Complexity Group, TU Wien, Vienna, Austria**
{tdepian, noellenburg, mwallinger}@ac.tuwien.ac.at
2 **Algorithms cluster, TU Eindhoven, Eindhoven, The Netherlands**
s.d.terziadis@tue.nl

Abstract

We can label dense sets of feature points by placing the labels along a rectangular boundary around the illustration and using non-crossing leader lines to connect each label with its feature. Although boundary labeling is well-studied, semantic constraints on the labels have not been investigated much. We consider *grouping* and *ordering constraints* for one-sided boundary labeling. Grouping constraints enforce that a subset of the labels must occupy a contiguous region on the boundary, and ordering constraints define a partial order on the features. While we show that it is weakly NP-hard to find an admissible labeling for non-uniform labels that can slide along the boundary, we present polynomial-time algorithms for the case of fixed candidate label positions or uniform-height labels.

Related Version arXiv:2402.12245

1 Introduction

One common guideline when creating labeled illustrations in technical and medical domains is to "not obscure important details with labels" [8, p. 35]. Therefore, designers tend to place the labels outside the illustrations, creating an *external labeling* as in Figure 1a. Short non-crossing polyline *leaders* are used to connect labels with the feature points, here called *sites*, they describe. *Boundary labeling* is a restricted setting, where we only allow placing labels along a rectangular boundary $\mathcal{B}$ around the illustration [3]. Initial work placed the labels on one or two sides of the boundary, usually right and left, but extensions to more sides are also possible [14, 18]. Different leader styles have been considered [2, 3], but we focus on so-called *po-leaders* that consist of two segments: one is *p*arallel and the other *o*rthogonal to the side of the boundary on which the label is placed [3]. See Figure 1b for an example.

Although extensions of boundary labeling have been considered [1, 11, 15, 16], little work has been performed to respect semantic constraints, such as those from Figure 1a. Here, the layers are labeled from inside-out and some labels are grouped together and thus placed next to each other. The survey of Bekos et al. [4] reports a handful of papers that group or cluster labels. Many rely on heuristics [17] or group (spatially close) sites together to label them with a single label [10, 21]. To the best of our knowledge, Niedermann et al. [22] are the first that support the grouping of labels while ensuring non-crossing leaders. They investigated contour labeling, a generalization of boundary labeling, and considered the grouping of labels as a possible extension without analyzing it further. Although grouping labels sees a growing interest [14, 20], recent results still heavily restrict the possible position of the labels and only support a limited set of grouping constraints.

* This research has been funded by the Vienna Science and Technology Fund (WWTF) [10.47379/ICT19035] and [10.47379/ICT22029], and received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement Grant Agreement No 101034253.

40th European Workshop on Computational Geometry, Ioannina, Greece, March 13–15, 2024.
This is an extended abstract of a presentation given at EuroCG'24. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

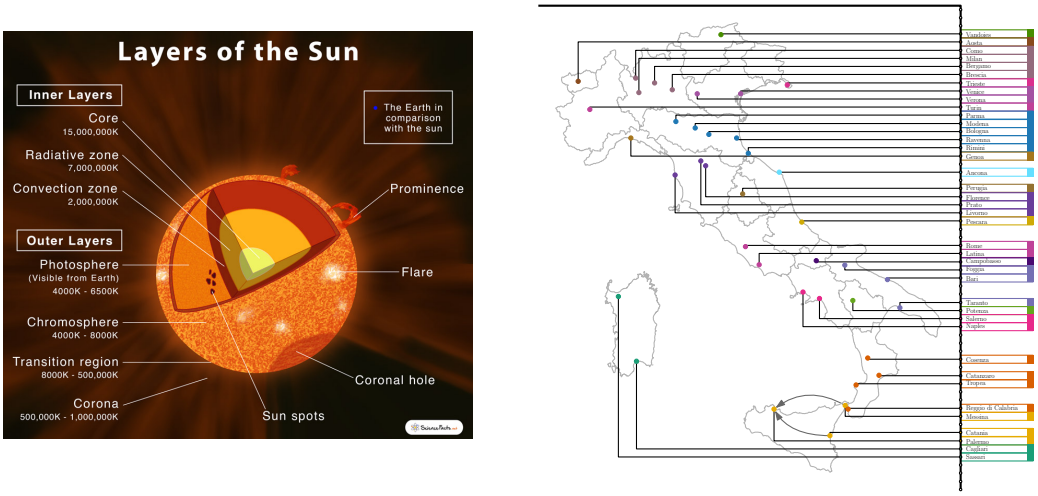


Figure 1 Labelings that adhere to semantic constraints. **(a)** © ScienceFacts.net [6]; **(b)** created by our algorithm from Section 2, colors indicate grouping and arrows ordering constraints.

Problem Description. Let $\mathcal{S}$ be a set of n sites in $\mathbb{R}^2$ in general position, i.e., no two sites share the same x- or y-coordinate, and enclosed in a bounding box $\mathcal{B}$. For each site $s_i \in \mathcal{S}$, we have an open rectangle ℓ_i of height $h(\ell_i)$ and some width, which we call the *label* of the site. The rectangles describe the bounding box of the (textual) labels, which are usually a single line of text in a fixed font size. Hence, we often restrict ourselves to uniform-height labels, but neglect their width. The *po*-leader $\lambda_i = (s_i, p_i)$ connects s_i with the *port* p_i of ℓ_i, which is the place where λ_i touches ℓ_i. We define the port for each label ℓ to be at half its height. In a one-sided boundary labeling $\mathcal{L}$ we place for each site $s \in \mathcal{S}$ a label ℓ such that p is on, w.l.o.g., the right side of $\mathcal{B}$ and connect s with ℓ using λ. If we are given a set of candidates for the ports $\mathcal{P}$, we say that we have *fixed ports*, otherwise *sliding ports*. Let Λ be the set of all possible leaders. A labeling is called *planar* if no two labels overlap and there is no leader-leader or -site crossing. We can access the x- and y-coordinate of a site or port with $x(\cdot)$ and $y(\cdot)$. Furthermore, we are given a set of constraints $\mathcal{C} = (\mathcal{G}, \preccurlyeq)$, consisting of a family of grouping constraints $\mathcal{G}$ and a partial order $\preccurlyeq$ on the sites. A *grouping* constraint $\emptyset \neq \mathcal{G} \subseteq \mathcal{S}$ enforces that the labels for the sites in $\mathcal{G}$ appear consecutively on the boundary, as in Figure 2a, i.e., it is not required that the labels are directly next to each other, as

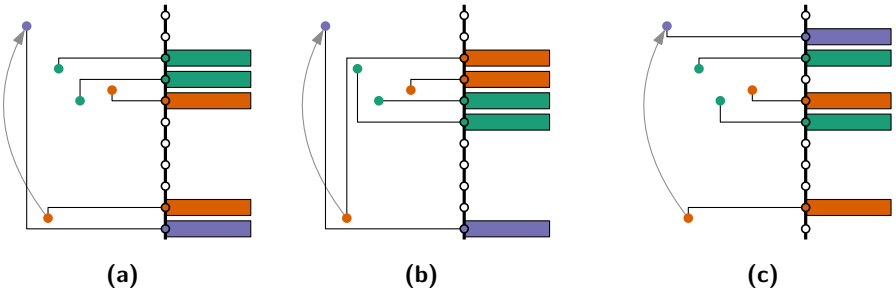


Figure 2 **(a)** Length- and **(b)** bend-minimal admissible labelings. **(c)** Planar but non-admissible length-minimal labeling. Colors indicate grouping and arrows ordering constraints.

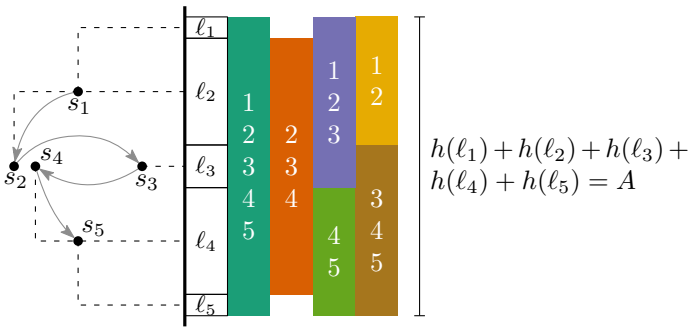


Figure 3 Creating an obstacle using grouping constraints as indicated by colored bands. An alternative way of forcing the same structure with only ordering constraints is shown with the arrows.

in Figure 2b. An *ordering* constraint $s_i \preccurlyeq s_j$ enforces that we have for the ports p_i and p_j $y(p_i) \geq y(p_j)$, i.e., the label for s_j must not appear above the label for s_i. We assume the existence of reflexive and transitive constraints in $\preccurlyeq$ and denote with r the number of constraints in the transitive reduction of $(\mathcal{S}, \preccurlyeq)$.

We say that a labeling *respects* the grouping/ordering constraints if all the grouping/ordering constraints are satisfied. Furthermore, we call the grouping/ordering constraints *consistent* if there exists a (not necessarily planar) labeling that respects them. A labeling is *admissible* if it is planar and respects the constraints. Furthermore, if an admissible labeling exists, we aim for one that optimizes a quality criterion expressed by a function $f\colon \Lambda \to \mathbb{R}_0^+$. In this paper, the optimization function f measures the length of a leader or expresses whether it has a bend or not. Figure 2 highlights the differences and shows in (c) that an optimal admissible labeling might be, w.r.t. f, worse than its planar (but non-admissible) counterpart.

In an instance $\mathcal{I}$ of the Constrained One-Sided Boundary Labeling problem[1] (1-CBL in short), we want to find an admissible one-sided *po*-labeling $\mathcal{L}^*$ for $\mathcal{I}$ (possibly on a set of m ports $\mathcal{P}$) that minimizes $\sum_{\lambda \in \mathcal{L}^*} f(\lambda)$ or report that no admissible labeling exists.

Computational Complexity of 1-CBL. Fink and Suri reduced the Partition problem to the problem of finding a planar labeling with non-uniform height labels and sliding ports in the presence of a single obstacle [11]. We can create with our constraints an obstacle on the boundary that serves the same purpose. Let $(\mathcal{A} = \{a_1, \dots a_N\}, w\colon \mathcal{A} \to \mathbb{N})$ be an instance of Partition with $\sum_{a \in \mathcal{A}} w(a) = 2A$ for some $A \in \mathbb{N}$[2] [13]. We create for each element a_i a site s_i whose corresponding label has a height of $w(a_i)$, and place the sites on a horizontal line next to each other. To mimic the obstacle, we place five sites in the configuration from Figure 3. Note that there is no alternative order for the labels of these sites nor room to slide them around. Hence, these labels must be placed contiguously, i.e., without any free space, at a fixed position on the boundary, i.e., they form a *block*. These blocks can be used to create two A-high free windows on the boundary where we can place the labels for the sites representing the elements of $\mathcal{A}$ in, see Figure 4. Thus, we can form an equivalence between partitioning the elements of $\mathcal{A}$ into two sets and placing the labels for the sites in the upper or lower window on the boundary. Finally, Figure 3 shows that we can replace the grouping constraints with ordering constraints, and thus we can prove Theorem 1.1.

[1] As we can show for a reasonable extension to two-sided labeling that finding an admissible labeling is NP-hard even for unit-height labels and fixed ports [9], we consider only one-sided labelings.
[2] Otherwise, $(\mathcal{A}, w)$ would be a trivial negative instance.

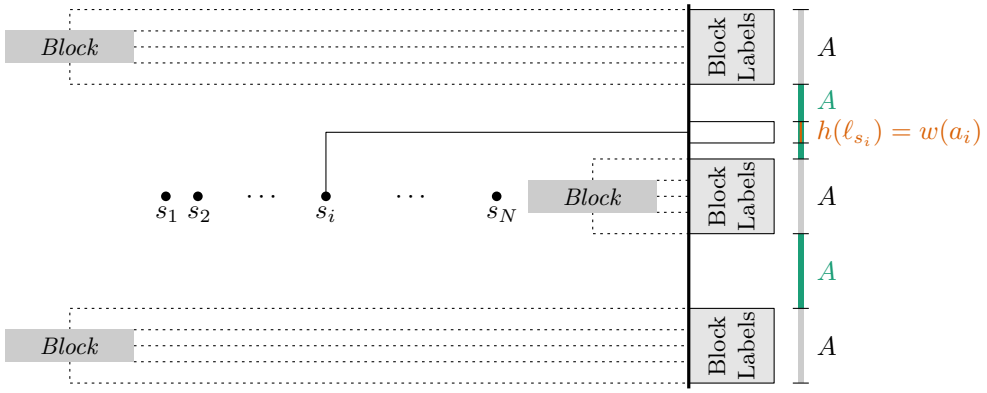


Figure 4 The placement of the sites in the reduction that shows weakly NP-hardness for 1-CBL.

▶ **Theorem 1.1.** *Deciding if an instance of* 1-CBL *has an admissible labeling is weakly NP-hard, even for a constant number of grouping or ordering constraints.*

Despite Theorem 1.1, we can solve 1-CBL in polynomial time if we use a pre-defined set of fixed ports (Section 2) or have uniform-height labels (Section 3).

Due to space constraints, all omitted details and proofs can be found in the full version [9].

2 Fixed Ports

We assume that we are given a set $\mathcal{P}$ of $m \geq n$ ports. Benkert et al. [5] observed that in a planar labeling $\mathcal{L}$, the leader λ_L connecting the leftmost site $s_L \in \mathcal{S}$ with some port p_L splits the instance $\mathcal{I}$ into two independent sub-instances, $\mathcal{I}_1$ and $\mathcal{I}_2$, excluding s_L and p_L. Therefore, we can describe a sub-instance I of $\mathcal{I}$ by two leaders (s_1, p_1) and (s_2, p_2) that bound the sub-instance from above and below, respectively. We denote the sub-instance as $I = (s_1, p_1, s_2, p_2)$ and refer with $\mathcal{S}(I)$ ($\mathcal{P}(I)$) to the sites (ports) in I, *excluding* those used in the definition of I, i.e., $\mathcal{S}(I) \coloneqq \{s \in \mathcal{S} \mid x(s_1) < x(s),\ x(s_2) < x(s),\ y(p_2) < y(s) < y(p_1)\}$ and $\mathcal{P}(I) \coloneqq \{p \in \mathcal{P} \mid y(p_2) < y(p) < y(p_1)\}$. Similarly, for a leader $\lambda = (s, p)$, we say that a site s' with $x(s) < x(s')$ is *above* λ if $y(s') > y(p)$ holds and *below* λ if $y(s') < y(p)$ holds. See also Figure 5 for a visualization of these definitions.

Two more observations about admissible labelings can be made: First, λ_L can never split sites $s, s' \in \mathcal{G}$ with $s_L \notin \mathcal{G}$. Second, λ_L never splits sites $s, s' \in \mathcal{S}$ with s above λ_L and s' below λ_L, for which we have $s' \preccurlyeq s_L$, $s' \preccurlyeq s$, or $s_L \preccurlyeq s$. Now, we could immediately define a dynamic programming (DP) algorithm that evaluates the induced sub-instances for each leader that adheres to these observations. However, we would then check every constraint in each sub-instance and not make use of implicit constraints given by, for example, overlapping groups. The following data structure makes these implicit constraints explicit.

PQ-A-Graphs. Every labeling $\mathcal{L}$ induces a permutation π of the sites by reading the labels from top to bottom. Let $k = |\mathcal{G}|$ and assume $k > 0$. Let $M(\mathcal{S}, \mathcal{G})$ be a $n \times k$ binary matrix with $m_{i,j} = 1$ iff $s_i \in \mathcal{G}_j$ for $\mathcal{G}_j \in \mathcal{G}$. We call $M(\mathcal{S}, \mathcal{G})$ the *sites vs. groups* matrix, and observe that $\mathcal{L}$ satisfies the constraint $\mathcal{G}_j$ iff the ones in the column j of $M(\mathcal{S}, \mathcal{G})$ are consecutive after we order the rows of $M(\mathcal{S}, \mathcal{G})$ according to π. If this holds for all columns of $M(\mathcal{S}, \mathcal{G})$, then the matrix has the so-called *consecutive ones property* (*C1P*) [12].

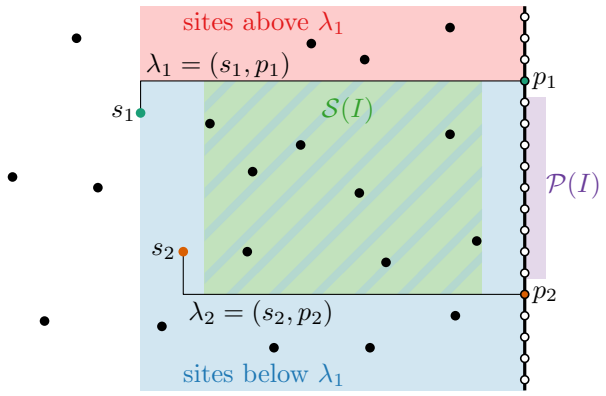


Figure 5 A sub-instance $I = (s_1, p_1, s_2, p_2)$ of our DP-algorithm and the used notation.

▶ **Lemma 2.1.** *$\mathcal{G}$ are consistent for $\mathcal{S}$ iff $M(\mathcal{S}, \mathcal{G})$ has the C1P.*

Booth and Lueker [7] propose an algorithm to check whether a binary matrix has the C1P. They use a PQ-Tree to keep track of the allowed row permutations. A PQ-Tree τ, for a given set $\mathcal{A}$ of elements, is a rooted tree with one leaf for each element of $\mathcal{A}$ and two different types of internal nodes t: P-nodes, that allow to freely permute the children of t, and Q-nodes, where the children of t can only be inversed [7]. Lemma 2.1 tells us that each family of consistent grouping constraints can be represented by a PQ-Tree. Note that we can interpret each subtree of the PQ-Tree as a grouping constraint and call them the *canonical groups*. However, not every grouping constraint results in a canonical group.

▶ **Definition 2.2** (PQ-A-Graph)**.** Let $\mathcal{S}$ be a set of sites, $\mathcal{G}$ be a family of consistent grouping constraints, and $\preccurlyeq$ be a partial order on $\mathcal{S}$. The *PQ-A-Graph* $\mathcal{T} = (\tau, A)$ consists of the PQ-Tree τ for $\mathcal{G}$, on whose leaves we embed the arcs A of a directed graph representing $\preccurlyeq$.

We denote with T_i the *subtree* in the underlying PQ-Tree τ rooted at the node t_i and with leaves(T_i) the leaf set. Figure 6 visualizes a PQ-A-Graph and the introduced terminology. Furthermore, observe that checking on the consistency of $\mathcal{C} = (\mathcal{G}, \preccurlyeq)$ is equivalent to solving the Reorder problem on τ and $\preccurlyeq$, i.e., asking whether we can re-order leaves(τ) such that the order induced by reading them from left to right extends the partial order $\preccurlyeq$ [19].

▶ **Lemma 2.3.** *We can check whether the constraints $\mathcal{C} = (\mathcal{G}, \preccurlyeq)$ are consistent for $\mathcal{S}$ and, if so, create the PQ-A-Graph $\mathcal{T}$ in $\mathcal{O}(n + |\mathcal{G}| + |\preccurlyeq| + \sum_{\mathcal{G} \in \mathcal{G}} |\mathcal{G}|)$ time. $\mathcal{T}$ uses $\mathcal{O}(n + |\preccurlyeq|)$ space.*

The DP-Algorithm. Let $I = (s_1, p_1, s_2, p_2)$ be a sub-instance and s_L the leftmost site in $\mathcal{S}$(I). Let $\mathcal{T}(s_1, s_2)$ denote the sub-graph of the PQ-A-Graph $\mathcal{T}$ rooted at the lowest common ancestor of s_1 and s_2 (in $\mathcal{T}$). Note that $\mathcal{T}(s_1, s_2)$ contains all the sites in $\mathcal{S}(I)$, together with s_1 and s_2, and hence represents all constraints relevant for the sub-instance I. Other constraints either do not affect sites in I or are trivially satisfied. Imagine we want to place the label ℓ_L for s_L at the port $p_L \in \mathcal{P}(I)$. We have to ensure that $\lambda_L = (s_L, p_L)$ does not violate planarity w.r.t. the already fixed labeling and that in the resulting sub-instances there are enough ports for the sites. Let Admissible$(I, \mathcal{T}, p_L)$ be a procedure that checks this and, in addition, verifies that p_L respects the constraints expressed by $\mathcal{T}(s_1, s_2)$.

To do the latter, we make use of the procedure RespectsConstraints$(I, \mathcal{T}, \lambda_L)$, which is defined as follows. Let t_L be the leaf for s_L in $\mathcal{T}(s_1, s_2)$. There is a unique path from t_L to the root of $\mathcal{T}(s_1, s_2)$, which we traverse bottom up and consider each internal node t on it.

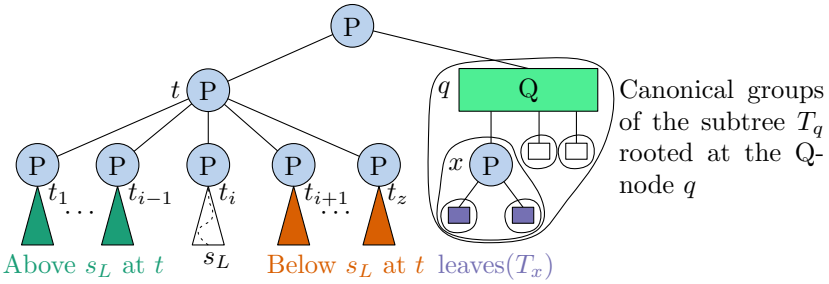


Figure 6 A sample PQ-A-Graph together with the used terminology.

Assume that t has the children $t_1, \ldots, t_z$ in this order from left to right. Let T_i, $1 \leq i \leq z$, be the subtree that contains the site s_L, rooted at t_i. The labels for all sites represented by leaves$(T_1), \ldots,$ leaves(T_{i-1}) will be placed above ℓ_L in any labeling $\mathcal{L}$ of $\mathcal{S}$ in which the children of t are ordered as stated. Therefore, we call these sites *above* s_L (at t). Analogously, the sites represented by leaves$(T_{i+1}), \ldots,$ leaves(T_z) are *below* s_L (at t). Figure 6 visualizes this. Note that the sites represented by leaves(T_i) are neither above nor below s_L at t.

If t is a P-node, there must exist at least one permutation π of the children of t in which *all* the sites in $\mathcal{S}$(I) above s_L at t (in π) are above λ_L (recall Figure 5), and all the sites in $\mathcal{S}$(I) below s_L at t (in the permutation π) are below λ_L, i.e., the sites are on the correct side of λ_L w.r.t. π. To not iterate through all possible permutations, we distribute the children of t, except t_i, into two sets, t_{above} and t_{below}, depending on whether they contain only leaves for sites that should be above or below s_L at t. If neither applies, we return with failure. We also have to ensure that this complies with the definition of I and the ordering constraints.

If t is a Q-node, we do the same, however, we only have to check which of the two orderings allowed by the Q-node complies with the position of the leader and whether it adheres to the definition of I, i.e., labels s_1 above s_2. RespectsConstraints$(I, \mathcal{T}, \lambda_L)$ performs the above checks for each of the $\mathcal{O}(n)$ nodes on the path from s_L to t_r in $\mathcal{O}(n\,(n + |\preccurlyeq|))$ time.

For a sub-instance $I = (s_1, p_1, s_2, p_2)$, we store in a table D the value $f(\mathcal{L}^*)$ of an optimal admissible labeling $\mathcal{L}^*$ on I or ∞ if none exists. If I does not contain a site we set $D[I] = 0$. Otherwise, we use the following relation, where the minimum of the empty set is ∞.

$$D[I] = \min_{\substack{p_L \in \mathcal{P}(I) \text{ where} \\ \textsc{Admissible}(I, \mathcal{T}, p_L) \text{ is true}}} (D[(s_1, p_1, s_L, p_L)] + D[(s_L, p_L, s_2, p_2)]) + f((s_L, p_L))$$

Correctness follows from the correctness of the approach from [5] combined with the fact that we consider only those ports that are admissible for s_L. By adding two auxiliary sites s_0, s_{n+1} and ports p_0, p_{m+1} above and below the sites and ports from $\mathcal{I}$, we can describe $\mathcal{I}$ by the sub-instance $I_0 = (s_0, p_0, s_{n+1}, p_{m+1})$. Hence, $D[I_0]$ will store in the end $f(\mathcal{L}^*)$, or ∞, if $\mathcal{I}$ does not possess an admissible labeling. We fill the $\mathcal{O}(n^2m^2)$ entries of D top-down using memoization. The time to evaluate an entry is dominated by the admissibility checks.

▶ **Theorem 2.4.** 1-CBL*, with fixed ports, can be solved in* $\mathcal{O}(n^5m^3 \log m + |\mathcal{G}| + \sum_{\mathcal{G} \in \mathcal{G}} |\mathcal{G}|)$ *time and* $\mathcal{O}(n^2m^2)$ *space.*

In the full version [9], we discuss an implementation of the algorithm for uniform-height labels. See Figure 1b for an example computed by this implementation.

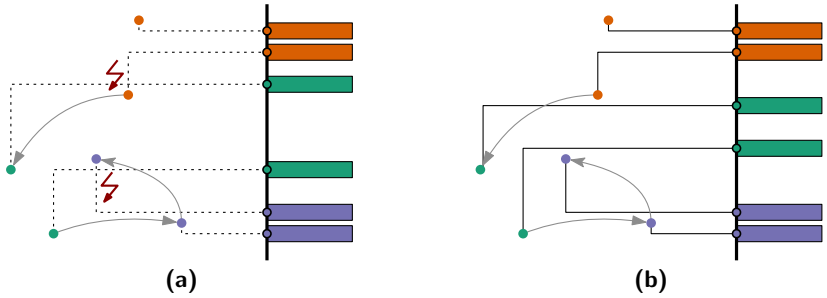


Figure 7 An instance whose admissibility depends on the position of the ports.

3 Sliding Ports with Uniform-Height Labels

Fixed ports have the limitation that the admissibility of an instance depends on the choice and position of the ports, as Figure 7 shows. By allowing the labels to slide along a vertical boundary line, we remove this limitation but require that all labels now have uniform height h. We would like to use the idea of Fink and Suri [11] and let each site s *induce* $\mathcal{O}(n)$ ports placed at multiples of h away from s, as in Figure 8a. Then, assuming that there exists an admissible labeling $\mathcal{L}$, we want to obtain a new one $\mathcal{L}'$ by sliding the labels along the boundary until each port in $\mathcal{L}'$ is induced by a site. To avoid the need for re-routing leaders in case of leader-site crossings, as this could violate constraints, we introduce $\mathcal{O}(n^2)$ additional ports placed sufficiently close to the induced ones. They guarantee us that there is a port (vertically) between any two sites, as Figure 8b shows. Hence, while sliding labels, we can reach a port before hitting a site with a leader, i.e., we never need to re-route leaders. As we defined $\mathcal{O}(n^2)$ *canonical ports*, for which we show in the full version [9] that they are sufficient for an admissible labeling, we can use our DP-Algorithm to obtain Theorem 3.1. In the full version [9], we further show that such canonical ports also exist for length- and bend-minimal labelings.

▶ **Theorem 3.1.** 1-CBL, *with uniform-height labels, can be solved in* $\mathcal{O}(n^{11} \log n + |\mathcal{G}| + \sum_{\mathcal{G} \in \mathcal{G}} |\mathcal{G}|)$ *time and* $\mathcal{O}(n^6)$ *space.*

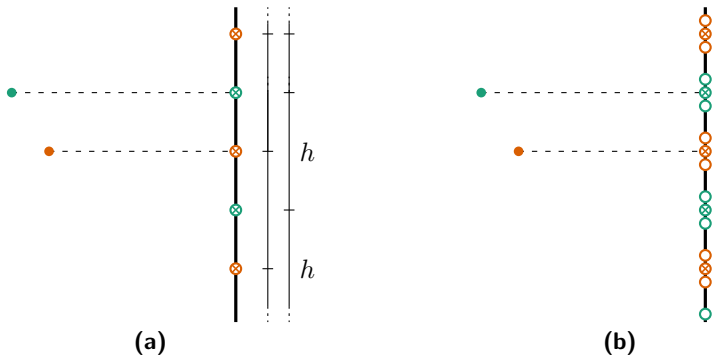


Figure 8 Set of ports **(a)** induced by the sites as in [11] and **(b)** extended to our canonical ports.

References

1 Michael A. Bekos, Sabine Cornelsen, Martin Fink, Seok-Hee Hong, Michael Kaufmann, Martin Nöllenburg, Ignaz Rutter, and Antonios Symvonis. Many-to-One Boundary Labeling with Backbones. *Journal of Graph Algorithms and Applications (JGAA)*, 19(3):779–816, 2015. doi:10.7155/jgaa.00379.

2 Michael A. Bekos, Michael Kaufmann, Martin Nöllenburg, and Antonios Symvonis. Boundary Labeling with Octilinear Leaders. *Algorithmica*, 57(3):436–461, 2010. doi:10.1007/s00453-009-9283-6.

3 Michael A. Bekos, Michael Kaufmann, Antonios Symvonis, and Alexander Wolff. Boundary labeling: Models and efficient algorithms for rectangular maps. *Computational Geometry*, 36(3):215–236, 2007. doi:10.1016/j.comgeo.2006.05.003.

4 Michael A. Bekos, Benjamin Niedermann, and Martin Nöllenburg. *External Labeling: Fundamental Concepts and Algorithmic Techniques*. Synthesis Lectures on Visualization. Springer, 2021. doi:10.1007/978-3-031-02609-6.

5 Marc Benkert, Herman J. Haverkort, Moritz Kroll, and Martin Nöllenburg. Algorithms for Multi-Criteria Boundary Labeling. *Journal of Graph Algorithms and Applications (JGAA)*, 13(3):289–317, 2009. doi:10.7155/jgaa.00189.

6 Satyam Bhuyan and Santanu Mukherjee (sciencefacts.net). Layers of the Sun, 2023. Accessed on 2023-09-07. URL: https://www.sciencefacts.net/layers-of-the-sun.html.

7 Kellogg S. Booth and George S. Lueker. Testing for the Consecutive Ones Property, Interval Graphs, and Graph Planarity Using PQ-Tree Algorithms. *Journal of Computer and System Sciences (JCSS)*, 13(3):335–379, 1976. doi:10.1016/S0022-0000(76)80045-1.

8 Mary Helen Briscoe. *A Researcher's Guide to Scientific and Medical Illustrations*. Springer Science & Business Media, 1990. doi:10.1007/978-1-4684-0355-8.

9 Thomas Depian, Martin Nöllenburg, Soeren Terziadis, and Markus Wallinger. Constrained Boundary Labeling, 2024. doi:10.48550/arXiv.2402.12245.

10 Martin Fink, Jan-Henrik Haunert, André Schulz, Joachim Spoerhase, and Alexander Wolff. Algorithms for Labeling Focus Regions. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2583–2592, 2012. doi:10.1109/TVCG.2012.193.

11 Martin Fink and Subhash Suri. Boundary Labeling with Obstacles. In *Proc. 28th Canadian Conference on Computational Geometry (CCCG)*, pages 86–92. Simon Fraser University, 2016.

12 Delbert Fulkerson and Oliver Gross. Incidence matrices and interval graphs. *Pacific Journal of Mathematics*, 15(3):835–855, 1965.

13 Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

14 Sven Gedicke, Lukas Arzoumanidis, and Jan-Henrik Haunert. Automating the external placement of symbols for point features in situation maps for emergency response. *Cartography and Geographic Information Science (CaGIS)*, 50(4):385–402, 2023. doi:10.1080/15230406.2023.2213446.

15 Sven Gedicke, Annika Bonerath, Benjamin Niedermann, and Jan-Henrik Haunert. Zoomless Maps: External Labeling Methods for the Interactive Exploration of Dense Point Sets at a Fixed Map Scale. *IEEE Transactions on Visualization and Computer Graphics*, 27(2):1247–1256, 2021. doi:10.1109/TVCG.2020.3030399.

16 Andreas Gemsa, Jan-Henrik Haunert, and Martin Nöllenburg. Multirow Boundary-Labeling Algorithms for Panorama Images. *ACM Transactions on Spatial Algorithms and Systems (TSAS)*, 1(1):1:1–1:30, 2015. doi:10.1145/2794299.

17 Timo Götzelmann, Knut Hartmann, and Thomas Strothotte. Contextual Grouping of Labels. In *Proc. 17th Simulation und Visualisierung (SimVis)*, pages 245–258. SCS Publishing House e.V., 2006.

18 Philipp Kindermann, Benjamin Niedermann, Ignaz Rutter, Marcus Schaefer, André Schulz, and Alexander Wolff. Multi-sided Boundary Labeling. *Algorithmica*, 76(1):225–258, 2016. doi:10.1007/s00453-015-0028-4.

19 Pavel Klavík, Jan Kratochvíl, Yota Otachi, Toshiki Saitoh, and Tomás Vyskocil. Extending Partial Representations of Interval Graphs. *Algorithmica*, 78(3):945–967, 2017. doi:10.1007/s00453-016-0186-z.

20 Jonathan Klawitter, Felix Klesen, Joris Y. Scholl, Thomas C. van Dijk, and Alexander Zaft. Visualizing Geophylogenies - Internal and External Labeling with Phylogenetic Tree Constraints. In *Proc. 12th International Conference Geographic Information Science (GIScience)*, volume 277 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 5:1–5:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPIcs.GIScience.2023.5.

21 Konrad Mühler and Bernhard Preim. Automatic Textual Annotation for Surgical Planning. In *Proc. 14th International Symposium on Vision, Modeling, and Visualization (VMV)*, pages 277–284. DNB, 2009.

22 Benjamin Niedermann, Martin Nöllenburg, and Ignaz Rutter. Radial Contour Labeling with Straight Leaders. In *Proc. 10th IEEE Pacific Visualization Symposium (PacificVis)*, Lecture Notes in Computer Science (LNCS), pages 295–304. IEEE Computer Society, 2017. doi:10.1109/PACIFICVIS.2017.8031608.

Exact solutions to the Weighted Region Problem*

Sarita de Berg[1], Guillermo Esteban[2,3], Rodrigo I. Silveira[4], and Frank Staals[1]

1 **Department of Information and Computing Sciences, Utrecht University, The Netherlands**
{s.deberg, f.staals}@uu.nl
2 **Departamento de Física y Matemáticas, Universidad de Alcalá, Spain**
g.esteban@uah.es
3 **School of Computer Science, Carleton University, Canada**
4 **Departament de Matemàtiques, Universitat Politècnica de Catalunya, Spain**
rodrigo.silveira@upc.edu

Abstract

In this paper, we consider the Weighted Region Problem. In the Weighted Region Problem, the length of a path is defined as the sum of the weights of the subpaths within each region, where the weight of a subpath is its Euclidean length multiplied by a weight $\alpha \geq 0$ depending on the region. We study a restricted version of the problem of determining shortest paths through a single weighted rectangular region. We prove that even this very restricted version of the problem is unsolvable within the Algebraic Computation Model over the Rational Numbers (ACM$\mathbb{Q}$). On the positive side, we provide the equations for the shortest paths that are computable within the ACM$\mathbb{Q}$. Additionally, we provide equations for the bisectors between regions of the Shortest Path Map for a source point on the boundary of (or inside) the rectangular region.

Related Version A full version of this paper is available at arXiv:2402.12028

1 Introduction

The Weighted Region Problem (WRP) [10] is a well-known geometric problem that, despite having been studied extensively, is still far from being well understood. Consider a subdivision of the plane into (usually polygonal) regions. Each region R_i has a weight $\alpha_i \geq 0$, representing the cost per unit distance of traveling in that region. Thus, a segment σ, of length $|\sigma|$, between two points in the same region has weighted length $\alpha_i|\sigma|$ when traversing the interior of R_i, or $\min\{\alpha_i, \alpha_j\}|\sigma|$ if it goes along the edge between R_i and R_j. Then, the weighted length of a path $\pi(s,t)$ through a subdivision is the sum of the weighted lengths of its subpaths through each face or edge. The resulting metric is called the *Weighted Region Metric*. The WRP entails computing a shortest path between two given points s and t under this metric. We denote the weighted length of $\pi(s,t)$ by $d(s,t)$. Figure 1 shows how the shape of a shortest path changes as the weight of one region varies.

Existing algorithms for the WRP—in its general formulation—are approximate. Since the seminal work by Mitchell and Papadimitriou [10], with the first $(1+\varepsilon)$-approximation, several algorithms have been proposed, with improvements on running times, but always keeping some dependency on the vertex coordinates sizes and weight ranges. These methods are usually based on the continuous Dijkstra's algorithm (e.g., [10]), or on adding Steiner points (e.g., see [1, 2, 3, 4, 13]). However, rather recently it has been proved that computing an exact shortest path between two points using the weighted region metric, even if there are

*Work by G. E. and R. I. S. has been supported by project PID2019-104129GB-I00 funded by MCIN/AEI/10.13039/501100011033. G. E. is also funded by an FPU of the Universidad de Alcalá.

40th European Workshop on Computational Geometry, Ioannina, Greece, March 13–15, 2024.
This is an extended abstract of a presentation given at EuroCG'24. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

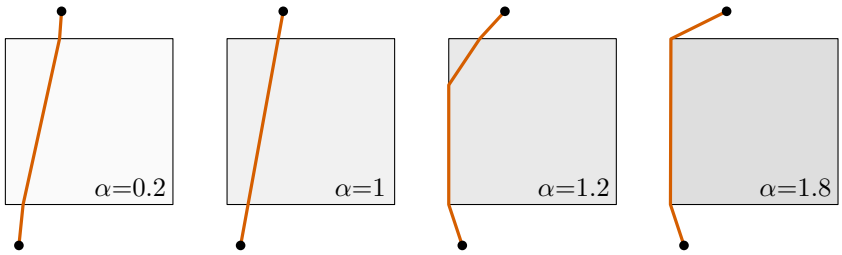


Figure 1 Examples of shortest paths between two points—shown in orange—for two weighted regions. The unbounded region has weight 1, the squares have varying weight α.

only three different weights, is an unsolvable problem in the Algebraic Computation Model over the Rational Numbers (ACM$\mathbb{Q}$) [5]. In the ACM$\mathbb{Q}$ one can compute exactly any number that can be obtained from rational numbers by applying a finite number of operations from $+, -, \times, \div$, and $\sqrt[k]{}$, for any integer $k \geq 2$. This provides a theoretical explanation for the lack of exact algorithms for the WRP, and justifies the study of approximation methods.

This also raises the question of which are the special cases for which the WRP can be solved exactly. Two natural ways to restrict the problem are by limiting the possible weights, and by restricting the shape of the regions. For example, computing a shortest path among polygonal obstacles can be seen as a variant of the WRP with weights in $\{1, \infty\}$. The case for weights in $\{0, 1, \infty\}$ can be solved in $O(n^2)$ time [6]. Other variants that can be solved exactly correspond to regions shaped as regular k-gons with weight ≥ 2 (since they can be considered as obstacles), or regions with two weights $\{1, \alpha\}$ consisting of parallel strips [11].

Our results. In light of the fact that the WRP is unsolvable in the ACM$\mathbb{Q}$ for three different weights, in this work we study the case of two arbitrary weights, that is, weights in $\{1, \alpha\}$. This case is particularly interesting, since an algorithm for weights $\{1, \alpha\}$ can be transformed into one for weights in $\{0, 1, \alpha, \infty\}$ [8]. However, the variant with weights $\{1, \alpha\}$ was conjectured to be as hard as the general WRP problem, see the first open problem in [6, Section 7]. (The results in [5] do not directly apply to weights $\{0, 1, \alpha, \infty\}$.)

This paper is organized as follows. First we present some preliminaries in Section 2. In Section 3 we consider two weights and one rectangular region R, with the source point s on its boundary or inside. For this setting, we figure out all types of possible optimal paths and give exact formulas to compute their lengths. In Section 3.3 we focus on the case where s is outside of R, and prove that in this case the WRP with weights $\{1, \alpha\}$ is already unsolvable in the ACM$\mathbb{Q}$. In Section 4 we explore the computation of the shortest path map for s. We finish with some conclusions in Section 5. Omitted proofs are in the full version of the paper.

2 Shortest paths and their properties

In this section we briefly review some key properties of shortest paths in weighted regions.

First, shortest paths in the weighted region problem will always be piecewise linear, see [10, Lemma 3.1]. Second, it is known that shortest paths must obey Snell's law of refraction. So we can think of a shortest path as a ray of light. We define the *angle of incidence* θ as the minimum angle between the incoming ray and the vector perpendicular to the region boundary. The *angle of refraction* θ' is defined as the minimum angle between the outgoing ray and the normal (see Figure 3). Snell's law states that whenever the ray goes from one region R_i to another region R_j, then $\alpha_i \sin\theta = \alpha_j \sin\theta'$. In addition, whenever

$\alpha_i > \alpha_j$, the angle θ_c at which $\frac{\alpha_i}{\alpha_j} \sin\theta_c = 1$ is called the *critical angle*. A ray that hits an edge at an angle of incidence greater than θ_c, will be totally reflected from the point at which it hits the boundary. In our problem, a shortest path will never be incident to an edge at an angle greater than θ_c. Finally, if the space only contains orthoconvex regions[1] with weight at least $\sqrt{2}$, they can be simply considered obstacles [11]. Thus, since we focus on a rectangular region R, we assume that its weight is $0 < \alpha < \sqrt{2}$. However, first we provide some general properties of shortest paths for arbitrary weighted regions that are interesting on their own.

▶ **Lemma 2.1.** *Let $\mathcal{S}$ be a polygonal subdivision for which each region has a weight in the set $\{1, \alpha\}$, with $\alpha > 0$. A shortest path $\pi(s,t)$ visits any edge of the subdivision at most once.*

Proof. Assume, for the sake of contradiction, that $\pi(s,t)$ intersects an edge e in at least two disjoint intervals I_1 and I_3. Moreover, let $p_1 \in I_1$ and $p_3 \in I_3$ be points for which the subpath $\pi(p_1,p_3) \subseteq \pi(s,t)$ does not intersect e in any points other than p_1 and p_3. Let p_2 be a point on $\pi(p_1,p_3)$ between p_1 and p_3, which thus does not lie on e. Now observe that there exists a path $\overline{p_1p_3}$ from p_1 to p_3 of length $\min\{1,\alpha\}|\overline{p_1p_3}|$. Since p_2 does not lie on $\overline{p_1p_3}$, it follows by the triangle inequality that the length of $\pi(p_1,p_3)$ is strictly larger than $\min\{1,\alpha\}|\overline{p_1p_3}|$. Hence, $\pi(s,t)$ is not a shortest path, and we obtain a contradiction. ◀

▶ **Corollary 2.2.** *Let $\mathcal{S}$ be a polygonal subdivision with n vertices, such that the weight of each region is within the set $\{1, \alpha\}$, with $\alpha > 0$. Any shortest path $\pi(s,t)$ is a polygonal chain with at most $O(n)$ vertices.*

Proof. Any shortest path is a polygonal chain whose interior vertices all lie on edges of $\mathcal{S}$, see [10, Proposition 3.8]. By Lemma 2.1, each edge contributes with at most two vertices. ◀

We observe that if the regions use only one of two weights $\{1, \alpha\}$, Corollary 2.2 implies that the time complexity of the algorithm proposed by Mitchell and Papadimitriou [10] can be improved by a quartic factor to $O(n^4 L)$, where L is the precision of the instance.

3 Computing a shortest path

We now consider the problem of computing a shortest path $\pi(s,t)$ from s to t when the region R is an axis-aligned rectangle of weight α. The exact shape of $\pi(s,t)$ depends on the position of s and t with respect to R, and on the value of α.

In Sections 3.1 and 3.2 we consider the case that s lies on the boundary or inside of R, respectively. We categorize the various types of shortest paths, and show that we can compute the shortest path of each type, and thus we can compute $\pi(s,t)$. In Section 3.3, we consider the case that s and t lie outside R. In this case $\pi(s,t)$ may have two vertices on the boundary of R. We show that the coordinates of these vertices cannot be computed exactly within the ACM$\mathbb{Q}$.

3.1 The source point s lies on the boundary of R

Throughout this section we consider the case where s is restricted to the boundary of R, a rectangle of unit height with top-left corner at $(0,0)$. Let $s = (s_x, 0)$, $s_x > 0$, be a point on the top side of R, see Figure 2. In addition, we assume that t is to the left of the line through s perpendicular to the top side of R. The other cases are symmetric.

[1] A region is orthoconvex if its intersection with every horizontal and vertical line is connected or empty [12].

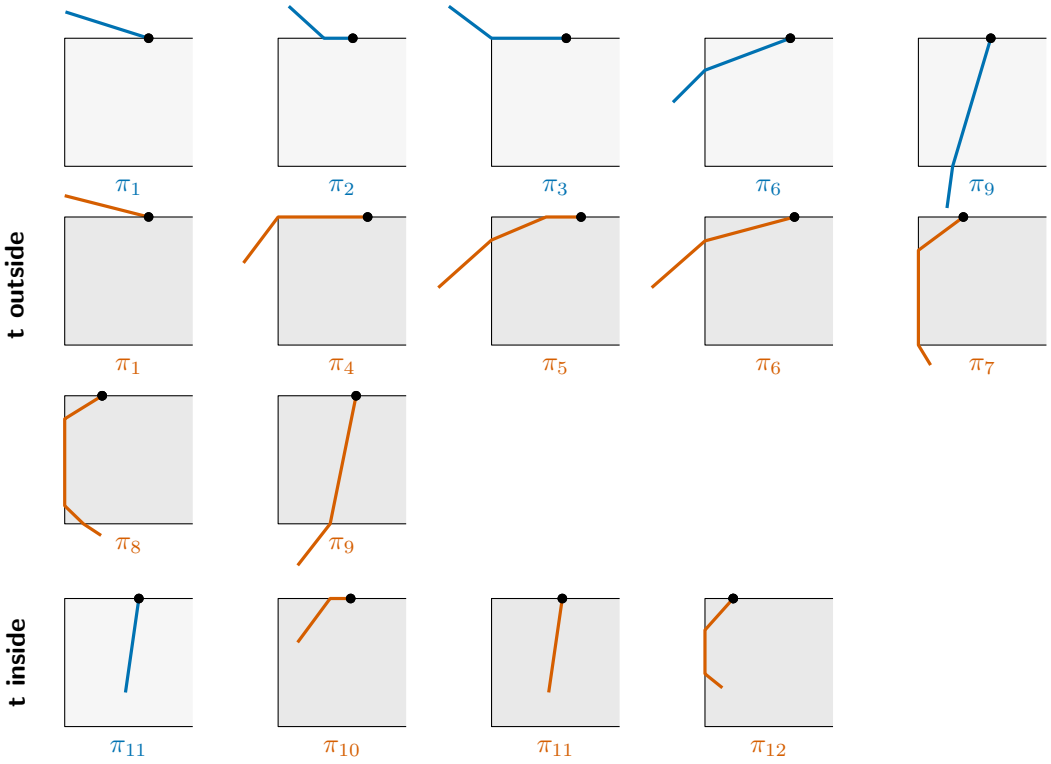


Figure 2 Path types for s on the boundary of R of weight $\alpha < 1$ (blue) and $1 < \alpha < \sqrt{2}$ (orange).

Shortest path types. Lemma 2.1 implies that in this setting, there are only $O(1)$ combinatorial types of paths that we have to consider. More precisely, we have that:

▶ **Observation 3.1.** *Let s be a point on the top boundary of a rectangle R with weight $0 < \alpha < \sqrt{2}$. There are 12 types of shortest paths $\pi_i(s,t)$, shown in Figure 2, up to symmetries.*

Length of $\pi_i(s,t)$. When s is on the boundary of R, there is at most one vertex of $\pi_i(s,t)$ without the critical angle property. This allows us to compute the exact coordinates of the vertices of $\pi_i(s,t)$. Theorem 3.2 gives the length $d_i(s,t)$ of the path $\pi_i(s,t)$. The proofs of the equations, which are based on Snell's law of refraction, are deferred to the full version.

▶ **Theorem 3.2.** *Let $s = (s_x, 0)$ be a point on the boundary of R with weight $0 < \alpha < \sqrt{2}$. A shortest path $\pi(s,t) = \pi_i(s,t)$ from s to a point $t = (t_x, t_y)$ and its length can be computed in*

$O(1)$ time in the ACM$\mathbb{Q}$. In particular, the length $d(s,t) = d_i(s,t)$ is given by

$$d_i(s,t) = \begin{cases} \sqrt{(s_x - t_x)^2 + t_y^2} & \text{if } i = 1 \\ \alpha(s_x - t_x) + \sqrt{1-\alpha^2}\,t_y & \text{if } i = 2 \\ \alpha s_x + \sqrt{t_x^2 + t_y^2} & \text{if } i = 3 \\ s_x + \sqrt{t_x^2 + t_y^2} & \text{if } i = 4 \\ s_x - \sqrt{2-\alpha^2}\,t_x - \sqrt{\alpha^2-1}\,t_y & \text{if } i = 5 \\ \alpha\sqrt{s_x^2 + y^2} + \sqrt{t_x^2 + (t_y - y)^2} & \text{if } i = 6 \\ \sqrt{\alpha^2-1}\,s_x + 1 + \sqrt{t_x^2 + (t_y+1)^2} & \text{if } i = 7 \\ \sqrt{\alpha^2-1}(s_x + t_x) - \sqrt{2-\alpha^2}(1+t_y) + 1 & \text{if } i = 8 \\ \alpha\sqrt{(s_x - x)^2 + 1} + \sqrt{(t_x - x)^2 + (t_y+1)^2} & \text{if } i = 9 \end{cases} \quad \left(\begin{array}{l}\text{and thus } t \text{ lies} \\ \text{outside } R\end{array}\right), \text{ and}$$

$$d_i(s,t) = \begin{cases} s_x - t_x - \sqrt{\alpha^2-1}\,t_y & \text{if } i = 10 \\ \alpha\sqrt{(s_x - t_x)^2 + t_y^2} & \text{if } i = 11 \\ \sqrt{\alpha^2-1}(s_x + t_x) - t_y & \text{if } i = 12 \end{cases} \quad \left(\begin{array}{l}\text{and thus } t \text{ lies} \\ \text{inside } R\end{array}\right),$$

in which x is the unique real solution in the interval (t_x, s_x) to the equation

$$\begin{aligned} &\beta x^4 - 2\beta(t_x + s_x)x^3 + [\beta(s_x^2 + t_x^2 + 4s_x t_x) + \alpha^2(1+t_y)^2 - 1]x^2 \\ &- 2[\beta(t_x s_x^2 + t_x^2 s_x) + \alpha^2(1+t_y)^2 s_x - t_x]x + \beta t_x^2 s_x^2 + \alpha^2(1+t_y)^2 s_x^2 - t_x^2 = 0, \end{aligned} \tag{1}$$

where $\beta = \alpha^2 - 1$, and y is the unique real solution in the interval $(t_y, 0)$ to the equation

$$\beta y^4 - 2t_y\beta y^3 + [\alpha^2 t_x^2 + \beta t_y^2 - s_x^2]y^2 + 2s_x^2 t_y y - s_x^2 t_y^2 = 0.$$

3.2 The source point s lies inside R

We now consider the case where s is restricted to the interior of the rectangle R.

▶ **Observation 3.3.** *Let s be a point in a rectangle R with weight $0 < \alpha < \sqrt{2}$. There are 6 types of shortest paths, up to symmetries, namely $\pi_i(s,t)$, for $i \in \{6,7,8,9,11,12\}$.*

The types of shortest paths are similar to the ones defined in Observation 3.1, see the paths in Figure 2 where the top side of R or the region above R is not intersected. As in Theorem 3.2, we can thus compute the (length of) a shortest path (of each type) exactly, albeit that the expressions for the length are dependent on the location of s in R. Theorem 3.2 gives exact lengths for all path types when R has height > 1 and s is at distance exactly 1 from the bottom boundary of R.

3.3 The source point s lies outside of R

When both the source and the target point are outside of R, the shortest path can again be of many different types. In particular, the types in Figure 2 can be generalized to this setting. There are two special cases where the shortest path bends *twice* at an angle that is not the critical angle: it can bend on two opposite sides of the rectangle, or on two incident sides. In the first case, the angles at both vertices of $\pi(s,t)$ are equal, and the shortest path can be computed exactly [11]. For the second case, we show that it is not possible to compute the coordinates of the vertices of $\pi(s,t)$ exactly in the ACM$\mathbb{Q}$. Hence, the WRP limited to two weights $\{1, \alpha\}$ is not solvable within the ACM$\mathbb{Q}$. Note that this path type can occur in an even simpler setting, where R is a single quadrant instead of a rectangle.

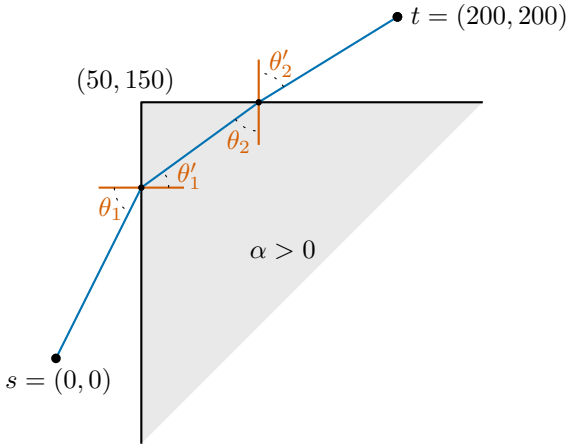


Figure 3 A shortest path from s to t that bends twice under different angles.

▶ **Theorem 3.4.** *The weighted region problem with weights in the set* $\{1, \alpha\}$*, with* $0 < \alpha < \sqrt{2}$*, cannot be solved exactly within the ACM*$\mathbb{Q}$*, even if* R *is a single quadrant.*

Proof sketch. We obtain this result by following the approach of De Carufel et al. [5] to show that the polynomial that represents a solution to the WRP in this situation is not solvable within the ACM$\mathbb{Q}$. We consider the situation in Figure 3. By applying Snell's law on both vertices of $\pi(s, t)$, and several trigonometric identities, we find the following expression for $u = \sin\theta_1$, where θ_1 is the angle at which the path leaves s, with respect to the x-axis:

$$\sqrt{\alpha^2 - u^2}\left(\frac{3}{u} - \frac{1}{\sqrt{1-u^2}} + \frac{1}{\sqrt{1-\alpha^2+u^2}}\right) = 3.$$

Note that a solution to this equation would give us the shortest path, as we can use Snell's law to find the other angles θ_1' and θ_2'. By squaring appropriately, this equation can be transformed into a degree 11 polynomial $p(u)$. Finally, we show that this polynomial is unsolvable within the ACM$\mathbb{Q}$ by applying a general lemma on the unsolvability of polynomials. ◀

4 Computing a Shortest Path Map

To find a shortest path from a source point s to *all* points at once, one can build a *Shortest Path Map* (*SPM*), see e.g., [7, 9, 10]. A *SPM* is a subdivision of the space for a given source s, where for each cell the paths $\pi(s, t)$, with t in the cell, have the same type. To compute the *SPM*, we consider computing the bisector $b_{i,j} = \{q \mid q \in \mathbb{R}^2 \wedge d_i(s, q) = d_j(s, q)\}$ for all relevant pairs of shortest path types π_i, π_j, i.e., pairs for which $b_{i,j}$ appears in the shortest path map. As before, we consider the setting where R is a rectangular region. In Section 4.1, we first consider the case when s lies on the boundary of R. In Section 4.2, we do the same for the case s lies inside R. The case that s lies outside R is not interesting, as we cannot even compute exactly a single shortest path in that case.

4.1 The source point s lies on the boundary of R

The *SPM* is given by the boundary of R and several bisector curves, expressed as points $(x, b_{i,j}(x))$. If $\alpha < 1$, these curves all lie outside R (the interior of R is a single region in the *SPM*). Bisectors involving $\pi_9(s, t)$ are of a much more complicated form, as might

be expected from the implicit representation used for $d_9(s,t)$ in Theorem 3.2. Therefore, Lemma 4.1 gives the bisector curves, excluding the ones related to $\pi_9(s,t)$. The proofs are deferred to the full version.

▶ **Lemma 4.1.** *The SPM for a point $s = (s_x, 0)$ on the boundary of R is defined by:*

$$b_{i,j}(x) = \begin{cases} \frac{\sqrt{1-\alpha^2}}{\alpha}(s_x - x) & \text{if } i = 1, j = 2 \\ -\frac{\sqrt{1-\alpha^2}}{\alpha}x & \text{if } i = 2, j = 3 \\ 0 & \text{if } i = 3, j = 6 \end{cases} \quad (\text{ when } \alpha < 1), \text{ and}$$

$$b_{i,j}(x) = \begin{cases} 0 & \text{if } i = 1, j = 4 \\ \frac{\sqrt{\alpha^2-1}}{\sqrt{2-\alpha^2}}x & \text{if } i = 4, j = 5 \\ \frac{\sqrt{\alpha^2-1}}{\sqrt{2-\alpha^2}}x - \sqrt{\alpha^2-1}s_x & \text{if } i = 5, j = 6 \\ x = 0 & \text{if } i = 6, j = 7 \\ -1 - \frac{\sqrt{2-\alpha^2}}{\sqrt{\alpha^2-1}}x & \text{if } i = 7, j = 8 \\ -\sqrt{\alpha^2-1}(s_x - x) & \text{if } i = 10, j = 11 \\ -\frac{(s_x+x)+2\alpha\sqrt{s_x x}}{\sqrt{\alpha^2-1}} & \text{if } i = 11, j = 12 \end{cases} \quad (\text{ when } 1 < \alpha < \sqrt{2}).$$

We conjecture the following on the bisectors involving $\pi_9(s,t)$.

▶ **Conjecture.** *No point on $b_{i,9} \setminus R$, $i \in \{4, \ldots, 8\}$, can be computed exactly within ACM$\mathbb{Q}$.*

We tried to prove this conjecture by taking a similar approach as in Theorem 3.4. However, the solution to Equation (1) already seems to be of high degree. We therefore did not manage to formulate a point on the bisector as a polynomial equation (not containing roots).

Note that in the more restrictive case where R is a single quadrant and s lies on the boundary, the only types of shortest paths that exist are $\pi_i(s,t)$, for $i \in \{1,2,3,4,5,6,10,11,12\}$. Thus, we can compute the *SPM* in the ACM$\mathbb{Q}$ (the bisectors are given by the equations in Lemma 4.1).

4.2 The source point s lies inside R

In this case we have shortest paths of type $\pi_i(s,t)$, for $i \in \{6,7,8,9,11,12\}$. Hence, the equations of the bisectors of the *SPM* are given by the sides of R, and bisector $b_{6,9}$ if $\alpha < 1$, and bisectors $b_{6,7}, b_{7,8}, b_{6,9}, b_{7,9}, b_{8,9}$ and $b_{11,12}$ if $1 < \alpha < \sqrt{2}$. See Lemma 4.1.

5 Conclusion

We analyzed the WRP when there is only one weighted rectangle R, and showed how to obtain the exact shortest path $\pi(s,t)$ and its length when s lies in or on R. When both s and t lie outside R the exact solution is unsolvable in ACM$\mathbb{Q}$. We obtain similar results in the case where R is a single quadrant. For future work, it would be interesting to analyze if or how we can generalize this to other convex shapes.

References

1 L. Aleksandrov, M. Lanthier, A. Maheshwari, and J.-R. Sack. An ε-approximation algorithm for weighted shortest paths on polyhedral surfaces. In *Scandinavian Workshop on Algorithm Theory*, pages 11–22. Springer, 1998.

2 L. Aleksandrov, A. Maheshwari, and J.-R. Sack. Approximation algorithms for geometric shortest path problems. In *Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing*, pages 286–295, 2000.

3 L. Aleksandrov, A. Maheshwari, and J.-R. Sack. Determining approximate shortest paths on weighted polyhedral surfaces. *Journal of the ACM*, 52(1):25–53, 2005.

4 P. Bose, G. Esteban, and A. Maheshwari. Weighted shortest path in equilateral triangular meshes. In *Proceedings of the 34th Canadian Conference on Computational Geometry*, pages 60–67, 2022.

5 J.-L. De Carufel, C. Grimm, A. Maheshwari, M. Owen, and M. H. M. Smid. A note on the unsolvability of the weighted region shortest path problem. *Computational Geometry*, 47(7):724–727, 2014. doi:10.1016/j.comgeo.2014.02.004.

6 L. Gewali, A. Meng, J. S. B. Mitchell, and S. Ntafos. Path planning in O/1/infinity weighted regions with applications. Technical report, Cornell University Operations Research and Industrial Engineering, 1988.

7 J. Hershberger and S. Suri. An optimal algorithm for Euclidean shortest paths in the plane. *SIAM Journal on Computing*, 28(6):2215–2256, 1999.

8 J. S. B. Mitchell. Shortest paths among obstacles, zero-cost regions, and roads. Technical report, Cornell University Operations Research and Industrial Engineering, 1987.

9 J. S. B. Mitchell. Shortest paths among obstacles in the plane. In *Proceedings of the Ninth Annual Symposium on Computational Geometry*, pages 308–317, 1993.

10 J. S. B. Mitchell and C. H. Papadimitriou. The weighted region problem: finding shortest paths through a weighted planar subdivision. *Journal of the ACM*, 38(1):18–73, 1991.

11 S. Narayanappa. *Geometric routing*. PhD thesis, 2006.

12 G. J. E. Rawlins and D. Wood. Ortho-convexity and its generalizations. In *Machine Intelligence and Pattern Recognition*, volume 6, pages 137–152. Elsevier, 1988.

13 Z. Sun and J. H. Reif. On finding approximate optimal paths in weighted regions. *Journal of Algorithms*, 58(1):1–32, 2006.

Clustering with Few Disks to Minimize the Sum of Radii*

Mikkel Abrahamsen[1], Sarita de Berg[2], Lucas Meijer[2], André Nusser[3], and Leonidas Theocharous[4]

1 **University of Copenhagen, Denmark**
`miab@di.ku.dk`
2 **Utrecht University, The Netherlands**
`s.deberg@uu.nl, l.meijer2@uu.nl`
3 **CNRS, Inria, I3S, Université Côte d'Azur, France**
`andre.nusser@inria.fr`
4 **Eindhoven University of Technology, The Netherlands**
`l.theocharous@tue.nl`

Abstract

Given a set of n points in the Euclidean plane, the k-MinSumRadius problem asks to cover this point set using k disks with the objective of minimizing the sum of the radii of the disks. A practically and structurally interesting special case of the k-MinSumRadius problem is that of small k. For the 2-MinSumRadius problem, a near-quadratic time algorithm with expected running time $\mathcal{O}(n^2 \log^2 n \log^2 \log n)$ was given over 30 years ago [Eppstein '92].

We present the first improvement of this result, namely, a near-linear time algorithm to compute the 2-MinSumRadius that runs in expected $\mathcal{O}(n \log^2 n \log^2 \log n)$ time. We generalize this result to any constant dimension d, for which we give an $\mathcal{O}(n^{2-1/(\lceil d/2 \rceil+1)+\varepsilon})$ time algorithm. Additionally, we give a near-quadratic time algorithm for 3-MinSumRadius in the plane that runs in expected $\mathcal{O}(n^2 \log^2 n \log^2 \log n)$ time.

Related Version Full version: `https://arxiv.org/abs/2312.08803`

1 Introduction

Clustering seeks to partition a data set in order to obtain a deeper understanding of its structure. There are different clustering notions that cater to different applications. An important subclass is geometric clusterings [6]. In their general form, as defined in [6], geometric clusterings try to partition a set of input points in the plane into k clusters such that some objective function is minimized. More formally, let f be a symmetric k-ary function and w a non-negative function over all subsets of input points. The geometric clustering problem is defined as follows: Given a set P of points in the Euclidean plane and an integer k, partition P into k sets $C_1, \ldots, C_k$ such that $f(w(C_1), \ldots, w(C_k))$ is minimized. Popular choices for the weight function w are the radius of the minimum enclosing disk, and the sum of squared distances from the points to the mean. The function f aggregates the weights of all clusters, for example using the maximum or the sum.

* MA is supported by Starting Grant 1054-00032B from the Independent Research Fund Denmark under the Sapere Aude research career programme and is part of Basic Algorithms Research Copenhagen (BARC), supported by the VILLUM Foundation grant 16582. LM is supported by the Netherlands Organisation for Scientific Research (NWO) under project no. VI.Vidi.213.150. AN was part of Basic Algorithms Research Copenhagen (BARC), supported by the VILLUM Foundation grant 16582 during part of this work. LT is supported by the Dutch Research Council (NWO) through Gravitation-grant NETWORKS-024.002.003. This work was initiated at the Workshop on New Directions in Geometric Algorithms, May 14–19 2023, Utrecht, The Netherlands.

40th European Workshop on Computational Geometry, Ioannina, Greece, March 13–15, 2024.
This is an extended abstract of a presentation given at EuroCG'24. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

Figure 1 The optimal 2-Center clustering (left) compared to the optimal 2-MinSumRadius clustering (right) for the same point set. In this example 2-MinSumRadius clustering better captures the structure of the point set than 2-Center clustering.

Arguably the most popular types of clustering in this setting are k-Center clustering (with f being the maximum, and w being the radius of the minimum enclosing disk) and k-Means clustering (with f being the sum, and w being the sum of squared distances to the mean). While geometric clusterings have the advantage that their underlying objective function can be very intuitive, unfortunately the cluster boundaries might sometimes be slightly more complex. For example, the disks defining the clusters can have a large overlap in k-Center clustering (see Figure 1), while in k-Means clustering the boundaries are defined by the Voronoi diagram on the mean points of the clusters (whose complexity has an exponential dependency on the dimension). An instance of geometric clustering for which the cluster boundaries implicitly consist of non-overlapping disks is k-MinSumRadius. In k-MinSumRadius clustering we want to minimize the sum of radii of the k disks with which we cover the input point set. In the geometric clustering setting, this means that the function f is the sum and w is the radius of the minimum enclosing disk. This is the notion of clustering that we consider in this work.

While k-Center and k-Means are both NP-hard in the Euclidean plane when k is part of the input [14, 16], the k-MinSumRadius problem can be solved in polynomial time [12]: $\mathcal{O}(n^{881})$ [1]. Although the running time of the known polynomial-time algorithm can likely be slightly improved using the same techniques, the balanced separators that are used in the algorithm inevitably lead to a high running time. Thus, we believe that further structural insights into the problem — especially with respect to separators — are needed to greatly reduce the exponent of the polynomial running time.

In order to obtain a deeper understanding of the problem and as clustering into a small number of clusters is practically more relevant, we consider the k-MinSumRadius problem for small values of k here. The importance of this setting is reflected in the extensive work that was conducted in the analogous setting for the k-Center and k-Means problem, especially for the case of two clusters [3, 4, 7, 8, 9, 11, 13, 17, 18] — also called *bi-partition*. Bi-partition problems are of interest on their own, but they can additionally be used as a subroutine in hierarchical clustering methods. Even more interestingly, while near-linear time algorithms for 2-Center clustering received a lot of attention [7, 11, 8, 17, 18], the best known algorithm for 2-MinSumRadius still has near-quadratic expected running time $\mathcal{O}(n^2 \log^2 n \log^2 \log n)$ [10], which has seen no improvement in the last 30 years, despite significant work on related problems.

[1] As the radii of minimum enclosing disks can contain square roots, the value of a solution is a sum of square roots. However, it is not known how to compare two sums of square roots in polynomial time in the number of elements. The running time merely counts the number of such comparisons.

Our results

In this work, we break the quadratic barrier for 2-MinSumRadius in the Euclidean plane by presenting a near-linear time algorithm with expected running time $\mathcal{O}(n \log^2 n \log^2 \log n)$. Our method actually extends to any constant dimension $d \geq 3$, again yielding a subquadratic algorithm with running time $\mathcal{O}(n^{2-1/(\lceil d/2 \rceil + 1)+\varepsilon})$. Moreover, we extend our structural insights to planar 3-MinSumRadius — matching the previously best 2-MinSumRadius running time — and give an algorithm with expected $\mathcal{O}(n^2 \log^2 n \log^2 \log n)$ running, which is the first non-trivial result on this special case that we are aware of. The running time for planar 2-MinSumRadius and 3-MinSumRadius can be made deterministic by using the deterministic algorithm to maintain a minimum enclosing ball [10], which increases the running times to $O(n \log^4 n)$ and $O(n^2 \log^4 n)$, respectively.

The main technical contribution leading to these results are structural insights that simplify the problem significantly. Concretely, we show that the points on the boundary of the minimum enclosing disk (or ball, in higher dimensions) of the point set, induce a constant number of directions such that there is a line (or hyperplane) with one of these directions that separates one cluster from the other clusters in an optimal solution. As there are only linearly many combinatorially distinct separator lines for each direction, we have linearly many separators in total that we have to consider. Note that this is the main difference to the previously best algorithm for planar 2-MinSumRadius [10], which considered quadratically many separators. We then check all clusterings induced by these separators using an algorithm from [10] to dynamically maintain a minimum enclosing disk and, in the $k = 3$ case, we use our 2-MinSumRadius algorithm as subroutine. For the higher-dimensional 2-MinSumRadius problem, we similarly use an algorithm to maintain a minimum enclosing ball in any dimension $d \geq 3$ [2]. While our algorithms are interesting in their own right, we additionally hope to enable a better understanding of the general case by uncovering this surprisingly simple structure of separators in the cases $k \in \{2, 3\}$.

Structure of the paper. Due to space limitations, in this paper we focus on presenting the algorithm and correctness of the planar $k = 2$ case, in Section 2 and Section 3. We give a brief overview of the $k = 3$ case in Section 4. As outlined above, the algorithms for the other cases than planar $k = 2$ are similar and rely on an analogous structural result for their respective case, namely that we can identify a linear number of separators out of which one separates one of the optimal clusters from the rest of the clusters. The extension to constant dimensions for the $k = 2$ case as well as the proof for the $k = 3$ case — which is significantly more technical — can be found in the full version [1].

2 Preliminaries

For a set of points $Q \subset \mathbb{R}^2$, let $\mathrm{MED}(Q)$ be the minimum enclosing disk that contains all points of Q and let $r(Q)$ be the radius of $\mathrm{MED}(Q)$.

▶ **Definition 2.1** (k-MinSumRadius)**.** Let P be a set of n points in $\mathbb{R}^d$ and k be a positive integer. The k-MinSumRadius problem asks to partition P into k clusters $C_1, C_2, ..., C_k$ such that $\sum_{i=1}^{k} r(C_i)$ is minimized.

Throughout the paper, let D denote the minimum enclosing disk of the input points of our k-MinSumRadius instance, i.e., $D := \mathrm{MED}(P)$, and let c be the center of D. We say that a point set Q defines a disk D' if $\mathrm{MED}(Q) = D'$. Let p be a point on the boundary of D,

we define p^* to be the diametrically-opposing point of p on D. The diameter $d(p)$ is then the segment from p to p^*.

The next lemma states that in an optimal solution clusters are well-separated, in the sense that their minimum enclosing disks are disjoint.

▶ **Lemma 2.2.** *Given a k-MinSumRadius instance, there exists an optimal solution with clusters $C_1, \ldots, C_k$ such that $MED(C_i) \cap MED(C_j) = \emptyset$ for all distinct $i, j \in [k]$.*

Proof. Consider an arbitrary optimal solution for which there are two clusters C_i, C_j where $\mathrm{MED}(C_i) \cap \mathrm{MED}(C_j) \neq \emptyset$. We replace the clusters C_i, C_j by a single cluster $C' := C_i \cup C_j$. Note that this does not increase the total cost, as $r(C') \leq r(C_i) + r(C_j)$. Hence, we obtain a clustering with one less cluster. We can recursively apply this argument until we either end up with a single cluster (which trivially satisfies the lemma) or all clusters have pairwise non-overlapping minimum enclosing disks. ◀

3 Near-Linear Algorithm for 2-MinSumRadius

In this section, we present our algorithm for 2-MinSumRadius for the plane. We generalize this to higher dimensions in the full version of this paper. For the plane, we prove the following result:

▶ **Theorem 3.1.** *For a set P of n points in the Euclidean plane, an optimal 2-MinSumRadius clustering can be computed in expected $\mathcal{O}(n \log^2 n \log^2 \log n)$ or worst-case $O(n \log^4 n)$ time.*

3.1 Algorithm

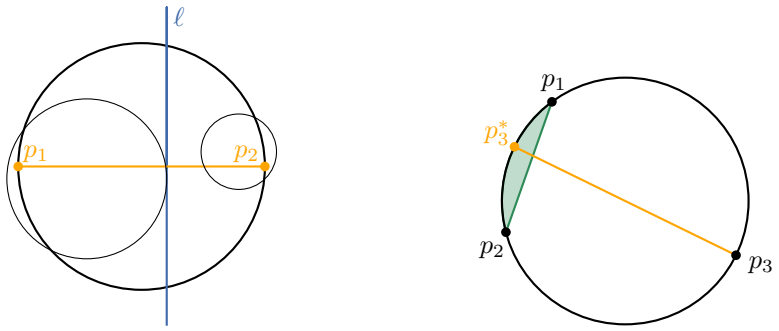
Our algorithm uses the insight that there exist a linear number of separators, such that one of them separates the points in cluster C_1 from those in cluster C_2.

▶ **Lemma 3.2.** *Given a point set P in the Euclidean plane, let C_1, C_2 be an optimal 2-MinSumRadius clustering of P. Furthermore, let p_1, p_2, p_3 be three points of P that define the minimum enclosing disk of P (with potentially $p_2 = p_3$). Then there exists a point $q \in \{p_1, p_2, p_3\}$ and a line ℓ orthogonal to $d(q)$ such that ℓ separates C_1 from C_2.*

We prove this result in Section 3.2. We now explain our algorithm relying on Lemma 3.2.

Algorithm description. Let p_1, p_2, p_3 denote a triple of points in P that define D (possibly $p_2 = p_3$). These points can be computed in $O(n)$ time [15]. We try out every combinatorially distinct line orthogonal to $d(p_i)$ for $i \in \{1, 2, 3\}$ as a separator. We consider the separators orthogonal to a specific $d(p_i)$ in sorted order such that in each step only one point or multiple collinear points switch sides with respect to the separator. This ensures that the minimum enclosing disk on one side of the separator is incremental while it is decremental on the other side. We can therefore use a data structure to dynamically maintain them. We then select the best solution found using these separators.

Correctness follows directly from Lemma 3.2, hence let us consider the running time. We can maintain the minimum enclosing disks A and B in expected $\mathcal{O}(\log^2 n \log^2 \log n)$ or worst-case $O(\log^4 n)$ amortized time per update [10]. So, checking all n separators requires expected $\mathcal{O}(n \log^2 n \log^2 \log n)$ or $O(n \log^4 n)$ worst-case time. As we only have diameters $d(p_1)$, $d(p_2)$, and $d(p_3)$ to handle, we can compute the optimal solution in the same time.



(a) Case 1: Two points define D.

(b) Case 2: At least three points define D.

Figure 2 Visualization of the two cases in the proof for the existence of a separator that is perpendicular to one of the diameters of p_1, p_2, p_3.

3.2 Linear number of cuts

In this subsection we prove Lemma 3.2, which states that it suffices to only check separators orthogonal to one of the diameters $d(p_1)$, $d(p_2)$, or $d(p_3)$. From now on, we assume that the optimal solution consists of two non-empty clusters. If the optimal solution contains only a single cluster, it must be D. So, only solutions that have sum of radii strictly smaller than $r(D)$ are relevant for us in the case of two non-empty clusters.

Proof of Lemma 3.2. We consider two different cases, depending on whether the minimum enclosing disk D of the input points is defined by two or more points.

Case 1: *Two points define D.* We illustrate this case in Figure 2a. Let p_1 and p_2 be the two points that define the minimum enclosing disk. Then, p_1 must be diametrically opposing p_2 on D, so $d(p_1) = d(p_2)$. As we assume that the optimal solution has value less than $r(D)$, we have that p_1 and p_2 must belong to different clusters; without loss of generality, let $p_1 \in C_1$ and $p_2 \in C_2$.

Let ℓ be the line tangent to $\text{MED}(C_1)$ perpendicular to $d(p_1)$ furthest in direction $\overrightarrow{c - p_1}$ (recall that c is the center of D). If ℓ does not separate $\text{MED}(C_1)$ and $\text{MED}(C_2)$, then the projections of the disks $\text{MED}(C_1)$ and $\text{MED}(C_2)$ on $d(p_1)$ cover all of the diameter, so $r(C_1) + r(C_2) \geq r(D)$, which is a contradiction. Hence, ℓ separates the clusters as desired.

Case 2: *At least three points define D.* We illustrate this case in Figure 2b. Let $p_1, p_2, p_3 \in P$ be any three points that jointly define the minimum enclosing disk. In any optimal solution, two out of these three points must be grouped together in a cluster. Without loss of generality, assume that p_1 and p_2 are in the same cluster.

Any disk containing p_1 and p_2 that is defined by points in D (thus, also has radius at most $r(D)$) must contain all of $\overset{\frown}{p_1p_2}$, the smallest of the two arcs on D connecting p_1 and p_2. We now add the artificial point p_3^* to the point set — the diametrically opposing point of p_3. We have that $p_3^* \in \overset{\frown}{p_1p_2}$ as otherwise p_1, p_2, p_3 would lie strictly inside one half of D and could therefore not define the minimum enclosing disk [5, Lemma 2.2]. Hence, adding p_3^* to the point set does not change the optimal solution. Thus, we reduced our problem to Case 1, where two points define the minimum enclosing disk. ◀

4 Near-Quadratic Algorithm for 3-MinSumRadius

Due to the space limit, we can only give a very brief description of the $k = 3$ case here and we refer to the full version for proofs and more details. Our near-quadratic algorithm for the 3-MinSumRadius problem again relies on the structural insight that there are only linearly many cuts that need to be considered in order to find one that separates one cluster from the two other clusters. For the separated cluster we can then simply compute the minimum enclosing disk, while for the other two clusters we use our near-linear time algorithm for 2-MinSumRadius. Hence, we obtain a near-quadratic time algorithm.

The following is the main structural lemma that our algorithm relies on:

▶ **Lemma 4.1.** *Given a point set P in the Euclidean plane, let C_1, C_2, C_3 be an optimal 3-MinSumRadius clustering of P. Furthermore, let p_1, p_2, p_3 be three points in P that define the minimum enclosing disk of P (with potentially $p_2 = p_3$). Then there exists a point $q \in \{p_1, p_2, p_3\}$ and a line ℓ orthogonal to $d(q)$ such that ℓ separates the cluster containing q from the other two clusters.*

The proof can be found in the full version. We prove this lemma by a case distinction on which cluster the points that define the minimum enclosing disk D belong to. We then either reduce to the case in which the endpoints of one of the diameters lie in different clusters (similar to the k = 2 case), or we show that non-existence of a separator results in a contradiction.

We then obtain the following theorem.

▶ **Theorem 4.2.** *For a set P of n points in the Euclidean plane an optimal 3-MinSumRadius can be computed in expected $\mathcal{O}(n^2 \log^2 n \log^2 \log n)$ or worst-case $\mathcal{O}(n^2 \log^4 n)$ time.*

References

1 Mikkel Abrahamsen, Sarita de Berg, Lucas Meijer, André Nusser, and Leonidas Theocharous. Clustering with few disks to minimize the sum of radii. *CoRR*, abs/2312.08803, 2023. arXiv:2312.08803, doi:10.48550/ARXIV.2312.08803.

2 Pankaj K. Agarwal and Jirí Matousek. Dynamic half-space range reporting and its applications. *Algorithmica*, 13(4):325–345, 1995. doi:10.1007/BF01293483.

3 Pankaj K. Agarwal and Micha Sharir. Planar geometric location problems. *Algorithmica*, 11(2):185–195, 1994. doi:10.1007/BF01182774.

4 Daniel Aloise, Amit Deshpande, Pierre Hansen, and Preyas Popat. NP-hardness of Euclidean sum-of-squares clustering. *Mach. Learn.*, 75(2):245–248, 2009. doi:10.1007/s10994-009-5103-0.

5 Mihai Badoiu, Sariel Har-Peled, and Piotr Indyk. Approximate clustering via core-sets. In *Proceedings of the 34th Annual ACM Symposium on Theory of Computing, STOC*, pages 250–257. ACM, 2002. doi:10.1145/509907.509947.

6 Vasilis Capoyleas, Günter Rote, and Gerhard J. Woeginger. Geometric clusterings. *J. Algorithms*, 12(2):341–356, 1991. doi:10.1016/0196-6774(91)90007-L.

7 Timothy M. Chan. More planar two-center algorithms. *Comput. Geom.*, 13(3):189–198, 1999. doi:10.1016/S0925-7721(99)00019-X.

8 Kyungjin Cho and Eunjin Oh. Optimal algorithm for the planar two-center problem. *CoRR*, abs/2007.08784, 2020. arXiv:2007.08784.

9 Sanjoy Dasgupta and Yoav Freund. Random projection trees for vector quantization. *IEEE Trans. Inf. Theory*, 55(7):3229–3242, 2009. doi:10.1109/TIT.2009.2021326.

10 David Eppstein. Dynamic three-dimensional linear programming. *INFORMS J. Comput.*, 4(4):360–368, 1992. doi:10.1287/ijoc.4.4.360.

11 David Eppstein. Faster construction of planar two-centers. In *Proceedings of the 8th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 131–138. ACM/SIAM, 1997.

12 Matt Gibson, Gaurav Kanade, Erik Krohn, Imran A. Pirwani, and Kasturi R. Varadarajan. On clustering to minimize the sum of radii. *SIAM J. Comput.*, 41(1):47–60, 2012. doi:10.1137/100798144.

13 Mary Inaba, Naoki Katoh, and Hiroshi Imai. Applications of weighted voronoi diagrams and randomization to variance-based k-clustering. In *Proceedings of the 10th Annual Symposium on Computational Geometry, SoCG*, pages 332–339. ACM, 1994. doi:10.1145/177424.178042.

14 Meena Mahajan, Prajakta Nimbhorkar, and Kasturi R. Varadarajan. The planar k-means problem is NP-hard. *Theor. Comput. Sci.*, 442:13–21, 2012. doi:10.1016/j.tcs.2010.05.034.

15 Nimrod Megiddo. Linear-time algorithms for linear programming in $\mathbb{R}^3$ and related problems. *SIAM J. Comput.*, 12(4):759–776, 1983. doi:10.1137/0212052.

16 Nimrod Megiddo and Kenneth J. Supowit. On the complexity of some common geometric location problems. *SIAM J. Comput.*, 13(1):182–196, 1984. doi:10.1137/0213014.

17 Micha Sharir. A near-linear algorithm for the planar 2-center problem. *Discret. Comput. Geom.*, 18(2):125–134, 1997. doi:10.1007/PL00009311.

18 Haitao Wang. On the planar two-center problem and circular hulls. *Discret. Comput. Geom.*, 68(4):1175–1226, 2022. doi:10.1007/s00454-021-00358-5.

Extending simple monotone drawings*

Jan Kynčl[1] and Jan Soukup[1]

1 **Department of Applied Mathematics, Charles University, Faculty of Mathematics and Physics, Malostranské nám. 25, 118 00 Praha 1, Czech Republic**
{kyncl,soukup}@kam.mff.cuni.cz

Abstract

We prove the following variant of Levi's Enlargement Lemma: for an arbitrary arrangement $\mathcal{A}$ of x-monotone pseudosegments in the plane and a pair of points a, b with distinct x-coordinates and not on the same pseudosegment, there exists a simple x-monotone curve with endpoints a, b that intersects every curve of $\mathcal{A}$ at most once. As a consequence, every simple monotone drawing of a graph can be extended to a simple monotone drawing of a complete graph.

Related Version arXiv:2312.17675

1 Introduction

Given $k \geq 1$, a finite set $\mathcal{A}$ of simple curves in the plane is called an *arrangement of k-strings* if every pair of the curves of $\mathcal{A}$ intersects at most k times, and every intersection point is a proper crossing or a common endpoint. An arrangement of 1-strings is also called an *arrangement of pseudosegments*, and each curve in the arrangement is called a *pseudosegment*. In this paper, we represent simple curves as subsets of the plane that are homeomorphic images of a closed interval.

A simple curve γ in the plane is *x-monotone*, shortly *monotone*, if γ intersects every line parallel to the y-axis at most once.

Given an arrangement $\mathcal{A}$ of monotone pseudosegments in the plane and a pair of points a, b with distinct x-coordinates and not on the same pseudosegment, we say that $\mathcal{A}$ is *(a, b)-extendable* if there exists a monotone curve with endpoints a, b that intersects every curve of $\mathcal{A}$ at most once. We say that $\mathcal{A}$ is *extendable* if it is (a, b)-extendable for all possible choices of a and b.

Our main result is the following.

▶ **Theorem 1.1.** *Every arrangement of monotone pseudosegments in the plane is extendable.*

The proof of Theorem 1.1 can be turned into an algorithm: the new pseudosegment extending an arrangement $\mathcal{A}$ and joining two given points a and b is constructed in at most $|\mathcal{A}|$ steps. Starting with an initial curve from a to b, in each step the curve is locally rerouted along one pseudosegment of $\mathcal{A}$.

A drawing of a graph in the plane is *simple* if every pair of edges has at most one common point, either a common endpoint or a proper crossing. A drawing of a graph is *monotone* if every edge is drawn as a monotone curve and no two vertices share the same x-coordinate. We have the following direct consequence of Theorem 1.1, illustrated in Figure 1.

▶ Corollary 1.2. Every simple monotone drawing of a graph in the plane can be extended to a simple monotone drawing of the complete graph with the same set of vertices.

* Supported by project 23-04949X of the Czech Science Foundation (GAČR) and by the grant SVV–2023–260699.

40th European Workshop on Computational Geometry, Ioannina, Greece, March 13–15, 2024.
This is an extended abstract of a presentation given at EuroCG'24. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

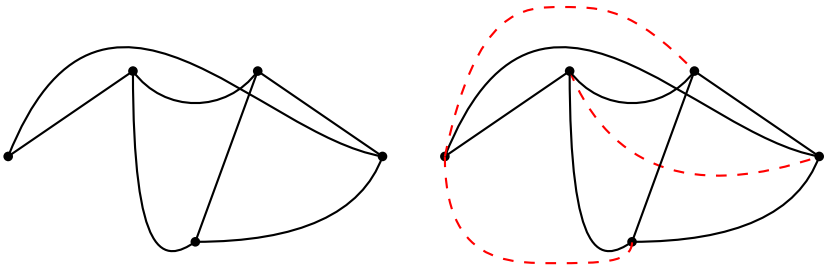


Figure 1 Left: a simple monotone drawing of a graph. Right: an extension of the drawing on the left to a simple monotone drawing of a complete graph. The added edges are dashed.

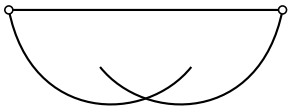


Figure 2 An example of an arrangement of three pseudosegments that cannot be extended to pseudolines forming a pseudoline arrangement.

In the full version of this article we also study the extendability problem for cylindrically monotone arrangements. We show that extending an arrangement of cylindrically monotone pseudosegments is not always possible; in fact, the corresponding decision problem is NP-hard.

We prove Theorem 1.1 in Section 2.

1.1 Related results

A *pseudoline* in the plane is an image of a Euclidean line under a homeomorphism of the plane; in other words, a pseudoline is a homeomorphic image of the set $\mathbb{R}$, unbounded in both directions. An *arrangement of pseudolines* is a finite set of pseudolines such that every pair of them has exactly one crossing, and no other common intersection point. Pseudolines are also often defined in the projective plane, as nonseparating simple closed curves.

Levi's Enlargement Lemma [9] states that for every arrangement of pseudolines and every pair of points a, b not on the same pseudoline, one can draw a new pseudoline through a and b, crossing every curve from the given arrangement exactly once. The lemma has several alternative proofs in the literature [3, 10].

By a classical result of Goodman [6], [5, Theorem 5.1.4], every arrangement of pseudolines can be transformed by a homeomorphism of the plane into an arrangement of monotone pseudolines, or a so-called *wiring diagram*. Therefore, monotone arrangements of pseudosegments can be considered as a generalization of pseudoline arrangements. On the other hand, Figure 2 shows an example that not every monotone arrangement of pseudosegments can be seen as a "restriction" of a pseudoline arrangement, and so Theorem 1.1 does not easily follow from Levi's Lemma. See Arroyo, Bensmail and Richter [1, Figure 2] for more examples. Since a pseudoline (in the projective plane) can be considered as a union of two pseudosegments, Theorem 1.1 can also be considered as a generalization of "a half" of Levi's Lemma.

A simple drawing of the disjoint union of two 2-paths that cannot be extended to a simple drawing of K_6 was constructed by Eggelton [4, Diagram 15(ii)] and later rediscovered by

the first author [8, Figure 9]. Later a few more examples of non-extendable simple drawings were constructed [7, Figures 1, 10]. None of these drawings are homeomorphic to monotone drawings, which follows, for example, from Theorem 1.1.

Arroyo et al. [2] showed that it is NP-hard to decide, given an arrangement $\mathcal{A}$ of pseudosegments and a pair of points a, b, whether a and b can be joined by a simple curve crossing each pseudosegment of $\mathcal{A}$ at most once. Our NP-hardness proof in the full version is a simple adaptation of this result to cylindrically monotone arrangements.

2 Monotone arrangements in the plane

We start with a few definitions and tools for analyzing x-monotone arrangements. Given a pair of points a, b in the plane, we write $a \prec b$ if a has a smaller x-coordinate than b. Clearly, $\prec$ is a strict linear order on the points of any monotone curve.

We can naturally talk about objects lying "below" and "above" monotone curves. Let a, b be points such that $a \prec b$. For any monotone curve γ we denote by $\gamma[a, b]$ and $\gamma(a, b)$ the subset of γ formed by the points x of γ satisfying $a \preceq x \preceq b$ and $a \prec x \prec b$, respectively. Similarly, for an arrangement $\mathcal{B}$ of monotone pseudosegments we denote by $\mathcal{B}[a, b]$ the arrangement of pseudosegments where we replace each $\gamma \in \mathcal{B}$ by $\gamma[a, b]$.

By *consecutive intersections* of two monotone curves with finitely many intersections we mean consecutive intersections with respect to their x-coordinates. Let α, β be two monotone curves with finitely many intersections. Let a, b be two consecutive intersections of α, β such that $a \prec b$. Then the only intersections of $\alpha[a, b]$ with $\beta[a, b]$ are the points a and b. In this case we say that the curves α and β form a *bigon*. Furthermore, if $\alpha(a, b)$ lies above $\beta(a, b)$ we say that α and β form an *α-top*, or equivalently, a *β-bottom* bigon.

The *lower envelope* $\mathrm{low}(\mathcal{U})$ of a set $\mathcal{U}$ of curves is the set of all points p of these curves such that no other point of any curve of $\mathcal{U}$ with the same x-coordinate as p is below p. Note that if $\mathcal{U}$ is an arrangement of monotone pseudosegments, then $\mathrm{low}(\mathcal{U})$ is a finite union of connected parts of pseudosegments.

2.1 Proof of Theorem 1.1

Let $\mathcal{A}$ be an arrangement of monotone pseudosegments. Let a, b, with $a \prec b$, be points that are not on the same pseudosegment of $\mathcal{A}$. We need to find a monotone curve from a to b that intersects every curve of $\mathcal{A}$ at most once. Since every curve of $\mathcal{A}$ is monotone, we can without loss of generality assume that $\mathcal{A} = \mathcal{A}[a, b]$.

Let $\mathcal{A}'$ be an arrangement of monotone pseudosegments formed by all pseudosegments of $\mathcal{A}$ together with three new segments τ_1, τ_2, τ_3, defined as follows. The segment τ_1 is an almost vertical segment starting in a and ending in some new point to the right of a and above all pseudosegments of $\mathcal{A}$. Similarly, τ_3 is an almost vertical segment ending in b and starting in some new point to the left of b and above all pseudosegments of $\mathcal{A}$. Finally, τ_2 is a horizontal segment crossing τ_1 and τ_3, and lying entirely above all pseudosegments of $\mathcal{A}$; see Figure 3. In this way, $\mathrm{low}(\{\tau_1, \tau_2, \tau_3\})$ is a monotone curve connecting a and b "from above", so that every pseudosegment $\gamma \in \mathcal{A}$ intersects it at most twice. Furthermore no $\gamma \in \mathcal{A}$ forms a γ-top bigon with $\mathrm{low}(\{\tau_1, \tau_2, \tau_3\})$ (it can only form a γ-bottom bigon).

In order to find an extending curve we do the following. We find a nonempty subset $\mathcal{U} \subseteq \mathcal{A}'$ of pseudosegments such that the lower envelope of $\mathcal{U}$ is a monotone curve connecting a to b, intersecting every pseudosegment of $\mathcal{A}' \setminus \mathcal{U}$ at most once. Furthermore, we find $\mathcal{U}$ so that no pseudosegment α touches $\mathrm{low}(\mathcal{U})$ from below in an inner point of α. After finding such $\mathcal{U}$, a new pseudosegment connecting a and b can clearly be drawn slightly below the

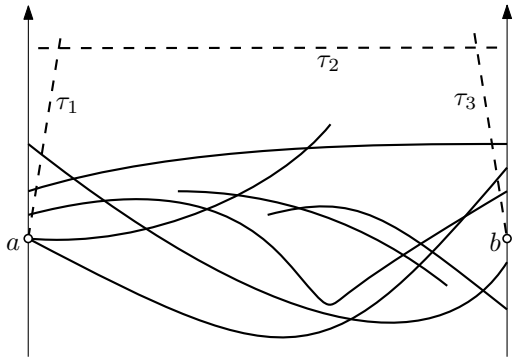


Figure 3 An arrangement of monotone pseudosegments with three added segments τ_1, τ_2, τ_3 connecting points a, b "from above".

lower envelope of $\mathcal{U}$ and will indeed intersect every pseudosegment of $\mathcal{A}'$ at most once. Thus, if such $\mathcal{U}$ exists, $\mathcal{A}'$, and consequently $\mathcal{A}$, is (a, b)-extendable.

We find $\mathcal{U}$ inductively. We start with $\mathcal{U}_0 = \{\tau_1, \tau_2, \tau_3\}$ and always look at the lower envelope of $\mathcal{U}_i$. In the ith step we select an arbitrary pseudosegment γ_i of $\mathcal{A}' \setminus \mathcal{U}_{i-1}$ intersecting $\text{low}(\mathcal{U}_{i-1})$ at least twice. If there is no such γ_i then $\mathcal{U} = \mathcal{U}_{i-1}$ and we are done. Otherwise, we set $\mathcal{U}_i = \mathcal{U}_{i-1} \cup \{\gamma_i\}$. The number of pseudosegments is finite, so this process finishes with a set $\mathcal{U}$ such that the lower envelope of $\mathcal{U}$ intersects every pseudosegment of $\mathcal{A}' \setminus \mathcal{U}$ at most once.

Additionally, we prove that the induction preserves the following invariants for every $\mathcal{U}_i$.

(I1) No pseudosegment α of $\mathcal{A}' \setminus \mathcal{U}_i$ forms an α-top bigon with $\text{low}(\mathcal{U}_i)$.

(I2) No pseudosegment α of $\mathcal{A}' \setminus \mathcal{U}_i$ touches $\text{low}(\mathcal{U}_i)$ from below in an inner point of α.

(I3) The lower envelope of $\mathcal{U}_i$ contains no endpoints of any pseudosegment of $\mathcal{A}'$ except for the points a and b.

(I4) The lower envelope of $\mathcal{U}_i$ is connected and contains a and b. Hence, it is a monotone curve connecting a to b.

In particular, by (I4), the lower envelope of $\mathcal{U}$ is a monotone curve connecting a to b and, by (I2), no pseudosegment α of $\mathcal{A}' \setminus \mathcal{U}$ touches $\text{low}(\mathcal{U})$ from below in an inner point of α. Since $\text{low}(\mathcal{U})$ intersects every pseudosegment of $\mathcal{A}' \setminus \mathcal{U}$ at most once by its construction, $\mathcal{A}$ is (a, b)-extendable by the previous discussion. Thus, it suffices to prove the correctness of these invariants to finish the proof.

The invariants hold for $\mathcal{U}_0$ by the construction of τ_1, τ_2 and τ_3. Suppose all invariants hold for $\mathcal{U}_{i-1}$. In particular, $\text{low}(\mathcal{U}_{i-1})$ is a monotone curve connecting a to b by invariant (I4). We show that all invariants also hold for $\mathcal{U}_i$.

The pseudosegment γ_i intersects $\text{low}(\mathcal{U}_{i-1})$ at least twice. We show that γ_i intersects $\text{low}(\mathcal{U}_{i-1})$ exactly twice. Suppose, for contradiction, that there are three consecutive intersections c, d and e of γ_i with $\text{low}(\mathcal{U}_{i-1})$ such that $c \prec d \prec e$. Then $\gamma_i[c, d]$ with $\text{low}(\mathcal{U}_{i-1})[c, d]$ forms a bigon and so does $\gamma_i[d, e]$ with $\text{low}(\mathcal{U}_{i-1})[d, e]$. By invariant (I1) both of these bigons must be $\text{low}(\mathcal{U}_{i-1})$-top bigons. However, in this case γ_i touches $\text{low}(\mathcal{U}_{i-1})$ from below in the point d. That is not possible by invariant (I2). Thus, γ_i intersects $\text{low}(\mathcal{U}_{i-1})$ exactly twice. Furthermore, by invariant (I1), γ_i and $\text{low}(\mathcal{U}_{i-1})$ form a γ_i-bottom bigon.

Let x and y be the two intersection points of γ_i and $\text{low}(\mathcal{U}_{i-1})$. Refer to Figure 4. Since γ_i and $\text{low}(\mathcal{U}_{i-1})$ form a γ_i-bottom bigon, the only part of the curve γ_i that lies below $\text{low}(\mathcal{U}_{i-1})$ is exactly $\gamma_i(x, y)$. Thus, the lower envelope of $\mathcal{U}_{i-1} \cup \{\gamma_i\}$ is a monotone curve

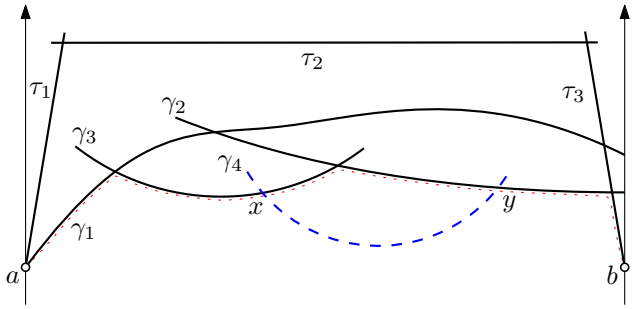


Figure 4 Induction step in the proof of Theorem 1.1. In the ith step (fourth step in the figure) we add pseudosegment γ_i (dashed) intersecting the lower envelope (dotted) of the previous segments twice. The lower envelope remains a connected curve connecting a with b and not containing any other endpoints of pseudosegments even after this addition.

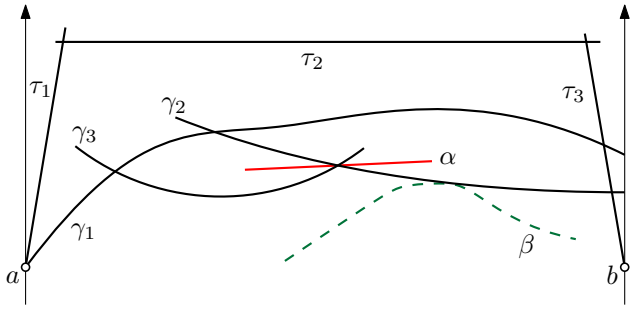


Figure 5 Induction step in the proof of Theorem 1.1. During the selection of $\mathcal{U}$ some pseudosegments may touch $\mathrm{low}(\mathcal{U})$ from above but never from below. Pseudosegment α touches $\mathrm{low}(\{\tau_1, \tau_2, \tau_3, \gamma_1, \gamma_2, \gamma_3\})$ from above. On the other hand, β cannot be in the same arrangement of pseudosegments since it touches γ_2.

connecting a and b. Therefore, invariant (I4) holds also for $\mathcal{U}_i$.

Since γ_i is a pseudosegment, its subset $\gamma_i(x, y)$ contains no endpoint of any pseudosegment of $\mathcal{A}'$. Since $\mathrm{low}(\mathcal{U}_i) \setminus \gamma_i(x, y) \subseteq \mathrm{low}(\mathcal{U}_{i-1})$ and invariant (I3) holds for $\mathcal{U}_{i-1}$, invariant (I3) also holds for $\mathcal{U}_i$.

Now, suppose that invariant (I2) does not hold, that is, there exists some pseudosegment β of $\mathcal{A}' \setminus \mathcal{U}_i$ that touches $\mathrm{low}(\mathcal{U}_i)$ from below in an inner point of β. Refer to Figure 5. By the definition of an arrangement of pseudosegments, the touching point is not an endpoint of any pseudosegment of $\mathcal{A}'$. Thus, β has to touch γ_i or $\mathrm{low}(\mathcal{U}_{i-1})$ in an inner point of β, a contradiction. Hence, invariant (I2) also holds for $\mathcal{U}_i$. Note that the analogous statement for touchings from above does not hold, that is, there may exist some pseudosegment α of $\mathcal{A}' \setminus \mathcal{U}_i$ that both touches $\mathrm{low}(\mathcal{U}_i)$ from above in an inner point of α and touches none of γ_i or $\mathrm{low}(\mathcal{U}_i)$ in an inner point of α.

Finally, suppose that invariant (I1) does not hold, that is, there exists some pseudosegment ρ of $\mathcal{A}' \setminus \mathcal{U}_i$ that together with $\mathrm{low}(\mathcal{U}_i)$ forms a ρ-top bigon. Call s and t the vertices of this bigon and assume $s \prec t$. See Figure 6.

If s and t both lie on $\gamma_i[x, y]$, then ρ and γ_i intersect twice, a contradiction. Otherwise s or t does not lie on $\gamma_i[x, y]$. Without loss of generality assume that t does not lie on $\gamma_i[x, y]$ and $y \prec t$. Then s either lies on $\mathrm{low}(\mathcal{U}_{i-1})$ or below it. In both cases $\rho[s, t]$ intersects

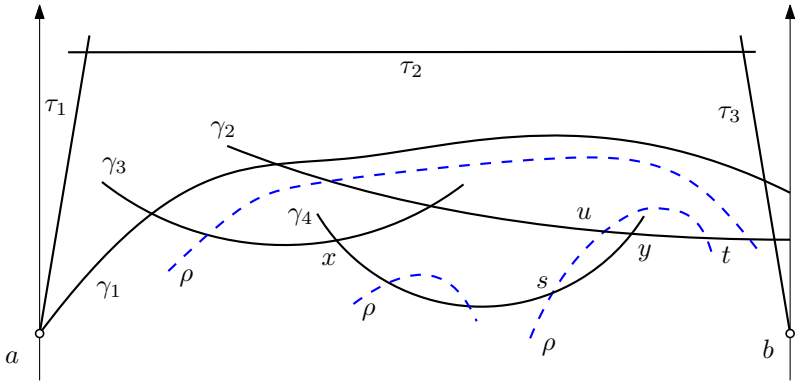


Figure 6 Induction step in the proof of Theorem 1.1. If there was some pseudosegment ρ that together with the lower envelope ($\text{low}(\{\tau_1, \tau_2, \tau_3, \gamma_1, \gamma_2, \gamma_3, \gamma_4\})$ in the picture) formed a ρ-top bigon, it would either form a ρ-top bigon with the previous lower envelope or intersect twice the segment that was added as the last. In the picture, there are three such possible ρ's.

$\text{low}(\mathcal{U}_{i-1})$ in some point other than t since $\rho[s,t]$ together with $\text{low}(\mathcal{U}_i)$ forms a ρ-top bigon. Denote the rightmost intersection of $\rho[s,t]$ and $\text{low}(\mathcal{U}_{i-1})$ other than t by u. Then $\rho(u,t)$ lies above $\text{low}(\mathcal{U}_{i-1})$ and so $\rho[u,t]$ together with $\text{low}(\mathcal{U}_i)$ forms a ρ-top bigon, a contradiction with invariant (I1) for $\mathcal{U}_{i-1}$. This concludes the proof of Theorem 1.1.

References

1 A. Arroyo, J. Bensmail and R. B. Richter, Extending drawings of graphs to arrangements of pseudolines, *J. Comput. Geom.* **12** (2021), no. 2, 3–24.

2 A. Arroyo, F. Klute, I. Parada, B. Vogtenhuber, R. Seidel and T. Wiedera, Inserting one edge into a simple drawing is hard, *Discrete Comput. Geom.* **69** (2023), no. 3, 745–770.

3 A. Arroyo, D. McQuillan, R. B. Richter and G. Salazar, Levi's Lemma, pseudolinear drawings of K_n, and empty triangles, *J. Graph Theory* **87** (2018), no. 4, 443–459.

4 Roger B. Eggelton, Crossing numbers of graphs, PhD thesis, University of Calgary, 1973.

5 S. Felsner and J. E. Goodman, Pseudoline arrangements, *Handbook of Discrete and Computational Geometry*, Third edition, Edited by Jacob E. Goodman, Joseph O'Rourke and Csaba D. Tóth, Discrete Mathematics and its Applications (Boca Raton), CRC Press, Boca Raton, FL, 2018. ISBN: 978-1-4987-1139. Electronic version: http://www.csun.edu/~ctoth/Handbook/HDCG3.html (accessed May 2023).

6 J. E. Goodman, Proof of a conjecture of Burr, Grünbaum, and Sloane, *Discrete Math.* **32** (1980), no. 1, 27–35.

7 J. Kynčl, J. Pach, R. Radoičić and G. Tóth, Saturated simple and k-simple topological graphs, *Comput. Geom.* **48** (2015), no. 4, 295–310.

8 J. Kynčl, Improved enumeration of simple topological graphs, *Discrete Comput. Geom.* **50**(3) (2013), 727–770.

9 F. Levi, Die Teilung der projektiven Ebene durch Gerade oder Pseudogerade, *Berichte Math.-Phys. Kl. Sächs. Akad. Wiss. Leipzig* **78** (1926), 256–267.

10 M. Schaefer, A proof of Levi's Extension Lemma, arXiv:1910.05388v1 (2019).

Hardness and modifications of the weak graph distance

Maike Buchin[1] **and Wolf Kißler**[1]

1 **Faculty of Computer Science, Ruhr-Universität Bochum**
{maike.buchin, wolf.kissler}@rub.de

Abstract

The weak graph distance is a distance measure for immersed graphs. We extend previous NP-hardness results for deciding this distance. Also, we present variations that we conjecture to be fixed-parameter tractable under regularity assumptions when parameterized in the number of crossings.

1 Introduction

Embedded and immersed graphs are widely used natural representations for many kinds of geometric networks. Given multiple models of the same network or representations of related networks, one is typically interested in comparing the models. In recent years, many different distance measures for embedded and immersed graphs have been proposed, cf. [2].

Two such distance measures are the strong and weak graph distance proposed by Akitaya et al. [1], which are based on the strong and weak Fréchet distance for polygonal curves, respectively. Both distance measures are metrics, cf. [2]. A key advantage of these measures is that they capture both geometric and topological (dis)similarity. As discussed in [1], first experiments on reconstructions of real road networks showed promising results.

The strong graph distance is NP-complete to approximate within a 1.10566 ratio even on plane graphs. The best known exact algorithm due to [1] runs in an XP-like time bound when parameterized in the number of faces.

For the weak graph distance, there is a quadratic-time decision algorithm on spike-free (i.e., cycles are embedded in a nice way) plane graphs. Akitaya et al. also showed that when both graphs are immersed in $\mathbb{R}^2$, the weak graph distance is NP-complete to decide.

Hence we are interested in whether (a variant of) the weak graph distance is tractable on realistic networks, in particular those with few edge crossings. For this, we first extend the hardness result of [1] in showing that deciding the directed weak graph distance remains NP-complete even if the source graph is plane. Moreover, we show that deciding the directed distance is NP-complete for G_1, G_2 embedded in $\mathbb{R}^d$ for $d \geq 3$. In both scenarios, constant-factor approximation is NP-complete as well.

Then we suggest the family of crossing-rigid weak graph distances as alternative distance measures. Under reasonable regularity assumptions, we conjecture that these measures admit fixed-parameter tractable algorithms when parameterized in the numbers of crossings.

1.1 The weak graph distance

Here, we introduce relevant notation from [1]. First, we recall the weak Fréchet distance:

▶ **Definition 1.1.** Let $s_1, s_2 \colon [0,1] \to \mathbb{R}^2$ be curves. Define their *weak Fréchet distance* by

$$\delta_{wF}(s_1, s_2) := \inf_{\alpha,\beta} \max_{t\in[0,1]} d(s_1(\alpha(t)), s_2(\beta(t))),$$

40th European Workshop on Computational Geometry, Ioannina, Greece, March 13–15, 2024.
This is an extended abstract of a presentation given at EuroCG'24. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

where α, β range over all continuous self-surjections of $[0,1]$ that keep the endpoints fixed and d is the standard Euclidean metric.

The weak graph distance is defined for embedded and immersed graphs. We use the terms embedded and immersed in the topological sense, that is, an embedding is (essentially) a crossing-free drawing in $\mathbb{R}^d$ and an immersion is any drawing that may also contain crossings. Moreover, we use the term plane graph for embeddings of planar graphs in $\mathbb{R}^2$.

In the following, let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be graphs immersed in $\mathbb{R}^2$ using straight-line immersions. We will slightly abuse notation and refer to the immersions of graphs, edges and vertices using the same notation as for the abstract graphs.

To define the weak graph distance, we first define graph mappings:

▶ **Definition 1.2.** A *graph mapping* $s\colon G_1 \to G_2$ is a map that maps
1. each vertex $v \in V_1$ to a point $s(v)$ on an edge of G_2 and
2. each edge $\{u, v\} \in E_1$ to a simple path from $s(u)$ to $s(v)$ in G_2.

The weak graph distance is now defined as the maximum weak Fréchet distance between an edge and its image under a (globally) optimal graph mapping:

▶ **Definition 1.3.** For immersed graphs G_1, G_2, define the *directed weak graph distance* via

$$\vec{\delta}_{wG}(G_1, G_2) := \min_{s:G_1 \to G_2} \max_{e \in E_1} \delta_{wF}(e, s(e)),$$

where s ranges over all graph mappings and e and $s(e)$ refer to the corresponding immersions as curves in $\mathbb{R}^2$. The *(undirected) weak graph distance* between G_1 and G_2 is defined as

$$\delta_{wG}(G_1, G_2) := \max(\vec{\delta}_{wG}(G_1, G_2), \vec{\delta}_{wG}(G_2, G_1)).$$

Lastly, we outline the general decision alg. described in [1]. For that, we define placements:

▶ **Definition 1.4.** An ε-placement of a vertex v is a connected component (w.r.t. the canonical topologization of G_2 as a simplicial complex) of $G_2 \cap B_\varepsilon(v)$. A *weak edge placement* of an edge $e = \{u, v\} \in E_1$ is a path P in G_2 that connects placements of u and v, respectively, such that $\delta_{wF}(e, P) \leq \varepsilon$. A *weak ε-placement* of G_1 is a graph mapping $s\colon G_1 \to G_2$ that maps each edge to a weak ε-placement.

Furthermore, we call a vertex placement C_v *weakly valid* if each adjacent vertex u has a placement C_u such that C_v and C_u are connected by a weak ε-placement of $\{u, v\}$. Otherwise, we call the placement *weakly invalid*.

The general decision algorithm now proceeds as follows:

Algorithm 1 General Decision Algorithm [1]

1: Compute vertex placements.
2: Compute mutual reachability information for vertex placements.
3: Prune invalid placements.
4: Decide if there exists a placement for the whole graph G_1.

As described in [1], steps 1-3 can be performed in quadratic time for general immersed graphs. However, existence of a weakly valid placement for each vertex does not imply existence of a weak placement of the whole graph, cf. Fig. 1. Thus, step 4 is non-trivial in general. In [1], it is shown that step 4 is in fact trivial if both graphs are plane and the embedding of G_1 meets the following regularity condition:

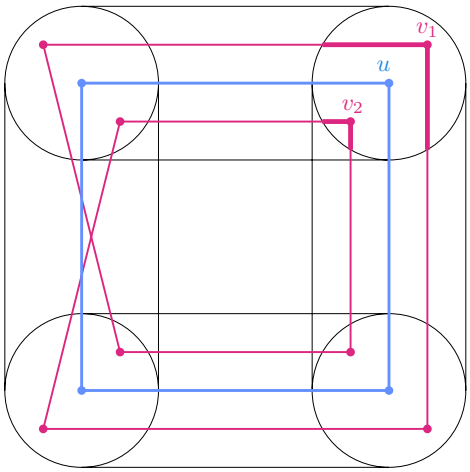


Figure 1 Graphs G_1 (blue) and G_2 (red) such that each vertex of G_1 has exactly two weakly valid placements, but no weak placement for G_1 exists.

▶ **Definition 1.5.** An immersed graph G is called *spike-free* if each cycle C of G is 2ε-thick and, for each three consecutive vertices $u, v, w \in C$, the ε-ball around u does not intersect the ε-tube around the edge $\{v, w\}$.

2 Hardness of deciding the weak graph distance

Here, we extend the hardness result from [1] to show that deciding the directed weak graph distance remains NP-hard even if the source graph is plane. Our proof idea resembles their original proof idea. However, their reduction is from binary CSP, where we cannot guarantee planarity of the resulting graphs. Instead, we reduce from 3-colorability of planar graphs.

▶ **Theorem 2.1.** *Deciding whether $\vec{\delta}_{wG}(G_1, G_2) \leq \varepsilon$ is NP-complete even if G_1 is plane.*

Proof idea. A more detailed version of this proof will be included in the full version due to space restrictions. We reduce from planar 3-coloring, which is NP-complete due to [6]. Given a planar graph $G = (V, E)$, we construct an embedding of G on a grid in linear time, cf. [3]. We construct a graph G_c by placing vertices $w_{v,i}$ in ε-balls around each $v \in V$ for $i \in \{1, 2, 3\}$. For adjacent vertices u, v in G, G_c has an edge $e_{u,v,i,j}$ connecting $w_{u,i}$ to $w_{v,j}$ iff $i \neq j$ and $u \prec v$ for some fixed linear order $\preceq$ on V.

By choosing ε sufficiently small, we can achieve that each $u \in V$ has exactly three placements in G_c corresponding to the $w_{u,i}$. Our idea is that placing u onto the placement corresponding to $w_{u,i}$ is comparable to coloring u with color i. However, since our distance measure is based on the weak Fréchet distance, multiple edges might be used to connect same colored placements of adjacent vertices.

This can be prevented by inserting a vertex in the middle of each edge of G. Denote the resulting immersed graph by G_s. Then, all edges of G must be placed essentially through some $e_{u,v,i,j}$, which exists iff $i \neq j$. Thus, a consistent ε-placement of G_s onto G_c must use globally consistent $e_{u,v,i,j}$, implying that G is 3-colorable. See Fig. 2 for an illustration. ◀

Starting with 4-regular planar graphs instead, the problem remains NP-complete due to [4] and we obtain slightly stronger results. Those observations will be included in the full version due to space restrictions, as well as detailed proofs of the following results:

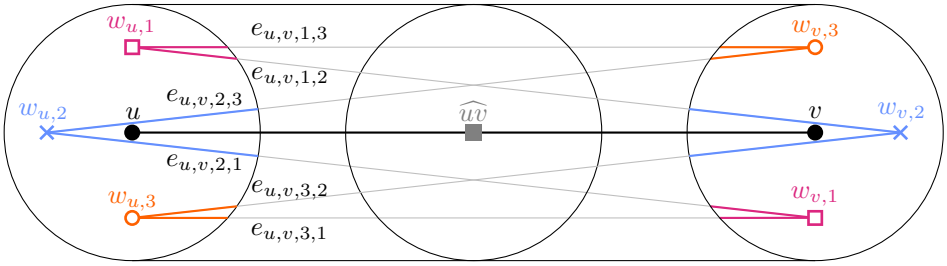


Figure 2 Illustration of the reduction in Thm. 2.1 on a single edge $\{u, v\}$ of the input graph.

▶ **Corollary 2.2.** *It is NP-hard to approximate $\vec{\delta}_{wG}(G_1, G_2)$ within any constant ratio $c \geq 1$ even if G_1 is embedded in $\mathbb{R}^2$.*

Proof idea. In the above construction, place the vertices of G_c within $\frac{\varepsilon}{c}$-balls instead. ◀

▶ **Theorem 2.3.** *The (directed) weak graph distance is NP-hard to approximate within any constant ratio $c \geq 1$ for graphs G_1, G_2 embedded in $\mathbb{R}^d$ for all $d \geq 3$.*

Proof idea. In a similar construction, embed the vertices of G, G_c on the moment curve. [5, Lemma 5.4.2] implies planarity. Approximation hardness is analogous to Cor. 2.2. ◀

3 Crossing-rigid weak graph distances

3.1 Definitions and properties

As seen in the previous results, the directed weak graph distance is NP-hard to decide if we do not restrict the crossings of G_2. Although the search space can be shown to have subexponential size in various cases, it is currently unknown whether there exists an FPT decision algorithm parameterized in the number of crossings of G_2 for the general case.

Hence, we propose modifying the distance measure by requiring crossings to be mapped onto crossings. This allows us to design FPT algorithms, and also captures the intuition that if two immersed graphs describe similar networks, they should have similar crossings.

In our notation, a crossing is a tuple (e, p) of an edge e and a point $p \in \mathbb{R}^2$ in which the immersion of e crosses the immersion of (at least) one other edge. First, we formalize the notion of mapping crossings onto crossings:

▶ **Definition 3.1.** Let $s\colon G_1 \to G_2$ be a graph mapping. We say that s is *loosely crossing-rigid* if s maps each edge $e = \{u, v\}$ that has crossings in points $p_1, \ldots, p_n$ to a sequence of (possibly constant) paths $P_0, \ldots, P_n$ in G_2 such that

1. all initial and terminal points of the P_i are $s(u), s(v)$ or crossings of G_2,
2. the P_i visit no crossings except for their initial and terminal points and
3. the concatenated path $P_0 P_1 \ldots P_n$ is defined and is a simple path from $s(u)$ to $s(v)$

s is *crossing-rigid* if there exists such a sequence such that P_0 and P_n are not constant. s is *strictly crossing-rigid* if there exists such a sequence such that none of the P_i are constant.

In other words: For edges without crossings, nothing is changed. For an edge e that has $n \geq 1$ crossings, the image of e

- under a loosely crossing-rigid graph mapping has at most n crossings,
- under a crossing-rigid graph mapping has at least one and at most n crossings,
- under a strictly crossing-rigid graph mapping has exactly n crossings.

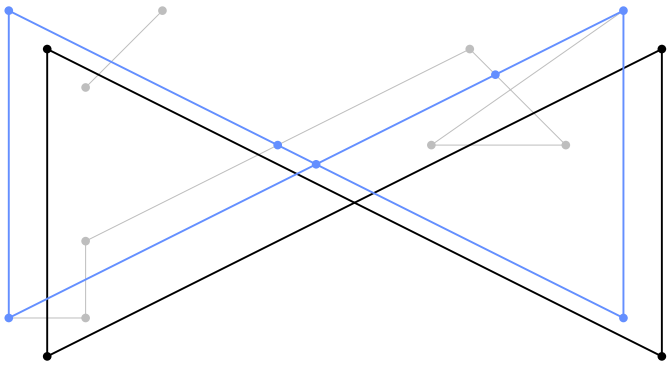


Figure 3 Graphs G_1 (black) and G_2 (blue and grey). The blue part is a valid image for G_1 under a crossing-rigid graph mapping. The diagonal grey path may not be an image of an edge of G_1 since it passes 3 crossings (the upper right crossing is passed through two edges).

As such, crossing-rigidity is of purely combinatorial rather than geometric nature.

▶ **Definition 3.2.** For immersed graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, the *directed crossing-rigid weak graph distance* is defined as

$$\vec{\delta}_{crwG}(G_1, G_2) = \inf_{s\colon G_1 \to G_2} \max_{e \in E_1} \delta_{wF}(e, s(e))$$

where s ranges over all crossing-rigid graph mappings $s\colon G_1 \to G_2$ and e and $s(e)$ are interpreted as the corresponding polygonal curves.

The *directed loosely crossing-rigid weak graph distance* $\vec{\delta}^l_{crwG}(G_1, G_2)$ and the *directed strictly crossing-rigid weak graph distance* $\vec{\delta}^s_{crwG}(G_1, G_2)$ are defined analogous.

Respective undirected versions $\delta_{crwG}, \delta^l_{crwG}, \delta^s_{crwG}$ can be defined as in Def. 1.3.

Note that when G_1 has crossings and G_2 is plane, there exist no (strictly) crossing-rigid graph mappings and as such, $\vec{\delta}_{crwG}(G_1, G_2) = \vec{\delta}^s_{crwG}(G_1, G_2) = \infty$. Moreover, since placements are no longer compact, we might have $\vec{\delta}_{crwG}(G_1, G_2) = \varepsilon$ (analogous for the loosely or strictly crossing-rigid versions) even if no ε-placement exists.

▶ **Observation 3.3.** *Without further restrictions, the above distance measures have several counterintuitive properties:*

1. *s may map crossing edges e_1 and e_2 such that $s(e_1)$ does not cross $s(e_2)$. Even if $s(e_1)$ and $s(e_2)$ cross, they need not cross in the corresponding crossings from Def. 3.1.*
2. *The implicit mapping $(e, p) \mapsto (e', p')$ of crossings of G_1 onto crossings of G_2 need not be one-to-one even for the strict distance.*
3. *A crossing p of an edge e may be mapped onto an edge crossing p' such that $d(p, p') > \varepsilon$.*

See Figs. 3 and 4 for illustrations.

However, our FPT approach is to assign a crossing of G_2 to each crossing of G_1 and for a fixed assignment, the above properties may be decided efficiently. Thus, we may require the mappings to not have any subset of the above properties without impacting tractability.

▶ **Lemma 3.4.** *Let G_1, G_2 be graphs immersed in $\mathbb{R}^2$. It holds that*

$$\vec{\delta}_{wG}(G_1, G_2) \leq \vec{\delta}^l_{crwG}(G_1, G_2) \leq \vec{\delta}_{crwG}(G_1, G_2) \leq \vec{\delta}^s_{crwG}(G_1, G_2). \tag{1}$$

If G_1 and G_2 are plane, $\vec{\delta}_{wG}(G_1, G_2) = \vec{\delta}^l_{crwG}(G_1, G_2) = \vec{\delta}_{crwG}(G_1, G_2) = \vec{\delta}^s_{crwG}(G_1, G_2)$.

▶ Remark. None of the ratios between the terms of eq. 1 are bounded.

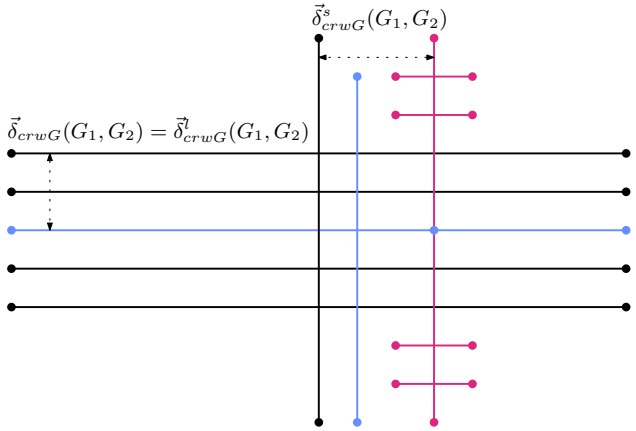


Figure 4 Graphs G_1 (black) and G_2 (blue and red). For the (loosely) crossing-rigid distance, G_1 can be mapped onto the blue edges. For the strict distance, the vertical edge needs to be mapped onto the red vertical edge, resulting in a larger distance.

3.2 Decision algorithm

Note that we have to amend the definitions in Def. 1.4: Placements of a vertex v are now connected components of $G_2 \backslash C(G_2)$ within $B_\varepsilon(v)$, where $C(G_2)$ is the set of points in $\mathbb{R}^2$ in which G_2 has crossings. Edge placements now have to adhere to Def. 3.1. We propose the following algorithmic approach for the crossing-rigid distances:

Algorithm 2 Decision algorithm for the existence of a crossing-rigid weak ε-placement

1: If not given, compute where the immersion of G_1 resp. G_2 has crossings.
2: Compute vertex placements.
3: **for each** valid assignment of crossings of G_2 to crossings of G_1
4: Compute reachability information for vertex placements under current assignment.
5: Prune invalid placements.
6: **if** there exists a placement for the whole graph G_1 **then return** true.
7: **return** false.

The definition of "validity" of an assignment in step 3 depends on whether we consider the loosely crossing-rigid, crossing-rigid or strictly crossing-rigid distance. Less assignments will be valid if we demand that the graph mapping does not have some of the properties from Obs. 3.3. There are at most $(k_2 + 1)^{k_1}$ such assignments. Restricting the assignments in the sense of Def. 3.1 or Obs. 3.3 takes polynomial time per assignment.

Steps 1, 2, 4 and 5 can be performed in polynomial time. Regarding step 6, we conjecture:

▶ **Conjecture 3.5.** *For the crossing-rigid and strictly crossing-rigid weak graph distance, step 6 can be performed in polynomial time if G_1 is spike-free. For the loosely crossing-rigid weak graph distance, step 6 can be performed in polynomial time if G_1 satisfies some slightly stronger regularity condition.*

Essentially, the idea is to have a similar situation to the plane case from [1, Lemma 7] for edges without crossings. For edges with crossings, after assigning crossings, all consistent weakly valid placements of the incident vertices are mutually reachable.

Verifying the above conjecture and developing a computation algorithm are natural next steps. Additionally, as mentioned above, it is currently unknown whether the weak graph

distance admits an FPT algorithm when parameterized in the respective number of crossings, which is also an interesting question. Lastly, more experimental work would give insight into the use of our distance measures for comparing realistic networks.

References

1 Hugo A. Akitaya, Maike Buchin, Bernhard Kilgus, Stef Sijben, and Carola Wenk. Distance measures for embedded graphs. *Computational Geometry*, 95:101743, 2021.

2 Maike Buchin, Erin Chambers, Pan Fang, Brittany Terese Fasy, Ellen Gasparovic, Elizabeth Munch, and Carola Wenk. Distances between immersed graphs: Metric properties. *La Matematica*, pages 1–26, 2023.

3 Marek Chrobak and Thomas H. Payne. A linear-time algorithm for drawing a planar graph on a grid. *Information Processing Letters*, 54(4):241–246, 1995.

4 David P. Dailey. Uniqueness of colorability and colorability of planar 4-regular graphs are NP-complete. *Discrete Mathematics*, 30(3):289–293, 1980.

5 Jiri Matousek. *Lectures on discrete geometry*, volume 212. Springer, 2013.

6 Larry Stockmeyer. Planar 3-colorability is polynomial complete. *ACM Sigact News*, 5(3):19–25, 1973.

The k-Transmitter Watchman Route Problem is NP-Hard Even in Histograms and Star-Shaped Polygons*

Anna Brötzner[1], Bengt J. Nilsson[1], and Christiane Schmidt[2]

1 **Department of Computer Science and Media Technology, Malmö University, Sweden**
{anna.brotzner,bengt.nilsson.TS}@mau.se

2 **Department of Science and Technology, Linköping University, Sweden**
christiane.schmidt@liu.se

Abstract

A k-transmitter g in a polygon P, with n vertices, k-sees a point $p \in P$ if the line segment $\overline{gp}$ intersects P's boundary at most k times. In the k-Transmitter Watchman Route Problem we aim to minimize the length of a k-transmitter watchman route along which every point in the polygon—or a discrete set of points in the interior of the polygon—is k-seen. We show that the k-Transmitter Watchman Route Problem for a discrete set of points is NP-hard for histograms, uni-monotone polygons, and star-shaped polygons given a fixed starting point. For histograms and uni-monotone polygons it is also NP-hard without a fixed starting point. Moreover, none of these versions can be approximated to within a factor $c \cdot \log n$, for any constant $c > 0$.

1 Introduction

k-transmitters were introduced as a generalization of the classical illumination problems [1]. Guards are replaced by *modems*, also called *k-transmitters*, who send a signal that can pass through up to k walls.

In the watchman route problem (WRP), introduced by Chin and Ntafos [3], instead of placing several stationary guards, we are given one mobile watchman who moves within the given environment, and want to compute a shortest watchman route such that all points in the environment are seen from some point on the route. This problem has shown to be solvable in polynomial time both with [3, 4, 10] and without [2, 9] a fixed starting point. Several different variations of the WRP have been considered, for example for polygons with and without holes [7], for lines and line segments [5], and using a k-transmitter as a watchman [8]. Nilsson and Schmidt proved NP-hardness for the k-transmitter watchman route problem for a discrete set of points within a simple polygon and provided a polylogarithmic approximation algorithm for that case [8]. In this paper, we show that it is also NP-hard for certain classes of simple polygons, namely histograms, uni-monotone, and star-shaped polygons, given a fixed starting point. Extending this, we also show NP-hardness without a fixed starting point for histograms and uni-monotone polygons.

* A. B., B. J. N. and C. S. are supported by grant 2021-03810 (Illuminate: provably good algorithms for guarding problems) and B. J. N. and C. S. are supported by grant 2018-04001 (New paradigms for autonomous unmanned air traffic management) from the Swedish Research Council (Vetenskapsrådet).

40th European Workshop on Computational Geometry, Ioannina, Greece, March 13–15, 2024.
This is an extended abstract of a presentation given at EuroCG'24. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

2 Notation and Preliminaries

Let P be a simple polygon having n vertices. P is called a *histogram* if it is rectilinear and has one horizontal edge (a base) that is equally long as the sum of lengths of all other horizontal edges. A *k-transmitter* is a modem that can see through up to k walls. Our goal is to see a set of points in the interior of P with one mobile k-transmitters. We say that point p *k-sees* point q (and p and q are *k-visible* to each other) if the segment between p and q intersects the polygon boundary at most k times. For a point $p \in P$ the k-visibility region of p is the set of all points in P that p k-sees.

k-Transmitter Watchman Route Problem with Starting Point (k-TrWRP(S, P, s)). Given a polygon P with n vertices, an integer $k \geq 2$, a starting point s in P, and a set of interior points S in P, find a minimum length watchman route that starts at s and lies within P such that all points in S are k-visible from the route.

k-Transmitter Watchman Route Problem (k-TrWRP(S, P)). Given a polygon P with n vertices, an integer $k \geq 2$ and a set of interior points S in P, find a minimum length watchman route that lies within P such that all points in S are k-visible from the route.

Since both the watchman and the points that need to be seen lie in the interior of P, it is sufficient to consider only even values for k. We therefore assume that k is even.

In Section 3, we prove that k-TrWRP(S, P, s) is NP-hard for histograms by providing a reduction from Set Cover. We then extend this reduction to uni-monotone and star-shaped polygons.

Set Cover Problem. Given a universe $\mathcal{U}$ and a family $\mathcal{R}$ of subsets of $\mathcal{U}$, find a subfamily $\mathcal{C} \subseteq \mathcal{R}$ that contains all elements of $\mathcal{U}$ and is of minimum cardinality.

Feige [6] showed that Set Cover cannot be approximated to within a factor $(1-o(1)) \ln |\mathcal{U}|$ in polynomial time. Thus, there exists no polynomial time algorithm that approximates k-TrWRP(S, P, s) for histograms, uni-monotone polygons or star-shaped polygons, and k-TrWRP(S, P) for histograms and uni-monotone polygons, within approximation ratio of $c \log |S|$ for any $c > 0$. Nevertheless, we can apply the approximation algorithm presented in [8] to a histogram, uni-monotone or star-shaped polygon P with n vertices. This algorithm has an approximation factor of $O(\log^2(|S| \cdot n) \log\log(|S| \cdot n) \log(|S| + 1))$.

3 NP-Hardness for Histograms: Reduction from Set Cover

▶ **Theorem 3.1.** *For any $k \geq 2$, k-TrWRP(S, P, s) is NP-hard for histograms and cannot be approximated within a factor $c \log n$, for any $c > 0$.*

Proof. We provide a reduction from Set Cover. Let $(\mathcal{U}, \mathcal{R})$ be an instance of the Set Cover Problem. We construct a bipartite graph G with $V(G) = \mathcal{U} \cup \mathcal{R}$ and $E(G) = \{(u, R) \mid u \in \mathcal{U}, R \in \mathcal{R}, u \in R\}$.

Given an integer k, we construct a histogram P such that k-visibility between points encodes the edges of the graph G: it contains a set of points $S = \mathcal{U}$, where point $u \in \mathcal{U}$ is k-visible from a region $R \in \mathcal{R}$ if and only if $(u, R) \in E(G)$.

The points $u_1, \ldots, u_{|\mathcal{U}|} \in \mathcal{U}$ are placed in the top of "towers", so-called *defensive towers*, that lie in the right part of P. Similarly, the regions $R_1, \ldots, R_{|\mathcal{R}|} \in \mathcal{R}$, in the following also

called *observation regions*, are placed from left to right in this order, in *watch towers* that lie close to the starting point s and to the left of the defensive towers. To ensure that the points in $\mathcal{U}$ are far to reach from s we place a long corridor between the two sets of towers. We call the part to the left of this corridor the *observational part of* P, and the part to the right of the corridor the *defensive part of* P. See Figure 1 for an example.

The horizontal boundary edges between the towers are all collinear. The longest line segment in P that contains all of these edges defines the so-called *supporting line*. We place the starting point s on the left end of the supporting line.

The watch towers are high enough such that walking up each of them is expensive. Moreover, we need to ensure that every point $u \in \mathcal{U}$ is k-visible from the observation regions whose corresponding subsets contain u. Therefore, we place the points and regions on different heights, such that the watch towers have decreasing height from left to right, and the defensive towers have increasing height. This means that R_1 will be in the highest tower in the observational part of P, and $u_{|\mathcal{U}|}$ will be in the highest tower in the defensive part of P. Let h be the height of the watch tower containing R_1, and let the height of the watch towers differ by $\varepsilon_1 \ll h$ only. Furthermore, let the width of the watch towers be $w < \varepsilon_2$, and denote the difference of the x-coordinates of R_1 an $R_{|\mathcal{R}|}$ by ℓ. We choose the horizontal distance of the watch towers such that ℓ is significantly smaller than h, $\ell \ll h$. The height of the defensive towers is adapted to the height of the watch towers to ensure k-visibility between observation regions and points in $\mathcal{U}$.

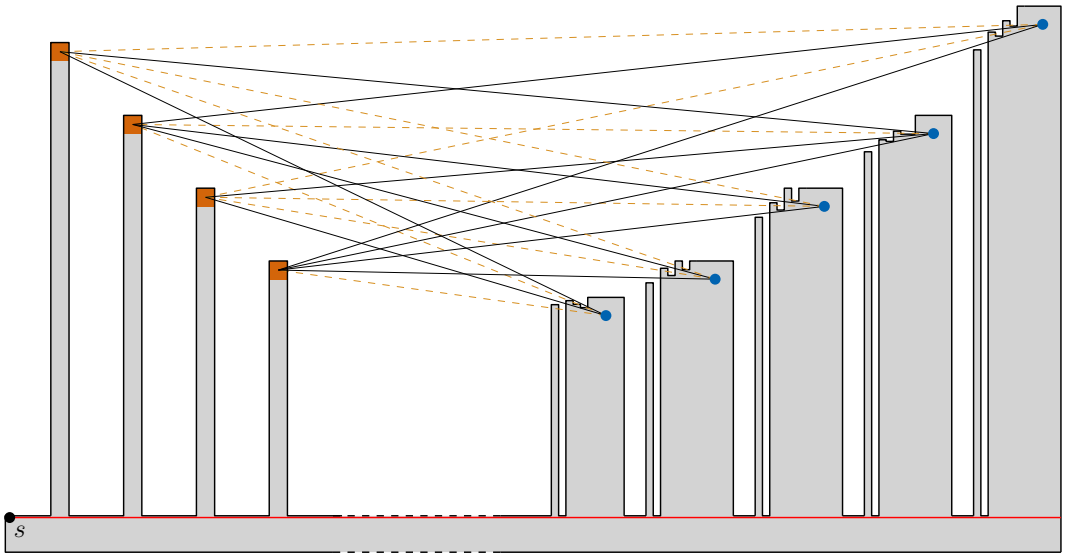


Figure 1 Construction of a histogram for $k = 2$. The regions $R_1, \ldots, R_{|\mathcal{R}|}$ are marked in orange. The points in $\mathcal{U}$, which lie in the defensive part of P, are colored blue. The continuous black line segments represent the edges of the graph G: the lines of k-visibility between the observation points and the points $\mathcal{U}$. If $(u, R) \notin E(G)$, then u is not k-visible from R. This is indicated by a dashed orange line segment. The red horizontal line segment in the bottom of P is the supporting line.

Let the length of corridor between the watch towers and the towers containing the points in $\mathcal{U}$ be $L \gg (h \cdot |\mathcal{R}| \cdot |\mathcal{U}| \cdot k + \ell) \log n$. Thus, walking through the corridor to reach the right part of P (and hence see the defensive points from their proximity) is much more expensive than climbing up every watch tower.

Since a point $u \in \mathcal{U}$ shall only be k-seen from an observation region R if $(u, R) \in E(G)$, and from its proximity, we need to block the k-visibility from every other point that lies

within a feasible walking distance from s. To do this, we add battlements to the defensive towers (see Figure 2). Consider the visibility cone from a point u to an observation region R, meaning the cone that arises when connecting u with every point in R by a straight line. For the sake of simplicity, let each observation region be a square of height w, and assume the visibility cone to be just a straight line. Then, between every two such lines emanating from u, we add $k/2$ battlements. We also add $k/2$ battlements in front of the tower, which stretch from the supporting line up to the first line of visibility, and—in case $(u, R_1) \notin E(G)$—we add $k/2$ battlements after the uppermost line of visibility, which intersect the line of visibility between u and R_1. Every battlement is sufficiently close to the next line of visibility, but will not touch it. For an example of this construction, see Figure 2.

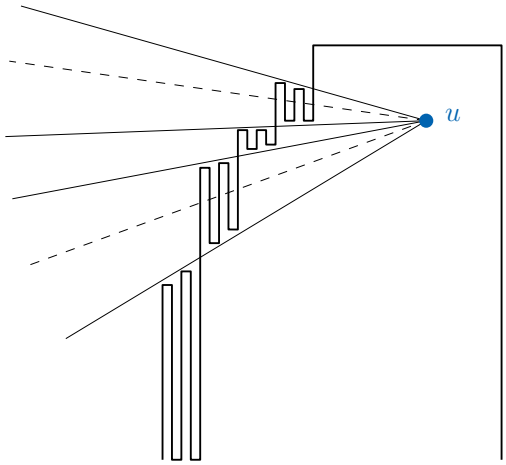


Figure 2 Adding battlements in front of a tower to block k-visibility. The lines suggest the line segment between u and the points corresponding to subsets in $\mathcal{R}$, where the continuous lines indicate $(u, R) \in E(G)$, and dashed lines indicate $(u, R) \notin E(G)$, for a region R representing a subset.

▶ Observation 3.1.1. The only points in the left part of the histogram P that k-see a point $u \in \mathcal{U}$ are exactly those that lie in the regions whose corresponding subsets contain u (that is, those $R \in \mathcal{R} : (u, R) \in E(G)$).

Since the watch towers are rather high, we obtain the shortest watchman route by climbing only few of the watch towers. More precisely, the watchman will visit exactly those regions in $\mathcal{R}$ that correspond to a minimum set cover of $\mathcal{U}$.

It remains to show that we can compute the histogram in polynomial time. To be more precise, this means that the histogram needs to have integer coordinates. The ratio between the distance of an observational point to its closest battlement and the total width of the polygon is at most $1/O\big(|\mathcal{R}| + k \cdot |\mathcal{U}| \cdot |\mathcal{R}|\big) = 1/O\big(k \cdot |\mathcal{U}| \cdot |\mathcal{R}|\big)$. We can construct the horizontal battlement edges using integer coordinates if we have $O(k \cdot |\mathcal{R}|)$ integral y-levels for each battlement. To achieve this, it is sufficient for the polygon to fit in a square of size $O\big(k^2 \cdot |\mathcal{U}| \cdot |\mathcal{R}|^2\big)$, hence we can restrict the construction to use coordinate values from 0 to $O\big(k^2 \cdot |\mathcal{U}| \cdot |\mathcal{R}|^2\big)$.

From the construction shown above, we conclude that our reduction is gap preserving, and thus, assuming $\mathsf{P} \neq \mathsf{NP}$, the problem cannot be polynomially approximated within a factor of $c \log n$, for any constant $c > 0$, where n is the total number of vertices. To see this, note that we can assume that $|\mathcal{R}| = |\mathcal{U}|^\alpha$, for some sufficiently large positive constant α, see also [6]. The number of vertices is bounded by $4 \cdot |\mathcal{R}| + 4k \cdot |\mathcal{U}| + 4 < n \leq 4 \cdot |\mathcal{R}| + 4k \cdot |\mathcal{R}| \cdot |\mathcal{U}| + 4$. Hence, $\Omega(n^{1/(2\alpha+1)}) \ni |\mathcal{U}| \in O(n^{1/\alpha})$ and since $|S| = |\mathcal{U}|$, the bound follows. ◀

We can substitute the starting point by the k-visibility region of a point u' and set $S = \mathcal{U} \cup \{u'\}$: We add $k/2$ towers of height $\gg h$ to the left of all watch towers, and one more tower of almost the same height to the left of these, in which we locate u', see Figure 3. All points in the k-visibility region of u' have a smaller x-coordinate than the leftmost watch tower. We then need to visit this region to k-see u'. Because of the length of the corridor, we again cannot afford to visit the defensive part of P. Thus, we have:

▶ **Corollary 3.2.** *For any $k \geq 2$, k-TrWRP(S, P) is* NP*-hard for histograms and cannot be approximated within a factor $c \log n$, for any $c > 0$.*

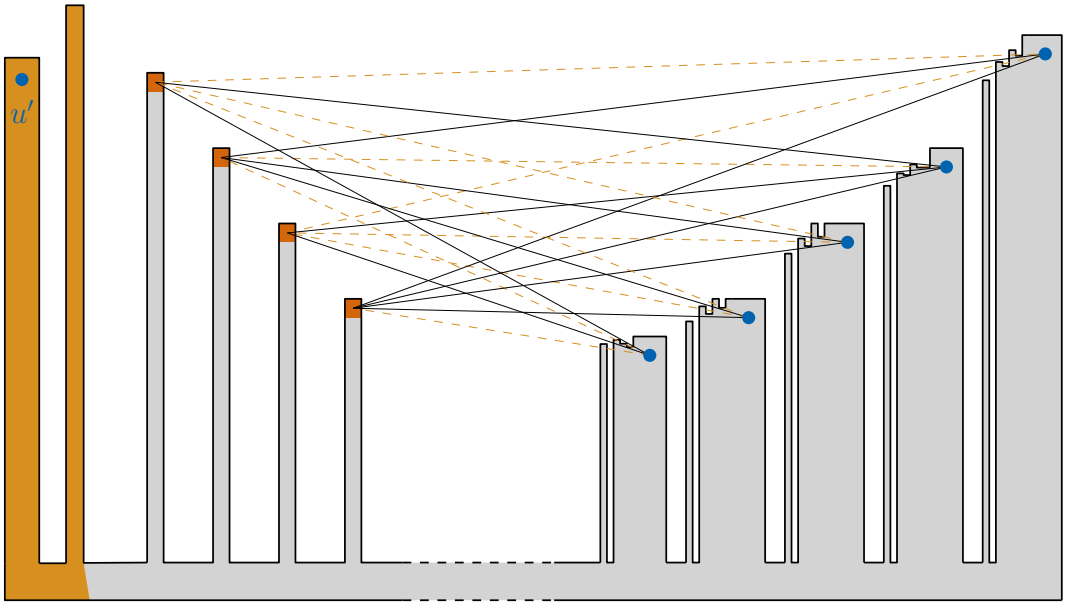


Figure 3 A modification of the histogram for $k = 2$: We add two more towers to the left of the watch towers and place u' in the leftmost tower. The 2-visibility region of u' is colored light orange.

A polygon P is called *x-monotone* if any vertical line intersects the boundary of P in at most two connected components. The boundary of an x-monotone polygon can be decomposed into two chains, splitting it at the (lowest) leftmost point and the (lowest) rightmost point of the boundary. An x-monotone polygon is called *uni-monotone* if either the upper or the lower chain is a horizontal segment. Clearly, a histogram is uni-monotone, hence, Theorem 3.1 and Corollary 3.2 yield:

▶ **Corollary 3.3.** *For any $k \geq 2$, k-TrWRP(S, P, s) and k-TrWRP(S, P) are* NP*-hard for uni-monotone polygons and cannot be approximated within a factor $c \log n$, for any $c > 0$.*

4 NP-Hardness for Star-Shaped Polygons

A polygon P is star-shaped if it contains a region (possibly a single point), called the *kernel*, from which every point in P is 0-seen. Given the histogram from the proof of Theorem 3.1, we can modify it into a star-shaped polygon for which the k-visibility properties again encode the bipartite graph G.

First, we stretch the towers such that they all "point" towards a common viewpoint—the kernel of P, which lies very far below the supporting line—preserving the k-visibility

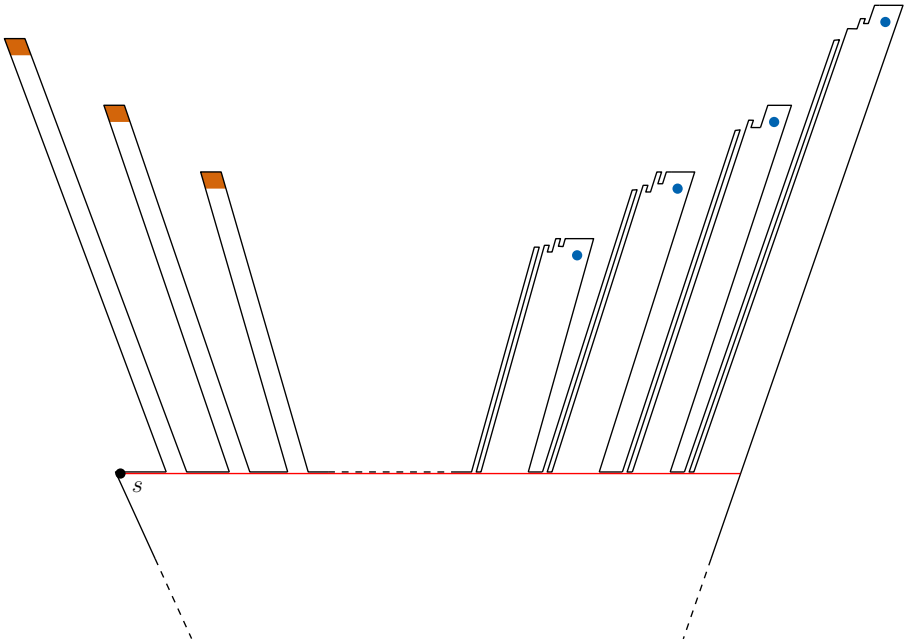


Figure 4 Sketch of a star-shaped polygon that evolves from slanting the histogram constructed previously. Note that the kernel is far below the supporting line.

properties. Moreover, we replace the base edge by two almost vertical edges that are slightly tilted towards the kernel and place the starting point s onto the left end of the supporting line. See Figure 4 for a sketch of this transformation.

To guarantee that the shortest watchman route will not be the direct path from s to the kernel of P, we ensure that the kernel is way too far away from s by choosing the angles along which the base edges and the towers are tilted accordingly. The construction can be made using integer coordinates inside a bounding box of polynomial range, similarly as in the proof of Theorem 3.1, yielding:

▶ **Theorem 4.1.** *For any $k \geq 2$, k-TrWRP(S, P, s) is* NP*-hard for star-shaped polygons and cannot be approximated within a factor $c \log n$, for any $c > 0$.*

Without a fixed starting point, k-TrWRP(S, P) clearly is easy to solve as then the shortest watchman route will be a route of length 0 somewhere in the kernel of P.

5 Conclusion

We establish NP-hardness of k-TrWRP(S, P, s) for histograms, uni-monotone polygons, and star-shaped polygons, as well as NP-hardness of k-TrWRP(S, P) for histogram and uni-monotone polygons. This is rather surprisingly, since these polygon classes seem to be fairly simple. The hardness reduction from Set Cover moreover yields inapproximability within a logarithmic factor in polynomial time. It would be of interest whether this result can be adapted to more polygon classes, like x-y-monotone polygons for example.

Acknowledgments. We want to thank Valentin Polishchuk for all the constructive discussions and for taking us to a very enjoyable ice cream spot! Another thank you goes to Daniel Perz, as well as the anonymous reviewers, who helped us to improve the results.

References

1 Oswin Aichholzer, Ruy Fabila-Monroy, David Flores-Peñaloza, Thomas Hackl, Clemens Huemer, Jorge Urrutia, and Birgit Vogtenhuber. Modem illumination of monotone polygons. *Computational Geometry: Theory and Applications*, 68:101–118, 2018. doi:10.1016/j.comgeo.2017.05.010.

2 Svante Carlsson, Håkan Jonsson, and Bengt J. Nilsson. Finding the shortest watchman route in a simple polygon. In Kam-Wing Ng, Prabhakar Raghavan, N. V. Balasubramanian, and Francis Y. L. Chin, editors, *Algorithms and Computation, 4th International Symposium, ISAAC '93, Hong Kong, December 15-17, 1993, Proceedings*, volume 762 of *Lecture Notes in Computer Science*, pages 58–67. Springer, 1993. doi:10.1007/3-540-57568-5_235.

3 Wei-pang Chin and Simeon Ntafos. Optimum Watchman Routes. In *Proceedings of the Second Annual Symposium on Computational Geometry*, SCG '86, page 24–33, New York, NY, USA, 1986. Association for Computing Machinery. doi:10.1145/10515.10518.

4 Moshe Dror, Alon Efrat, Anna Lubiw, and Joseph S. B. Mitchell. Touring a sequence of polygons. In *Symposium on the Theory of Computing*, 2003. URL: https://api.semanticscholar.org/CorpusID:951535.

5 Adrian Dumitrescu, Joseph S.B. Mitchell, and Paweł Żyliński. Watchman routes for lines and line segments. *Computational Geometry*, 47(4):527–538, 2014. doi:10.1016/j.comgeo.2013.11.008.

6 Uriel Feige. A threshold of ln n for approximating set cover. *J. ACM*, 45(4):634–652, jul 1998. doi:10.1145/285055.285059.

7 Joseph S. B. Mitchell. *Approximating Watchman Routes*, pages 844–855. 2013. doi:10.1137/1.9781611973105.60.

8 Bengt J. Nilsson and Christiane Schmidt. k-Transmitter Watchman Routes. In Chun-Cheng Lin, Bertrand M. T. Lin, and Giuseppe Liotta, editors, *WALCOM: Algorithms and Computation*, pages 202–213, Cham, 2023. Springer Nature Switzerland.

9 Xuehou Tan. Fast computation of shortest watchman routes in simple polygons. *Information Processing Letters*, 77(1):27–33, 2001. doi:10.1016/S0020-0190(00)00146-0.

10 Xuehou Tan, Tomio Hirata, and Yasuyoshi Inagaki. Corrigendum to "an incremental algorithm for constructing shortest watchman routes". *Int. J. Comput. Geometry Appl.*, 9:319–323, 06 1999. doi:10.1142/S0218195999000212.

Deltahedral Domes over Equiangular Polygons*

MIT CompGeom Group[1], Hugo A. Akitaya[2], Erik D. Demaine[3], Adam Hesterberg[4], Anna Lubiw[5], Jayson Lynch[6], Joseph O'Rourke[7], Frederick Stock[8], and Josef Tkadlec[9]

1 **Artificial 1st author to highlight that coauthors worked as an equal group.**
2 **U. Mass. Lowell, hugo_akitaya@uml.edu**
3 **MIT, edemaine@mit.edu**
4 **Harvard U., achesterberg@gmail.com**
5 **U. Waterloo, alubiw@uwaterloo.ca**
6 **MIT, jaysonl@mit.edu**
7 **Smith College, jorourke@smith.edu.**
8 **U. Mass. Lowell, fbs9594@rit.edu**
9 **Charles U., josef.tkadlec@iuuk.mff.cuni.cz**

Abstract

A *polyiamond* is a polygon composed of unit equilateral triangles, and a *generalized deltahedron* is a convex polyhedron whose every face is a convex polyiamond. We study a variant where one face may be an exception. For a convex polygon P, if there is a convex polyhedron that has P as one face and all the other faces are convex polyiamonds, then we say that P *can be domed*. Our main result is a complete characterization of which equiangular n-gons can be domed: only if $n \in \{3, 4, 5, 6, 8, 10, 12\}$, and only with some conditions on the integer edge lengths.

1 Introduction

In the study of what can be built with equilateral triangles, the most well-known result is that there are exactly eight convex ***deltahedra***—polyhedra where every face is an equilateral triangle—with $n = 4, 5, 6, 7, 8, 9, 10, 12$ vertices. See references in [3] or Wikipedia.[1] What if coplanar triangles are allowed? In the plane, the polygons built of equilateral triangles are the ***polyiamonds***. Convex polyiamonds have 3, 4, 5, or 6 vertices. The convex polyhedra with polyiamond faces are the "non-strictly convex deltahedra", or ***generalized deltahedra***, following the nomenclature of Bezdek [3]. See the above cited Wikipedia article for some examples. There are an infinite number of generalized deltahedra, though the number of combinatorial types is finite since they have at most 12 vertices. There is no published characterization, though a forthcoming one is mentioned in [3].

Our goal (only partially achieved) is to characterize when a convex polygon can be "domed" with a convex surface composed of equilateral triangles. For a convex polygon P, if there is a convex polyhedron that has P as one face and all the other faces are convex polyiamonds, then we say that P ***can be deltahedrally domed***, or just ***domed*** for short. Here the ***deltahedral dome*** (***dome*** for short), denoted by $\mathcal{D}$, is the part of the polyhedron excluding face P, and P is called the ***base*** of the dome. Note that P itself may or may not be a polyiamond.

We assume that all the equilateral triangles have unit edge length, so P must be an integer polygon (with integer side lengths). Here is a simple example:

* J.T. was supported by the Center for Foundations of Modern Computer Science (Charles University project UNCE/SCI/004) and by project PRIMUS/24/SCI/012 from Charles University.

[1] https://en.wikipedia.org/wiki/Deltahedron

40th European Workshop on Computational Geometry, Ioannina, Greece, March 13–15, 2024.
This is an extended abstract of a presentation given at EuroCG'24. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

▶ **Lemma 1.1.** *Every integer rectangle can be domed.*[2]

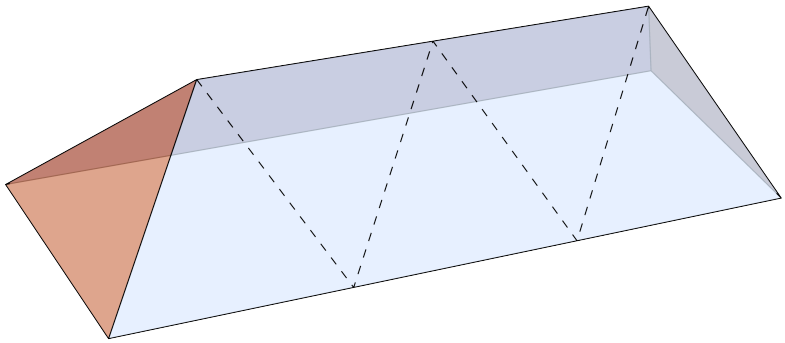


Figure 1 "Roof" dome over a 3×1 rectangle.

1.1 Main Theorem

▶ **Theorem 1.2.** *(a) The only equiangular polygons that can be domed have n vertices, where $n \in \{3, 4, 5, 6, 8, 10, 12\}$; for each such n, any regular integer n-gon can be domed. (b) Moreover, for $n = 3, 4, 5, 6$ every equiangular integer polygon can be domed, and for $n = 8, 10, 12$, an equiangular integer n-gon can be domed iff the odd edge lengths are equal and the even edge lengths are those of an equiangular $\frac{n}{2}$-gon.*

For small n, edge-length conditions for an equiangular integer polygon are known [2]: for $n = 4$ these are rectangles; for $n = 5$ there is only the regular pentagon; and for $n = 6$ the edge lengths must be integers a, b, c, a', b', c' with $a - a' = b' - b = c - c'$ (a 6-sided polyiamond).

Part (a) of Theorem 1.2 is proved in Sections 2 and 3. We have established several results beyond the main theorem (for example, that all polyiamonds, equiangular or not, are domeable, and that there is no domeable polygon with 25 or more vertices)—see Section 4.

1.2 Glazyrin and Pak

The source of our work derives from a paper by Glazyrin and Pak: "Domes over Curves" [4], which answers a question posed by Richard Kenyon in 2005.[3] In [4], a "curve" P is a closed polygonal chain in $\mathbb{R}^3$, and a dome is a PL-surface composed of unit equilateral triangles whose boundary is ∂P. Then they say that P can be *spanned*. We note the following two differences with our definitions:

(1) Our P is a 2D convex polygon; theirs is a 3D possibly self-intersecting polygonal chain.
(2) Our dome $\mathcal{D}$ is embedded (non-self-intersecting) and convex. Their PL-surface is (in general) nonconvex, immersed, and self-intersecting.

Under their conditions, they show that certain nonplanar rhombi cannot be spanned, which answers Kenyon's question in the negative.[4] More interesting for our purposes, they prove

[2] Due to space limitations, several proofs appear only in the full version of this paper.
[3] https://gauss.math.yale.edu/~rwk25/openprobs/.
[4] Recent work [1] extends the Glazyrin-Pak negative result to show that "generic" integer polygons cannot be spanned.

that every planar regular polygon can be spanned (their Theorem 1.4). In contrast, our Theorem 1.2 says that the regular 7-, 9-, and 11-gons cannot be domed, nor can any regular n-gon for $n > 12$. And here "regular" can be strengthened to "equiangular." Compared to their results, our conditions constrain the geometry and limit what can be domed.

2 Domed Regular Polygons

We prove one part of Theorem 1.2(a) by exhibiting domes over regular integer n-gons, for $n \in \{3, 4, 5, 6, 8, 10, 12\}$. We will use $\bar{P}_n$ to denote a regular integer n-gon.

- $3, 4, 5$: $\bar{P}_n$ for $n = 3, 4, 5$ can each be domed by a pyramid: Fig. 2.
- 6 : Hexagonal antiprism: Fig. 3(a).
- 8 : A slice through a gyroelongated square diprism: Fig. 3(b).
- 10 : A slice through an icosahedron: Fig. 3(c).
- 12 : A slice through a hexagonal antiprism: Fig. 3(d).

A few remarks. The pyramid pattern for $n = 3, 4, 5$ cannot be extended to $\bar{P}_6$, for that would result in a doubly-covered hexagon, not a dome by our definition. For $n = 8, 10, 12$, we show $\bar{P}_n$ as a slice of a convex polyhedron, with the dome the upper half of the surface. But we have established that not every doming of an equiangular polygon derives from a slice.

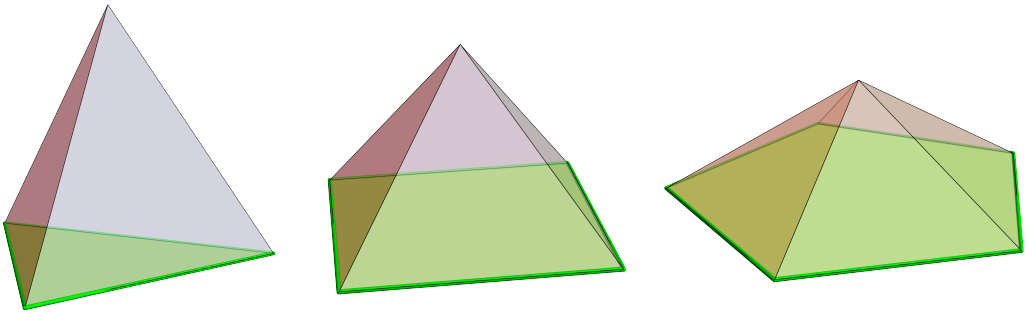


Figure 2 Pyramids over $\bar{P}_n$, $n = 3, 4, 5$.

Figures 2 and 3 show one way to dome each regular polygon $\bar{P}_n$, but there are other solutions. For example, $\bar{P}_5$ can be domed by a low slice through the icosahedron as shown in Fig. 4(a). And again, these figures illustrate regular polygons, special cases of equiangular polygons. To give a hint of the further possibilities, Fig. 4(b) shows an equiangular decagon P_{10} whose edge lengths alternate 1 and 3.

3 Proof of Theorem 1.2(a): Restrictions on n

In this section we complete the proof of the first half of Theorem 1.2: The only equiangular n-gons that can be domed have $n \in \{3, 4, 5, 6, 8, 10, 12\}$. For a dome over an equiangular n-gon, $n \geq 6$, we use the following steps:

(1) Each base vertex has three incident dome triangles.
(2) Curvature constraints imply that the number of (non-base) dome vertices is at most 6.

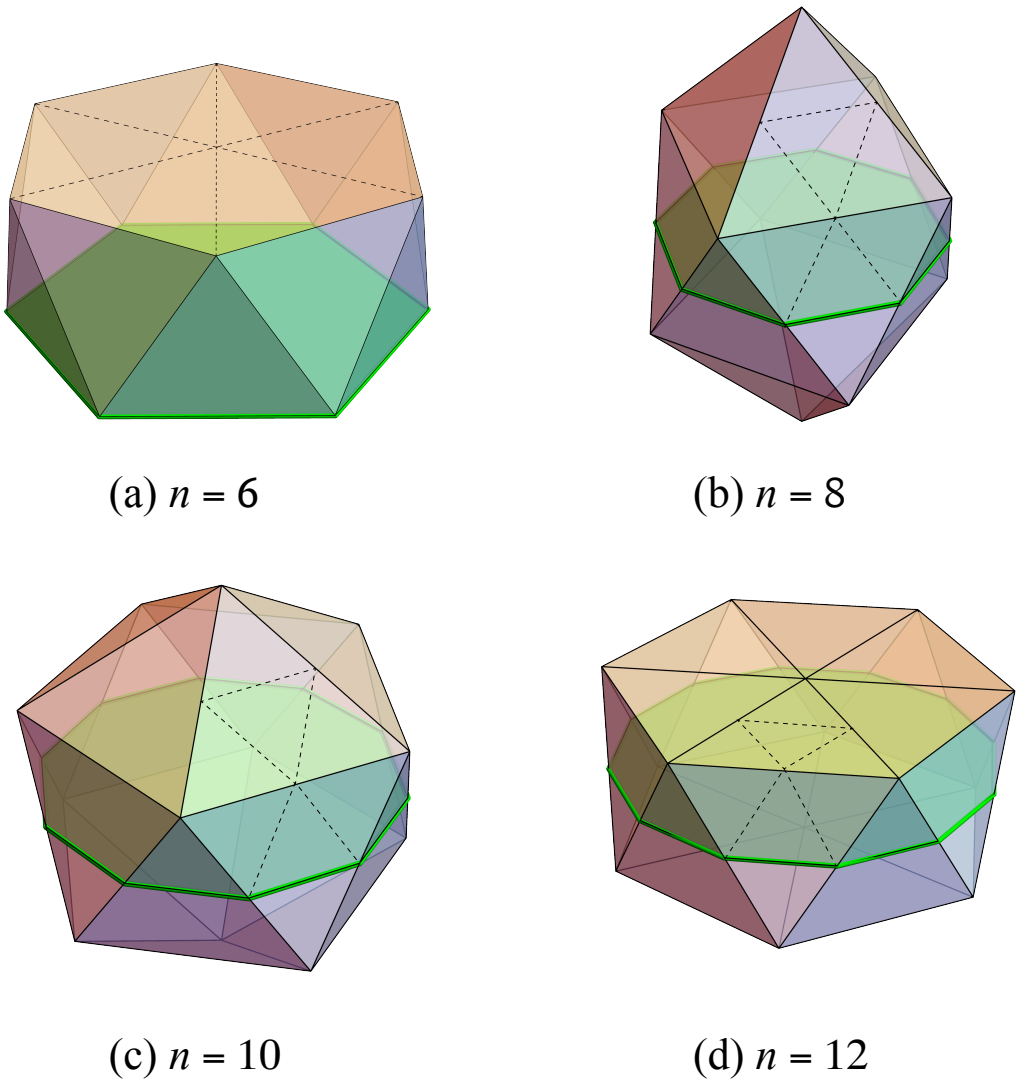
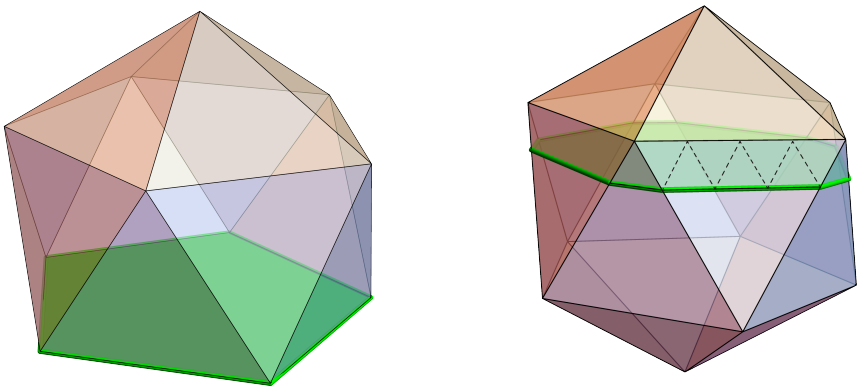


Figure 3 Examples of $\bar{P}_n$ domes for $n = 6, 8, 10, 12$.



(a) $n = 5$ (b) $n = 10$

Figure 4 (a) A different dome over $\bar{P}_5$. (b) Equiangular decagon with edge lengths alternating $1, 3$.

(3) Of the n dome faces incident to base edges, at least half tilt toward the outside of the base and have a "private" dome vertex. Furthermore, for n odd we strengthen this to *all* dome faces incident to base edges.
(4) Thus, since there are at most 6 dome vertices, $n \leq 12$, and for n odd, there are no solutions for $n \geq 6$.

Note that the base angle β of an equiangular n-gon is $\frac{n-2}{n}180°$ so if $n \geq 6$, then every base angle is $\geq 120°$. This weaker assumption on a domeable convex n-gon is enough for most of our argument.

▶ **Lemma 3.1.** *If a base vertex b_i has base angle $\beta_i \geq 120°$, then it is incident to three dome vertices.*

Proof. Base vertex b_i cannot be incident to just one or two triangles, otherwise the total face angle is $\leq 120°$, which is not enough to span β_i. Vertex b_i cannot be incident to four triangles, because $\beta_i + 240° \geq 360°$, and similarly for five (or more) triangles. ◀

From this we can analyze the base curvature:

▶ **Lemma 3.2.** *If every base vertex b_i is incident to three dome triangles, then the sum of the curvatures at the base vertices is 2π.*

Proof. Let β_i be the angle of P at vertex b_i. Then the curvature at b_i is $\omega_i = 2\pi - (\beta_i + \pi)$, where the final π term follows from the assumption that b_i is incident to three triangles. Recalling that $\sum_i \beta_i = \pi(n-2)$ for any simple polygon, we have:

$$\sum_i \omega_i \;=\; \sum_i (\pi - \beta_i) \;=\; n\pi - \sum_i \beta_i \;=\; n\pi - \pi(n-2) \;=\; 2\pi.$$

◀

▶ **Lemma 3.3.** *If $\mathcal{D}$ is a dome over a convex polygon P that has all angles $\geq 120°$, then $\mathcal{D}$ has at most 6 dome vertices.*

Proof. Let V_3, V_4, V_5 be the number of (non-base) dome vertices with $3, 4, 5$ incident triangles, respectively. By the Gauss-Bonnet theorem, the total curvature of a convex polyhedron is 4π. The curvature of a V_k vertex is $2\pi - k\frac{\pi}{3}$. By Lemmas 3.1 and 3.2 (this is where we use the assumption that all base angles are $\geq 120°$) the curvature at the base vertices of any dome over P is 2π. Thus, in units of π:

$$V_3 + \frac{2}{3}V_4 + \frac{1}{3}V_5 = 2\,.$$

Therefore the number of dome vertices of $\mathcal{D}$ is $V_3 + V_4 + V_5 \leq 3V_3 + 2V_4 + V_5 = 6$. ◀

3.1 Face Normals and Private Dome Vertices

It remains to show step (3): that, of the n dome faces incident to base edges, at least half of them [and for odd n, all of them] tilt toward the outside of the base and each have a "private" dome vertex. We say that a dome vertex v is ***private*** if there is a unique dome face incident to v and to a base edge.

Orient the dome with the base in the horizontal xy-plane. A dome triangle/face has an ***upward normal*** if its normal has a positive z-component, and a ***downward normal*** if its normal has a negative z-component. (This formalizes "tilting towards the outside").

▶ **Lemma 3.4** (±Normals). *Consider a base vertex b_i with base angle $\geq 120°$. Suppose the three dome triangles incident to b_i are t_1, t_2, t_3 where t_1 and t_3 are incident to the base edges at b_i (possibly t_2 is coplanar with t_1 or with t_3, but not both). Then t_2 has an upward normal and at least one of t_1, t_3 has a downward normal.*

Lemma 3.4 is proved in the full version of this paper using the Gauss map and spherical trigonometry.

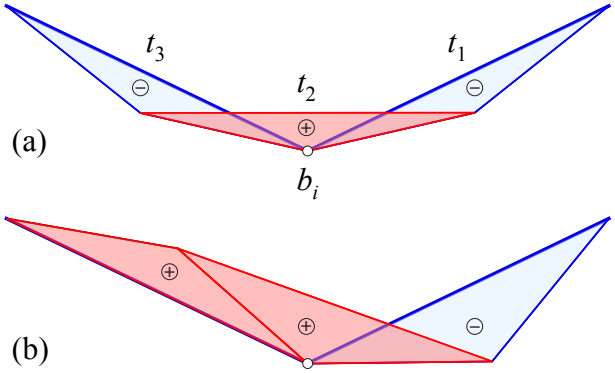


Figure 5 Overhead view of three triangles incident to base vertex b_i. Triangles with upward normals pink, downward normals blue. (a) Both t_1 and t_3 downward. (b) Only t_1 downward.

▶ **Observation 3.5.** *In Lemma 3.4, if t_1 has a downward normal, then it cannot be coplanar with t_2 (which has an upward normal), and thus the dome face containing t_1 has a face angle of $60°$ at the base vertex b_i.*

▶ **Lemma 3.6.** *If P is a domeable convex n-gon with all angles $\geq 120°$, then $n \leq 12$. Furthermore, there is no domeable equiangular n-gon for odd $n \geq 6$.*

Proof. For the second statement in the lemma, note that an equiangular n-gon, $n \geq 6$ has all angles $\geq 120°$. So consider an n-gon P with all angles $\geq 120°$, and suppose P has a dome $\mathcal{D}$. By Lemma 3.3, $\mathcal{D}$ has at most 6 dome vertices. We will prove that $n \leq 12$, and derive a contradiction for odd $n \geq 6$.

Consider the n faces of $\mathcal{D}$ incident to a base edge. Let d be the number of those faces with downward normals. By Observation 3.5, a downward pointing face f incident to base edge e has 60° face angles at the the endpoints of e, so it must include a dome vertex whose projection to the xy-plane lies in the equilateral xy-triangle on edge e. See Fig. 5 and Fig. 6. Such a dome vertex is unique to base edge e so we call it a "private" dome vertex. Thus there are at least d private vertices. Since $\mathcal{D}$ has at most 6 dome vertices, we have $d \leq 6$.

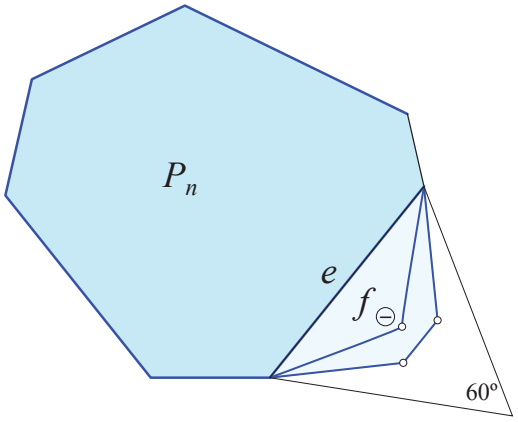


Figure 6 Projection of downward face f lies within an equilateral triangle outside base edge e. Here $P_n = P_7$.

Now Lemma 3.4 implies that $d \geq \frac{n}{2}$. Thus $\frac{n}{2} \leq d \leq 6$ so $n \leq 12$. Furthermore:

▶ **Lemma 3.7.** *For any dome over an equiangular n-gon with odd $n \geq 6$, all the dome faces incident to base edges have downward normals, i.e., $d \geq n$.*

Therefore there is no domeable equiangular n-gon with odd $n \geq 6$, since we would need $n \leq d \leq 6$. ◀

Proof outline for Lemma 3.7. Each base vertex b_i has four incident faces and thus, considered in isolation, has only one degree of freedom for the dihedral angles of incident edges. Let δ_i be the dihedral angle of $\mathcal{D}$ at base edge $e_i = b_i b_{i+1}$. Because b_i and b_{i+1} share edge e_i and have the same face angles, we can show that $\delta_{i-1} = \delta_{i+1}$. Since n is odd, this implies that all the δ_i's are equal. Lemma 3.4 implies that at least one δ_i is $> 90°$, so they all are. ◀

4 Further Results and Open Questions

We have characterized equiangular domeable polygons. Many open problems remain. Here are a few.

(1) Is there any convex n-gon with $n > 12$ that can be domed? A rough bound is $n \leq 55$: every dome vertex has curvature at least $\frac{\pi}{3}$ so there are at most 11 dome vertices; every

base vertex must be adjacent to a dome vertex and dome vertices have degree at most 5. We can prove the stronger bound $n \leq 24$.

(2) Is there any convex 7-gon that can be domed? We have constructed 9- and 11-gons (non-equiangular) that can be domed. See Fig. 7.

(3) Is there any non-equilateral triangle that can be domed? Glazarin and Pak conjectured [4] that, even under their looser conditions, an isosceles triangle with edge lengths $2, 2, 1$ cannot be spanned.

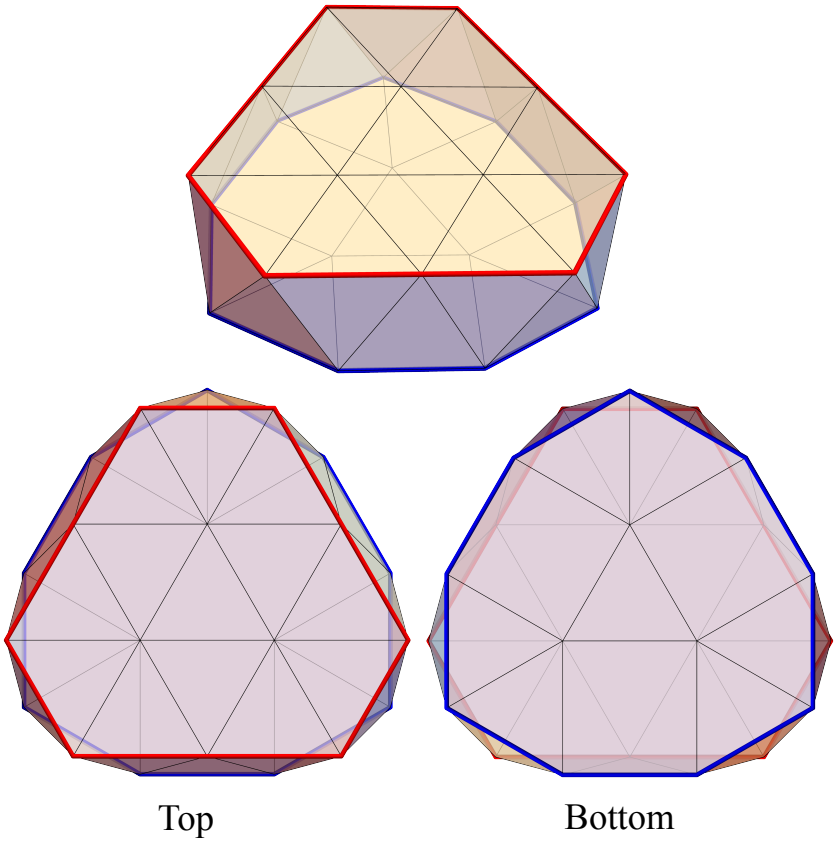


Figure 7 The top (red) is a polyiamond of 13 equilateral triangles. The base (blue) is a 9-gon with base angles 120° and 150°.

Acknowledgments. We benefitted from suggestions from four anonymous reviewers.

References

1 Sasha Anan'in and Dmitrii Korshunov. Moduli spaces of polygons and deformations of polyhedra with boundary. *Geometriae Dedicata*, 218(1):1–19, 2024.

2 Derek Ball. Equiangular polygons. *Math. Gazette*, 86(507):396–407, 2002.

3 Daniel Bezdek. A proof of an extension of the icosahedral conjecture of Steiner for generalized deltahedra. *Contributions Discrete Math.*, 2(1), 2007.

4 Alexey Glazyrin and Igor Pak. Domes over curves. *International Mathematics Research Notices*, 2022(18):14067–14104, 2022.

Clustered Planarity Variants for Level Graphs*

Simon D. Fink[1,2], Matthias Pfretzschner[2], Ignaz Rutter[2], and Marie Diana Sieper[3]

1 **Algorithms and Complexity Group, Technische Universität Wien, Austria**
sfink@ac.tuwien.ac.at
2 **Faculty of Computer Science and Mathematics, University of Passau, Germany**
{finksim,pfretzschner,rutter}@fim.uni-passau.de
3 **Institute of Computer Science, University of Würzburg, Germany**
marie.sieper@uni-wuerzburg.de

Abstract

We consider variants of the clustered planarity problem for level-planar drawings. So far, only convex clusters have been studied in this setting. We introduce two new variants that both insist on a level-planar drawing of the input graph but relax the requirements on the shape of the clusters. In unrestricted Clustered Level Planarity (uCLP) we only require that they are bounded by simple closed curves that enclose exactly the vertices of the cluster and cross each edge of the graph at most once. The problem y-monotone Clustered Level Planarity (y-CLP) requires that additionally it must be possible to augment each cluster with edges that do not cross the cluster boundaries so that it becomes connected while the graph remains level-planar, thereby mimicking a classic characterization of clustered planarity in the level-planar setting.

We give a polynomial-time algorithm for uCLP if the input graph is biconnected and has a single source. By contrast, we show that y-CLP is hard under the same restrictions and it remains NP-hard even if the number of levels is bounded by a constant and there is only a single non-trivial cluster.

Related Version *full version including missing proofs*: arXiv:2402.13153

1 Introduction

A *level graph* (G, γ) is a graph $G = (V, E)$ and a function $\gamma\colon V \to \{1, 2, \dots, k\}$ with $k \in \mathbb{N}$ that assigns vertices to levels such that no two adjacent vertices are assigned to the same level. A *level planar drawing* of a level graph (G, γ) is a crossing-free drawing of G that maps each vertex v to a point on the line $y = \gamma(v)$ and each edge to a y-monotone curve between its endpoints. A level graph is *level planar* if it has a level planar drawing. Level planarity can be tested in linear time [11].

Let $G = (V, E)$ be a graph. A *clustering* T of G is a rooted tree whose leaves are the vertices V. Each inner node μ of T represents a *cluster*, which encompasses all leaves V_μ of the subtree rooted at μ. The pair (G, T) is called a *clustered graph*. A *clustered planar drawing* of a clustered graph (G, T) is a planar drawing of G that also maps every cluster μ to a region R_μ that is enclosed by a simple closed curve such that (i) R_μ contains exactly the vertices V_μ, (ii) no two region boundaries intersect, and (iii) no edge intersects the boundary of a cluster region more than once. The combination of (i) and (iii) implies that an edge may intersect a cluster boundary if and only if precisely one of its endpoints lies inside the cluster. A clustered graph is *clustered planar* if it has a clustered planar drawing. The problem of testing this property and finding such drawings is called Clustered Planarity. In a

* Simon D. Fink was funded by the Deutsche Forschungsgemeinschaft (German Research Foundation, DFG) grant RU-1903/3-1 and by the Vienna Science and Technology Fund (WWTF) [10.47379/ICT22029].

40th European Workshop on Computational Geometry, Ioannina, Greece, March 13–15, 2024.
This is an extended abstract of a presentation given at EuroCG'24. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

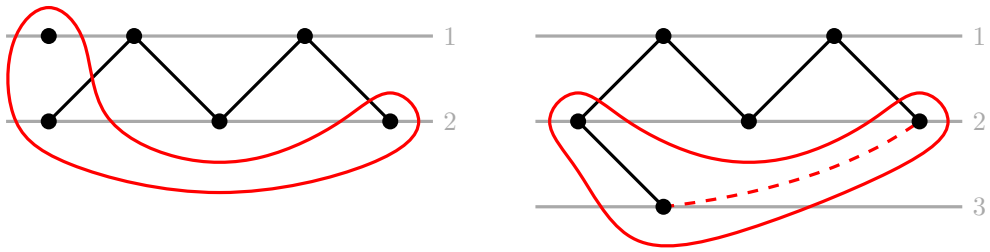


Figure 1 **(a)** A drawing that is level planar and clustered planar and thus cl-planar, but not convex cl-planar or y-cl-planar. **(b)** A drawing that is y-cl-planar (with the augmentation edge in E' shown dashed in red) and thus also cl-planar, but not convex cl-planar.

recent breakthrough, Fulek and Tóth gave the first efficient algorithm for this problem [9], which was soon after improved to a quadratic-time solution [2].

In this paper, we seek to explore the combination of the two concepts of level planarity and clustered planarity. Namely, our input is a *clustered level graph (cl-graph)*, which is a tuple (G, γ, T) such that (G, γ) is a level graph and (G, T) is a clustered graph. We insist on a level-planar drawing of G. However, it is not immediately clear which conditions the cluster boundaries should fulfill. Forster and Bachmaier [8] proposed the problem variant Convex Clustered Level Planarity (short cCLP), which requires to draw the clusters as convex regions[1]. They showed that cCLP can be solved in linear time if the graph is proper (i.e., all edges connect vertices on adjacent levels) and the clusters are level-connected (i.e., each cluster contains an edge between any pair of adjacent levels it spans). Angelini et al. [1] showed that testing cCLP is NP-complete, but can be tested in quadratic time if the input graph is proper, thereby dropping the requirement of level-connectedness.

In this paper we consider two new variants that relax the conditions on the drawing of the cluster. In unrestricted Clustered Level Planarity (short uCLP) we keep the conditions (i)–(iii) as stated above, i.e., the shapes of clusters are not restricted by the levels. Our second variant y-monotone Clustered Level Planarity (short y-CLP) is based on the characterization that a planar drawing $\mathcal{G}$ of a graph G is clustered planar w.r.t. to a clustering T if and only if it is possible to insert a set of *augmentation edges* into $\mathcal{G}$ in a planar way such that each cluster becomes connected and no cycle formed by vertices of a cluster μ encloses a vertex not in μ in its interior [5]. In analogy to this, we define a level-planar drawing to be *y-cl-planar* if it satisfies these conditions but additionally the augmentation edges can be added as y-monotone curves. Figure 1 shows that cCLP, uCLP, y-CLP are indeed different problems. We are not aware of work that concerns uCLP or y-CLP.

We show that uCLP can be solved in polynomial time if the input graph is biconnected and has a single *source*, i.e., a vertex that does not have neighbors on a lower level. On the other hand we show that y-CLP is NP-complete under the same conditions and also if the number of levels is 5 and there is only a single non-trivial cluster, i.e., that not contains all vertices.

[1] they only considered this convex setting and used the name CLP instead of cCLP

2 Single-Source Biconnected (unrestricted) Clustered Level Planarity

We show that UCLP can be solved efficiently if G is a biconnected graph with a single source. To this end, we combine the polynomial-time solution for Clustered Planarity [2] with a combinatorial description of all level-planar drawings of a biconnected single-source graph [4].

Note that whether a drawing of G is clustered planar depends only on its combinatorial embedding, rather than the precise drawing. Thus, we call an embedding $\mathcal{E}$ of G *clustered planar* if the corresponding drawings are. We call an embedding $\mathcal{E}$ of G *level planar* if G admits a level-planar drawing with embedding $\mathcal{E}$. To solve UCLP for an instance (G, γ, T), we need to find an embedding of G that is both cluster planar and level planar.

We first introduce yet another type of constraints called *synchronized fixed-vertex constraints* (*sfv-constraints* for short). For a graph $G = (V, E)$ an *sfv-constraint* is a set Q of pairs (v, σ_v), where $v \in V$ and σ_v is a fixed cyclic order of the edges incident to v, called its *default rotation*. An embedding $\mathcal{E}$ of G *satisfies* the constraint Q if for each pair $(v, \sigma_v) \in Q$ the rotation of v in $\mathcal{E}$ is its default rotation or if for each pair $(v, \sigma_v) \in Q$ the rotation of v in $\mathcal{E}$ is the reverse of its default rotation. Given a set $\mathcal{Q}$ of sfv-constraints, we say that an embedding of G *satisfies* $\mathcal{Q}$ if it satisfies each $Q \in \mathcal{Q}$.

We use sfv-constraints to bridge from level-planarity to usual planarity. To this end, we introduce a slightly generalized version of Clustered Planarity, where we seek a clustered-planar embedding that additionally satisfies a given set $\mathcal{Q}$ of sfv-constraints. This problem is called Sync CP. The point is that the algorithm of Bläsius et al. [2] reduces Clustered Planarity to the intermediate problem Synchronized Planarity, which includes the option to directly express sfv-constraints. The reduction that shows the following lemma can be found in the full version of this paper at `arXiv:2402.13153` together with all further missing proofs.

▶ **Lemma 2.1.** Sync CP *can be solved in* $O(n^3)$ *time.*

We now turn to the second ingredient. Let (G, γ) be a biconnected single-source level-planar graph and let Γ be an arbitrary level-planar embedding of G. Brückner and Rutter [4] showed that there exists a data structure, called LP-tree, very similar to the famous SPQR-tree, that represents precisely the level-planar embeddings of G; see fig. 2 for an example. Like the SPQR-tree, the embeddings decisions for the LP-tree are made by (i) arbitrarily reordering parallel subgraphs between a pair of vertices and (ii) flipping the embeddings of some disjoint and otherwise rigid structures. Hence, the possible orderings of the edges around each vertex v of G in any level-planar embedding can be described by a PQ-tree [3, 7] T_v, called *level PQ-tree* that is straightforwardly derived from the LP-tree; it contains one P-node $u_{v,\mu}$ for each parallel structure μ in which v occurs and one Q-node $u_{v,\rho}$ for each rigid structure ρ, in which v occurs; see fig. 2a. The level-planar embedding Γ is used as reference to determine a *default rotation* $\sigma_{v,\rho}$ for each Q-node $u_{v,\rho}$.

If an embedding $\mathcal{E}$ of G is level-planar, the rotation of each vertex v is necessarily represented by its level PQ-tree T_v. It further holds that all Q-nodes $u_{v,\rho}$ with $v \in V$ that stem from the same rigid structure either all have their default orientation or all its reversal. An embedding where the last condition holds for each rigid structure is called *ρ-consistent*.

▶ **Lemma 2.2.** *An embedding $\mathcal{E}$ is level-planar if and only if the rotation of each vertex v is represented by its level PQ-tree T_v and moreover $\mathcal{E}$ is ρ-consistent.*

For a biconnected single-source level-planar graph G with level-planar embedding Γ, we derive a new graph G^+ and a set $\mathcal{Q}$ of synchronized fixed-vertex constraints. We replace

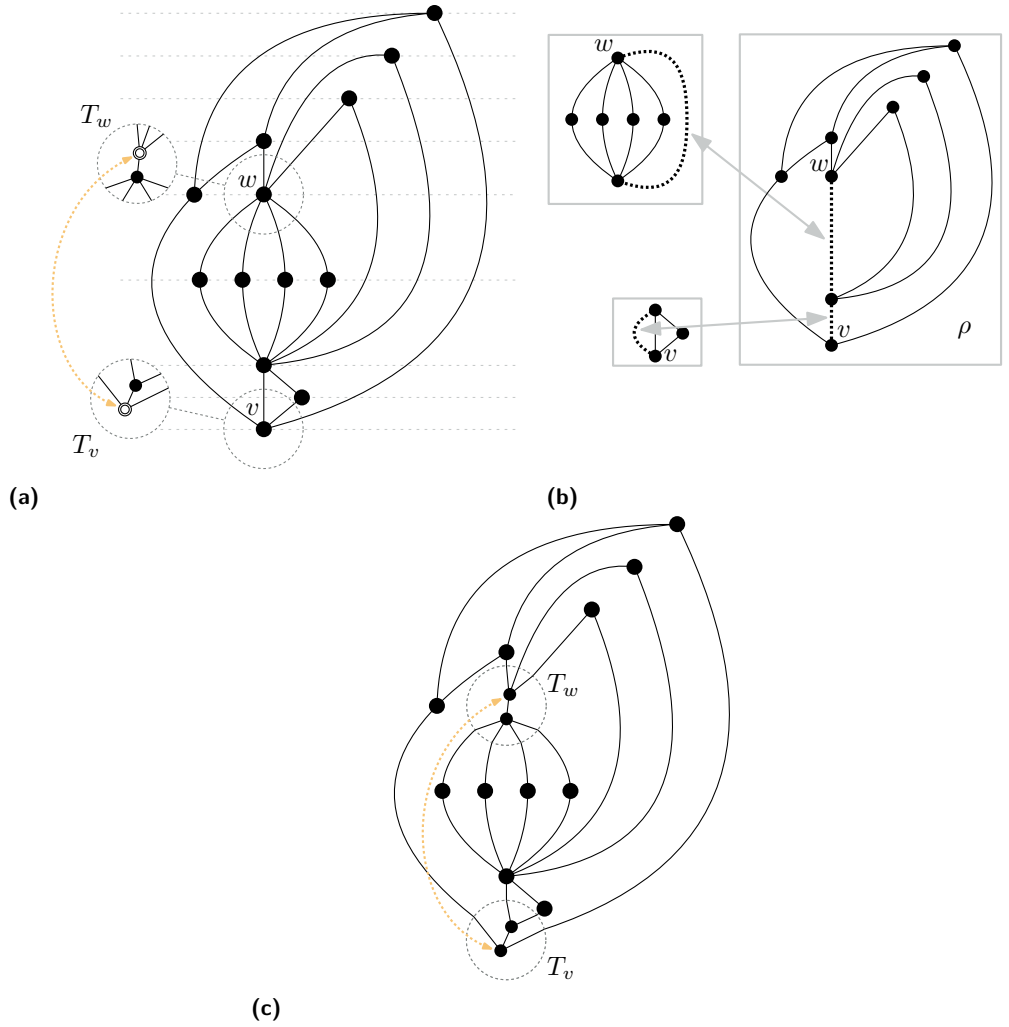


Figure 2 **(a)** A level graph G with two level PQ-trees T_w and T_v derived from **(b)** its LP-tree. P-nodes are represented by black disks, Q-nodes as white double disks. **(c)** The graph after replacing w, v by T_w, T_v; the orange arrow indicates the sfv-constraint due to ρ.

each vertex v of G by a tree isomorphic to its level PQ-tree T_v; see Figure 2c. In order to enforce ρ-consistency, we additionally create for each rigid structure ρ of the LP-tree an sfv-constraint $Q_\rho = \{(u_{v,\rho}, \sigma_{v,\rho}) \mid v \text{ occurs in the rigid structure } \rho\}$. Let $\mathcal{Q}$ denote the set of these constraints for all rigid structures. Clearly, we can obtain a planar embedding of G by taking a planar embedding of G^+ that satisfies $\mathcal{Q}$ and contracting each tree T_v back into a single vertex. The embeddings we can obtain in this way are precisely those where the rotation of each vertex v is represented by its level PQ-tree T_v and that are ρ-consistent.

Finally, it is time to connect clusters and level planarity. To this end, consider a clustering T on G. We naturally obtain a corresponding clustering T^+ of G^+ by placing each vertex of G^+ into the cluster of the vertex of G it replaces.

▶ **Lemma 2.3.** *(G, γ, T) admits a planar embedding that is level-planar and clustered-planar if and only if (G^+, T^+) admits a clustered-planar embedding that satisfies $\mathcal{Q}$.*

Altogether, this reduces the problem uCLP of biconnected single-source graphs to Sync CP, which can be solved efficiently by Lemma 2.1.

▶ **Theorem 2.4.** uCLP *can be solved in $O(n^3)$ time for biconnected single-source level graphs.*

3 Hardness of y-monotone Clustered Level Planarity

It is easy to see that y-CLP lies in NP by guessing and verifying the augmentation edges and an embedding. We show that it is NP-hard even for inputs with very restricted properties.

▶ **Theorem 3.1.** *y-monotone Clustered Level Planarity is NP-complete, even if the input is a biconnected graph with just one source.*

Proof Sketch. We reduce from the NP-complete problem Planar Monotone 3-SAT [6], which asks for the satisfiablity of 3-SAT formulas whose incidence graph has a drawing where all variables lie on a vertical line ℓ, each clause contains only positive or only negative literals, and the positive and negative clauses lie on opposing sides of ℓ; see Figure 3a.

Given such a drawing for a formula ϕ, we first reorient the clauses horizontally above the variables as illustrated in Figure 3b. From this drawing, we construct an equivalent instance of y-CLP as illustrated in Figure 3c. The variables and literals are represented by triconnected pillars (a $(3 \times k)$-grid for suitable k) that extend vertically towards the clause gadgets. The horizontal flip of a pillar represents its truth value. To synchronize pillars of the same variable, we use wedges that enclose a vertex of a disconnected cluster and, due to the required y-monotonicity for cluster connections, prohibit corresponding wedges of adjacent pillars to face each other, as such a connection would have to bypass both wedges. Using two clusters per variable, we can thus ensure consistent flips for all variables; see Figure 3c.

Using a similar approach with wedges, we can construct a clause gadget that allows all assignments for its literals except when all three literals are false; see Figure 3c for the structure of the gadget. As shown in Figure 4, there exists one configuration for the literal pillars of a clause where no valid embedding is possible, as the cluster of the gadget cannot connect with only y-monotone curves. Figure 5 shows valid embeddings for all other configurations. This way, we can construct in polynomial time an equivalent instance of y-CLP where the graph is biconnected and only has a single source. ◀

Our second reduction is from 3-Partition, whose input is a multiset $A = \{a_1, \ldots, a_{3m}\}$ positive integers and a bound $B \in \mathbb{N}^+$ with $B/4 < a_i < B/2$ and $\sum_{a \in A} a = m \cdot B$.

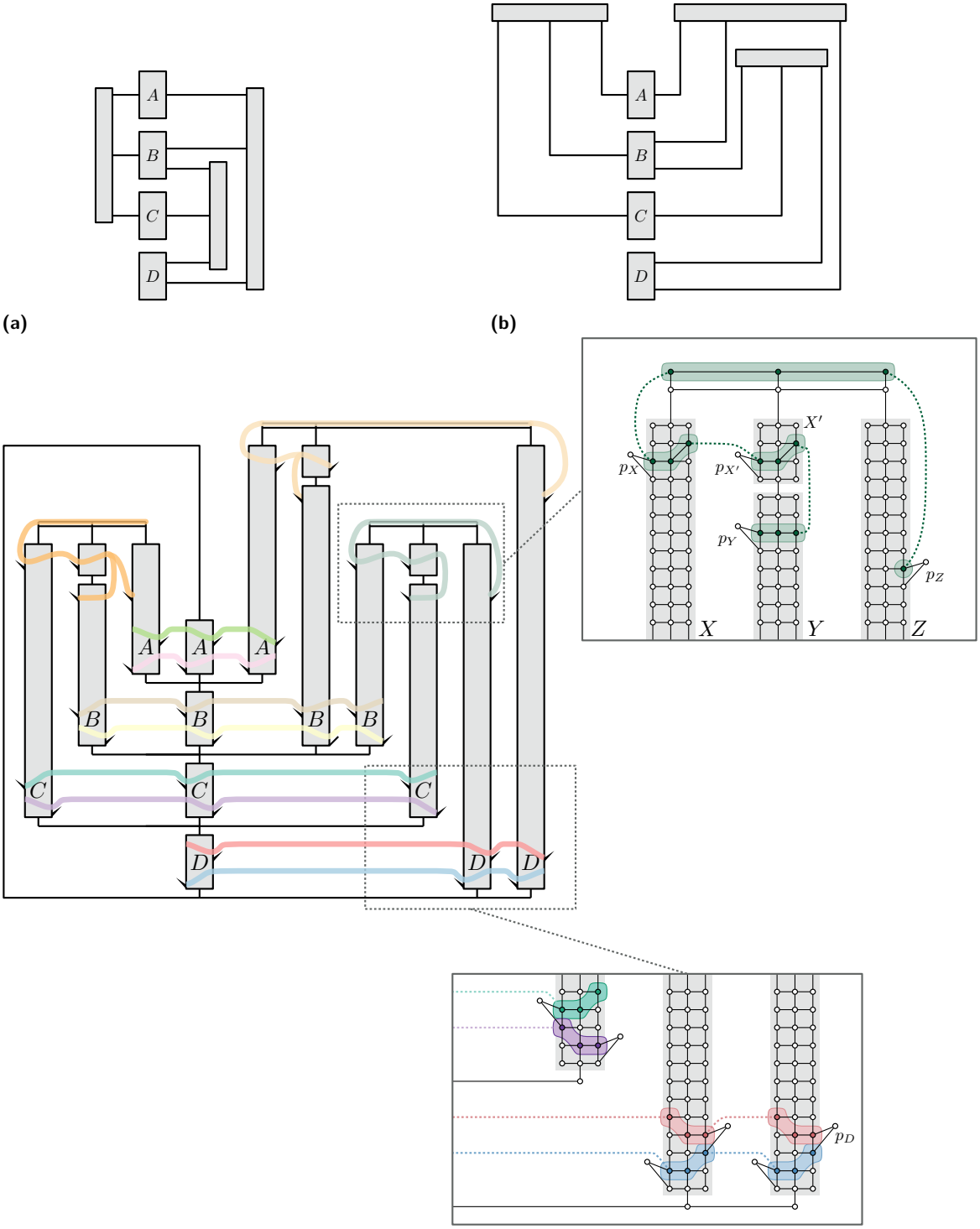


Figure 3 **(a)** An instance of Planar Monotone 3-SAT. **(b)** The modified incidence graph. **(c)** The structure of the corresponding y-CLP instance. Highlighted are a clause gadget (top right) and the gadget for propagating variable assignments (bottom right).

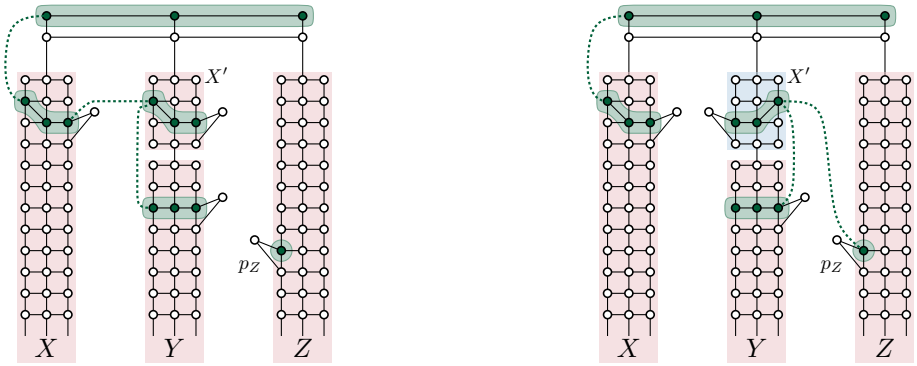


Figure 4 Neither flip of X' admits a valid embedding of the clause gadget if all literals are false.

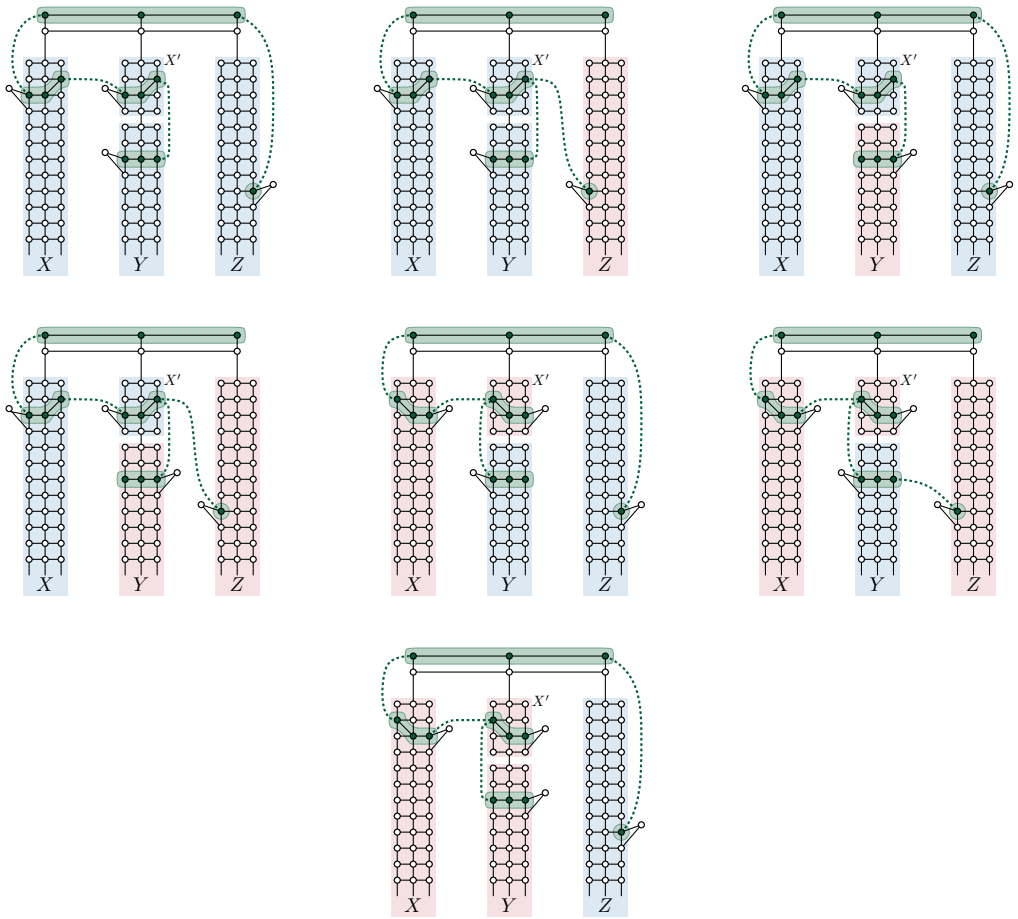


Figure 5 The seven variable assignments for which the clause gadget admits a valid embedding.

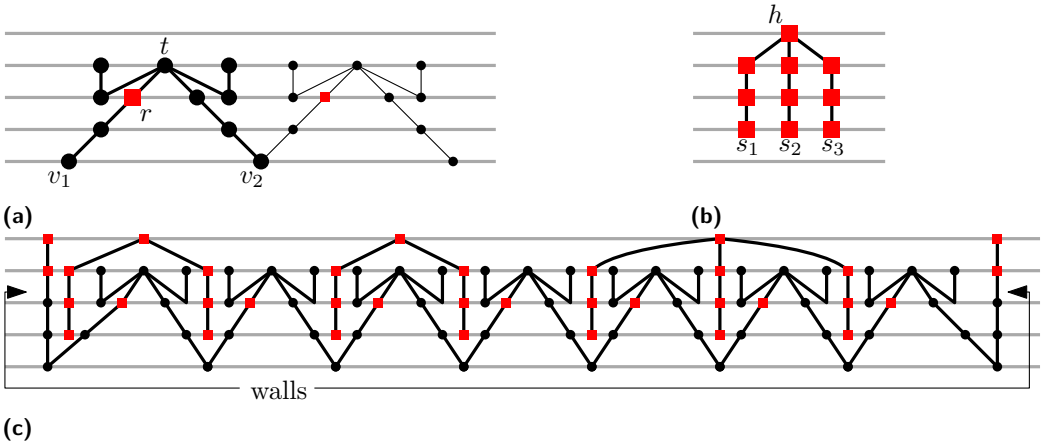


Figure 6 **(a)** A receiver (bold) with a marked vertex r (box), and a second receiver (non-bold) chained to the first one. **(b)** A 3-plug. **(c)** A bucket of size 7, filled with two 2-plugs and a 3-plug. Every marked bucket vertex can be connected to a pin with a y-monotone curve.

The question is whether A can be partitioned into m sets $A_1, A_2, \ldots, A_m$, such that for every $j \in \{1, \ldots, m\}$ it is $\sum_{a \in A_j} a = B$. 3-Partition is strongly NP-complete, i.e., it remains NP-complete even if B is polynomial in m [10].

▶ **Theorem 3.2.** *y-monotone Clustered Level Planarity is NP-complete, even if the input contains only one non-trivial cluster, the number of levels is at most 5, and all vertices have a fixed rotation.*

Proof Sketch. Let (m, A, B) be an instance of 3-Partition. We construct an instance of uCLP with a single non-trivial cluster μ. The main idea is to build m *buckets* (the structure in Figure 6c) by chaining B *receivers* (the structure in Figure 6a, each of which contains a *connector vertex* r that belongs to μ), and closing the sides of each bucket with two paths (also called *walls*). Note that each of the B connector vertices of a bucket must be connected to the rest of μ due to the paths of length 2 attached to the vertices marked as t in the receiver; see Figure 6a. For every $a \in A$, we generate an *a-plug*; the structure in Figure 6b). The leaves of a plug are called *pins* and these are the only vertices that can link the connector vertices to the remainder of cluster μ. Given a plug and a bucket, either all of the vertices of the plug are drawn between the two bucket walls, or none of them. Thus, we can model a solution $A_1, \ldots, A_m$ with the m buckets, and a drawing assigns $a \in A$ to A_i if and only if the corresponding a-plug is in the i-th bucket. Since every pin can join at most one connector vertex, there are at least B pins inside a bucket in a valid drawing. Since there are m buckets and a total of $m \cdot B$ pins, the instance is valid if and only if we can distribute the plugs in such a way that there are precisely B pins per bucket, which corresponds directly to a solution of (m, A, B). ◀

4 Conclusion

We have introduced the problems unrestricted Clustered Level Planarity and y-monotone Clustered Level Planarity, gave an polynomial-time algorithm for unrestricted Clustered Level Planarity if restricted to biconnected single-source graphs, and showed that y-monotone Clustered Level Planarity is NP-complete under very restricted conditions.

We conclude by providing some open questions. On the one hand, these are inspired by the restrictions imposed by our algorithm. The LP-trees we use in Theorem 2.4 only exist for biconnected single-source instances and it is unlikely that this concept can be extended to multiple sources [4, Section 5]. Is it possible to extend our algorithm to non-biconnected graphs? More generally, what is the complexity of unrestricted Clustered Level Planarity? On the other hand, the NP-hardness results on y-monotone Clustered Level Planarity and (non-proper) Convex Clustered Level Planarity [1] raise the question whether these problems are FPT with respect to natural parameters.

References

1 Patrizio Angelini, Giordano Da Lozzo, Giuseppe Di Battista, Fabrizio Frati, and Vincenzo Roselli. The importance of being proper: (in clustered-level planarity and t-level planarity). *Theoretical Computer Science*, 571:1–9, 2015. doi:10.1016/j.tcs.2014.12.019.

2 Thomas Bläsius, Simon D. Fink, and Ignaz Rutter. Synchronized planarity with applications to constrained planarity problems. *ACM Transactions on Algorithms*, 2023. doi:10.1145/3607474.

3 Kellogg S. Booth and George S. Lueker. Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms. *Journal of Computer and System Sciences*, 13(3):335–379, 1976. doi:10.1016/S0022-0000(76)80045-1.

4 Guido Brückner and Ignaz Rutter. An SPQR-tree-like embedding representation for level planarity. *Journal of Computational Geometry*, 14(1):48–77, 2023. doi:10.20382/JOCG.V14I1A3.

5 Sabine Cornelsen and Dorothea Wagner. Completely connected clustered graphs. *Journal of Discrete Algorithms*, 4(2):313–323, 2006. doi:10.1016/j.jda.2005.06.002.

6 Mark de Berg and Amirali Khosravi. Optimal binary space partitions for segments in the plane. *Int. J. Comput. Geom. Appl.*, 22(3):187–206, 2012. doi:10.1142/S0218195912500045.

7 Simon D. Fink, Matthias Pfretzschner, and Ignaz Rutter. Experimental comparison of PC-trees and PQ-trees. *ACM Journal of Experimental Algorithmics*, 28(1):10:1–10:24, 2023. doi:10.1145/3611653.

8 Michael Forster and Christian Bachmaier. Clustered level planarity. In Peter van Emde Boas, Jaroslav Pokorný, Mária Bieliková, and Julius Stuller, editors, *Proceedings of the 30th Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM'04)*, volume 2932 of *Lecture Notes in Computer Science*, pages 218–228. Springer, 2004. doi:10.1007/978-3-540-24618-3_18.

9 Radoslav Fulek and Csaba D. Tóth. Atomic embeddability, clustered planarity, and thickenability. *Journal of the ACM*, 69(2):13:1–13:34, 2022. arXiv:1907.13086v1, doi:10.1145/3502264.

10 Michael R. Garey and David S. Johnson. Complexity results for multiprocessor scheduling under resource constraints. *SIAM Journal on Computing*, 4(4):397–411, 1975. doi:10.1137/0204035.

11 Michael Jünger, Sebastian Leipert, and Petra Mutzel. Level planarity testing in linear time. In Sue Whitesides, editor, *Proceedings of the 6th International Symposium on Graph Drawing (GD'98)*, volume 1547 of *Lecture Notes in Computer Science*, pages 224–237. Springer, 1998. doi:10.1007/3-540-37623-2_17.

Robust Algorithms for Finding Triangles and Computing the Girth in Unit Disk and Transmission Graphs

Katharina Klost[1] **and Wolfgang Mulzer**[1]

1 **Institut für Informatik, Freie Universität Berlin**
{kathklost, mulzer}@inf.fu-berlin.de

Abstract

We describe optimal *robust* algorithms for finding a triangle and the unweighted girth in a unit disk graph, as well as finding a triangle in a transmission graph. In the robust setting, the input is not given as a set of sites in the plane, but rather as an abstract graph. The input may or may not be realizable as a unit disk graph or a transmission graph. If the graph is realizable, the algorithm is guaranteed to give the correct answer. If not, the algorithm will either give a correct answer or correctly state that the input is not of the required type.

1 Introduction

Suppose we are given a set $S \subseteq \mathbb{R}^2$ of n *sites* in the plane, where each site $s \in S$ has an *associated radius* $r_s > 0$. The *disk graph* $D(S)$ on S is defined as $D(S) = (S, E)$, where $E = \{st \mid \|st\| \leq r_s + r_t\}$, with $\|st\|$ being the Euclidean distance between the sites s and t. If all associated radii are 1, $D(S)$ is called a *unit disk graph*. The *transmission graph* on S is the *directed* graph with vertex set S and a directed edge st from site s to site t if and only if $\|st\| \leq r_s$, i.e., if and only if the t lies inside the disk of radius r_s centered at s. If all radii are equal, the edges st and ts are always either both present or both absent, for any two sites $s, t \in S$, and the resulting transmission graph is equivalent to a unit disk graph. Thus, for transmission graphs, the interesting case is that the associated radii are not all the same.

There is a large body of literature on (unit) disk graphs, see, e.g., [3,5,8,12,13]. Transmission graphs are not as widely studied, but recently they have received some attention [13,14]. Even though disk graphs and transmission graphs may have up to $\Omega(n^2)$ edges, they can be described succinctly with $O(n)$ numbers, namely the coordinates of the sites and the associated radii. Thus, the underlying geometry often makes it possible to find efficient algorithms whose running time depends only on n.

In the setting where a unit disk graph or a transmission graph is given as an abstract graph, not much is known. One possible explanation is that the problem of deciding whether an abstract graph is a (unit) disk graph or a transmission graph is $\exists\mathbb{R}$-hard [11, 15]. In fact, Kang and Müller [11] show that there are unit disk graphs whose coordinates need an exponential number of bits in their representation, so even if it is known that the input is a unit disk graph, it is not clear that a realization of the graph can be efficiently computed. As transmission graphs with unit radii are equivalent to unit disk graphs, the result carries over to transmission graphs.

Raghavan and Spinrad [19] introduced a notion of *robust* algorithms in *restricted domains*. A restricted domain is a subset of the possible inputs. In our case, it will be the domain of unit disk graphs or transmission graphs as a subdomain of all abstract graphs. Contrary to the *promise* setting, in which the algorithm only gives guarantees for inputs from the restricted domain, the output in the robust setting must always be useful. If the input comes

40th European Workshop on Computational Geometry, Ioannina, Greece, March 13–15, 2024.
This is an extended abstract of a presentation given at EuroCG'24. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

from the restricted domain, the algorithm must always return a correct result. If the input is not from the restricted domain, the algorithm may either return a correct result, or correctly state that the input does not meet the requirement. Raghavan and Spinrad [19] give a robust polynomial-time algorithm for the CLIQUE-problem in unit disk graphs.

The problem of finding a triangle or of computing the *girth* (the shortest unweighted cycle) in a graph is a basic algorithmic question in graph theory. The best know algorithm for general graphs uses matrix multiplication and runs in either $O(n^\omega)$ or $O(n^{2\omega/(\omega+1)})$ time, where $\omega \leq 2.371552$ is the matrix multiplication constant [2,7,10,20]. The best combinatorial algorithm needs $O\left(n^3/2^{\Omega(\sqrt[7]{\log n})}\right)$ time [1,21].

For special graph classes, better results are known. In the case of *planar* graphs, Itai and Rodeh [10], and, independently, Papadimitriou and Yannakakis [17] show that a triangle can be found in $O(n)$ time, if it exists. Chang and Lu [4] give an $O(n)$ time algorithm for computing the unweighted girth in an undirected planar graph. Kaplan et al. [13] show that in the geometric setting, finding a triangle and computing the unweighted girth can be done in $O(n \log n)$ time for general disk graphs and in $O(n \log n)$ expected time for transmission graphs. They also give algorithms with the same expected running time, for finding the smallest weighted triangle in disk graphs and transmission graphs, as well as for computing the weighted girth of a disk graph. In the geometric setting, there are $\Omega(n \log n)$ lower bounds for finding (short) triangles and computing the (weighted) girth in the algebraic decision tree model [16,18].

In this paper, we show that there are $O(n)$ time algorithms for finding a triangle and computing the girth in unit disk graph, in the robust setting. Furthermore, we extend the ideas to an algorithm for finding a triangle in a transmission graph in $O(n+m)$ time. The running times for the algorithms in the unit disk graph setting can be sublinear in the input size, as the input in the robust setting consists of a representation of all vertices and edges. In particular, the result is better than the $\Omega(n \log n)$ lower bound for the geometric setting, because this lower bound stems from the difficulty of finding the edges of the graph. The running time for transmission graphs is linear in the input size, and it is significantly faster than the currently fastest algorithm for general graphs [11].

2 Preliminaries

We assume that the input is an abstract unweighted graph $G = (V, E)$, given as an adjacency list. For an undirected graph, given a vertex v, we denote by $N(v)$ the set of vertices that are adjacent to v, and by $\deg(v) = |N(v)|$ the degree of v. In the adjacency list representation, a set of k neighbors of v can be reported in $O(k)$ time, and testing if two vertices u and v are adjacent takes $O(\min(\deg(v), \deg(u)))$ time. For a directed graph, let $N_{\text{in}}(v)$ and $N_{\text{out}}(v)$ be the vertices connected by an incoming or outgoing edge to a vertex v, respectively. Let $N_{\text{bi}} = N_{\text{in}} \cap N_{\text{out}}$ be the vertices connected by both an incoming and an outgoing edge. We call an edge (u, v) such that $u \in N_{\text{bi}}(v)$ a *bidirected edge*.

▶ **Definition 2.1** (Raghavan and Spinrad [19]). A *robust* algorithm for a problem P on a domain C solves P correctly, if the input is from C. If the input is not in C, the algorithm may either produce a correct answer for P or report that the input is not in C.

3 Unit disk graphs

Our algorithms make use two key properties. Both properties are well known. For completeness, we include proofs for the variants we use.

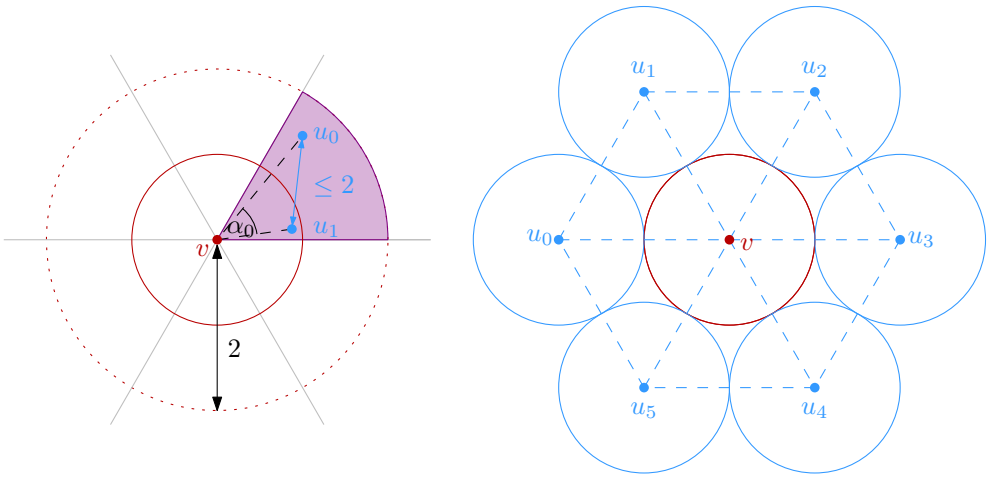


Figure 1 Left: The disk defined by v is marked in solid red. There are two sites in the shaded sector. The diameter of the sector is 2, so u_0 and u_1 are connected by an edge.
Right: The mutual distance between the u_i is maximized on the vertices of a regular polygon. The side length of each equilateral triangle is 2, and thus the desired edges exist.

▶ **Lemma 3.1.** *Let $G = (V, E)$ be a graph that is realizable as an unit disk graph, and let $v \in V$ be a vertex with $\deg(v) > 5$. Then, the subgraph induced by v and any six adjacent vertices contains a triangle.*

Proof. Consider a realization of the unit disk graph in the plane. We identify the vertices with the corresponding sites. Let $u_0, \ldots, u_5 \in N(v)$ be any six neighbors of v, labelled in clockwise order around v.

Let $\alpha_i = \sphericalangle u_i v u_{i+1}$, $i = 0, \ldots 5$, be the angles between the consecutive neighbors with respect to v, where the indices are taken modulo 6. Note that $\sum_{i=0}^{5} \alpha_i = 2\pi$, and suppose that α_0 is a minimum angle in this sequence.

First, suppose that $\alpha_0 < \pi/3$. Then, there is a cone with opening angle $\pi/3$ and apex v that contains the sites u_0 and u_1. Since u_0 and u_1 are adjacent to v, we have $\|u_0 v\| \leq 2$ and $\|u_1 v\| \leq 2$, so u_0 and u_1 both lie inside a circular sector with angle $\pi/3$, apex v, and radius 2. This circular sector has diameter 2. Thus, all points in it, in particular u_0 and u_1, have mutual distance at most 2. If follows that u_0 and u_1 are connected by an edge, closing a triangle, see Figure 1, left.

Second, suppose that $\alpha_0 \geq \pi/3$. Then, since α_0 is minimum and since the α_i sum to 2π, it follows that $\alpha_i = \pi/3$, for $i = 0, \ldots, 5$, so the u_i lie on six concentric, uniformly spaced rays that emanate from v. In this case, the maximum possible mutual distance between the u_i's is achieved when the sites constitute the corners of a regular hexagon with center v and $\|u_i v\| = 2$, for $i = 0, \ldots, 5$. This hexagon decomposes into six equilateral triangles of side length 2, and thus the unit disk graph contains all consecutive edges $\{u_i, u_{i+1}\}$, closing a triangle, see Figure 1, right. ◀

▶ **Lemma 3.2.** *If a (unit) disk graph does not contain a triangle, it is planar.*

Proof. Given the geometric representation of a unit disk graph, the natural embedding of this unit disk graph into the plane is by connecting the sites that represent the vertices by line segment. Evans et al. [6], as well as Kaplan et al. [13] show that if this embedding is not plane, there has to be a triangle in the unit disk graph. Conversely, this implies that if there

is no triangle in the graph, the natural embedding is crossing free, directly implying that the graph is planar. ◀

3.1 Finding a Triangle

▶ **Theorem 3.3.** *There is a robust algorithm to find a triangle in a disk graph in $O(n)$ time.*

Proof. The algorithm works as follows. If there is no vertex v with $\deg(v) > 5$, then we check explicitly for every vertex whether two of its neighbors are adjacent. If so, we have found a triangle. If not, there is none. As all degrees are constant, this takes $O(1)$ time per vertex, for a total of $O(n)$ time.

Now, assume there is a vertex v with $\deg(v) > 5$. Let $N'(v)$ be a set of any seven neighbors of v. For every pair of neighbors u, w from $N'(v)$, explicitly check if there is an edge between u and w. If an edge is found, report the triangle u, v, w. Otherwise, report that the input is not a unit disk graph. This step takes $O(n)$ time to identify v and then at most $O(n)$ time to check the adjacencies for each of the $O(1)$ vertices in $N'(v)$, summing up to $O(n)$ total time. Note that in the case that not all degrees are at most five, only one vertex is considered in detail.

To see that the algorithm is correct, we consider all possible cases. If the maximum degree of the graph is at most 5, all vertices and their neighbors are explicitly checked. So if there is a triangle in the graph, the algorithm will find it and correctly report it. Furthermore, no triangles can be missed. Otherwise, there is a vertex v with degree larger than 5. If the input is a disk graph, Lemma 3.1 guarantees that there is a triangle in $N'(v)$. The algorithm explicitly searches for such edge between vertices of $N(v')$. If such an edge is found, the triangle is correctly reported. In the other case, Lemma 3.1 implies that the input is not a unit disk graph, as reported by the algorithm. ◀

3.2 Computing the Girth

▶ **Theorem 3.4.** *There is a robust algorithm to compute the girth of a graph in the domain of unit disk graphs that runs in $O(n)$ time.*

Proof. First, run the algorithm from Theorem 3.3 on the input. If the algorithm determines that the input graph is not a unit disk graph, report this and finish. If the algorithm found a triangle, the girth of the graph is three and can be reported.

If the algorithm from Theorem 3.3 did not find a triangle and did not report that the graph is not a unit disk graph, we use a linear time planarity testing algorithm on the graph, e.g., the algorithm described by Hopcroft and Tarjan [9]. If the graph is not planar, report that it is not a unit disk graph. In the other case, the algorithm for computing the girth in a planar graph by Chang and Lu [4] can be used to compute the girth of the graph in $O(n)$ time.

By combining the running times for each step, the overall running time follows. Correctness follows from Lemma 3.2 and the correctness of the algorithm by Chang and Lu. ◀

4 Finding a directed triangle in a transmission graph

The following key lemma needed for the robust algorithm for transmission graphs was previously shown by Klost [16].

▶ **Lemma 4.1.** *If G is a transmission graph and v is a vertex with $|N_{bi}(v)| > 6$, then G contains a triangle.*

▶ **Lemma 4.2.** *The sets $N_{bi}(v)$, for all $v \in V$ can be found in $O(n+m)$ time.*

Proof. We assume that the adjacency list representation has the following standard form: on the top level, there is an array that can be indexed by the vertices in V. Each entry points to the head and the tail of a linked list. This linked list contains all vertices u such that $(v,u) \in E$, in no particular order. Note that the order of the array directly induces a total order $\preceq$ on V.

We compute adjacency list representations $L_{\preceq}^{\top}$ and $L_{\preceq}$ of the transposed graph $G^{\top}$ and the original graph G, with the additional property that linked list are sorted according to $\preceq$. For this, we initialize a new array and traverse the original adjacency list representation of G, by considering the vertices according to $\preceq$. For every edge (v,u) that is encountered, we append v to the linked list of u, in $O(1)$ time. As the source vertices v are traversed according to $\preceq$, this gives the desired representation $L_{\preceq}^{\top}$ of $G^{\top}$, in time $O(n+m)$. After that, we can obtain the representation $L_{\preceq}$ of G by the same procedure, using $L_{\preceq}^{\top}$ as the initial adjacency list. Finally, to identify the sets $N_{\mathrm{bi}}(v)$, for each $v \in V$, it suffices to merge the associated lists $L_{\preceq}$ and $L_{\preceq}^{\top}$, in total time $O(n+m)$. ◀

▶ **Observation 4.3.** *In every directed cycle C in a transmission graph, there is least one vertex v with $N_{bi}(v) \neq \emptyset$.*

Proof. Let v be the site with the smallest radius on C and let u be the successor of v on C. Then the edge (v,u) exists by definition. Furthermore, the existence of this edge directly implies that $\|uv\| \leq r_v \leq r_u$ and thus (u,v) is also an edge of the transmission graph. ◀

▶ **Theorem 4.4.** *There is a robust algorithm that finds a directed triangle in a transmission graph in $O(n+m)$ time.*

Proof. Preprocess the input as described in the proof of Lemma 4.2 in $O(n+m)$ time, such that the set $N_{\mathrm{bi}}(v)$ is known for every vertex v. This also gives the representations $L_{\preceq}$ and $L_{\preceq}^{\top}$ which can be used to compute a subgraph G_{uni} of G by removing all bidirected edges.

We consider two cases. In the first case, all vertices have $|N_{\mathrm{bi}}(v)| \leq 6$. Test in $O(n+m)$ time if G_{uni} is acyclic. If yes, report that G is not a transmission graph. If not, consider for each vertex v the edges (u,v) with $u \in N_{\mathrm{bi}}(v)$ explicitly and check if there is a vertex w in $N_{\mathrm{in}}(v) \cap N_{\mathrm{out}}(u) \neq \emptyset$. If yes, report the triangle vuw. If no triangle was found after considering all vertices, report that G does not have a triangle. This step take $O(n+m)$ overall time as the adjacency list of every vertex is traversed a constant number of times. By Observation 4.3, the first check makes sure that no triangles without bidirected edges are missed, as in this case G_{uni} is not acyclic, and it is correctly reported that the graph is not a transmission graph. In the second step, all possible triangles with at least one bidirected edge are explicitly checked.

In the second case, let v be a vertex with $|N_{\mathrm{bi}}(v)| > 6$. Find a set of 7 vertices from this set and explicitly check if any pair of them closes a triangle. If there is no triangle, then report that the input is not a transmission graph. Otherwise, report the triangle. This again takes $O(n+m)$ time. The correctness follows directly from Lemma 4.1. ◀

5 Conclusion

We showed that there are robust sublinear algorithms for finding a triangle and computing the girth in unit disk graph as well as a linear time algorithm for finding a triangle in a transmission graph. Extending the arguments to general disk graphs seems to be hard, as the properties given in Lemma 3.1 and Lemma 4.1 do not easily carry over to general disk

graphs. It would be interesting to see if there are properties of transmission graphs that allow a sublinear running time similar to the unit disk graph case.

References

1 Amir Abboud, Nick Fischer, Zander Kelley, Shachar Lovett, and Raghu Meka. New graph decompositions and combinatorial boolean matrix multiplication algorithms. arXiv:2311.09095, 2023.

2 Noga Alon, Raphael Yuster, and Uri Zwick. Finding and counting given length cycles. *Algorithmica*, 17(3):209–223, 1997. doi:10.1007/BF02523189.

3 Sergio Cabello and Miha Jejčič. Shortest paths in intersection graphs of unit disks. *Comput. Geom. Theory Appl.*, 48(4):360–367, 2015. doi:10.1016/j.comgeo.2014.12.003.

4 Hsien-Chih Chang and Hsueh-I Lu. Computing the girth of a planar graph in linear time. *SIAM J. Comput.*, 42(3):1077–1094, 2013. doi:10.1137/110832033.

5 Brent N. Clark, Charles J. Colbourn, and David S. Johnson. Unit disk graphs. *Discrete Mathematics*, 86(1):165–177, 1990. doi:10.1016/0012-365X(90)90358-O.

6 William Evans, Mereke van Garderen, Maarten Löffler, and Valentin Polishchuk. Recognizing a DOG is hard, but not when it is thin and unit. In *Proc. 8th FUN*, pages 16:1–16:12, 2016. doi:10.4230/LIPIcs.FUN.2016.16.

7 François Le Gall. Powers of tensors and fast matrix multiplication. In *Proc. 39th International Symposium on Symbolic and Algebraic Computation (ISSAC)*, pages 296–303, 2014. doi:10.1145/2608628.2608664.

8 A. Gräf, M. Stumpf, and G. Weißenfels. On coloring unit disk graphs. *Algorithmica*, 20(3):277–293, 1998. doi:10.1007/PL00009196.

9 John Hopcroft and Robert Tarjan. Efficient planarity testing. *J. ACM*, 21(4):549–568, 1974. doi:10.1145/321850.321852.

10 Alon Itai and Michael Rodeh. Finding a minimum circuit in a graph. *SIAM J. Comput.*, 7(4):413–423, 1978.

11 Ross J. Kang and Tobias Müller. Sphere and dot product representations of graphs. *Discrete Comput. Geom.*, 47(3):548–568, 2012. doi:10.1007/s00454-012-9394-8.

12 Haim Kaplan, Alexander Kauer, Katharina Klost, Kristin Knorr, Wolfgang Mulzer, Liam Roditty, and Paul Seiferth. Dynamic connectivity in disk graphs. *Discrete Comput. Geom.*, 71(1):214–277, 2024. doi:10.1007/s00454-023-00621-x.

13 Haim Kaplan, Katharina Klost, Wolfgang Mulzer, Liam Roditty, Paul Seiferth, and Micha Sharir. Triangles and girth in disk graphs and transmission graphs. In *Proc. 27th Annu. European Sympos. Algorithms (ESA)*, pages 64:1–64:14, 2019. doi:10.4230/LIPIcs.ESA.2019.64.

14 Haim Kaplan, Wolfgang Mulzer, Liam Roditty, and Paul Seiferth. Spanners and reachability oracles for directed transmission graphs. In *Proc. 31st Int. Sympos. Comput. Geom. (SoCG)*, pages 156–170, 2015. doi:10.4230/LIPIcs.SOCG.2015.156.

15 Katharina Klost. Complexity of recognizing generalized transmission graphs. Master's thesis, Freie Universität Berlin, 2017.

16 Katharina Klost. *Geometric Graphs: Reachability, Long Trees and Short Cycles*. PhD thesis, Freie Universität Berlin, 2021.

17 Christos H. Papadimitriou and Mihalis Yannakakis. The clique problem for planar graphs. *Inform. Process. Lett.*, 13(4):131–133, 1981. doi:10.1016/0020-0190(81)90041-7.

18 Valentin Polishchuk. Personal communication. 2017.

19 Vijay Raghavan and Jeremy Spinrad. Robust algorithms for restricted domains. *J. Algorithms*, 48(1):160–172, 2003. doi:10.1016/S0196-6774(03)00048-8.

20 Virginia Vassilevska Williams, Yinzhan Xu, Zixuan Xu, and Renfei Zhou. New bounds for matrix multiplication: From alpha to omega. In *Proc. 35th Annu. ACM-SIAM Sympos. Discrete Algorithms (SODA)*, pages 3792–3835, 2024. doi:10.1137/1.9781611977912.

21 Huacheng Yu. An improved combinatorial algorithm for Boolean matrix multiplication. *Inform. and Comput.*, 261:240–247, 2018. doi:10.1016/J.IC.2018.02.006.

Connected Matchings*

Oswin Aichholzer[1], Sergio Cabello[2], Viola Mészáros[3], and Jan Soukup[4]

1 **Institute of Software Technology, Graz University of Technology, Graz, Austria**
`oaich@ist.tugraz.at`
2 **Faculty of Mathematics and Physics, University of Ljubljana, Slovenia**
Institute of Mathematics, Physics and Mechanics, Slovenia
`sergio.cabello@fmf.uni-lj.si`
3 **Bolyai Institute, University of Szeged, Szeged, Hungary**
`meszaros.viola@gmail.com`
4 **Faculty of Mathematics and Physics, Charles University, Prague, Czech Republic**
`soukup@kam.mff.cuni.cz`

Abstract

We show that each set of $n \geqslant 2$ points in the plane in general position has a straight-line matching with at least $(5n+1)/27$ edges whose segments form a connected set, while for some point sets the best one can achieve is $\lceil \frac{n-1}{3} \rceil$.

1 Introduction

Consider a set P of n points in the plane in general position, meaning that no three points of P are collinear. A (straight line) *matching* M for P is a set of segments with endpoints in P such that no two segments share an endpoint. A matching M for P is *connected* (via their crossings) if the union of the segments of M forms a connected set. Equivalently, a matching is connected when the intersection graph of its segments is connected. The *size* of the matching M is the number of edges (or segments) in M. In this paper, we study the following problem.

▶ **Question 1.1** (Connected Matching). *Find for each n the largest value $f(n)$ with the following property: each set of n points in general position in the plane has a connected matching with $f(n)$ edges.*

We provide upper and lower bounds for the function $f(n)$. Our upper bounds are constructive and lead to effective algorithms to compute the connected matching. In this short version we focus on our constructions. Missing details and proofs of the algorithmic claims will appear in the full paper, where we also consider a colored version of the problem.

The problem can be seen as a variant of the problem on crossing families of Aronov et al. [3], where one wants to find as many segments as possible with endpoints in P such

* Research on this work has been initiated at the 17th European Geometric Graph Week which was held from August 15th to 19th in Leipzig. We thank all participants for the good atmosphere as well as for inspiring discussions on the topic. O. A. supported by FWF grant W1230. Research funded in part by the Slovenian Research and Innovation Agency (P1-0297, J1-2452, N1-0218, N1-0285). Research funded in part by the European Union (ERC, KARST, project number 101071836). Views and opinions expressed are however those of the authors only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them. J. S. supported by the grant SVV–2023–260699.

40th European Workshop on Computational Geometry, Ioannina, Greece, March 13–15, 2024.
This is an extended abstract of a presentation given at EuroCG'24. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

that any pair of segments crosses in their interior. While in our setting we are asking for a connected subgraph in the intersection graph of the segments, the crossing families problem asks that the intersection graph is a complete graph. The best lower bound, showing an almost linear lower bound for crossing families, has been a recent breakthrough by Pach, Rubin and Tardos [5]. Aichholzer et al. [2] have the currently best upper bound.

2 Balanced separation with a short path

In this section we provide a structural result about splitting the convex hull of a point set with a single edge or with a 2-edge path in such a way that both sides contain a large fraction of the point set. A very similar result can be found in Ábrego and Fernández-Merchant [1, Lemma 2]. We include a proof because their bound has a small error[1], our approach is different in the treatment of the triangular case (Theorem 2.1), and we discuss the algorithmic counterpart, a part that is not considered in [1] and that forces us to rework a proof.

We first consider the case when the convex hull is a triangle and the partition can be with different number of points. This will be a tool for the general case. See Figure 1.

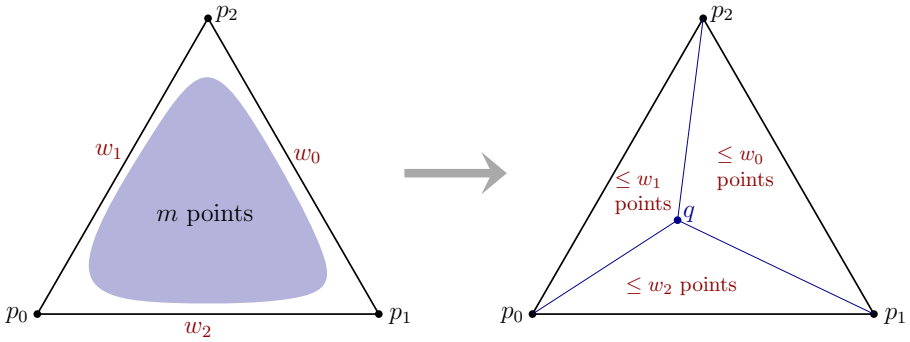


Figure 1 Statement in Theorem 2.1.

▶ **Theorem 2.1.** *Assume that we have a triangle with vertices p_0, p_1 and p_2 and in its interior there is a set P of $m \geqslant 1$ points such that $P \cup \{p_0, p_1, p_2\}$ is in general position. For any integer weights w_0, w_1, w_2 such $0 \leqslant w_0, w_1, w_2 < m$ and $\ell := w_0 + w_1 + w_2 > 2m - 3$, there exist at least $\ell - 2m + 3 > 0$ points $q \in P$ such that, for each $i \in \{0, 1, 2\}$, the triangle $\triangle(p_i q p_{i+1})$ contains at most w_{i+2} points of P in its interior, where all indices are modulo 3.*

We can find $\ell - 2m + 3$ points with this property in linear time.

Proof. In this proof, all indices are modulo 3. For $i \in \{0, 1, 2\}$, consider a ray r_i that starts at p_{i-1} and goes through p_i. We rotate r_i around p_{i-1} in the direction towards p_{i+1} until we pass r_i over $m - w_i - 1$ points of P. See Figure 2, left, to visualize the case $i = 1$. For any of the points $q \in P$ we did not scan over, the triangle $\triangle(p_{i-1} q p_{i+1})$ contains at most w_i points of P in its interior; note that q is not in the interior of $\triangle(p_{i-1} q p_{i+1})$.

Some points of P may be scanned more than once, but in total we scan at most $3m - w_1 - w_2 - w_3 - 3 = 3m - \ell - 3$ points. So there are at least $m - (3m - \ell - 3) = \ell - 2m + 3 > 0$ points remaining, and each of them satisfies the desired property. ◀

As a special case we state the following corollary, which might be of its own interest.

[1] Lemma 2 in [1] is not correct for $n = 4$.

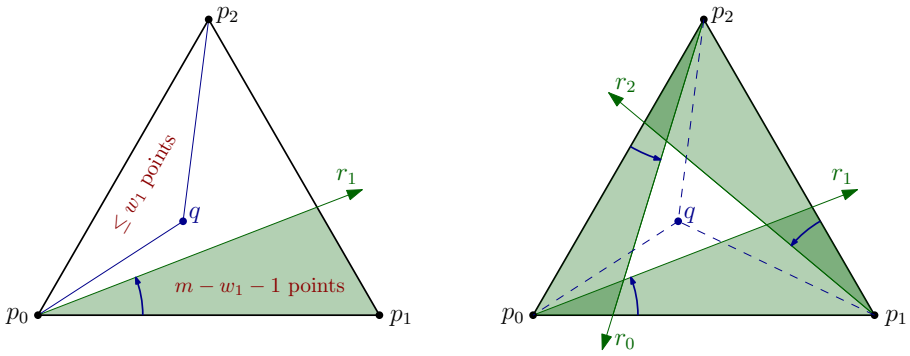


Figure 2 Left: rotating r_1 until we pass over $m - w_1 - 1$ points. Right: the part of the triangle that is not shadowed contains at least $\ell - 2m + 3$ points.

▶ **Corollary 2.2.** *Let Δ be a triangle with a set P of $m \geqslant 1$ points in its interior. Then there is a point of P that splits Δ into three triangles, such that none of these parts contains more than $\lceil (2m-2)/3 \rceil$ points of P in its interior.*

This result resembles the classical Centerpoint theorem [4, Section 1.4], which tells that for each set P of n points in the plane there exists a so-called centerpoint q with the property that each open halfplane that does not contain q has at most $2n/3$ of the points of P inside. However, the centerpoint does not need to be a point of P, and for some point sets it cannot be an element of P.

Denote by $CH(P)$ the convex hull of P. A point $p \in P$ is *extremal* for P if it lies on the boundary of $CH(P)$. A *k-separating path* for P is a plane path π spanned by vertices of P and connecting two different extremal points of P such that $CH(P) \setminus \pi$ has two parts, each containing at least k points; the points on the path are counted in no part. See Figure 3. The *length* of such a path is its number of edges.

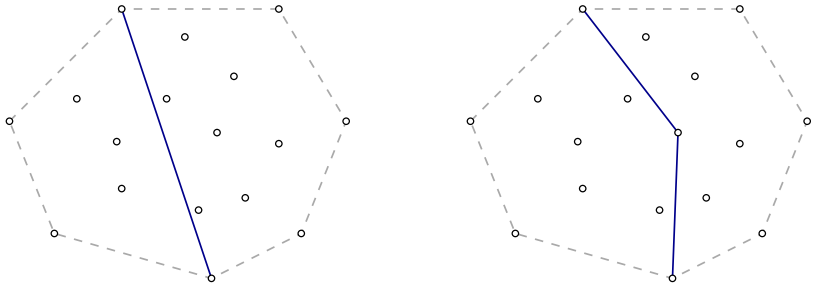


Figure 3 Left: 5-separating path of length 1. Right: 7-separating path of length 2.

▶ **Theorem 2.3.** *Let P be a set of $n \geqslant 2$ points in general position in the plane. There exists a $\lceil \frac{n-4}{3} \rceil$-separating path for P of length 1 or 2 and it can be found in time linear in n.*

Proof sketch. For $n \leqslant 4$ the statement is trivially true. So for the reminder of the proof assume that $n \geqslant 5$. Let us set $r = \lceil (2(n-3)-2)/3 \rceil = \lceil (2n-8)/3 \rceil$. The intuition is that r is the bound of Corollary 2.2 for $n - 3 \geqslant 1$ points; in our current setting, n is also counting the vertices of the triangle. We also set $k = \lceil (n-4)/3 \rceil \geqslant 1$ as $n \geqslant 5$.

Choose an extremal point $q_0 \in P$ with the smallest y-coordinate. Let $q_1, \ldots, q_{n-1}$ be the points $P \setminus \{q_0\}$ sorted increasingly by the angle $\overline{q_0 q_i}$ makes with the horizontal rightward ray from q_0. See Figure 4, left.

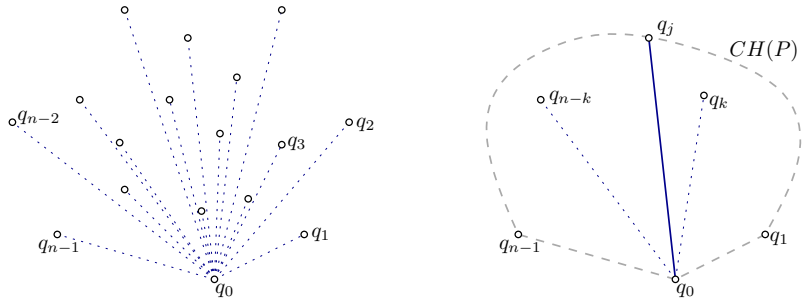


Figure 4 Proof of Theorem 2.3.

If between q_k and q_{n-k} there is some extremal point q_j for P, which implies that $k < j < n-k$, then the segment q_0q_j is a k-separating path of length 1 and we are done. See Figure 4, right. Otherwise, the rays q_0q_k and q_0q_{n-k} intersect the same edge e of $CH(P)$. Let q_aq_b be the edge e, with $a < b$. This means $a \leqslant k < n-k \leqslant b$ and the triangle $\triangle(q_0q_aq_b)$ has exactly $b-a-1$ points in its interior. See Figure 5, left.

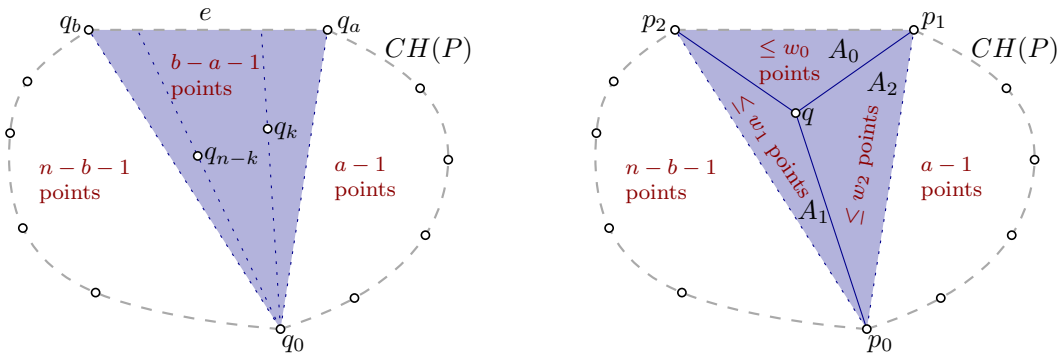


Figure 5 Continuation of the proof of Theorem 2.3.

We want to apply Theorem 2.1 to $\triangle(q_0q_aq_b)$ and the $m = b-a-1 \geqslant (n+1)/3 \geqslant 1$ points of P in its interior. To this end, set $p_0 = q_0$, $p_1 = q_a$, $p_2 = q_b$, $w_0 = r$, $w_1 = r-(n-b-1)$, and $w_2 = r-(a-1)$. See Figure 5, right. After checking that indeed $w_0+w_1+w_2 > 2m-3$, which we skip, Theorem 2.1 implies the existence of a point $q \in P$ in the interior of $\triangle(p_0p_1p_2) = \triangle(q_0q_aq_b)$ that splits it into three triangular pieces such that the interior of the triangle $\triangle(p_{i-1}qp_{i+1})$ has at most w_i points of P (for $i = 0, 1, 2$ and indices modulo 3).

We split $CH(P)$ into three parts A_0, A_1, A_2 by removing the segments $qq_0 = qp_0$, $qq_a = qp_1$ and $qq_b = qp_2$. See Figure 5, right. The points q, q_0, q_a, q_b belong to no part, while all the other points of P belong to exactly one part. From the choices of weights w_i, each part contains at most r points of P. Any part among A_0, A_1, A_2 with most points has at least $\lceil (n-4)/3 \rceil = k$ points and its boundary defines a k-separating path of length 2. ◀

3 Upper bound

Consider n points split into three sets A_0, A_1, A_2 of size $\sim \frac{n}{3}$, where each A_i lies on its own slightly curved blade of a three-bladed windmill; see Figure 6. We use indices modulo three in the discussion. We can form such a configuration so that each line determined by two points of A_i separates A_{i+1} from A_{i+2}, and no segment connecting one point of A_i with

one point of A_{i+1} crosses any segment connecting two points of A_{i+1}. Hence, the set of all segments is separated into three parts where each part consists of segments connecting two points of A_i or one point of A_i and one point of A_{i+1}, and segments from different parts do not cross. Clearly, the size of the largest matching spanning $A_i \cup A_{i+1}$, if their sizes differ by at most one, is $\min\{|A_i|, |A_{i+1}|\}$, and the largest of those values over $i \in \{0, 1, 2\}$ gives the largest connected matching. Treating carefully the modulus of n, we get for each $n \geqslant 1$ a point set where the maximum connected matching has size $\lceil \frac{n-1}{3} \rceil$.

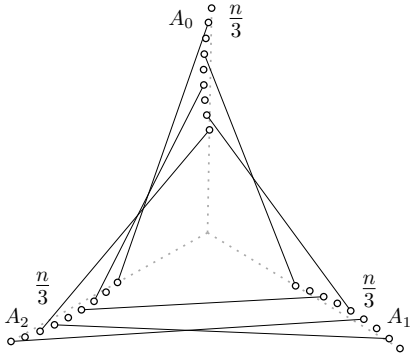


Figure 6 Upper bound for connected matchings.

4 Lower bound

We first consider the following special setting, depicted in Figure 7, left.

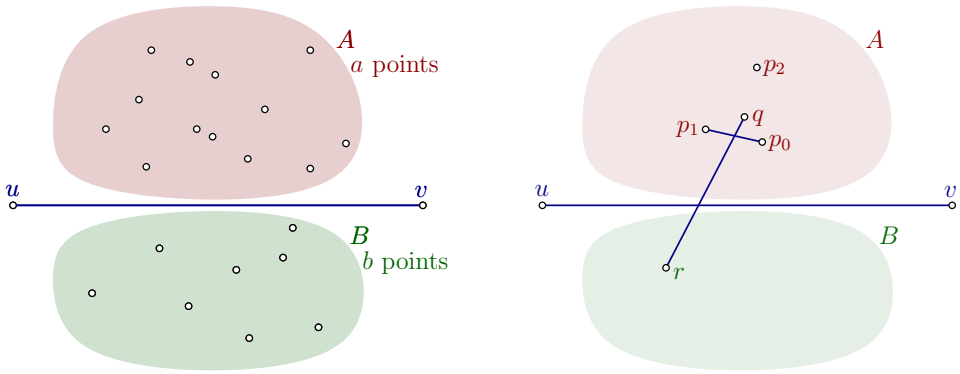


Figure 7 Left: Situation in Lemma 4.1. Right: edges added to the matching when A has four points not in convex position.

▶ **Lemma 4.1.** *Assume that we have a horizontal segment uv, a set A of a points above the line supporting uv, and a set B of $b \leqslant a$ points below the line supporting uv such that, for all $(a, b) \in A \times B$, the segment ab intersects uv, and $A \cup B \cup \{u, v\}$ consists of $a + b + 2$ points in general position. Then, $A \cup B \cup \{u, v\}$ has a connected matching of size at least*

$$m(a,b) := \begin{cases} 1+b & \text{if } b \leqslant a \leqslant 2b+3, \\ (a+3b+2)/5, & \text{if } 2b+3 \leqslant a \leqslant 7b+3, \\ 1+2b, & \text{if } a \geqslant 7b+3. \end{cases}$$

Such a connected matching can be computed in $O(1 + a \log a)$ time.

Proof sketch. We first make two easy observations that will come in handy:

(a) A matching of B onto A with b edges together with the edge uv to "connect" them is a connected matching of size $b + 1$. We want to improve upon this when the sides are unbalanced, in particular when a is larger than $2b \pm O(1)$.
(b) If A has a large subset A' in convex position, then we can get a connected matching of size $\lfloor \frac{|A'|}{2} \rfloor$, for example by connecting "antipodal" points along the boundary of $CH(A')$.

We construct a connected matching M iteratively as follows. At the start we add uv to M. While $|A| > |B| > 0$ and A has four points p_0, p_1, p_2, q such that q is in the interior of $\triangle(p_0p_1p_2)$, we take an arbitrary point $r \in B$, add the edge qr to M, and to M the edge pp' of $\triangle(p_0p_1p_2)$ crossed by qr. See Figure 7, right. Note that $\{pp', uv, qr\}$ is a connected matching. Then we remove p, p', q from A, and r from B. With each repetition of this operation, we increase the size of the matching by two, remove three points from A, and remove a point from B. We repeat this operation until B is empty, $|A| \leqslant |B|$, or A is in convex position, whatever happens first. Let k be the number of repetitions of this operation, let A' and B' be the subsets of A and B, respectively, that remain at the end. Therefore, M is a connected matching with $1 + 2k$ edges, A' has $a - 3k$ points, and B' has $b - k$ points.

We now consider the different conditions that hold at the end:

- If we finish because B' is empty, then $k = b$ and the matching M has $1 + 2b$ edges.
- If we finish because $|A'| \leqslant |B'|$, we match the remaining points of A' to B' arbitrarily and add those edges $|A'|$ to M; since they cross uv, M keeps being a connected matching. Using that the cardinality of A decreases at steps of size 3 and the cardinality of B decreases at steps of size 1, it is possible to show that the size of the connected matching M is in this case $1 + \lfloor (a + b)/2 \rfloor$.
- If we finish because A' does not have any 4 points with the desired condition, the key observation is to note that A' is in convex position. (This is also true if $|A'| \leqslant 3$.) We consider two connected matchings and take the best of both.
  The first matching is obtained by adding to M a matching between all the vertices of B' and any subset of A' with $|B'|$ points. The second matching, which we denote by M', is obtained by taking a connected matching of the points A'; they are in convex position.
  A comparison between M and M' shows that the larger one has size at least

$$\begin{cases} 1 + b & \text{if } b \leqslant a \leqslant 2b + 3, \\ (a + 3b + 2)/5, & \text{if } 2b + 3 \leqslant a \leqslant 7b + 3, \\ (a - 3b - 1)/2, & \text{if } a \geqslant 7b + 3. \end{cases}$$

Since we have given a construction that can finish with 3 different conditions, one has to show that in each scenario $m(a, b)$ is a lower bound on the size of the connected matching. We skip this computation. ◀

Note that the bound $m(a, b)$ of Lemma 4.1 is monotone increasing in a and in b, also when we take a and b as real values (with $b \leqslant a$ always.) Moreover, when $a + b$ is kept constant, $m(a, b)$ is larger for larger b. This means that $m(a, b) \leqslant m(a - 1, b + 1)$, if $b \leqslant a - 2$.

▶ **Theorem 4.2.** *Let P be a set of $n \geqslant 2$ points in general position in the plane. Then P has a connected matching set of size at least $(5n + 1)/27$ and it can be computed in $O(n \log n)$ time.*

Proof sketch. By Theorem 2.3 we know that there is a $\lceil \frac{n-4}{3} \rceil$-separating path P of length 1 or 2 for P. Let A and B be the sets of points of P on each side of π, such that $|A| \geqslant |B|$. Note that the vertices of π do not go to any of the sides, which means that $n-3 \leqslant |A|+|B| \leqslant n-2$ and $\lceil \frac{n-4}{3} \rceil \leqslant |B| \leqslant |A|$. Each edge connecting a point of A to a point of B crosses π.

If π consists of a single edge e, then we match all points of B to points of A arbitrarily, and include e also in the matching. Since all these edges intersect e, they form a connected matching of size $1 + |B| \geqslant \lceil \frac{n-1}{3} \rceil \geqslant \frac{5n+1}{27}$. (This last inequality holds for $n \geqslant 2$.)

For the remainder of this proof we assume that π has length two, and denote its edges by e_1 and e_2. We build a *maximal* matching M_1 from $B_1 \subseteq B$ to $A_1 \subseteq A$ with edges that cross e_1. This means that $|A_1| = |B_1|$ and there is no point in $A \setminus A_1$ that can be connected to a point in $B \setminus B_1$ by crossing e_1. Set $A_2 = A \setminus A_1$ and $B_2 = B \setminus B_1$. Each segment connecting a point in A_2 to a point of B_2 must cross e_2 because it does not cross e_1. We make an arbitrary matching M_2 connecting each point of B_2 to points of A_2; this can be done because $|B_2| = |B| - |M_1| \leqslant |A| - |M_1| = |A_2|$. We add e_1 to M_1 and e_2 to M_2 so that M_1 and M_2 become connected matchings with $|M_1| + |M_2| = 2 + |B|$.

If M_1 or M_2 has size at least $\frac{5n+1}{27}$, then we are done. Therefore, we restrict our attention to the case when $|M_1|, |M_2| \leqslant \frac{5n+1}{27}$. Since $|A_1| = |B_1| = |M_1| - 1 \leqslant \frac{5n-26}{27}$, we have

$$|B_2| \;=\; |B| - |B_1| \;\geqslant\; \left\lceil \frac{n-4}{3} \right\rceil - \frac{5n-26}{27} \;\geqslant\; \frac{4n-10}{27}.$$

We apply Lemma 4.1 to the segment e_2 with A_2 and B_2 to get a connected matching, where $a = |A_2|$ and $b = |B_2|$. Using properties of the lower bound $m(a, b)$ of Lemma 4.1, we can see that a worst-case lower bound is obtained evaluating $m(a, b)$ at the values $a = \frac{13n-19}{27}$ and $b = \frac{4n-10}{27}$. With these values one obtains the lower bound $(5n + 1)/27$. ◀

References

1 Bernardo M. Ábrego and Silvia Fernández-Merchant. The rectilinear local crossing number of K_n. *J. Comb. Theory, Ser. A*, 151:131–145, 2017. doi:10.1016/j.jcta.2017.04.003.

2 Oswin Aichholzer, Jan Kyncl, Manfred Scheucher, Birgit Vogtenhuber, and Pavel Valtr. On crossing-families in planar point sets. *Comput. Geom.*, 107:101899, 2022. doi:10.1016/j.comgeo.2022.101899.

3 Boris Aronov, Paul Erdös, Wayne Goddard, Daniel J. Kleitman, Michael Klugerman, János Pach, and Leonard J. Schulman. Crossing families. *Comb.*, 14(2):127–134, 1994. doi:10.1007/BF01215345.

4 Jiří Matoušek. *Lectures on discrete geometry*, volume 212 of *Graduate texts in mathematics*. Springer, 2002. doi:10.1007/978-1-4613-0039-7.

5 János Pach, Natan Rubin, and Gábor Tardos. Planar point sets determine many pairwise crossing segments. *Adv. Math.*, 386:Paper No. 107779, 21, 2021. doi:10.1016/j.aim.2021.107779.

A Clique-Based Separator for Intersection Graphs of Geodesic Disks in $\mathbb{R}^2$

Boris Aronov[1], Mark de Berg[2], and Leonidas Theocharous[3]

1 **Tandon School of Engineering, New York University, Brooklyn NY, USA**
`boris.aronov@nyu.edu`
2 **Department of Mathematics and Computer Science, TU Eindhoven**
`M.T.d.Berg@tue.nl`
3 **Department of Mathematics and Computer Science, TU Eindhoven**
`l.theocharous@tue.nl`

Abstract

A *clique-based separator* of a graph is a (balanced) separator consisting of cliques, and its size is measured as the total number of cliques it contains. Clique-based separators were introduced by De Berg *et al.* [3], who showed that the intersection graph of a set $\mathcal{D}$ of convex fat objects in the plane admits a separator of size $O(\sqrt{n})$. We extend this result by showing that *for any well-behaved shortest-path metric* d defined on a path-connected and closed subset $F \subset \mathbb{R}^2$, a set of geodesic disks with respect to that metric admits a separator consisting of $O(n^{4/5})$ cliques.

1 Introduction

The Planar Separator Theorem states that any planar graph with n nodes has a *balanced separator* of size $O(\sqrt{n})$. In other words, for any planar graph $\mathcal{G} = (V, E)$ there exists a subset $S \subset V$ of size $O(\sqrt{n})$ with the following property: $V \setminus S$ can be split into subsets A and B with $|A| \leqslant 2n/3$ and $|B| \leqslant 2n/3$ such that there are no edges between A and B. This fundamental result was first proved in 1979 by Lipton and Tarjan [9] and has been refined in several ways, see e.g. [5, 6]. It has proved to be extremely useful for obtaining efficient divide-and-conquer algorithms for a large variety of problems on planar graphs.

In this paper we are interested in *geometric intersection graphs* in the plane. These are graphs whose node set corresponds to a set $\mathcal{D}$ of objects in the plane and that have an edge between two nodes iff the corresponding objects intersect. We will denote this intersection graph by $\mathcal{G}^{\times}(\mathcal{D})$. *(Unit) disk graphs* and *string graphs*—where the set $\mathcal{D}$ consists of (unit) disks and curves, respectively—are among the most popular types of intersection graphs. Unit disk graphs in particular have been studied extensively, because they serve as a model for wireless communication networks. It is well known that disk graphs are generalizations of planar graphs, because by the Circle Packing Theorem (also known as the Koebe–Andreev–Thurston Theorem) every planar graph is the intersection graph of a set of disks with disjoint interiors [10]. It is therefore natural to try to extend the Planar Separator Theorem to intersection graphs. A direct generalization is clearly impossible, however, since intersection graphs can contain arbitrarily large cliques. There are ways to still obtain separator theorems for intersection graphs. One can, for example, allow the size of the separator to depend on m, the number of edges, instead of on the number of vertices. It's known that any string graph admits a separator of size $O(\sqrt{m})$ [7].

Recently, De Berg *et al.* [3] introduced *clique-based separators*. A clique-based separator is a (balanced) separator consisting of cliques, and its size is not measured as the total number of nodes of the cliques but as the total number of cliques it contains. De Berg *et al.* showed that the intersection graph of a set $\mathcal{D}$ of convex fat objects in the plane admits a

40th European Workshop on Computational Geometry, Ioannina, Greece, March 13–15, 2024.
This is an extended abstract of a presentation given at EuroCG'24. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

separator consisting of $O(\sqrt{n})$ cliques. Clique-based separators are useful because cliques can be handled efficiently for many problems. De Berg *et al.* used their separators to solve many classic graph problems, including Independent Set, Dominating Set and more.

De Berg *et al.* [4] proved clique-based separator theorems for various other classes of objects, including map graphs and intersection graphs of pseudo-disks. They also showed that intersection graphs of *geodesic disks inside a simple polygon*—that is, geodesic disks induced by the standard shortest-path metric inside the polygon—admit a clique-based separator consisting of $O(n^{2/3})$ cliques. They left the case of geodesic disks in a polygon with holes as an open problem. We note that string graphs, which subsume the class of intersection graphs of geodesic disks, do *not* admit clique-based separators of sublinear size, since string graphs can contain arbitrarily large bipartite cliques.

Our results. We show that intersection graphs of geodesic disks in a polygon with holes admit a clique-based separator consisting of a sublinear number of cliques. Our result is actually much more general, as it shows that *for any well-behaved shortest-path metric d* defined on a path-connected and closed subset $F \subset \mathbb{R}^2$, a set $\mathcal{D}$ of geodesic disks with respect to that metric admits a separator consisting of $O(n^{4/5})$ cliques. Roughly speaking, we call a metric well-behaved if any two shortest paths under that metric meet at finitely many connected components, and any two disjoint components have some minimum positive clearance between them. This includes the shortest-path metric defined by a set of (possibly curved) obstacles in the plane, the shortest-path metric defined on a terrain, and the shortest-path metric among weighted regions in the plane. (The formal requirements on a well-behaved shortest-path metric can be found in the full version of the paper.) Note that we do not require shortest paths to be unique, nor do we put a bound on the number of intersections between the boundaries of two geodesic disks, nor do we require the metric space to have bounded doubling dimension.

The idea of our new approach is as follows. Recall that the *ply* of $\mathcal{D}$ is defined as the maximum number of objects from $\mathcal{D}$ with a common intersection. We first reduce the ply of the set $\mathcal{D}$ by removing all cliques of size $\Omega(n^{1/5})$, thus obtaining a set $\mathcal{D}^*$ of ply $O(n^{1/5})$. (The removed cliques will eventually be added to the separator. A similar preprocessing step was used in [4] to handle pseudo-disks.) The remaining arrangement can still be arbitrarily complex, however. To overcome this, we ignore the arrangement induced by the disks, and instead focus on the realization of the graph $\mathcal{G}^\times(\mathcal{D}^*)$ obtained by drawing a shortest path π_{ij} between the centers of any two intersecting disks $D_i, D_j \in \mathcal{D}^*$. We then prove that the number of edges of $\mathcal{G}^\times(\mathcal{D}^*)$ must be $O(n^{8/5})$; otherwise there will be an intersection point of two shortest paths $\pi_{ij}, \pi_{k\ell}$ that has large ply, which is not possible due to the preprocessing step. Since the number of edges of $\mathcal{G}^\times(\mathcal{D}^*)$ is $O(n^{8/5})$ we can use the separator result on string graphs to obtain a separator of size $O(\sqrt{n^{8/5}}) = O(n^{4/5})$. Adding the cliques that were removed in the preprocessing step then yields a separator consisting of $O(n^{4/5})$ cliques. In the full version we show how to improve the bound to $O(n^{3/4+\varepsilon})$ using a bootstrapping scheme. In this EuroCG paper, however, we only prove the weaker bound of $O(n^{4/5})$.

Clique-based separators give sub-exponential algorithms for Maximum Independent Set, Feedback Vertex Set, and q-Coloring for constant q [4]. When using our clique-based separator, the running times for Maximum Independent Set and Feedback Vertex Set are inferior to what is known for string graphs. For q-Coloring with $q \geqslant 4$ this is not the case, since it does not admit a sub-exponential algorithm, assuming ETH [2]. Our clique-based separator, however, yields an algorithm for geodesic disks running in $2^{O(n^{4/5})}$ time, if the boundaries of the disks D_i can be computed in polynomial time. In the full version we also

show how to obtain an efficient distance-oracle for intersection graphs of geodesic disks.

2 A clique-based separator for geodesic disks in $\mathbb{R}^2$

Let d be a metric defined on a closed path-connected subset $F \subset \mathbb{R}^2$, and let $\mathcal{D} = \{D_1, \ldots, D_n\}$ be a set of geodesic disks in F, with respect to the metric d. Thus each disk D_i is defined as $D_i := \{q \in F : d(q, p_i) \leqslant r_i\}$, where $p_i \in F$ is the center of D_i an $r_i \geqslant 0$ is its radius. Let $\mathcal{D}_0 := \mathcal{D}$ and let $\mathcal{G}^\times(\mathcal{D}_0)$ denote the intersection graph of $\mathcal{D}_0$. We denote the set of edges of $\mathcal{G}^\times(\mathcal{D}_0)$ by E and define $m := |E|$.

2.1 Constructing the separator

We proceed in three steps: in a preprocessing step we reduce the ply of the set of disks we work with, in the second step we prove that if the ply is sublinear then the number of edges in the intersection graph is subquadratic, and in the third step we construct the separator.

Step 1: Reducing the ply. For a point $p \in F$, let $\mathcal{D}_0(p) := \{D_i \in \mathcal{D}_0 : p \in D_i\}$ be the set of disks from $\mathcal{D}_0$ containing p—note that $\mathcal{D}_0(p)$ forms a clique in $\mathcal{G}^\times(\mathcal{D}_0)$—and define $\text{ply}(p) := |\mathcal{D}_0(p)|$ to be the ply of p with respect to $\mathcal{D}_0$. The ply of the set $\mathcal{D}_0$ is defined as $\text{ply}(\mathcal{D}_0) := \max\{p \in F : \text{ply}(p)\}$.

We reduce the ply of $\mathcal{D}_0$ in the following greedy manner. Let α be a fixed constant with $0 < \alpha < 1$. In the basic construction we will use $\alpha = 1/5$, but in our bootstrapping scheme we will work with other values as well. We check whether there exists a point p such that $|\mathcal{D}_0(p)| \geqslant \frac{1}{4}n^\alpha$. If so, we remove $\mathcal{D}_0(p)$ from $\mathcal{D}_0$ and add it as a clique to $\mathcal{S}$. We repeat this process until $\text{ply}(\mathcal{D}_0) < \frac{1}{4}n^\alpha$. Thus in the first step at most $4n^{1-\alpha}$ cliques are added to $\mathcal{S}$.

To avoid confusion with our initial set $\mathcal{D}_0$, we denote the set of disks remaining at the end of Step 1 by $\mathcal{D}_1$ and we denote the set of edges of $\mathcal{G}^\times(\mathcal{D}_1)$ by E_1.

Step 2: Bounding the size of E_1. To bound the size of E_1, we draw a shortest path π_{ij} between the centers p_i, p_j of every two intersecting disks $D_i, D_j \in \mathcal{D}_1$, thus obtaining a geometric realization of the graph $\mathcal{G}^\times(\mathcal{D}_1)$. Slightly abusing notation, we will not distinguish between an edge (D_i, D_j) in $\mathcal{G}^\times(\mathcal{D}_1)$ and its geometric realization π_{ij}. Let $\Pi(\mathcal{D}_1) := \{\pi_{ij} : (D_i, D_j) \in E_1\}$ be the resulting set of paths. To focus on the main idea behind our proof we will assume that $\Pi(\mathcal{D}_1)$ is a *proper path set*: a set of paths such that any two paths $\pi_{ij}, \pi_{k\ell}$ have at most two points in common, each intersection point is either a shared endpoint or a proper crossing, and no proper crossing coincides with another proper crossing or with an endpoint. The proof also works without this assumption, given that the metric we are working with is well-behaved. We defer this discussion to the full version. We need the following result about the number of crossings in dense graphs, known as the Crossing Lemma [1, 8]. The term *planar drawing* here refers to a drawing where no edge interior passes through a vertex and all intersections are proper crossings. Thus it applies to a proper path set.

▶ **Lemma 2.1** (Crossing Lemma). *There exists a constant $c > 0$, such that every planar drawing of a graph with n vertices and $m \geqslant 4n$ edges contains at least $c\frac{m^3}{n^2}$ crossings.*

Using the Crossing Lemma we will show that $|E_1| = O(n^{\frac{3+\alpha}{2}})$, as follows. If $|E_1| = O(n^{\frac{3+\alpha}{2}})$ does not hold, then by the Crossing Lemma there must be many crossings between the edges $\pi_{ij} \in E_1$. We will show show that this implies that there is a crossing of ply greater than $\frac{1}{4}n^\alpha$, thus contradicting that $\text{ply}(\mathcal{D}_1) < \frac{1}{4}n^\alpha$. We now make this idea precise.

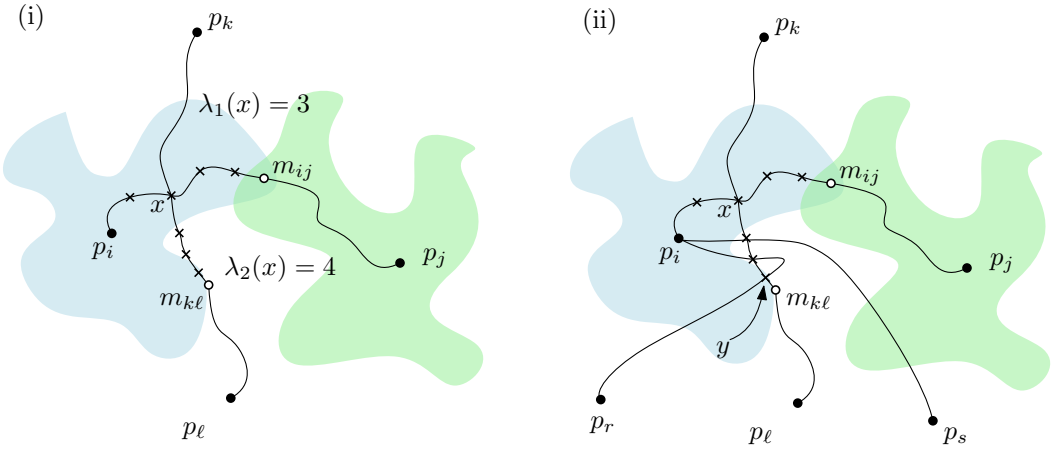


Figure 1 (i) A labeling of a crossing $x \in \mathcal{X}$. (ii) The crossing y is assigned to D_i four times.

▶ **Lemma 2.2.** *Let $\mathcal{G}^\times(\mathcal{D}_1) = (\mathcal{D}_1, E_1)$ be the intersection graph of a set $\mathcal{D}_1$ of disks such that $\text{ply}(\mathcal{D}_1) < \frac{1}{4}n^\alpha$. Then $|E_1| \leqslant \sqrt{\frac{4}{c}} \cdot n^{\frac{3+\alpha}{2}}$, where c is the constant appearing in the Crossing Lemma.*

Proof. Consider the proper path set $\Pi(\mathcal{D}_1)$ and let $\mathcal{X}$ be the set of crossings between the paths in $\Pi(\mathcal{D}_1)$. Assume for a contradiction that $|E_1| > \sqrt{\frac{4}{c}} \cdot n^{\frac{3+\alpha}{2}}$. We will show that then there has to exist a crossing $x \in \mathcal{X}$ of ply at least $\frac{n^\alpha}{4}$, which contradicts that $\text{ply}(\mathcal{D}_1) < \frac{1}{4}n^\alpha$.

We start by giving a lower bound on the total ply of all crossings in the drawing. To this end, we split each edge $\pi_{ij} \in E_1$ in two *half-edges* as follows. For two points $x, y \in \pi_{ij}$, let $\pi_{ij}[x, y]$ denote the subpath of π_{ij} between x and y. Recall that p_i is the center of disk D_i. We now pick an arbitrary point $m_{ij} \in \pi_{ij} \cap (D_i \cap D_j)$ and split π_{ij} at m_{ij} into a half-edge $\pi_{ij}[p_i, m_{ij}]$ connecting p_i to m_{ij} and a half-edge $\pi_{ij}[p_j, m_{ij}]$ connecting p_j to m_{ij}. For brevity, we will denote these two half-edges by h_{ij} and h_{ji}, respectively. Clearly each half-edge has length at most the radius of the disk it lies in, and so $h_{ij} \subset D_i$ and $h_{ji} \subset D_j$. We denote the resulting set of half-edges by $\mathcal{E}_1$.

We label each crossing $x \in \mathcal{X}$ with an unordered pair of integers $\{\lambda_1(x), \lambda_2(x)\}$, defined as follows: if x is the crossing between the half-edges $h_{ij}, h_{k\ell}$, then $\lambda_1(x)$ is the number of crossings contained in $\pi_{ij}[x, m_{ij}]$ and $\lambda_2(x)$ is the number of crossings contained in $\pi_{k\ell}[x, m_{k\ell}]$; see Fig. 1(i). This labeling is useful to obtain a rough bound on the total ply of all crossings, because of the following observation, which immediately follows from the triangle inequality.

Observation 1. Consider a crossing $x = h_{ij} \cap h_{k\ell}$. If $d(x, m_{k\ell}) \leqslant d(x, m_{ij})$ then all crossings $y \in \pi_{k\ell}[x, m_{k\ell}]$ are contained in D_i, and otherwise all crossings $y \in \pi_{ij}[x, m_{ij}]$ are contained in D_k.

Let $K := \sum_{x \in \mathcal{X}} \text{ply}(x)$ denote the total ply of all crossings. The following claim bounds K in terms of the labels $\{\lambda_1(x), \lambda_2(x)\}$.

Claim 1. $K > \frac{1}{2|\mathcal{D}_1|} \sum_{x \in \mathcal{X}} \min\{\lambda_1(x), \lambda_2(x)\}$.

Proof. Define $K(D_i) := \big|\{x \in \mathcal{X} : x \in D_i\}\big|$ to be the contribution of D_i to the total

ply K, and note that

$$K = \sum_{x \in \mathcal{X}} \text{ply}(x) = \sum_{x \in \mathcal{X}} \big|\{D_i \in \mathcal{D}_1 : x \in D_i\}\big| = \sum_{D_i \in \mathcal{D}_1} \big|\{x \in \mathcal{X} : x \in D_i\}\big| = \sum_{D_i \in \mathcal{D}_1} K(D_i).$$

Now consider a crossing $x = h_{ij} \cap h_{k\ell}$. If $d(x, m_{k\ell}) \leqslant d(x, m_{ij})$ then we assign x to D_i, and otherwise we assign x to D_k. Let $\mathcal{X}(D_i)$ be the set of crossings assigned to D_i. By Observation 1 and the definition of the label $\{\lambda_1(x), \lambda_2(x)\}$, the disk D_i contains at least $\min\{\lambda_1(x), \lambda_2(x)\}$ crossings $y \in \pi_{k\ell}[x, m_{k\ell}]$. Thus, summing over all crossings $x \in \mathcal{X}(D_i) \cap h_{ij}$ and all half-edges h_{ij} incident to D_i, we find that

$$K(D_i) \geqslant \frac{1}{2\deg(D_i)} \sum_{h_{ij}} \sum_{x \in \mathcal{X}(D_i) \cap h_{ij}} \min\{\lambda_1(x), \lambda_2(x)\} > \frac{1}{2|\mathcal{D}_1|} \sum_{x \in \mathcal{X}(D_i)} \min\{\lambda_1(x), \lambda_2(x)\},$$

where $\deg(D_i)$ denotes the degree of D_i in $\mathcal{G}^\times(\mathcal{D}_1)$. The factor $\frac{1}{2\deg(D_i)}$ arises because a crossing $y \in h_{k\ell}$ can be counted up to $2\deg(D_i)$ times in the expression $\sum_{x \in \mathcal{X}(D_i)} \min\{\lambda_1(x), \lambda_2(x)\}$, namely at most twice for every half-edge incident to D_i; see Fig. 1(ii). (Twice, because a pair of paths in a proper path set may cross twice.) Since each crossing is assigned to exactly one set $\mathcal{X}(D_i)$, we obtain

$$K = \sum_{D_i \in \mathcal{D}_1} K(D_i) > \sum_{D_i \in \mathcal{D}} \left(\frac{1}{2|\mathcal{D}_1|} \sum_{x \in \mathcal{X}(D_i)} \min\{\lambda_1(x), \lambda_2(x)\} \right) = \frac{1}{2|\mathcal{D}_1|} \sum_{x \in \mathcal{X}} \min\{\lambda_1(x), \lambda_2(x)\}.$$

◁

From the Crossing Lemma and our initial assumption that $|E_1| > \sqrt{\frac{4}{c}} \cdot n^{\frac{3+\alpha}{2}}$, we have that

$$|\mathcal{X}| \geqslant c\frac{|E_1|^3}{n^2} > 4|E_1|n^{1+\alpha} = 2|\mathcal{E}_1|n^{1+\alpha}. \tag{1}$$

In order to get a rough bound for $\sum_{x \in \mathcal{X}} \min\{\lambda_1(x), \lambda_2(x)\}$, we will ignore crossings with small labels, while ensuring that we don't ignore too many crossings in total. More precisely, for every half-edge h_{ij}, we disregard its first $n^{1+\alpha}$ crossings, starting from the one closest to m_{ij}. We let $\mathcal{X}^*$ denote the set of remaining crossings. Note that $|\mathcal{X}^*| \geqslant |\mathcal{X}| - |\mathcal{E}_1|n^{1+\alpha}$, and $\min\{\lambda_1(x), \lambda_2(x)\} \geqslant n^{1+\alpha}$ for every $x \in \mathcal{X}^*$. Therefore

$$K > \frac{1}{2|\mathcal{D}_1|} \sum_{x \in \mathcal{X}^*} \min\{\lambda_1(x), \lambda_2(x)\} = \frac{1}{2|\mathcal{D}_1|} \cdot |\mathcal{X}^*| \cdot n^{1+\alpha} \geqslant \frac{\big(|\mathcal{X}| - |\mathcal{E}_1|n^{1+\alpha}\big)\, n^{1+\alpha}}{2|\mathcal{D}_1|} \geqslant \frac{|\mathcal{X}|}{4} n^\alpha.$$

This means that there exists a crossing $x \in \mathcal{X}$ of ply at least $\frac{1}{4}n^\alpha$, which contradicts the condition of the lemma and thus finishes the proof. ◀

Step 3: Applying the separator theorem for string graphs. Lee's separator theorem for string graphs [7] states that any string graph on m edges admits a balanced separator of size $O(\sqrt{m})$. It is well known that any intersection graph of path-connected sets in the plane is a string graph. Hence, we can apply Lee's result to $\mathcal{G}^\times(\mathcal{D}_1)$ which has $O(n^{\frac{3+\alpha}{2}})$ edges. Thus, $\mathcal{G}^\times(\mathcal{D}_1)$ has a separator of size $O(n^{\frac{3+\alpha}{4}})$. By adding the vertices of this separator as singletons to our separator $\mathcal{S}$ then, together with the cliques added in Step 1, we obtain a separator consisting of $O(n^{\frac{3+\alpha}{4}} + n^{1-\alpha})$ cliques. Picking $\alpha = 1/5$, and anticipating the extension to the case where $\Pi(\mathcal{D}_1)$ is not a proper path set, we obtain the following result.

▶ **Theorem 2.3.** *Let d be a well-behaved shortest-path metric on a closed and path-connected subset $F \subset \mathbb{R}^2$ and let $\mathcal{D}$ be a set of n geodesic disks with respect to the metric d. Then $\mathcal{G}^\times(\mathcal{D})$ has a balanced clique-based separator consisting of $O(n^{4/5})$ cliques.*

De Berg *et al.* [4] showed that if a class of graphs admits a clique-based separator consisting of $S(n)$ cliques, and the separator can be constructed in polynomial time, then one can solve q-Coloring in $2^{O(S(n))}$ time. Note that if we can compute the boundaries ∂D_i in polynomial time,[1] then we can also compute our separator in polynomial time. Indeed, we can then compute $\mathcal{G}^\times(\mathcal{D})$ and the arrangement of the disk boundaries in polynomial time, which allows us to do Step 1 (reducing the ply) in polynomial time. Since a separator for string graphs can be computed in polynomial time [7], this is easily seen to imply that the separator construction runs in polynomial time. Thus we obtain the following result.

▶ **Corollary 2.4.** *Let d be a shortest-path metric on a connected subset $F \subset \mathbb{R}^2$ and let $\mathcal{D}$ be a set of n geodesic disks with respect to the metric d, where d is such that the boundaries ∂D_i of the disks in $\mathcal{D}$ can be computed in polynomial time. Let $q \geqslant 1$ and $\varepsilon > 0$ be fixed constants. Then q-Coloring can be solved in $2^{O(n^{\frac{4}{5}})}$ time on $\mathcal{G}^\times(\mathcal{D})$.*

References

1. M. Ajtai, V. Chvátal, M.M. Newborn, and E. Szemerédi. Crossing-free subgraphs. In Peter L. Hammer, Alexander Rosa, Gert Sabidussi, and Jean Turgeon, editors, *Theory and Practice of Combinatorics*, volume 60 of *North-Holland Mathematics Studies*, pages 9–12. North-Holland, 1982. doi:doi.org/10.1016/S0304-0208(08)73484-4.
2. Édouard Bonnet and Pawel Rzazewski. Optimality program in segment and string graphs. *Algorithmica*, 81(7):3047–3073, 2019. doi:10.1007/s00453-019-00568-7.
3. Mark de Berg, Hans L. Bodlaender, Sándor Kisfaludi-Bak, Dániel Marx, and Tom C. van der Zanden. A framework for Exponential-Time-Hypothesis-tight algorithms and lower bounds in geometric intersection graphs. *SIAM J. Comput.*, 49:1291–1331, 2020. doi:10.1137/20M1320870.
4. Mark de Berg, Sándor Kisfaludi-Bak, Morteza Monemizadeh, and Leonidas Theocharous. Clique-based separators for geometric intersection graphs. *Algorithmica*, 85(6):1652–1678, 2023. doi:10.1007/S00453-022-01041-8.
5. Hristo Djidjev and Shankar M. Venkatesan. Reduced constants for simple cycle graph separation. *Acta Informatica*, 34(3):231–243, 1997. doi:10.1007/s002360050082.
6. Jacob Fox and János Pach. Separator theorems and Turán-type results for planar intersection graphs. *Advances in Mathematics*, 219(3):1070–1080, 2008. doi:doi.org/10.1016/j.aim.2008.06.002.
7. James R. Lee. Separators in region intersection graphs. In *8th Innovations in Theoretical Computer Science Conference, ITCS 2017*, volume 67 of *LIPIcs*, pages 1:1–1:8, 2017. doi:10.4230/LIPIcs.ITCS.2017.1.
8. Thomas Leighton. *Complexity Issues in VLSI*. Foundations of Computing Series. MIT Press, 2003.
9. Richard J. Lipton and Robert Endre Tarjan. A separator theorem for planar graphs. *SIAM J. Appl. Math*, 36(2):177–189, 1977. doi:doi/10.1137/0136016.
10. William Thurston. *The geometry and topology of 3-manifolds*. Princeton Lecture Notes, 1978–1981.

[1] Typically the time to do this would not only depend on n, but also on the complexity of F and the distance function d. For simplicity we state our results in terms of n only. (This, of course, puts restrictions on the complexity of F and d.)

On k-Plane Insertion into Plane Drawings

Julia Katheder[1], Philipp Kindermann[2], Fabian Klute[3], Irene Parada[3], and Ignaz Rutter[4]

1 **Wilhelm-Schickard-Institut für Informatik, Universität Tübingen, Germany**
julia.katheder@uni-tuebingen.de
2 **FB IV – Informatikwissenschaften, Universität Trier, Germany**
kindermann@uni-trier.de
3 **Universitat Politècnica de Catalunya, Spain**
fabian.klute@upc.edu, irene.parada@upc.edu
4 **Faculty of Computer Science and Mathematics, University of Passau, Germany**
rutter@fim.uni-passau.de

Abstract

We introduce the k-Plane Insertion into Plane drawing (k-PIP) problem: given a plane drawing of a planar graph G and a set of edges F, insert the edges in F into the drawing such that the resulting drawing is k-plane. In this paper, we focus on the 1-PIP scenario. We present a linear-time algorithm for the case that G is a triangulation, while proving NP-completeness for the case that G is biconnected and F forms a path or a matching.

1 Introduction

Inserting edges into planar graphs is a long-studied problem in Graph Drawing. Most commonly, the goal is to find a way to insert the edges while minimizing the number of crossings and maintaining the planarity of the prescribed subgraph. This problem is a core step in the planarization method to find graph drawings with few crossings [22]. Gutwenger et al. [15] have studied the case of a single edge. For multiple edges the picture is more complicated. In case the edges are all incident to one vertex previously not present in the graph, the problem can again be solved in polynomial time [7]. However, the problem is NP-hard even when the given drawing is fixed and its corresponding graph biconnected [23, 25]. Assuming a fixed drawing, Hamm and Hliněný presented an FPT-algorithm parameterized by the number of crossings [16]. Finally, Chimani and Hliněný [8] gave an FPT-algorithm for the fixed and variable embedding settings with the number of inserted edges as a parameter.

In this paper, we take a slightly different viewpoint and do not restrict the overall number of created crossings, but instead their structure. Moreover, we focus on the case when the drawing of the given planar graph is fixed. Then our goal is, given a plane drawing Γ and a set F of edges with its endpoints being vertices of this graph, to find a k-plane drawing containing Γ as a subdrawing plus the edges of F. Here, a *k-plane drawing* of a graph is one in which no edge is crossed more than k times. The class of *k-planar graphs*, which are those admitting a k-plane drawing, is widely studied in Graph Drawing [10, 17].

▶ Problem 1 (k-Plane Insertion into Plane drawing (k-PIP)). Given a plane drawing Γ of a graph $G = (V, E)$ and a set of edges F with endpoints in V, find a k-plane drawing of the graph $(V, E \cup F)$ that contains Γ as a subdrawing.

Our results. In this paper, we focus on 1-PIP. Note that we may assume that a solution to an instance of 1-PIP is a simple 1-plane drawing, i.e., no two edges share more than one point, since in a 1-plane drawing simplicity only affects the crossings of adjacent edges.

40th European Workshop on Computational Geometry, Ioannina, Greece, March 13–15, 2024.
This is an extended abstract of a presentation given at EuroCG'24. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

In Section 2, we present an $O(|V|)$ algorithm for the case that G is a triangulation. To accomplish this, we first reduce the number of possible ways one edge can be inserted into the given drawing to at most two per edge in F and then use a 2-SAT formulation to compute a solution if possible. In Section 3, we show that 1-PIP is NP-complete even if G is biconnected and the edges in F form a path or a matching.

Related work. k-PIP is related to the problem of extending a partial drawing of a graph to a drawing of the full graph. Usually, the goal in such problems is to maintain certain properties of the given drawing. For example, in works by Angelini et al. [1], Eiben et al. [13, 12], Ganian et al. [14], or Arroyo et al. [3] the input is a plane, 1-plane, k-plane, or simple drawing, respectively, and the desired extension must maintain the property of being plane, 1-plane, k-plane, or simple. Restrictions of the drawing such as it being straight-line [24], level-planar [4], upward [20], or orthogonal [2] have been explored. Other results consider the number of bends [6] or assume that the partially drawn subgraph is a cycle [5, 21].

2 Extending a triangulation

We assume standard notation and concepts from graph theory; compare, e.g., [11]. Given an instance (G, Γ, F) of 1-PIP, an edge $e \in F$ might be inserted into Γ in different ways. Note that e cannot be inserted without crossings in a triangulation. An *option* for e is an edge γ of G such that e can be inserted into Γ crossing only γ. Note that in a triangulation, a pair of faces uniquely defines an edge γ that must be crossed if e is inserted into said pair of faces. Thus, with the term option we might also refer to a pair of faces. An option for e is *safe* if, in case the instance admits a solution, there is one solution in which e is inserted according to this option. Two options for two edges e and e' of F *clash* if inserting e and e' according to these options violates 1-planarity. Examples of safe options are those of edges with a single option and an option without clashes. An immediate solution can be found if each edge in F has a non-clashing option. However, it is not sufficient for each edge in F to have a safe option in order to find a solution, e.g. in the case that two single options are clashing. Finally, observe that in a triangulation, each edge of Γ can only be an option for one edge of F and clashes with at most four other options.

▶ **Theorem 2.1.** *Given a plane drawing Γ of a triangulation $G = (V, E)$ and a set of edges F with endpoints in V, 1-PIP on (G, Γ, F) can be solved in time $O(|V|)$.*

Proof. The idea is to preprocess the instance until we are left with a set of edges $F' \subseteq F$ with two options each. The resulting instance can then be solved using a 2SAT formula. Begin by computing all options for every $e \in F$. Since Γ is a plane drawing, we can get the triangles incident to each $v \in V$ in cyclic order and also the options for edges in F incident to v. This way, we get the overall $O(|V|)$ options for edges in F in $O(|V|)$ time. For an edge $(u, v) \in F$, $u, v \in V$, with two or more options we say that two options are *consecutive* if the corresponding faces are consecutive in the cyclic order around u (or v); see the options for (u, v) in Fig. 1(c) for an illustration. We say a set of options is *cyclically consecutive* if the corresponding edges induce a cycle in G; see the options for (u, v) in Fig. 1(d).

Whenever an edge e has no option left, we stop and output `no` and if e has exactly one option left, we insert it into Γ. Every time we insert an edge, we need to remove at most four options of other edges plus all the options of the just inserted edge. Consider an edge $e = (u, v) \in F$, $u, v \in V$, that has three or more options. We consider four cases.

(a) There are at least three options for e such that no two of them are consecutive. Given one of these three options, say σ_i, we claim that it is either safe or never possible in a

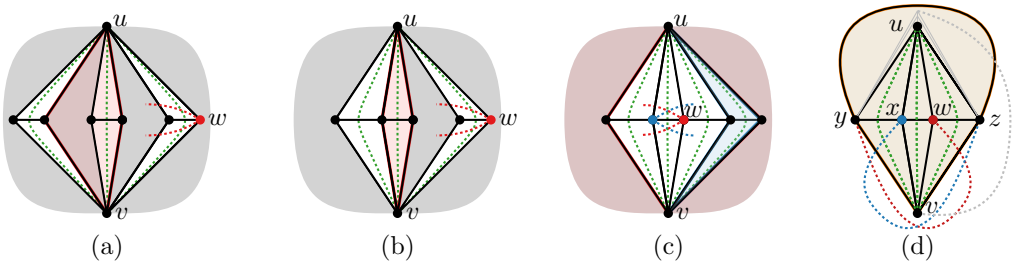


Figure 1 Cases with three or more options in a triangulation.

solution; see Fig. 1(a). If σ_i is not clashing with any other option, it is safe and we add it. Otherwise, let w and x be the two endpoints of σ_i. Option σ_i can only be clashing with two options for edges in F incident to w and two options for edges in F incident to x. Moreover, any option for those edges clashes with σ_i. To see this, consider w.l.o.g. such an edge in F incident to w (illustrated in red in Fig. 1(a)). The other options for e define a non-empty region in which this edge cannot be drawn (see red region in Fig. 1(a)), restricting its options to options that clash with σ_i.

(b) There are at least three options for e, and one of them, σ_i, is not consecutive to any of the other two; see Fig. 1(b). As in Case (a), σ_i is either safe or never possible.

(c) There are at least four consecutive non-cyclic options for e; see Fig. 1(c). Let σ_i be one of the central options. As in the previous cases, σ_i is either safe or never possible.

(d) There are three consecutive or four cyclically consecutive options for e; see Fig. 1(d). Consider the middle option σ_i (or any option if there were four). If it is safe, we just add it. Else, let w and x be the endpoints of σ_i and y, z the other endpoints of options for e. Assume, w.l.o.g., that σ_i clashes with an option of an edge e_w incident to w and to vertex y. For σ_i to be a possible option in a solution, e_w must have an option that does not clash with it. There is only one possibility, and it implies that v, y, z or u, y, z form a triangle. Assume, w.l.o.g., the former, so (y, z) is an edge in Γ. Let $\diamond$ be the set of vertices $\{u, v, w, x, y, z\}$ and $G_\diamond$ the octahedron subgraph of G induced by $\diamond$.
Edges in F with exactly one endpoint in $\diamond \setminus \{u\}$ have at most one option. Thus, we can insert them first and see whether we are still in Case (d). Edges incident to u and to a vertex not in $\diamond$ cannot clash with any option of an edge between vertices in $\diamond$. Thus, we can solve the constant-size subinstance consisting of inserting such edges into $G_\diamond$ independently, taking into account the single-option edges that we might have inserted.

Once each edge has exactly two options we create a 2SAT formula containing one variable per option and clauses that ensure exactly one option per edge in F' and exclude clashes. This formula has size $O(|V|)$ and is satisfiable iff the original instance has a solution. ◀

3 Inserting a path or a matching is NP-complete

We prove NP-hardness by reduction from Planar Monotone 3-SAT; the membership of the problem in NP is straightforward. Let ϕ be a Boolean formula in CNF with variables $V = \{x_1, \ldots, x_n\}$ and clauses $C = \{c_1, \ldots, c_m\}$. Each clause has at most three literals and is either *positive* (all literals are positive) or *negative* (all literals are negative). Furthermore, there is a rectilinear representation Γ_ϕ of the variable-clause incidence graph of ϕ, such that all variables and clauses are depicted as axis-aligned rectangles connected via vertical segments and all variables are positioned on the x-axis, all positive clauses lie above and all

negative clauses lie below the x-axis; see Fig. 2 for an example. This problem is known to be NP-complete [9, 19].

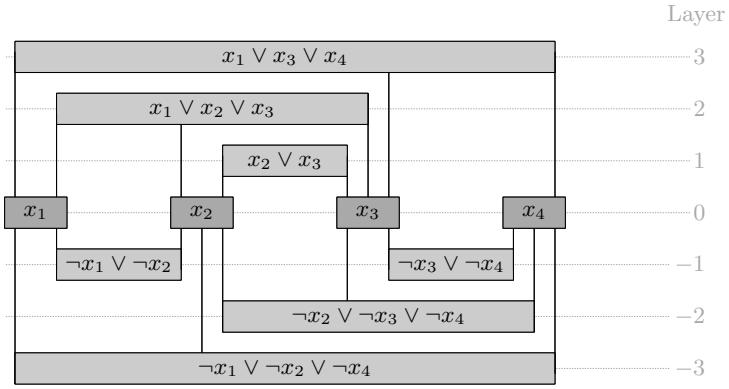


Figure 2 Rectilinear representation of the variable-clause incidence graph of a PLANAR MONOTONE 3-SAT instance.

In the following, starting from Γ_ϕ, we construct a graph $G = (V, E)$, its plane drawing Γ, and the edge set F, which will be inserted into Γ in a specific way. We start with the case of F forming a path (see Theorem 3.4) and describe the changes to our construction for F being a matching afterwards (see Corollary 3.5). The bars in Γ_ϕ can be layered decreasingly from top to bottom. We set the layer of the variables as layer zero and denote by $L(c)$ the layer of clause c; see Fig. 2. We denote by H the graph K_4 missing one arbitrary edge and create chains of copies of H, that are connected via their degree 2 vertices, such that the chord is drawn inside the C_4 cycle of H. This provides the structure for routing the path F in the drawing. We say that an edge $e \in F$ is *ℓ-spanning* if there are ℓ chords of H in the chain between its endpoints.

The variable gadget. We replace each bar of a variable x in Γ_ϕ by a variable gadget which consists of a H-chain of $4a + 1$ copies where a is the maximum over the number of positive and negative occurrences of x in ϕ. Let $u_1, \ldots, u_{4a+2}$ be the vertices where the H copies are joined from left- to rightmost copy. Moreover, we mark for $i \in \{0, \ldots, a-1\}$ the vertices u_{3+4i} as *variable endpoints* (c.f. the squares in Fig. 3). Each such vertex gets two additional incident edges, called *literal edges*, which are going to connect the variable gadgets to adjacent layers and encode the truth-value of the respective variables. We call a literal edge exiting its variable endpoint upwards (downwards) *positive* (*negative*).

For each $i \in \{0, \ldots a-1\}$, we define the path F to pass through u_{1+4i}, u_{4+4i}, u_{3+4i}, u_{6+4i}, u_{5+4i}, except for $a-1$ where we omit the last vertex; cf. red path in Fig. 3. Then, for a variable gadget its edges in F consist of an alternation of 3-spanning and 1-spanning edges. By 1-planarity and simplicity, F cannot cross the H-chain. We refer to the 3-spanning edges that cross literal edges as *blocking 3-spanners* and the others as *forcing 3-spanners*.

For the remainder, we depict literal edges representing the value `true` in blue and the ones representing `false` in orange, while the edges in F are colored in red; c.f. Figs. 3 to 6. The proofs of statements marked with a ($\star$) are available in the long version on arXiv [18].

▶ **Lemma 3.1** ($\star$). *Let v be a variable gadget described as above, then in any 1-planar drawing containing it, its literal edges, and the edges $F_v \subseteq F$ incident to vertices in v either all negative or all positive literal edges are crossed.*

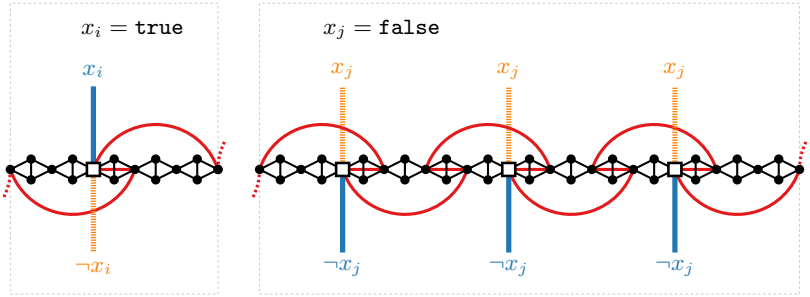


Figure 3 Drawing of the variable gadget.

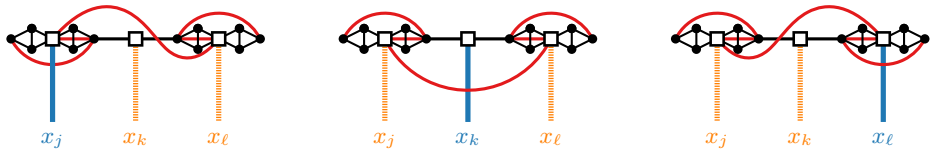


Figure 4 Drawing of the clause gadget.

If the negative literal edges are crossed, we think of the variable corresponding to the gadget as set to `true` and `false` otherwise. We connect the variable gadgets by adding one copy of H with a 1-spanning edge added to F in between them; see Fig. 5.

The clause gadget. We describe the construction only for the positive clauses, as the construction for the negative ones is symmetric. The clause gadget is depicted in Fig. 4. It consists of a chain of two copies of H, followed by two edges, followed by two more copies of H. We mark the middle vertices of each of the two copies and the two edges as *variable endpoints* and add one additional edge to them, their *literal edge*. Assume that all literal edges are drawn on the same side as shown in Fig. 4 and add edges to F as shown in red.

▶ **Lemma 3.2** (⋆). *Let c be a clause gadget drawn as describe above, then in any 1-planar drawing containing c, its literal edges, and the edges $F_c \subseteq F$ incident to vertices of c, at least one literal edge has to be crossed by an edge in F_c.*

Propagating the variable state. In order to ensure that there is no interaction between the path F and other parts of the drawing, we insert H-chains with 1-spanning edges added to F on every layer of Γ_ϕ and insert the clause gadgets into the respective layers as shown in Fig. 5. We create further variable endpoints on the layers $i > 0$, in order to propagate the state of the variable gadgets to clauses in higher layers. For each pair of corresponding variable endpoints of a variable gadget and clause gadget, we create a variable endpoint at a merged vertex in the H-chain in each layer i with $0 < i < L(c)$ and prescribe F to span the two neighboring copies of H. Further, we connect every two consecutive variable endpoints on layer j and $j+1$ with $0 \leq j < L(c)$ via a literal edge, as illustrated in Fig. 5.

▶ **Lemma 3.3** (⋆). *Let $P = e_1, \ldots, e_{L(c)}$ be a path of literal edges such that e_1 is incident to a variable endpoint of variable gadget v and $e_{L(c)}$ to one clause gadget c, if e_1 is crossed in v, then $e_{L(c)}$ is crossed by an edge of F incident to vertices on layer $L(c) - 1$.*

Note that if the first edge of P is not crossed in the variable gadget, this does not imply that its last edge is uncrossed in layer $L(c)-1$. In fact, this is possible when multiple literals

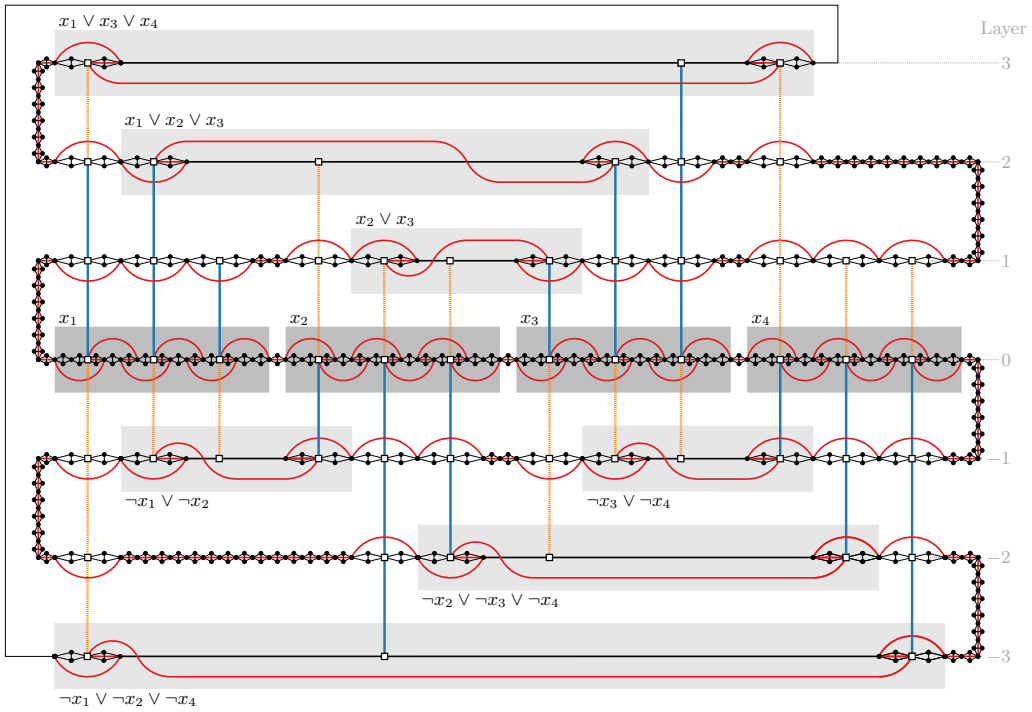


Figure 5 Solution (in red) of the 1-PIP instance coming from the graph given in Fig. 2.

evaluate to `true` for a clause gadget; e.g. the top- and leftmost orange edge in Fig. 5. If a variable is not contained in the same number of negative and positive clauses we ensure the alternating pattern of F on layer zero by connecting the remaining variable endpoints to ones on layer one, e.g., see x_1 in Fig. 5. Lastly, the subpath of F on every layer is joined to the one of the previous layer in an alternating fashion by an H-chain and 1-spanning edges.

▶ **Theorem 3.4** ($\star$). *1-PIP is* NP*-complete even if G is biconnected and F forms a path.*

We can use the same construction but replacing the alternating connections between the layers by edges to prove NP-hardness also for the case that F is a matching; see Fig. 6.

▶ **Corollary 3.5.** *1-PIP is* NP*-complete even if G is biconnected and F forms a matching.*

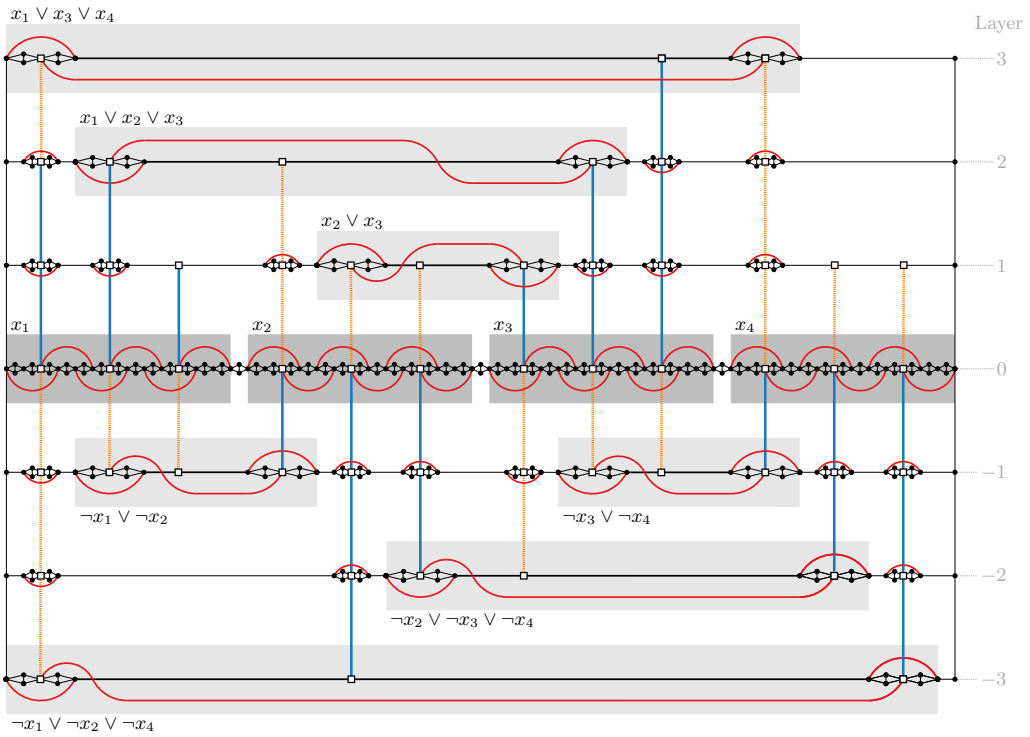


Figure 6 1-PIP instance where F is a matching reduced from the graph given in Fig. 2

4 Conclusion

We introduced the k-PIP problem and showed that it is NP-complete even when the given graph is biconnected and the inserted edges form a path or matching. We also presented a linear-time algorithm when the given graph is triangulated. This naturally raises the question if the triconnected case of k-PIP is polynomial-time solvable or NP-complete.

References

1 Patrizio Angelini, Giuseppe Di Battista, Fabrizio Frati, Vít Jelínek, Jan Kratochvíl, Maurizio Patrignani, and Ignaz Rutter. Testing planarity of partially embedded graphs. *ACM Transactions on Algorithms*, 11(4):32:1–32:42, 2015. doi:10.1145/2629341.

2 Patrizio Angelini, Ignaz Rutter, and Sandhya T. P. Extending partial orthogonal drawings. In *Proceedings of the 28th International Symposium on Graph Drawing and Network Visualization (GD'20)*, volume 12590 of *LNCS*, page 265–278. Springer, 2020. doi:10.1007/978-3-030-68766-3_21.

3 Alan Arroyo, Fabian Klute, Irene Parada, Birgit Vogtenhuber, Raimund Seidel, and Tilo Wiedera. Inserting one edge into a simple drawing is hard. *Discrete & Computational Geometry*, 69(3):745–770, 2023. doi:10.1007/S00454-022-00394-9.

4 Guido Brückner and Ignaz Rutter. Partial and constrained level planarity. In *Proceedings of the 28th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'17)*, page 2000–2011. SIAM, 2017. doi:10.1137/1.9781611974782.130.

5 Erin W. Chambers, David Eppstein, Michael T. Goodrich, and Maarten Löffler. Drawing graphs in the plane with a prescribed outer face and polynomial area. *Journal of Graph Algorithms and Applications*, 16(2):243–259, 2012. doi:10.7155/jgaa.00257.

6 Timothy M. Chan, Fabrizio Frati, Carsten Gutwenger, Anna Lubiw, Petra Mutzel, and Marcus Schaefer. Drawing partially embedded and simultaneously planar graphs. *Journal of Graph Algorithms and Applications*, 19(2):681–706, 2015. doi:10.7155/jgaa.00375.

7 Markus Chimani, Carsten Gutwenger, Petra Mutzel, and Christian Wolf. Inserting a vertex into a planar graph. In Claire Mathieu, editor, *Proceedings of the 20th Annual ACM-SIAM Symposium on Discrete Algorithms, (SODA'09)*, page 375–383. SIAM, 2009. doi:10.1137/1.9781611973068.42.

8 Markus Chimani and Petr Hlinený. Inserting multiple edges into a planar graph. *Journal of Graph Algorithms and Applications*, 27(6):489–522, 2023. doi:10.7155/JGAA.00631.

9 Mark de Berg and Amirali Khosravi. Optimal binary space partitions for segments in the plane. *International Journal of Computational Geometry and Applications*, 22(3):187–206, 2012. doi:10.1142/S0218195912500045.

10 Walter Didimo, Giuseppe Liotta, and Fabrizio Montecchiani. A survey on graph drawing beyond planarity. *ACM Computing Surveys*, 52(1):4:1–4:37, 2019. doi:10.1145/3301281.

11 Reinhard Diestel. *Graph Theory, 5th Edition*, volume 173 of *Graduate texts in mathematics*. Springer, 2012. doi:10.1007/978-3-662-53622-3.

12 Eduard Eiben, Robert Ganian, Thekla Hamm, Fabian Klute, and Martin Nöllenburg. Extending nearly complete 1-planar drawings in polynomial time. In *Proceedings of the 45th International Symposium on Mathematical Foundations of Computer Science, (MFCS'20)*, volume 170 of *LIPIcs*, page 31:1–31:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.MFCS.2020.31.

13 Eduard Eiben, Robert Ganian, Thekla Hamm, Fabian Klute, and Martin Nöllenburg. Extending partial 1-planar drawings. In *Proceedings of the 47th International Colloquium on Automata, Languages, and Programming (ICALP'20)*, volume 168 of *LIPIcs*, page 43:1–43:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.ICALP.2020.43.

14 Robert Ganian, Thekla Hamm, Fabian Klute, Irene Parada, and Birgit Vogtenhuber. Crossing-optimal extension of simple drawings. In Nikhil Bansal, Emanuela Merelli, and James Worrell, editors, *48th International Colloquium on Automata, Languages, and Programming, ICALP 2021, July 12-16, 2021, Glasgow, Scotland (Virtual Conference)*, volume 198 of *LIPIcs*, page 72:1–72:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPICS.ICALP.2021.72.

15 Carsten Gutwenger, Petra Mutzel, and René Weiskircher. Inserting an edge into a planar graph. *Algorithmica*, 41(4):289–308, 2005. doi:10.1007/S00453-004-1128-8.

16 Thekla Hamm and Petr Hliněný. Parameterised partially-predrawn crossing number. In *Proceedings of the 38th International Symposium on Computational Geometry (SoCG'22)*, volume 224 of *LIPIcs*, page 46:1–46:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.SoCG.2022.46.

17 Seok-Hee Hong and Takeshi Tokuyama, editors. *Beyond Planar Graphs, Communications of NII Shonan Meetings*. Springer, 2020. doi:10.1007/978-981-15-6533-5.

18 Julia Katheder, Philipp Kindermann, Fabian Klute, Irene Parada, and Ignaz Rutter. On k-plane insertion into plane drawings. arXiv preprint, 2024. URL: https://arxiv.org/abs/2402.14552, arXiv:2402.14552.

19 Donald E. Knuth and Arvind Raghunathan. The problem of compatible representatives. *SIAM Journal on Discrete Mathematics*, 5(3):422–427, 1992. doi:10.1137/0405033.

20 Giordano Da Lozzo, Giuseppe Di Battista, and Fabrizio Frati. Extending upward planar graph drawings. *Computational Geometry: Theory and Applications*, 91:101668, 2020. doi:10.1016/j.comgeo.2020.101668.

21 Tamara Mchedlidze, Martin Nöllenburg, and Ignaz Rutter. Extending convex partial drawings of graphs. *Algorithmica*, 76(1):47–67, 2016. doi:10.1007/s00453-015-0018-6.

22 Petra Mutzel and Thomas Ziegler. The constrained crossing minimization problem. In Jan Kratochvíl, editor, *Proceedings of the 7th International Symposium on Graph Drawing (GD'99)*, volume 1731 of *Lecture Notes in Computer Science*, page 175–185. Springer, 1999. doi:10.1007/3-540-46648-7_18.

23 Petra Mutzel and Thomas Ziegler. The constrained crossing minimization problem a first approach. In *Proceedings of the 1999 International Conference on Operations Research*, page 125–134. Springer, 1999. doi:10.1007/978-3-642-58409-1_11.

24 Maurizio Patrignani. On extending a partial straight-line drawing. *International Journal of Foundations of Computer Science*, 17(5):1061–1070, 2006. doi:10.1142/S0129054106004261.

25 Thomas Ziegler. *Crossing minimization in automatic graph drawing*. PhD thesis, Saarland University, Saarbrücken, Germany, 2001.

Delaunay Triangulation and Convex Polygons with Predictions*†

Sergio Cabello[1] and Panos Giannopoulos[2]

1 **Faculty of Mathematics and Physics, University of Ljubljana, Slovenia**
Institute of Mathematics, Physics and Mechanics, Slovenia
sergio.cabello@fmf.uni-lj.si

2 **Department of Computer Science, City, University of London, UK**
Panos.Giannopoulos@city.ac.uk

Abstract

We show that given a triangulation T of a set P of n points in the plane, the Delaunay triangulation $DT(P)$ of P can be built in $O(n + k \log^4 k)$ time, where k is the number of edges of $DT(P)$ not present in T. We also show that several problems about convex polygons can be solved in $O(\log k)$ time, when the vertices of the polygons are given in an array and we are given a pointer to vertices that in the array are at distance at most k from the vertices defining the solution.

1 Introduction

We give algorithms for some fundamental geometric problems under the so-called 'predictions' paradigm; see Mitzenmacher and Vassilvitskii [9] for an overview. Given a problem Π and a prediction P for a solution of an instance of Π, we would like to devise an algorithm that computes a solution S in time at most $f(n, d(P, S))$, where n the size of the instance and d is an appropriate measure of how close the prediction is to the actual solution. Roughly speaking, we would like f to be such that when d is small (good prediction) the running time is better than the best known time complexity, and when d is large (bad prediction) the running time approaches that of the best known algorithms, or at least it does not get much worse. A usual example here is binary search in a sorted array with a given starting position; the binary search can be done in $O(1 + \log k)$ steps using exponential search, where k is the distance in the array between the starting and the final position of the search.

Our purpose is to initiate the study of problems in Computational Geometry under the lens of algorithms with predictions. As an example, consider the classical problem of computing the closest pair in a set P of n points in $\mathbb{R}^d$, where d is constant. Assume that the prediction is a pair of points $(p, q) \in P^2$, $p \neq q$. We can then decompose the space into cube cells of diagonal length $|pq|$ and assign each point of P to the cube cell that contains it. If δ^* is the distance of the closest pair, then inside each cube cell there are at most $O((|pq|/\delta^*)^d)$ points. We then iterate over the points in P and, for each point $p_i \in P$, check the distance from p_i to the points that are in the $O(1)$ cube cells that may have some point at distance at most $|pq|$ from p_i. In total, it takes $O(n \cdot (|pq|/\delta^*)^d)$ time, assuming the floor function is

* Research funded in part by the Slovenian Research and Innovation Agency (P1-0297, J1-2452, N1-0218, N1-0285). Research funded in part by the European Union (ERC, KARST, project number 101071836). Views and opinions expressed are however those of the authors only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them.

† The authors would like to thank Kostas Tsakalidis for suggesting the area of algorithmic problems with predictions.

40th European Workshop on Computational Geometry, Ioannina, Greece, March 13–15, 2024.
This is an extended abstract of a presentation given at EuroCG'24. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

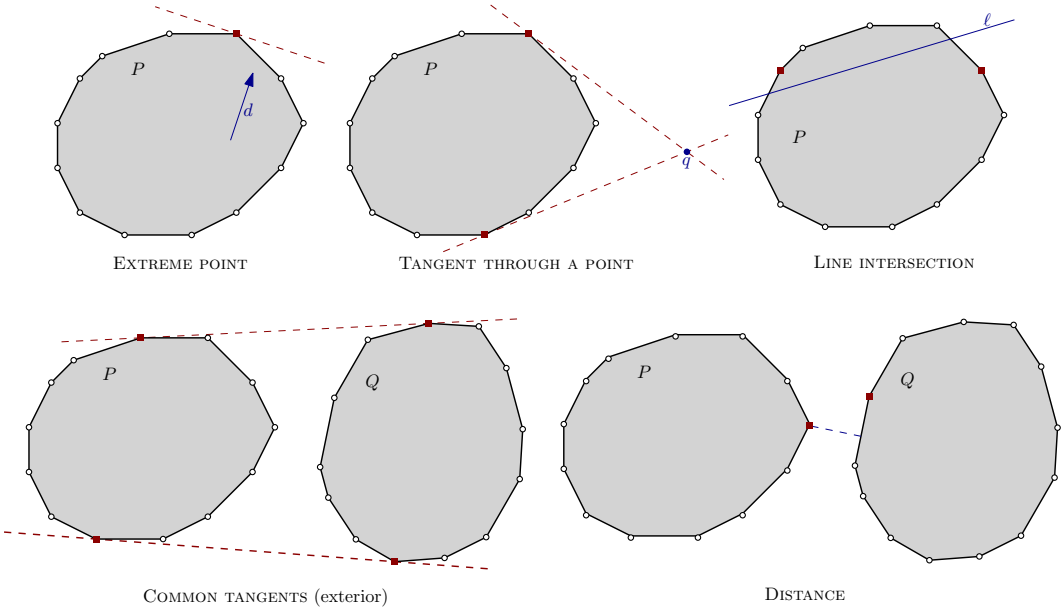


Figure 1 Problems considered in Section 2. In all cases the vertices defining the optimum are marked with red squares.

available. When the prediction is good, i.e., $|pq|/\delta^*$ is constant, the running time will be linear. As the prediction deteriorates, the running time may become quadratic and it is better to turn to other algorithms.

A geometric result that falls in this paradigm is the data structure of Iacono and Langerman [8] for point location in the plane. They consider the problem of storing a plane triangulation for point location with a finger: preprocess the triangulation such that, for a given point p and a finger to a triangle Δ of the triangulation, we can locate the triangle Δ_p containing p in $O(\log d(\Delta, \Delta_p))$ time, where d is a distance-like metric in the subdivision.

In this paper, we provide the following results:

- Several problems for convex polygons described by an array of vertices and with some vertices as prediction can be solved in $O(1 + \log k)$, where k is the distance in the array of the prediction to the vertices describing the solution.
- Given a triangulation T of a planar n-point set in the plane, we can compute its Delaunay triangulation in $O(n + k \log^4 k)$ time, where k is the number of non-Delaunay edges in T.
- Computing the Delaunay triangulation of P from a given triangulation of P requires $\Omega(n \log n)$ operations.

2 Convex polygons

We reconsider early computational geometry problems on convex polygons given by an array under the lens of predictions. Here, predictions are indices of the array that, in the case of a good prediction, are close to the indices of the elements that define the solution object.

Let P and Q be two polygons with m and n vertices respectively, described by the arrays $P[0, \ldots, m-1]$ and $Q[0, \ldots, n-1]$ containing their vertices in counterclockwise order. For any two indices $0 \leqslant i, j \leqslant m-1$ for P, let $d_m(i, j) = \min\{|i-j|, |i+m-j|, |i-j-m|\}$ be their cyclic distance. The cyclic distance $d_n(\cdot, \cdot)$ for Q is defined similarly.

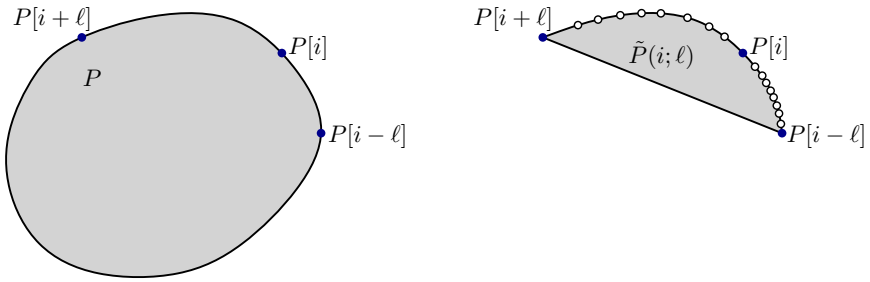


Figure 2 The polygon $\tilde{P}(i,\ell)$.

We consider the following problems with predictions; see Figure 1. In all cases we use k to measure the distance between the prediction and the vertices defining the optimal solution.

- Extreme point: Given P and a direction d, find a extremal vertex of P in the direction d. The prediction is an index i and the measure is

$$k \;=\; \min\{d_m(i,i^*) \mid P[i^*] \text{ extreme point of } P \text{ in direction } d\}.$$

- Tangent through a point: Given P and a point q external to P, find the tangents from q to P. The prediction is two indices i_1 and i_2 and the measure is

$$k \;=\; \min\{d_m(i_1,i_1^*) + d_m(i_2,i_2^*) \mid qP[i_1^*] \text{ and } qP[i_2^*] \text{ define the tangents from } q \text{ to } P\}.$$

- Line intersection: Given P and a line ℓ, find the intersections of ℓ with the boundary of P. The prediction is two indices i_1 and i_2 and the measure is

$$k \;=\; \min\{d_m(i_1,i_1^*)+d_m(i_2,i_2^*) \mid i_1^* \neq i_2^*,\ \ell \text{ intersects } P[i_1^*]P[i_1^*+1] \text{ and } P[i_2^*]P[i_2^*+1]\}.$$

- Common tangents: Given P and Q that are disjoint, find the common exterior/interior tangents of P and Q. For the exterior tangents, the prediction is two indices $i_1, i_2 \in \{0,\ldots,m-1\}$ and two indices $j_1, j_2 \in \{0,\ldots,n-1\}$ and the measure is

$$k \;=\; \min\{\sum_{t=1,2} (d_m(i_t,i_t^*) + d_n(j_t,j_t^*)) \mid P[i_1^*]Q[j_1^*] \text{ and } P[i_2^*]Q[j_2^*] \text{ define the exterior tangents of } P \text{ and } Q\}.$$

  For the interior tangents the definition is similar.
- Distance: Given P and Q, find the minimum distance between P and Q. The prediction is two indices $i_1 \in \{0,\ldots,m-1\}$ and $j_1 \in \{0,\ldots,n-1\}$ and the measure is

$$k \;=\; \min\{d_m(i_1,i_1^*)+d_n(j_1,j_1^*) \mid \text{the distance between } P[i_1^*]P[i_1^*+1] \text{ and } Q[j_1^*]Q[j_1^*+1] \text{ defines the distance between } P \text{ and } Q\}.$$

▶ **Theorem 2.1.** Extreme point, Tangent through a point, Line intersection, Common tangents *and* Distance *with predictions can be solved in* $O(\log k)$ *time.*

Proof sketch. The idea is to perform a doubly exponential search on the indices. For index i and value ℓ, let $\tilde{P}(i;\ell)$ be the portion of the polygon with vertices $P[i-\ell,\ldots,i+\ell]$, where indices are taken cyclically; see Figure 2. Let $\tilde{P}(i_1,i_2;\ell)$ be the polygon obtained by joining $\tilde{P}(i_1;\ell)$ and $\tilde{P}(i_2;\ell)$. Each of $\tilde{P}(i_1;\ell)$ and $\tilde{P}(i_1,i_2;\ell)$ have at most $O(\ell)$ vertices.

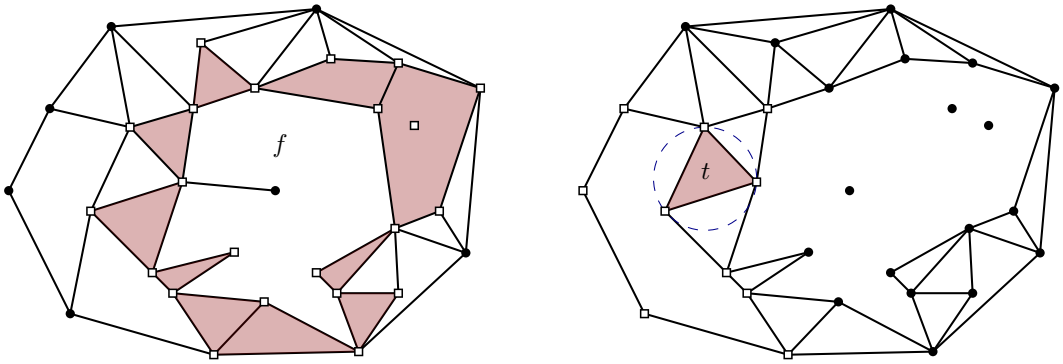


Figure 3 Left: $N_G(f)$ are shaded for the face f, and the points $V(N_G(f))$ are marked with squares. Right: $V(N_G(t))$ are marked with squares for the shaded triangle t.

We start with $\ell = 2$ and use the standard algorithm for the subpolygons $\tilde{P}(i_1;\ell)$ or $\tilde{P}(i_1,i_2;\ell)$, and/or $\tilde{Q}(j_1;\ell)$ or $\tilde{Q}(j_1,j_2;\ell)$, depending on the problem [2, 6, 7, 10]. We get a candidate solution for the subpolygons in $O(\log \ell)$ time, which we can test for optimality for the whole P and Q in constant time. If it is optimal, we have finished, otherwise we square the value of ℓ and repeat. The values of ℓ follow the sequence 2^{2^i} for $i = 0, 1, 2, \ldots, \lceil \log_2 \log_2 k \rceil$ and therefore the total running time is $\sum_{i=0}^{\lceil \log_2 \log_2 k \rceil} O\left(\log(2^{2^i})\right) = O(\log k)$. ◀

3 Delaunay triangulation

A *plane straight-line graph*, shortened to *PSLG*, is a planar embedding of a graph where all edges are drawn as straight-line edges. A *triangulation* of a planar point set P is a connected PSLG such that: (i) its vertex set is precisely P, (ii) all its bounded faces are triangles, and (iii) the unbounded face is the complement of the convex hull $CH(P)$. We remark that a triangle with an extra vertex inside, possibly isolated, is not a triangular face of a PSLG.

Let P be a set of n points in the plane. For simplicity, we assume that the points in P are in general position, i.e., no four points are co-circular and no three points are collinear. For a triangle t, let $C(t)$ be its circumcircle, and for $\{p,q,r\} \in \binom{P}{3}$, let $C(p,q,r) = C(\triangle(p,q,r))$.

The Delaunay triangulation of P, denoted by $DT(P)$, can be defined as follows: for any three distinct points $p,q,r \in P$, the triangle $\triangle(p,q,r)$ belongs to $DT(P)$ if and only if the circle $C(p,q,r)$ has no point of P in its interior. The Delaunay triangulation of a point set in general position is unique. We assume familiarity with $DT(P)$; see [4, Chapter 9].

Let G be a PSLG and f be a face of G. Let $N_G(f)$ be the set of neighbor faces of f in the dual graph of G and $V(N_G(f))$ be the vertex set of the faces of $N_G(f)$. Any isolated vertices in the interior of a face of $N_G(f)$ belong also to $V(N_G(f))$; see Figure 3.

A standard property of $DT(P)$ is that it can be tested locally: a triangulation T of P is $DT(P)$ if and only if, for each triangle $t = \triangle(p_i,p_j,p_k)$ of T it holds that no point $p \in V(N_G(t))$ is in the interior of the circle $C(p_i,p_j,p_k)$. We will need the following extension to identify subgraphs of $DT(P)$. See Figure 3, right, for the hypothesis.

▶ **Lemma 3.1.** *Let G be a PSLG with vertex set P and such that each edge of G is on the boundary of $CH(P)$ or adjacent to a triangular face. Assume that, for each triangle t of G, all the points of $V(N_G(t))$ are outside $C(t)$. Then each edge of G is an edge of $DT(P)$.*

Our algorithm will identify and delete an appropriate subset of the edges of a given triangulation of P. For this, we will need the following bounds.

▶ **Lemma 3.2.** *Let T be a triangulation of P and let E' be a subset of $E(T)$ with k edges. Remove from T the edges of E' and then remove also all edges that have bounded, non-triangular faces on both sides. In total, we remove $O(k)$ edges.*

We will use the following data structures that readily follow from Chan [1]. For the circles, one uses the lift to the paraboloid and point-plane duality in $\mathbb{R}^3$.

▶ **Lemma 3.3.** *We can maintain a dynamic set P of n points in the plane such that:*

- *insertion or deletion of a point takes $O(\log^4 n)$ amortized time;*
- *for a query circle C, we can detect whether some point of P is inside C in $O(\log^2 n)$ time.*

We can maintain a dynamic set $\mathcal{C}$ of n circles in the plane such that:

- *insertion or deletion of a circle takes $O(\log^4 n)$ amortized time;*
- *for a query point p, we can detect whether some circle of $\mathcal{C}$ contains p in $O(\log^2 n)$ time.*

We next show our main result, where we assume that the prediction is an arbitrary PSLG.

▶ **Theorem 3.4.** *Assume that we have a set P of n points in general position in the plane and we are given a PSLG G with vertex set P. Let k be the (unknown) number of edges of $DT(P)$ that are not in G. In $O(n + k \log^4 k)$ time we can compute $DT(P)$.*

Proof sketch. The first step is identifying a subset E' of $O(k)$ edges of G in $O(n + k \log^4 k)$ time such that all edges of $G - E'$ are from the Delaunay triangulation $DT(P)$. The subset E' may contain also some edges of $DT(P)$; thus, we compute a superset of the "wrong" edges. The criteria to identify E' is that $G - E'$ should satisfy the assumptions of Lemma 3.1. There are two basic ideas that are being used for this:

- Test circumcircles of triangular faces to see whether they contain some point of a neighbor face. The same triangle may have to be checked multiple times because its neighbor faces may change, and we employ data structures to do this efficiently.
- If an edge is on the boundary of two non-triangular faces, we have no triangle to test the edge. Then, we just add the edge to E'; by Lemma 3.2, this can happen $O(k)$ times.

For any subgraph G' of G, let $F_{\geqslant 4}(G')$ be the faces of G' that are non-triangular and bounded. At the start, $F_{\geqslant 4}(G)$ has $O(k)$ vertices and edges; even if G would be a subgraph of $DT(P)$, we still have to add k edges to G to obtain $DT(P)$.

We start with $E' = \emptyset$ and the PSLG $G' = G$, and maintain the invariant that $G = G' + E'$. Thus, E' is the set of edges that are being removed. Let Q be the set of points that lie in the faces of $F_{\geqslant 4}(G')$. We maintain Q in the first data structure of Lemma 3.3. We also maintain the set τ of triangles that share an edge with a face of $F_{\geqslant 4}(G')$. For each triangle of τ, its circumcircle is stored in the second data structure of Lemma 3.3.

We iterate over the triangles t of G' and check whether $C(t)$ contains some point of $V(N_{G'}(t))$. The adjacent triangles are checked explicitly, while the adjacent non-triangular faces are checked by querying Q with $C(t)$. If $C(t)$ contains some point of $V(N_{G'}(t))$, we delete *all* edges of t from G' and add them to E'. This makes a new non-triangular face of G'. We have to update Q, τ and the data structures that store them; there are $O(1)$ new elements. The circumcircles of the old triangles of τ may contain some new point of Q, and the new triangles of τ (adjacent to the new face of G') have to be retested. See Figure 4. This can be done efficiently using $O(1)$ queries and editions in the data structures. More precisely, for each of the $O(1)$ new triangles in τ we check whether they contain some point

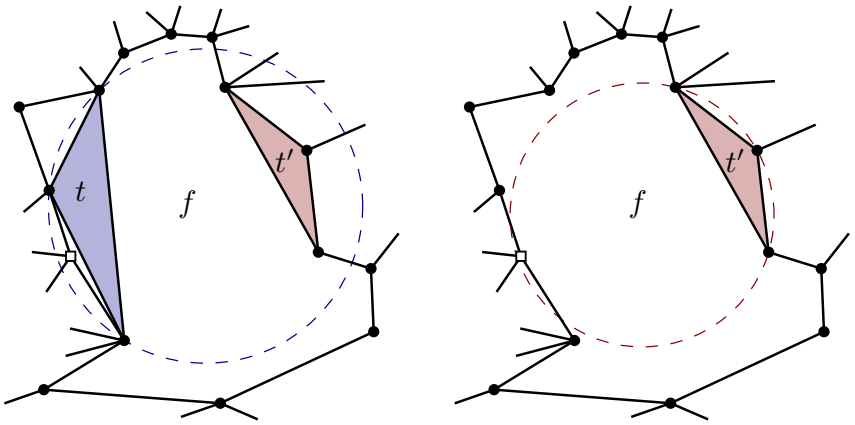


Figure 4 When we check t, we decide to delete $E(t)$ because $C(t)$ contains some point of $V(f) \subseteq Q$. The point marked with an empty square becomes a new point of Q that is in the interior of $C(t')$.

of Q, and for each of the $O(1)$ new points in Q we check whether it is contained in some circumcircle of $\{C(t') \mid t' \in \tau\}$.

For each triangle we delete, at least one of its edges is not in $DT(P)$. Therefore, we delete at most $3k$ edges. Each triangle that is deleted triggers $O(1)$ additional operations in the data structures of Lemma 3.3 storing Q and τ and triggers that $O(1)$ new triangles of τ have to be tested again. Finally, we remove from the final G' all the edges that have on both sides non-triangular faces, and add them to E'. This does not change the set Q nor τ, and therefore we keep having that, for each triangle t of G', $C(t)$ contains no point of $V(N_{G'}(t))$. Because of Lemma 3.2, we have deleted $O(3k) = O(k)$ additional edges.

We have obtained $E' \subseteq E(G)$ such that $|E'| = O(k)$ and Lemma 3.1 applies to $G - E'$. Thus all edges of $G - E'$ are edges of $DT(P)$. We then compute the constrained Delaunay triangulation for the faces of $G - E'$ using the algorithm of Chew [3] in $O(k \log k)$ time. ◀

▶ **Corollary 3.5.** *Assume that we have a set P of n points in general position in the plane and we are given a triangulation T of P. Let k be the (unknown) number of edges of T that are not in $DT(P)$. In $O(n + k \log^4 k)$ time we can compute $DT(P)$.*

Finally we can easily show the following lower bound.

▶ **Theorem 3.6.** *Computing $DT(P)$ for a set P of n points in the plane from a triangulation of P takes $\Omega(n \log n)$ time in the decision tree model of computation.*

4 Conclusions

We conjecture that the Delaunay triangulation $DT(P)$ can be computed from any other triangulation T of P in $O(n + k \log k)$ time, where k is the number of edges of $DT(P)$ that are not in T.

References

1 Timothy M. Chan. Dynamic geometric data structures via shallow cuttings. *Discret. Comput. Geom.*, 64(4):1235–1252, 2020. doi:10.1007/s00454-020-00229-5.

2 Bernard Chazelle and David P. Dobkin. Intersection of convex objects in two and three dimensions. *J. ACM*, 34(1):1–27, 1987. doi:10.1145/7531.24036.

3 L. Paul Chew. Constrained Delaunay triangulations. *Algorithmica*, 4(1):97–108, 1989. doi:10.1007/BF01553881.

4 Mark de Berg, Otfried Cheong, Marc J. van Kreveld, and Mark H. Overmars. *Computational geometry: algorithms and applications, 3rd Edition*. Springer, 2008.

5 Hristo N. Djidjev and Andrzej Lingas. On computing Voronoi diagrams for sorted point sets. *Int. J. Comput. Geom. Appl.*, 5(3):327–337, 1995. doi:10.1142/S0218195995000192.

6 Herbert Edelsbrunner. Computing the extreme distances between two convex polygons. *J. Algorithms*, 6(2):213–224, 1985. doi:10.1016/0196-6774(85)90039-2.

7 Leonidas J. Guibas, John Hershberger, and Jack Snoeyink. Compact interval trees: a data structure for convex hulls. *Int. J. Comput. Geom. Appl.*, 1(1):1–22, 1991. doi:10.1142/S0218195991000025.

8 John Iacono and Stefan Langerman. Proximate planar point location. In *Proceedings of the 19th ACM Symposium on Computational Geometry, 2003*, pages 220–226. ACM, 2003. doi:10.1145/777792.777826.

9 Michael Mitzenmacher and Sergei Vassilvitskii. Algorithms with predictions. In Tim Roughgarden, editor, *Beyond the Worst-Case Analysis of Algorithms*, pages 646–662. Cambridge University Press, 2020. doi:10.1017/9781108637435.037.

10 Mark H. Overmars and Jan van Leeuwen. Maintenance of configurations in the plane. *J. Comput. Syst. Sci.*, 23(2):166–204, 1981. doi:10.1016/0022-0000(81)90012-X.

2-Coloring Point Guards in a k-Guarded Polygon*

Stephane Durocher[1], Myroslav Kryven[1], Fengyi Liu[1], Amirhossein Mashghdoust[1], and Ikaro Penha Costa[1]

1 **Department of Computer Science, University of Manitoba**
{stephane.durocher,myroslav.kryven,fengyi.liu,amirhossein.mashghdoust,
ikaro.penhacosta}@umanitoba.ca

Abstract

For every $k \geq 2$, we describe how to construct a polygon P and a set G of points in P, such that P is *k-guarded* by G (i.e., every point in P is visible to at least k points in G) and for every *2-coloring* of G (i.e., for every bipartition of G) at least one of the colors does not guard P. This answers an open question posed by Morin [10].

1 Introduction

The *art gallery problem*, introduced by Klee [11] in 1973, is a well-known and extensively studied classical problem in the field of Computational Geometry. Given a simple polygon P (without holes) in the plane, the objective is to find a set G of points in P, called *guards*, such that every point $p \in P$ is *visible* to at least one guard $g \in G$; that is, the line segment $\overline{pg}$ does not pass outside P. Chvátal [6] showed that $\lfloor n/3 \rfloor$ guards suffice to guard any n-vertex simple polygon P, and that there exist polygons that require $\lfloor n/3 \rfloor$ guards. Fisk [7] later gave a simplified proof (one of the *Proofs from THE BOOK* [2]) using a 3-coloring argument. The optimization problem of finding a set G of points of minimum cardinality that guards a given simple polygon P is NP-hard [8], and was recently shown to be $\exists\mathbb{R}$-complete [1].

To introduce robustness and redundancy to the model, the art gallery problem generalizes to the *k-guarding problem*, in which each point in the input polygon P must be visible to at least k guards. Belleville et al. [3] examined a variant of k-guarding, in which guards are placed at the interior of the edges of P. Salleh [12] studied k-guarding with the constraint that guards are placed on the vertices of P, called *k-vertex guarding*. Salleh showed that $\lfloor 2n/3 \rfloor$ guards are sometimes necessary when $k = 2$, and $\lfloor 3n/4 \rfloor$ guards are sometimes necessary when $k = 3$ (see also [9]). Bereg [4] showed that Fisk's coloring argument can be used to prove these bounds. The k-guarding problem has also been studied from an algorithmic perspective; Busto et al. [5] gave a polynomial-time $O(k \log\log OPT_k(P))$-approximation algorithm for the k-guarding problem, where $OPT_k(P)$ is the optimal number of guards. As observed by Busto et al., if guards must be placed at different vertices of P, then there exist simple polygons that cannot be k-vertex guarded for $k \geq 4$ because some points in P are seen by fewer than k vertices. In k-guarding, this problem is naturally resolved by placing multiple guards arbitrarily close to each other.

During the open problem session at WADS 2023, Morin [10] asked whether there exists a positive integer k such that for all polygons P and all sets G of points that k-guard P, there exists a bipartition of G (equivalently, a 2-coloring of G) that gives two sets that each guard (1-guard) P. Morin presented counter-examples for $k = 2$ and $k = 3$ for which no such bipartition exists (see Figure 1) and asked whether this property generalizes to higher

* This work was funded in part by the Natural Sciences and Engineering Council of Canada (NSERC).

40th European Workshop on Computational Geometry, Ioannina, Greece, March 13–15, 2024.
This is an extended abstract of a presentation given at EuroCG'24. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

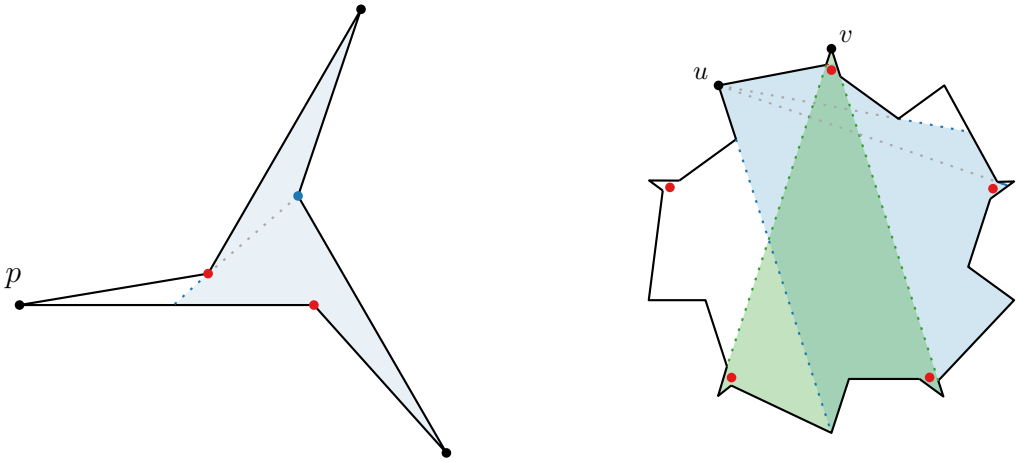


Figure 1 Examples for $k = 2$ and $k = 3$ [10]. The polygon P on the left is 2-guarded by the set G of three guards (red and blue points). Any 2-coloring of G partitions G either into 3 and 0, or 1 and 2. In both cases, at least one of the three convex vertices of P is not seen by any guard of the color with fewer guards. In this example, the blue guard, whose visibility region is shaded blue, cannot see the vertex p [10]. The polygon P' on the right is 3-guarded by the set G' of five guards (red points). There are $\binom{5}{3} = 10$ subsets of G' of cardinality three. Observe that each of these 10 subsets uniquely 3-guards exactly one of the 10 convex vertices of P'. E.g., the vertex v is 3-guarded by the three guards that are not consecutive on the boundary of P' in the visibility region shaded green, whereas the vertex u is 3-guarded by the three guards that are consecutive on the boundary of P' in the visibility region shaded blue. Any 2-coloring of G will result in one color class containing at most two guards. Consequently, some convex vertex of P is visible only by guards of the same color [10].

values of k. We answer this question in this paper. Observe that for any set G_1 that guards P, k copies of G_1 k-guard P and can be partitioned into k sets (and, therefore, into two sets) that each guard P. Consequently, Morin's question asks whether *every* set G that k-guards P can be partitioned into two sets that each guard P.

We formally define k-guarding as considered in this paper.

▶ **Definition 1.1** (k-guarding)**.** Given a simple polygon P, an integer $k \geq 1$, and a set G of points (guards) in P, P is *k-guarded* by G if for all $p \in P$, there exists $G' \subseteq G$, such that $|G'| = k$ and for all $g \in G'$, the line segment $\overline{gp}$ does not pass outside P. That is, every point in P is visible to at least k guards in G.

We say that the set of guards G is *2-colorable* if there exists a bipartition of G that partitions G into two sets such that each 1-guards P. The notions of k-guardability and 2-colorability characterize the degree to which a set G of guards sees the polygon P. Intuitively, a larger value of k should increase the probability that a set G of guards that k-guards a polygon is 2-colorable. We show that it is not the case in general. For every $k \geq 2$, we describe (see Section 2) how to construct a polygon P and a set G of guards, such that G k-guards P, but G is not 2-colorable.

Before presenting details of our construction, we first introduce some helpful definitions.

▶ **Definition 1.2.** A *k-ary tree* is a tree in which every non-leaf vertex has exactly k children.

▶ **Definition 1.3.** Given a simple polygon P and a set G of guards in P, a region $R \subseteq P$ is *uniquely guarded* by $G' \subseteq G$ if every point in R is visible (relative to P) to every guard in G', and there exists a point in R that is not visible (relative to P) to any guard in $G \setminus G'$.

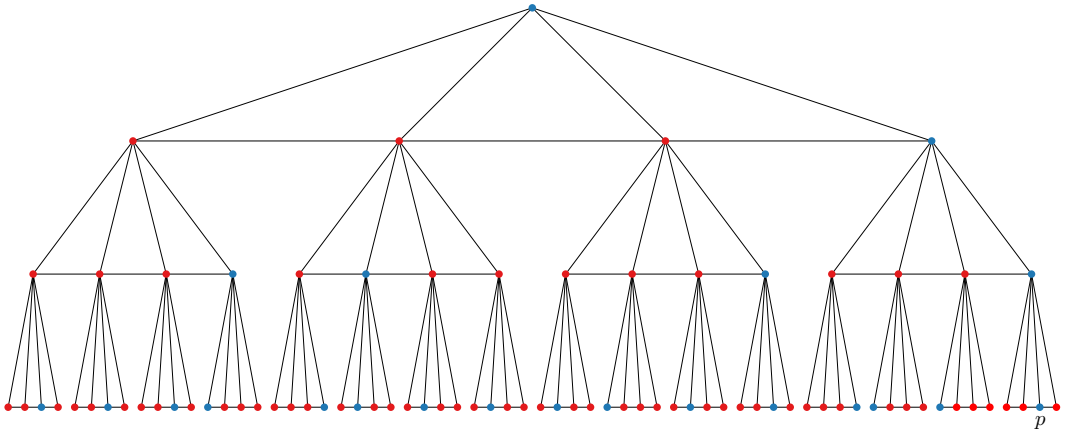


Figure 2 Proof idea. Our construction embeds a set G of guards in a polygon P_k, where G forms a perfect k-ary tree T_k of height $k-1$. Every path from root to leaf in T_k is a set of k guards in G that has an associated uniquely guarded region in P_k. Similarly, every set of siblings in T_k is a set of k guards in G that has an associated uniquely guarded region in P_k. Consequently, every set of siblings must include at least one node of each color. Therefore, there exists a monochromatic path from the root node to some leaf node. In this example, $k = 4$ and the path from the root to node p is monochromatic.

▶ **Definition 1.4.** For a simple polygon P, a set G of guards in P, and a region $R \subseteq P$ uniquely guarded by $G' \subseteq G$, we call a point in R that is only visible to G' a *witness point*, and a region composed of witness points a *witness region*.

2 Guards of a k-Guarded Polygon Are Not Always 2-Colorable

In this section, we prove our main result. The key idea is sketched in Figure 2, then proved formally in Lemma 2.1 and Theorem 2.2.

▶ **Lemma 2.1.** *For any $k \geq 1$, there exists a polygon P_k and a set G of guards in P_k that form a perfect k-ary tree T_k of height $k-1$ such that:*

1. *The polygon P_k is k-guarded by G.*
2. *For every root-to-leaf path in T_k, the points of G on that path uniquely guard some region of P_k.*
3. *For each internal node of T_k, its children uniquely guard some region of P_k.*

Proof. We will prove existence of a polygon, Π_k, defined below, that satisfies Properties 1–3 above. Consider a polygon Π_h with a set of guards G arranged in Π_h as a perfect k-ary tree of height $h-1$ (guards in each level are aligned horizontally, see Figure 4a) such that:

A The polygon Π_h is h-guarded by G.

B For every root-to-leaf path $g_{v_r} g_v$ (i.e. the path from the root v_r with the guard g_{v_r} to the leaf v with the guard g_v in the tree), the points (guards) of G on that path uniquely guard a convex region Q_v of Π_h with a witness triangle $\Delta_v = A_v B_v C_v$ of Q_v such that:

(1) Δ_v does not contain any of the guards;
(2) B_v is the bottommost point of Q_v;
(3) $A_v \in K_l B_v$, where K_l is the first vertex on Q_v after B_v clockwise;
(4) $C_v \in B_v K_r$, where K_r is the first vertex on Q_v after B_v counter clockwise.

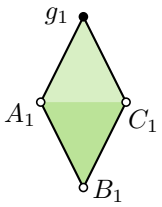
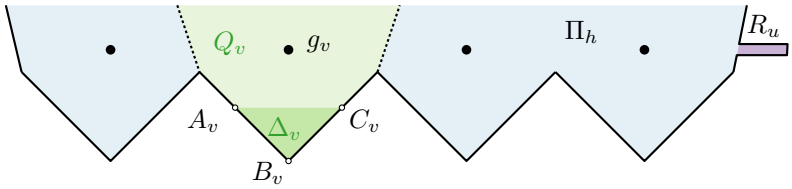
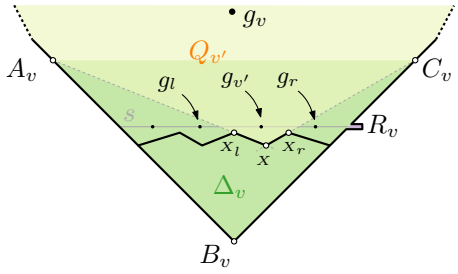


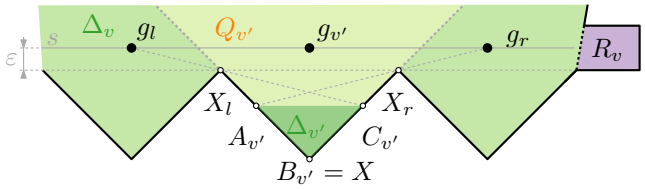
Figure 3 Base case, Π_1



(a) the polygon Π_h



(b) the region Q_v with the witness triangle Δ_v



(c) extending Q_v to $Q_{v'}$ with the witness triangle $\Delta_{v'}$

Figure 4 Illustration in support of the proof of Lemma 2.1

C For each internal node g_u in the tree, the children of g_u uniquely guard a trapezoidal region R_u.

Observe that any polygon P and set G of guards that satisfy Properties A–C also satisfy Properties 1–3. In what follows, we show how to construct Π_k by induction.

Base case. Let Π_1 be a diamond polygon with a single guard g_1 at its topmost vertex; see Figure 3. The entire polygon Π_1 is uniquely guarded by a single guard g_1, that defines a perfect k-ary tree of height 0. Therefore, Properties A–C are trivially satisfied.

Induction step. Now we show how to extend the polygon Π_h to Π_{h+1} so that Properties A–C hold. Place k guards on a horizontal segment s strictly contained in Δ_v; see Figure 4b. For a new guard $g_{v'}$, we reshape Q_v by drawing rays from A_v and C_v that cross at some point X in Δ_v below s. We ensure that A_vX crosses s between $g_{v'}$ and the guard g_l immediately to the left, and that C_vX crosses s between $g_{v'}$ and the guard g_r immediately to the right. Let $Q_{v'}$ denote the convex polygon obtained from Q_v by adding the edges A_vX and C_vX and

removing away from Q_v the parts that are below these two edges. Let X_lX and XX_r be the new edges forming Π_{h+1} by placing $X_l \in A_vX$ and $X_r \in C_vX$ right below s (a sufficiently small distance $\varepsilon > 0$); see Figure 4c.

We let $\Delta_{v'} = A_{v'}B_{v'}C_{v'}$, where $B_{v'} = X$, $A_{v'}$ is the point where the ray from g_r through X_r hits $X_lB_{v'}$, and $C_{v'}$ is the point where the ray from g_l through X_l hits $X_rB_{v'}$; see Figure 4c.

Let us show that Property B is satisfied. First, observe that all the guards on the root-to-leaf path $g_{v_r}g_{v'}$ are contained in the convex region $Q_{v'}$ (this holds, because by induction the guards on the root-to-leaf path $g_{v_r}g_v$ are inside Q_v, $Q_{v'} \subset Q_v$, and the guard $g_{v'}$ is inside $Q_{v'}$); therefore, all the guards on the root-to-leaf path $g_{v_r}g_{v'}$ see $Q_{v'}$. Second, notice that $\Delta_{v'} \subset \Delta_v$; therefore, no guards from the previous levels (guarding Π_h), except the root-to-leaf path $g_{v_r}g_v$ and $g_{v'}$ can see $\Delta_{v'}$. Let us show that out of the new guards (added at level $h+1$) only $g_{v'}$ can see $\Delta_{v'}$. Observe that all these guards are arranged horizontally and $\Delta_{v'}$ is contained below the line through g_l (that is, a guard immediately to the left of $g_{v'}$) and $C_{v'}$, that is, an endpoint of $\Delta_{v'}$. Therefore, $\Delta_{v'}$ is not seen by g_l, nor by any guard left of g_l. By an analogous argument, $\Delta_{v'}$ is not seen by g_r, nor by any guard right of g_r. Therefore, $\Delta_{v'}$ is a witness triangle of $Q_{v'}$ guarded by the root-to-leaf path $g_{v_r}g_{v'}$. Properties B.(1)–B.(4) are satisfied by construction with the vertices $B_{v'}$, $A_{v'}$, and $C_{v'}$ respectively; see Figure 4c.

To satisfy Property C, we make a trapezoidal pocket R_v of height 2ε and width $\delta(\varepsilon)$ aligned with s (so that every point of the pocket is visible to the children of g_v) on the right side of B_vC_v; see Figures 4b and 4c. For sufficiently small ε, the width $\delta(\varepsilon)$ of R_v can be made arbitrarily small, so that it does not interfere with the rest of the polygon Π_h and the right end of R_v is only seen to the guards that are children of g_v.

Finally, to see that Property A is satisfied (that is, that Π_h is h-guarded) observe that every point of the polygon is either contained in at least one convex region Q_v that contains h guards or it is contained in some trapezoidal pocket R_v that is seen by $k \geq h$ children of g_v. ◀

▶ **Theorem 2.2.** *There exists a polygon P and a set of guards G such that P is k-guarded by G but there is no 2-coloring of G.*

Proof. Consider a k-guarded polygon P_k from Lemma 2.1 with a set of guards G embedded in P_k as a perfect k-ary tree T_k of height $k-1$. Suppose there exists a 2-coloring of G. For each internal node g_u in T_k, the children of g_u uniquely guard some region of P_k. Since G is 2-colored, this set of siblings must include at least one blue guard and at least one red guard. Suppose, without loss of generality, that the root is colored blue. Therefore, there is a root-to-leaf path $g_{v_r}g_v$ that follows only the blue guards. According to Property 2, that path is uniquely guarding some region of P_k, and, therefore, there is a point in P_k that is only seen by blue guards, contradicting our assumption that there exists a 2-coloring of G. ◀

3 Directions for Future Research

We conclude with some open questions.

In the construction of polygon P_k in the proof of Lemma 2.1, the ratio of the lengths of the longest edge and the shortest edge is exponential in k. Consequently, we ask the following questions.

▶ Question 1. Is there a polygon P that is k-guarded by a set of guards G that is not 2-colorable for which the ratio of the lengths of the longest edge and the shortest edge is polynomial in k?

Is there a simpler construction than P_k? For example, does there exist a *weakly visible* polygon P (that is, every point of P is visible from some point on a given line segment in P) such that P is k-guarded by some set G of guards, but no bipartition of G exists such that each part guards P?

▶ Question 2. Is there a weakly visible polygon P that is k-guarded by a set of guards G that is not 2-colorable?

We can also examine the complexity (number of vertices) of P_k in terms of k. Our construction for P_k has $\Theta(k^k)$ vertices.

▶ Question 3. Can we show that P_k always needs $\omega(k)$ vertices?

References

1 Mikkel Abrahamsen, Anna Adamaszek, and Tillmann Miltzow. The art gallery problem is $\exists\mathbb{R}$-complete. *Journal of the ACM*, 69(1), 2021.

2 Martin Aigner and Günter M. Ziegler. *Proofs from THE BOOK*. Springer, 4th edition, 2009.

3 Patrice Belleville, Prosenjit Bose, Jurek Czyzowicz, Jorge Urrutia, and Joseph Zaks. *K*-guarding polygons on the plane. In *Proc. 6th Canadian Conference on Computational Geometry (CCCG)*, pages 381–386, 1994.

4 Sergey Bereg. On k-vertex guarding simple polygons. Technical report, Kyoto University, 2009. Computational Geometry and Discrete Mathematics.

5 Daniel Busto, William S. Evans, and David G. Kirkpatrick. On k-guarding polygons. In *Proc. 25th Canadian Conference on Computational Geometry (CCCG)*, pages 283–288, 2013.

6 Václav Chvátal. A combinatorial theorem in plane geometry. *Journal of Combinatorial Theory, Series B*, 18(1):39–41, 1975.

7 Steve Fisk. A short proof of Chvátal's Watchman Theorem. *Journal of Combinatorial Theory, Series B*, 24(3):374, 1978.

8 D. T. Lee and Arthur K. Lin. Computational complexity of art gallery problems. *IEEE Transactions on Information Theory*, 32(2):276–282, 1986.

9 Kurt Mehlhorn, Jörg Sack, and Joseph Zaks. Note on the paper "K-vertex guarding simple polygons" [Computational Geometry 42 (4) (May 2009) 352–361]". *Computational Geometry: Theory and Applications*, 42:722, 2009.

10 Pat Morin, Prosenjit Bose, and Paz Carmi. Open problem session. 18th Algorithms and Data Structures Symposium (WADS), 2023.

11 J. O'Rourke. *Art Gallery Theorems and Algorithms*. International series of monographs on computer science. Oxford University Press, 1987.

12 Ihsan Salleh. K-vertex guarding simple polygons. *Computational Geometry: Theory and Applications*, 42(4):352–361, 2009.

Flips in Odd Matchings*

Oswin Aichholzer[1], Anna Brötzner[2], Daniel Perz[3], and Patrick Schnider[4]

1 **Graz University of Technology**
oaich@ist.tugraz.at
2 **Department of Computer Science and Media Technology, Malmö University, Sweden**
anna.brotzner@mau.se
3 danielperz@gmx.at
4 **Department of Computer Science, ETH Zürich**
patrick.schnider@inf.ethz.ch

Abstract

Let P be a set of $n = 2m + 1$ points in the plane in general position. We define the graph GM_P whose vertex set is the set of all plane matchings on P with exactly m edges. Two vertices in GM_P are connected if the two corresponding matchings have $m - 1$ edges in common. In this work we show that GM_P is connected.

1 Introduction

Reconfiguration is the process of changing a structure into another—either through continuous motion or through discrete changes. Concentrating on plane graphs and discrete reconfiguration steps of bounded complexity, like exchanging one edge of the graph for another edge such that the new graph is in the same graph class, a single reconfiguration step is often called an *edge flip*. The *flip graph* is then defined as the graph having a vertex for each configuration and an edge for each flip. Flip graphs have several applications, for example morphing [6] and enumeration [8]. Three questions are central: studying the connectivity of the flip graph, its diameter, and the complexity of finding the shortest flip sequence between two given configurations. The topic of flip graphs has been well studied for different graph classes like triangulations [3, 14, 15, 16, 17, 19, 20], plane spanning trees [11, 12], plane spanning paths [2, 5], and many more. For a nice survey see [10].

For matchings usually other types of flips were considered since a perfect matching cannot be transformed to another perfect matching with a single edge flip. A natural flip in perfect matchings is to replace two matching edges with two other edges, such that the new graph is again a perfect matching. These flips were studied mostly for convex point sets [9, 18]. While the according flip graph is connected on convex point sets it is open whether this flip graph is connected for any set of points in general position. Other types of flips in perfect matchings can be found in [1, 4, 7].

In this work we study a setting where single edge flips are possible for matchings. Let P be a set of $n = 2m + 1$ points in the plane in general position (that is, no 3 points on a line). An *almost perfect matching* on P is a set M of m line segments whose endpoints are pairwise disjoint and in P. The matching M is called *plane* if no two segments cross.

* Research on this work has been initiated at the 18th European Geometric Graph Week which was held from September 4th to 8th in Alcalá de Henares. We thank all participants for the good atmosphere as well as for inspiring discussions on the topic.

40th European Workshop on Computational Geometry, Ioannina, Greece, March 13–15, 2024.
This is an extended abstract of a presentation given at EuroCG'24. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

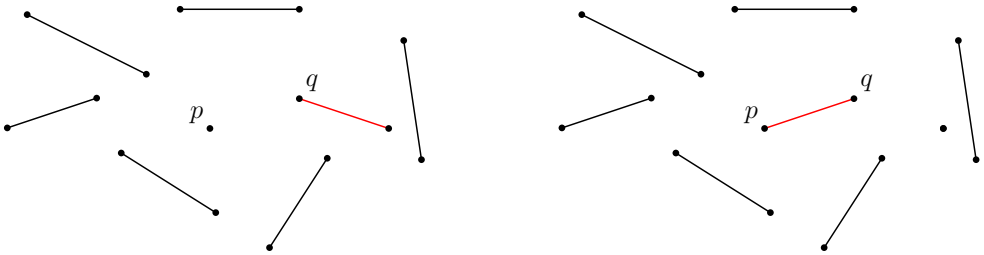


Figure 1 Flipping a matching edge: the previously unmatched point p is matched to q.

Let $\mathcal{M}_P$ denote the set of all plane almost perfect matchings on P. We define the flip graph GM_P with vertex set $\mathcal{M}_P$ through the following flip operation. Consider a matching M_1 and let p be the unmatched point. Let q be a point in P such that the segment pq does not cross any segment in M_1. The flip now consists of removing the segment incident to q from the matching and adding pq instead, see Figure 1. Note that this gives another plane almost perfect matching M_2. In the graph GM_P, the vertices corresponding to M_1 and M_2 are adjacent.

In this paper, we prove the following theorem.

▶ **Theorem 1.1.** *For any set P of $n = 2m + 1$ points in general position in the plane the flip graph GM_P is connected.*

In Section 2 we give an overview of the used techniques and the proof of Theorem 1.1. Then in Section 3 we prove the lemmata used for the proof of Theorem 1.1.

2 Overview and Proof of Theorem 1.1

In this section, we give an overview of our used techniques and the proof of Theorem 1.1.

Let $G = (V, E)$ be a graph G and let M be a matching in G. We call a path P in G an *alternating path* if the edges of P lie alternately in M and in $E \setminus M$. In the following, we consider so-called *segment endpoint visibility graphs*: graphs that encode the visibility between the endpoints of a set of segments. More precisely, given a set S of (non-intersecting) segments in the plane, its segment endpoint visibility graph is the graph that contains a vertex for every segment endpoint, and an edge between two vertices if the corresponding segment endpoints either (1) are connected by a segment in S, or (2) "see" each other, meaning that the open segment between them does not intersect any segment from S. Hoffmann and Tóth [13] proved that segment endpoint visibility graphs always admit a simple Hamiltonian polygon—this is a plane Hamiltonian cycle—, and moreover presented an algorithm to find such a polygon. This result is crucial for us, as a plane perfect matching can be considered as a set of segments in the plane. Hence, for every plane matching M there exists a plane subgraph of the segment endpoint visibility graphs of M that is the (not necessarily disjoint) union of a Hamiltonian cycle and M. Even disregarding planarity, we prove

▶ **Lemma 2.1.** *Let G be an undirected graph that is the union of a Hamiltonian cycle C and a perfect matching M. Let $e_1 = (a, b)$ and $e_2 = (c, d)$ be two matching edges. Then there exists an alternating path P that starts with the vertex a and the edge e_1 and ends with the vertex c.*

We denote the *symmetric difference* of two graphs A, B with $A \triangle B$. Given the setup of Lemma 2.1, we can compute another matching $M_2 = M \triangle P$ in which both a and d are

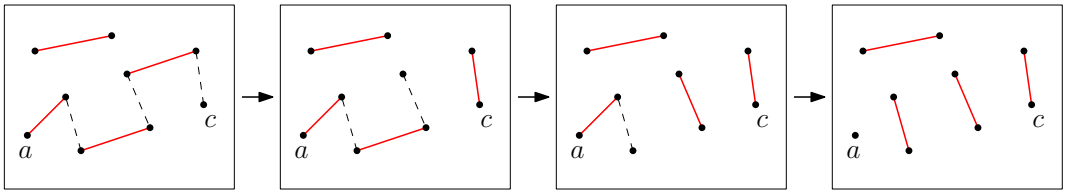


Figure 2 A plane alternating path in the visibility graph gives rise to a sequence of flips.

unmatched. Ignoring the point d, this augmentation corresponds to a sequence of flips in a point set of odd size. See Figure 2 for an illustration. This flip sequence starts with the matching $M_1 = M \setminus \{e_2\}$ and point c being unmatched, and ends with the matching M_2 and point a being unmatched.

To prove that the flip graph GM_P is connected, we show that there always exists a sequence of flips which transforms a given plane almost perfect matching into a plane almost perfect matching, where the unmatched point lies on the boundary of the convex hull.

▶ **Lemma 2.2.** *Let M_1 be a plane almost perfect matching and let t be a point on the convex hull of P. Then there exists a sequence of flips to a matching M_2 in which the unmatched point is t.*

We use Lemma 2.2 to show that we can flip every matching M to a *canonical matching* M_C, which we now define. Let $P = \{p_1, p_2, \ldots, p_{2m+1}\}$, where the points are labeled from left to right. The canonical matching M_C now consists of the edges $p_1p_2, p_3p_4, \ldots, p_{2m-1}p_{2m}$ with p_{2m+1} remaining unmatched. It follows from the ordering of the points that this matching is plane.

Proof of Theorem 1.1. Let M be any plane almost perfect matching on P. Let i be the smallest index for which the edge $p_{2i-1}p_{2i}$ is not in M. We show that there is a sequence of flips on the point set $\{p_{2i-1}, p_{2i}, \ldots, p_{2m}, p_{2m+1}\}$ after which $p_{2i-1}p_{2i}$ is in the resulting matching. In the following, for simplicity of notation, we set $i = 1$.

Using Lemma 2.2, we first flip to a matching M_2 in which the point p_1 is unmatched. As the segment p_1p_2 cannot be crossed by any other segment, we can thus do one more flip which puts p_1p_2 into the resulting matching. Now we can inductively continue the argument on the point set $P' = \{p_3, \ldots, p_{2m+1}\}$ and eventually reach the canonical matching M_C. ◀

▶ Remark. From our proof it follows directly that not more than $O(n^2)$ flips are needed to transform any plane almost perfect matching on P into any other plane almost perfect matching on P. Or in other words, the diameter of the flip graph GM_P is in $O(n^2)$.

3 Proofs of the Lemmata

In this section, we prove Lemma 2.1 and Lemma 2.2. We begin this section with presenting a procedure to find an alternating path in an abstract graph.

▶ **Lemma 2.1.** *Let G be an undirected graph that is the union of a Hamiltonian cycle C and a perfect matching M. Let $e_1 = (a, b)$ and $e_2 = (c, d)$ be two matching edges. Then there exists an alternating path P that starts with the vertex a and the edge e_1 and ends with the vertex c.*

Proof. In a first step, we reduce to the situation where no matching edge except possibly e_1 or e_2 lies on the cycle C, that is, $C \cap M \subseteq \{e_1, e_2\}$. To this end, assume that there is a

matching edge $f = \{f_1, f_2\}$ lying on the path (f_0, f_1, f_2, f_3) of the cycle C. We define the graph G' with vertex set $V(G') = V(G) \setminus \{f_1, f_2\}$ by keeping all edges of G induced by $V(G')$ and adding the edge $\{f_0, f_3\}$. It follows from the construction that G' is again the union of a Hamiltonian cycle and a perfect matching and that G' contains an alternating path starting at a and ending at c if and only if G contains an alternating path starting at a and ending at c. Thus, in the following we may assume that $C \cap M \subseteq \{e_1, e_2\}$.

We now describe an algorithm that explicitly constructs a required alternating path. The algorithm constructs a sequence of graphs $G_2, G_3, \ldots, G_p$, starting with $G_2 = \{e_1\}$, with the following properties:

(1) the graph G_k has the k vertices $v_1, \ldots, v_k$;
(2) G_k has two vertices of degree 1, namely v_1 and v_k;
(3) all other vertices of G_k have degree 2 and are incident to one edge in M and one edge in $C \setminus M$;
(4) $v_1 = a$, $v_2 = b$ and $v_p = c$.

From these properties it follows that the last graph G_p is the disjoint union of cycles and the required alternating path P. It remains to describe the algorithm and prove that the constructed sequence of graphs satisfies the above properties. We start by setting $G_2 = \{e_1\}$, which trivially satisfies all the properties. In order to construct G_{k+1} from G_k we distinguish two cases, depending on whether in G_k the (unique) edge e incident to v_k is in M or not.

Case 1: $e \in C \setminus M$**.** Let $m = \{v_k, w\}$ be the matching edge incident to v_k. We define G_{k+1} by adding m to G_k. By Property (3) for G_k, all vertices in G_k except v_k are incident to an edge in M, and as M is a perfect matching, this implies that w is not a vertex of G_k. Thus, G_{k+1} has one more vertex, proving Property (1) for G_{k+1}. The only vertices whose degrees have changed are $w = v_{k+1}$, which now has degree 1, and v_k which is now also incident to an edge in M. This proves properties (2) and (3).

Case 2: $e \in M$**.** For an illustration of this case, see Figure 3. Consider the unique path Q in C from v_k to c which does not pass through a and let w be the first vertex on this path that is not a vertex of G_k. Set $v_{k+1} = w$. For any edge e in Q, add e to G_{k+1} if and only if it is not in G_k and remove it otherwise. Properties (1) and (2) follow directly by definition. For Property (3), note that the only vertices whose neighborhoods have changed are the vertices on Q. As Q is a path on C and we assumed that C contains no matching edge other than e_1 and e_2, it follows that no matching edge was removed. All vertices are thus still incident to exactly one matching edge. Further, as C is a cycle, every vertex in Q is incident to exactly two edges in $C \setminus M$. It follows from the construction that exactly one of these edges is removed while the other one is added, proving Property (3).

Finally, we stop the procedure as soon as we add the vertex c, which has to happen for some G_p, $p \leq n$, where n is the number of vertices of G. This proves the last part of Property (4) and thus finishes the proof. ◀

▶ **Lemma 2.2.** *Let M_1 be a plane almost perfect matching and let t be a point on the convex hull of P. Then there exists a sequence of flips to a matching M_2 in which the unmatched point is t.*

Proof. Let p be the unmatched point in M_1. If $p = t$ then we are trivially done, so assume for the remainder that $p \neq t$. We duplicate p such that the two points p, p' have the same neighborhood in the segment endpoint visibility graph. Moreover, we add the edge pp' to M_1. By [13] there is a plane Hamiltonian cycle C that spans all segment endpoints of M_1 and $M_1 \cup C$ is plane. Let u be the vertex that is matched to t in M_1. By Lemma 2.1, there is an

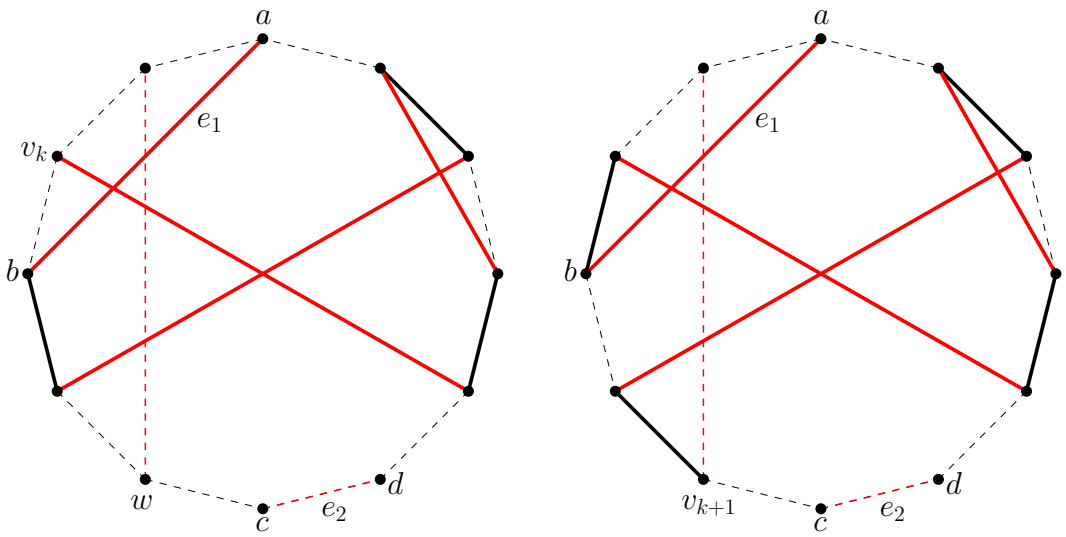


Figure 3 Constructing G_{k+1} (right) from G_k (left). The paths G_k and G_{k+1} are depicted with lines, while unused edges of G are dashed. The matching edges are red, the cycle edges are black.

alternating path P from t to p in $C \cup M_1$ which starts with the edge tu. Since the underlying graph is plane, P is also plane. If p and p' are in P, then the edge pp' is also in P because pp' is a matching edge. Hence, we can contract p and p' to a single point p such that P is still an alternating path.

Now, we construct a matching M_2 by augmenting M_1 via a sequence of flips along P to get $M_2 = M_1 \triangle P$. M_2 is an almost perfect matching in which p is matched, and t is the unmatched point. ◀

4 Conclusion

We considered the flip graph GM_P of plane matchings for point sets of odd size, and showed that GM_P is connected. In the course of the proof, we also showed that the union of a Hamiltonian cycle and a perfect matching always contains an alternating path from an arbitrary matching edge to any other arbitrary point.

While we showed that the flip graph is connected, it would be interesting to determine more precise bounds for the diameter of GM_P. Another interesting setting might be to study this problem in two colored point sets with plane almost perfect bicolored matchings and determine whether the according flip graph is still connected.

References

1 Oswin Aichholzer, Sergey Bereg, Adrian Dumitrescu, Alfredo García, Clemens Huemer, Ferran Hurtado, Mikio Kano, Alberto Márquez, David Rappaport, Shakhar Smorodinsky, Diane Souvaine, Jorge Urrutia, and David. Wood. Compatible geometric matchings. *Computational Geometry*, 42(6-7):617–626, 2009. doi:10.1016/j.comgeo.2008.12.005.

2 Oswin Aichholzer, Kristin Knorr, Wolfgang Mulzer, Johannes Obenaus, Rosna Paul, and Birgit Vogtenhuber. Flipping plane spanning paths. In *International Conference and Workshops on Algorithms and Computation*, pages 49–60. Springer, 2023. doi:10.1007/978-3-031-27051-2_5.

3 Oswin Aichholzer, Wolfgang Mulzer, and Alexander Pilz. Flip Distance Between Triangulations of a Simple Polygon is NP-Complete. *Discrete Comput. Geom.*, 54(2):368–389, 2015. doi:10.1007/s00454-015-9709-7.

4 Oswin Aichholzer, Julia Obmann, Pavel Paták, Daniel Perz, Josef Tkadlec, and Birgit Vogtenhuber. Disjoint compatibility via graph classes. In *International Workshop on Graph-Theoretic Concepts in Computer Science*, pages 16–28. Springer, 2022. doi:10.1007/978-3-031-15914-5_2.

5 Selim G. Akl, Md. Kamrul Islam, and Henk Meijer. On planar path transformation. *Information Processing Letters*, 104(2):59–64, 2007. doi:10.1016/j.ipl.2007.05.009.

6 Soroush Alamdari, Patrizio Angelini, Fidel Barrera-Cruz, Timothy M. Chan, Giordano Da Lozzo, Giuseppe Di Battista, Fabrizio Frati, Penny Haxell, Anna Lubiw, Maurizio Patrignani, Vincenzo Roselli, Sahil Singla, and Bryan T. Wilkinson. How to morph planar graph drawings. *SIAM Journal on Computing*, 46(2):824–852, 2017. doi:10.1137/16M1069171.

7 Greg Aloupis, Luis Barba, Stefan Langerman, and Diane L. Souvaine. Bichromatic compatible matchings. *Computational Geometry*, 48(8):622–633, 2015. doi:10.1016/j.comgeo.2014.08.009.

8 David Avis and Komei Fukuda. Reverse search for enumeration. *Discrete Applied Mathematics*, 65(1-3):21–46, 1996. doi:10.1016/0166-218X(95)00026-N.

9 Ahmad Biniaz, Anil Maheshwari, and Michiel Smid. Flip distance to some plane configurations. *Computational Geometry*, 81:12–21, 2019. doi:10.1016/j.comgeo.2019.01.008.

10 Prosenjit Bose and Ferran Hurtado. Flips in planar graphs. *Computational Geometry*, 42(1):60–80, 2009. doi:10.1016/j.comgeo.2008.04.001.

11 Nicolas Bousquet, Lucas De Meyer, Théo Pierron, and Alexandra Wesolek. Reconfiguration of plane trees in convex geometric graphs. *arXiv preprint arXiv:2310.18518*, 2023.

12 M. Carmen Hernando, Ferran Hurtado, Alberto Márquez, Merce Mora, and Marc Noy. Geometric tree graphs of points in convex position. *Discrete Applied Mathematics*, 93(1):51–66, 1999. doi:10.1016/S0166-218X(99)00006-2.

13 Michael Hoffmann and Csaba D. Tóth. Segment endpoint visibility graphs are Hamiltonian. *Computational Geometry*, 26(1):47–68, 2003. doi:10.1016/S0925-7721(02)00172-4.

14 Ferran Hurtado, Marc Noy, and Jorge Urrutia. Flipping edges in triangulations. *Discrete & Computational Geometry*, pages 333–346, 1999. doi:10.1007/PL00009464.

15 Iyad Kanj, Eric Sedgwick, and Ge Xia. Computing the flip distance between triangulations. *Discrete & Computational Geometry*, 58(2):313–344, 2017. doi:10.1007/s00454-017-9867-x.

16 Charles L. Lawson. Transforming triangulations. *Discrete Mathematics*, 3(4):365–372, 1972. doi:10.1016/0012-365X(72)90093-3.

17 Anna Lubiw and Vinayak Pathak. Flip distance between two triangulations of a point set is NP-complete. *Computational Geometry*, 49:17–23, 2015. doi:10.1016/j.comgeo.2014.11.001.

18 Marcel Milich, Torsten Mütze, and Martin Pergel. On flips in planar matchings. *Discrete Applied Mathematics*, 289:427–445, 2021. doi:10.1016/j.dam.2020.10.018.

19 Alexander Pilz. Flip distance between triangulations of a planar point set is APX-hard. *Computational Geometry*, 47(5):589–604, 2014. doi:10.1016/j.comgeo.2014.01.001.

20 Uli Wagner and Emo Welzl. Connectivity of triangulation flip graphs in the plane. *Discrete & Computational Geometry*, 68(4):1227–1284, 2022. doi:10.1007/s00454-022-00436-2.

A variant of backwards analysis applicable to order-dependent sets*

Evanthia Papadopoulou[1] and Martin Suderland[2]

1 **Faculty of Informatics, Università della Svizzera italiana (USI), Lugano, Switzerland**
`evanthia.papadopoulou@usi.ch`

2 **Courant Institute, New York University (NYU), New York, NY, USA**
`martin.suderland@cs.nyu.edu`

Abstract

Backwards analysis is a simple, yet powerful technique used in analyzing the expected performance of randomized algorithms: a randomized incremental algorithm is seen as if it were running backwards in time, from output to input [Seidel 1993]. To be applicable, a key requirement is that the algorithm's output is independent of the randomization order. This needs to hold even for intermediate structures, assuming the same set of elements have been processed. In this note we illustrate a variant, which can be applied to algorithms with order-dependent output. As an example, we use the *randomized incremental triangulation* of a point set in $\mathbb{R}^d$, a generalization of QUICKSORT in higher dimensions, which has been cited by Seidel as a negative example, where backwards analysis could not be applied. We prove that the expected running time of this algorithm is $O(n \log n)$. This variant of backwards analysis was introduced by [Junginger and Papadopoulou, DCG 2023].

1 Introduction

Backwards analysis was popularized in Computational Geometry by Seidel [7]. It is a simple, yet powerful technique to analyze the expected performance of a randomized algorithm. Backwards analysis is based on the observation that the cost of the last step of an algorithm can be often expressed as a function of the complexity of the final output; and thus, the algorithm can be analyzed as if it were running backwards in time, from output to input [7].

In computational geometry, the first algorithm analyzed by backwards analysis was the construction of the Delaunay triangulation of a set of points in convex position in the plane [2]. Since then, backwards analysis has been applied to a plethora of problems, and has become a standard trick in analyzing the expected performance of randomized algorithms, see e.g., [1] and references therein. This includes a simplified analysis of the influential randomized incremental construction paradigm, introduced by Clarkson and Shor [3], and a particularly simple approach to analyze QUICKSORT, both presented by Seidel in [7]. A key requirement to apply backwards analysis, however, is that the algorithm's output is independent of the randomization order. To illustrate this fact, Seidel pointed out a negative example, where backwards analysis could not be applied. This is the *randomized incremental triangulation* of a point set in $\mathbb{R}^d$, which generalizes QUICKSORT in higher dimensions. Seidel concluded the section with an open problem: "It remains to be seen whether for fixed $d > 1$ the expected running time of this triangulation algorithm is indeed $O(n \log n)$".

* This research was supported by the Swiss National Science Foundation, Projects 200021E_201356 (Evanthia Papadopoulou) and P500PT_206736/1 (Martin Suderland).

40th European Workshop on Computational Geometry, Ioannina, Greece, March 13–15, 2024.
This is an extended abstract of a presentation given at EuroCG'24. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

Contribution. In this paper we advertise a new variant of backwards analysis, which indeed can prove the aforementioned $O(n \log n)$ time complexity for the randomized incremental triangulation algorithm. Problems where the final output or intermediate structures depend on the randomization order can benefit from this approach. It was first described in [5] for analyzing a randomized incremental algorithm to perform deletion in abstract Voronoi diagrams in linear time The intermediate *Voronoi-like diagrams* computed by this algorithm were order-dependent and thus ordinary backwards analysis could not be correctly applied[1]. In this paper we present another example where this new variant of backwards analysis can analyse the running time of a randomized incremental algorithm while ordinary backwards analysis cannot be applied.

2 Backwards analysis

Preliminaries Given $n \in \mathbb{N}$, we write $[n]$ for $\{1, 2, ..., n\}$. Let S_n be the set of all permutations of length n. We use the one-line notation of permutations.

For a permutation $\sigma = (\sigma_1, \sigma_2..., \sigma_n)$ and index $1 \leqslant i \leqslant n$ let $\sigma(i) = \sigma_i$. Further, we define the *shift permutation* $\sigma^{(i)} = (\sigma_1, ..., \sigma_{i-1}, \sigma_{i+1}, ..., \sigma_n, \sigma_i)$, which moves the i-th element to the end.

The standard backwards analysis is based on the following idea: "Analyze an algorithm as if it were running backwards in time, from output to input. This is based on the observation that often the cost of the last step of an algorithm can be expressed as a function of the complexity of the final product output of the algorithm" [7].

The expected running time of a randomized algorithm is computed step by step. We assume that the algorithm is randomized over a set of n elements. Denote by T_i the running time expended for the i-th step and by O_i the computed structure at step $1 \leqslant i \leqslant n$. Then $T_i(\sigma)$, the time required for the i-th step for permutation σ, is getting bounded by a function of the complexity of the output object $O_i(\sigma)$).

Considering that each permutation is equally likely, we write for the expected time $E(T_i)$:

$$E(T_i) = \frac{\sum_{\sigma \in S_i} T_i(\sigma)}{i!} = \frac{\sum_{j=1}^{i} \sum_{\substack{\sigma \in S_i \\ \sigma(i)=j}} T_i(\sigma)}{i!}.$$

Because the standard backwards analysis is used for order-independent structures, the time needed for the i-th step does not depend on the entire permutation σ but only on the last processed element, i.e. $\sigma(i) = j$.

The variant of backwards analysis exploits the key idea of looking at groups of similar permutations, where the order of elements is almost the same. A group has one representative permutation σ and consists of $G_\sigma = \{\sigma^{(1)}, \sigma^{(2)}, ..., \sigma^{(n)}\}$, where $\sigma^{(n)} = \sigma$.

Essentially each group has one representative from which all other permutations can be derived, by moving one element to the end of the one-line representation, see Fig. 1. Let $R_n \subset S_n$ be a set of $(n-1)!$ many representative permutations such that the groups of representatives is a partition of all permutations of length n, i.e. $\bigcup_{\sigma \in R_n} G_\sigma = S_n$.

[1] The preliminary version of this paper [4] had originally assumed otherwise, after expressing the time complexity of each step as a function of the output structure. The new variant in [5] successfully completed the analysis.

$$G_\sigma = \left\{ \begin{array}{l} \sigma = (\sigma_1, \sigma_2, \sigma_3, ..., \sigma_{n-1}, \sigma_n) = \sigma^{(n)} \\ \sigma^{(1)} = (\sigma_2, \sigma_3, ..., \sigma_{n-1}, \sigma_n, \sigma_1) \\ \sigma^{(2)} = (\sigma_1, \sigma_3, ..., \sigma_{n-1}, \sigma_n, \sigma_2) \\ ... \\ \sigma^{(i)} = (\sigma_1, ..., \sigma_{i-1}, \sigma_{i+1}..., \sigma_n, \sigma_i) \\ ... \\ \sigma^{(n-1)} = (\sigma_1, \sigma_2, ..., \sigma_{n-2}, \sigma_n, \sigma_{n-1}) \end{array} \right\}$$

Figure 1 The group G_σ for representative $\sigma = (\sigma_1, \sigma_2, \sigma_3, ..., \sigma_{n-1}, \sigma_n)$.

For brevity we write $T_i(G_\sigma) = \sum_{\rho \in G_\sigma} T_i(\rho)$. Thus, for the expected time needed for step i we can derive:

$$E(T_i) = \frac{\sum_{\sigma \in S_i} T_i(\sigma)}{i!} = \frac{\sum_{\sigma \in R_i} T_i(G_\sigma)}{i!}.$$

The existence of such a set R_n, for all $n \in \mathbb{N}$, was shown by Levenshtein [6] for problems that were not related to backwards analysis or this variant, and this was originally pointed out to us by Stefan Felsner. For $n = 4$, the set

$$\{(1,2,3,4), (2,1,4,3), (3,1,4,2), (3,2,4,1), (4,1,3,2), (4,2,3,1)\}$$

can be chosen as R_4. Only the existence of a set R_n is important for the variant to work without having to know a particular representation.

Why choose this grouping? The grouping is chosen with the following ideas in mind.

- Each element appears exactly once as last element: this is important for computing the overall expected running time, where each element should appear as last the same number of times.
- The grouping minimizes the number of inversions between the base permutation σ and the permutations within its group G_σ.

Among all groupings satisfying the first property, the one that we chose with G_σ induces the least number of inversions. Minimizing inversions is desirable because it can drastically simplify the derivation compared to other alternative groupings that may satisfy the first item, such as the one generated by swapping elements $\sigma^{(i)}$ and $\sigma^{(n)}$.

To evaluate $T_i(G_\sigma)$ we try to bound each $T_i(\sigma^{(j)})$ by a portion of the output structure $O_i(\sigma)$ for the base permutation σ of the group. We thus evaluate $T_i(G_\sigma)$ as a function involving $O_i(\sigma)$. A minimum number of inversions between $\sigma^{(j)}$ and σ is hense essential for simplifying this task. For our triangulation example this is done in Lemma 3.2.

3 Example: Triangulation algorithm

Given a set of n points $P = \{p_1, p_2, ..., p_n\} \subset \mathbb{R}^d$ for some $d \in \mathbb{N}$ within a d-simplex Δ. We are recalling a higher-dimensional triangulation algorithm of P within Δ, see [7]. Initially, the triangulation T consists of just one simplex Δ. The points are getting inserted one-by-one while we keep the triangulation updated. If point p_i is inserted we are looking for the simplex τ of T, which contains p_i. We replace this simplex τ by the $d + 1$ many simplices, which have p_i as corner each and which partition τ. An overview of the algorithm is given in Algorithm 1 together with an example in Fig. 2. The randomized version is achieved by

Algorithm 1: Triangulating a point set

Input : Set of n points $P = \{p_1, p_2, ..., p_n\}$ within simplex $\Delta \subset \mathbb{R}^d$
Output : Triangulation T of P and Δ

```
T ← Δ;
for i = 1 → n do
    Remove simplex τ, which contains p_i, from T;
    Partition τ into d + 1 simplices, each having p_i as corner;
    Add the new d + 1 simplices to T;
return T;
```

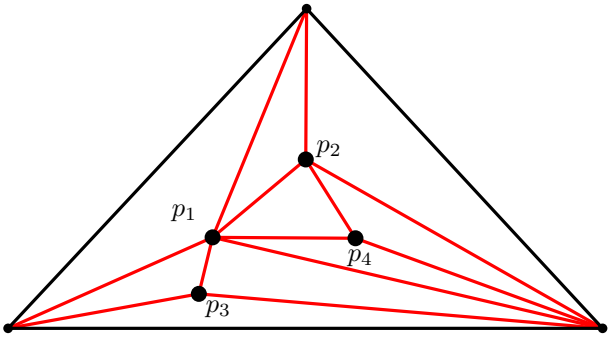


Figure 2 Triangulation of Algorithm 1 for the insertion order $\sigma = (1, 2, 3, 4)$.

initially permuting the set of points P by some random permutation $\sigma \in S_n$, i.e. processing the points in the order $p_{\sigma(1)}, p_{\sigma(2)}, ..., p_{\sigma(n)}$.

At any time during the algorithm the following information is kept:

① for each simplex we know the points of P, which are contained in it, and
② for each point $p \in P$ we know the simplex containing p.

Expected time complexity In this part we show how to apply the variant of the backwards analysis to derive a bound on the time complexity for the triangulation algorithm. For simplicity we prove the following bound for the 2-dimensional version, even though the analysis can easily be generalized to higher dimensions.

▸ **Theorem 3.1.** *For any fixed d, Algorithm 1 has $O(n \log n)$ expected time complexity.*

The expected running time of the i-th step is dominated by updating the information in Items ① and ②, i.e. rebucketing the remaining points of P which were contained in the deleted triangle τ. These are at most $n - i$ many points, but we show that this number is much less in expectation. For $0 \leqslant j \leqslant i \leqslant n$ and permutation $\sigma \in S_n$ denote by T^i_σ the partial triangulation derived by only processing the first i many points specified by σ. Moreover, let ${}^jA^i_\sigma$ be the union of triangles of T^i_σ, which are incident to point $p_{\sigma(j)}$. Finally, if $i < n$ we write Δ^i_σ for the triangle of T^i_σ, which contains point $p_{\sigma(n)}$.

The main idea for the variant of the backwards analysis is to bound the time complexity needed to process an entire group G_σ by some features of the output derived from a single permutation σ. In the triangulation algorithm case, we are putting the points, which need rebucketing in the i-th step for some permutation $\sigma^{(j)}$ for some $j \in [n]$, in relation with the output triangulation T^i_σ corresponding to permutation σ.

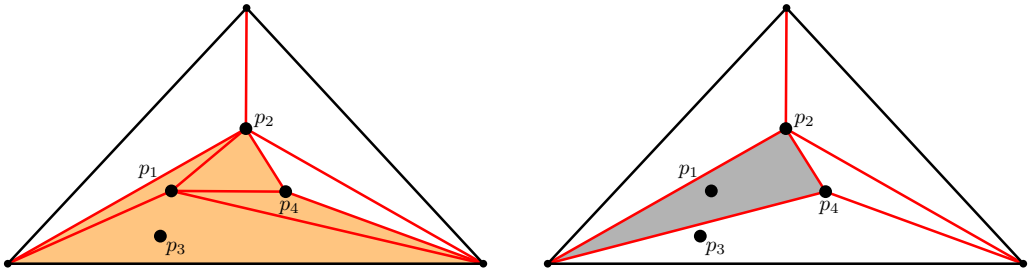


Figure 3 (Left) Partial triangulation T^3_σ for insertion order $\sigma = (2, 1, 4, 3)$ and orange highlighted area ${}^2A^3_\sigma$ and (Right) partial triangulation $T^2_{\sigma^{(2)}}$ for insertion order $\sigma^{(2)} = (2, 4, 3, 1)$ with gray highlighted triangle $\Delta^2_{\sigma^{(2)}}$. Note the subset relation between the orange and gray highlighted sets, which are proven in Lemma 3.2.

▸ **Lemma 3.2.** *For all $1 \leqslant j \leqslant i \leqslant n$ and any permutation $\sigma \in S_n$ it holds: $\Delta^{i-1}_{\sigma^{(j)}} \subset {}^jA^i_\sigma$.*

Proof. The proof works by induction over i. The base case $i = j$ holds because $\Delta^{j-1}_{\sigma^{(j)}} = {}^jA^j_\sigma$. The first difference between $\sigma^{(j)}$ and σ occurs at the j-th position. Thus the triangle in $\mathrm{T}^{j-1}_{\sigma^{(j)}}$, which contains point $p_{\sigma(j)}$, equals the union of three triangles in T^j_σ incident to $p_{\sigma(j)}$.

Let us therefore assume that the induction hypothesis $\Delta^{i-1}_{\sigma^{(j)}} \subset {}^jA^i_\sigma$ holds for some $i \in \{j, j+1, ..., n-1\}$. We want to prove that also $\Delta^i_{\sigma^{(j)}} \subset {}^jA^{i+1}_\sigma$.

If $p_{\sigma(i+1)} \notin {}^jA^i_\sigma$ then we have $\Delta^{i-1}_{\sigma^{(j)}} = \Delta^i_{\sigma^{(j)}}$ and ${}^jA^i_\sigma = {}^jA^{i+1}_\sigma$ and thus the claim $\Delta^i_{\sigma^{(j)}} \subset {}^jA^{i+1}_\sigma$ trivially follows. For the rest of the proof we assume the opposite, i.e. $p_{\sigma(i+1)} \in {}^jA^i_\sigma$. See Fig. 4 for an illustration of the proof. Consider the ray from $p_{\sigma(j)}$ to $p_{\sigma(i+1)}$ and denote q the point of intersection with the boundary of ${}^jA^i_\sigma$. In counterclockwise order around the boundary of ${}^jA^i_\sigma$ we call a (resp. b) the vertex directly after (resp. before) q. Let r_a (resp. r_b) be the ray from $p_{\sigma(i+1)}$ to a (resp. b). Finally, we call W the wedge bounded by r_b and r_a, which does not contain the segment $\overline{p_{\sigma(j)}p_{\sigma(i+1)}}$. Note that none of the points $P \cap {}^jA^i_\sigma$ lies in the interior of W. By construction, we have that ${}^jA^{i+1}_\sigma = {}^jA^i_\sigma \backslash W$. On the other hand, the boundary of the triangle $\Delta^i_{\sigma^{(j)}}$ intersects the segment $\overline{p_{\sigma(j)}p_{\sigma(i+1)}}$, or at least passes through $p_{\sigma(i+1)}$; at the same time it contains the point $p_{\sigma(j)}$ and none of its endpoints lies in W. Therefore, $\Delta^i_{\sigma^{(j)}} \cap W = \varnothing$.

Combining the induction hypothesis $\Delta^{i-1}_{\sigma^{(j)}} \subset {}^jA^i_\sigma$ with the properties $\Delta^i_{\sigma^{(j)}} \subset \Delta^{i-1}_{\sigma^{(j)}}$, $\Delta^i_{\sigma^{(j)}} \cap W = \varnothing$, and ${}^jA^{i+1}_\sigma = {}^jA^i_\sigma \backslash W$, we can indeed verify that also $\Delta^i_{\sigma^{(j)}} \subset {}^jA^{i+1}_\sigma$ holds:

$$\begin{aligned} & \Delta^i_{\sigma^{(j)}} \subset \Delta^{i-1}_{\sigma^{(j)}} \subset {}^jA^i_\sigma \\ \Rightarrow \quad & \Delta^i_{\sigma^{(j)}} \backslash W \subset {}^jA^i_\sigma \backslash W \\ \Rightarrow \quad & \Delta^i_{\sigma^{(j)}} \subset {}^jA^{i+1}_\sigma \end{aligned}$$

◂

▸ **Corollary 3.3.** *In the i-th insertion step of Algorithm 1, each of the $n - i$ remaining points has to be rebucketed for at most 3 permutations within a group G_σ for any $\sigma \in S_i$.*

Proof. Due to Lemma 3.2 we have the property $\Delta^{i-1}_{\sigma^{(j)}} \subset {}^jA^i_\sigma$ holding true for any $j \in [i]$ and permutation $\sigma \in S_i$. The term $\Delta^{i-1}_{\sigma^{(j)}}$ describes exactly the region which has to be re-partitioned in the i-th step of the triangulation algorithm if the permutation $\sigma^{(j)}$ is chosen, i.e. any point of P falling into that region would have to be rebucketed. Thus, the region which needs to be repartitioned for permutation $\sigma^{(j)}$ is a subset of all triangles in T^i_σ, which are incident to point $p_{\sigma(j)}$. For a point $p \in P$ let τ be the triangle of T^i_σ containing p. Then p

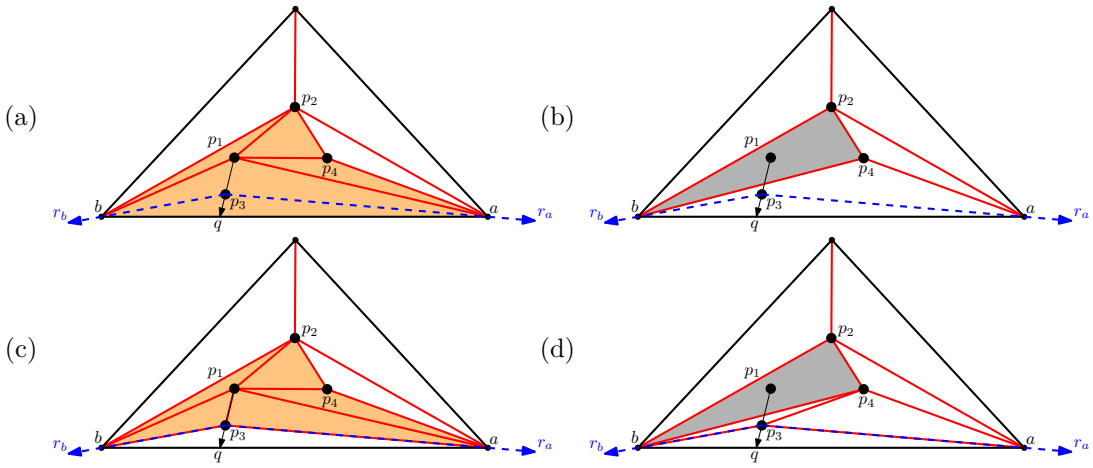


Figure 4 For insertion order $\sigma = (2, 1, 4, 3)$ we show: (a) T^3_σ with highlighted ${}^2A^3_\sigma$ (b) $T^2_{\sigma(2)}$ with highlighted $\Delta^2_{\sigma(2)}$ (c) T^4_σ with highlighted ${}^2A^4_\sigma$ (d) $T^3_{\sigma(2)}$ with highlighted $\Delta^3_{\sigma(2)}$. From $\Delta^2_{\sigma(2)} \subset {}^2A^3_\sigma$ we derive that also $\Delta^3_{\sigma(2)} \subset {}^2A^4_\sigma$.

only needs to be rebucketed for permutations of G_σ, in which one of τ's corners is processed last. Thus p has to be rebucketed at most three times in total for the entire group G_σ; at most once for each permutation in G_σ with one of τ's corners being processed last. ◂

The proof of Theorem 3.1 follows immediately from the previous corollary:

$$E(T(n)) = \frac{\sum_{\sigma \in R_i} T_i(G_\sigma)}{i!} = O\left(\frac{\sum_{\sigma \in R_i} 3(n-i)}{i!}\right) = O\left(\sum_{i=1}^{n} \frac{n}{i}\right) = O(n \log n).$$

4 Conclusion

The first example using this variant of backwards analysis appeared in the context of abstract Voronoi-like diagrams. These were introduced in [5] serving as intermediate structures in a randomized incremental algorithm to perform site-deletion in an abstract Voronoi diagram in expected linear time. These intermediate structures depended on the permutation order of the randomized algorithm, while the final output did not.

The triangulation algorithm, presented in this abstract, is a very simple algorithm, which we mostly used to illustrate this new variant of backwards analysis. We believe that this variant can be of interest to many other problems, when analyzing the expected time complexity of randomized algorithms for order-dependent structures.

References

1 Jean-Daniel Boissonnat and Mariette Yvinec. *Algorithmic Geometry*. Cambridge University Press, New York, NY, USA, 1998.

2 L. Paul Chew. Building Voronoi diagrams for convex polygons in linear expected time. Technical report, Dartmouth College, Hanover, USA, 1990.

3 Kenneth L. Clarkson and Peter W. Shor. Applications of random sampling in computational geometry, II. *Discrete & Computational Geometry*, 4:387–421, 1989.

4 Kolja Junginger and Evanthia Papadopoulou. Deletion in abstract Voronoi diagrams in expected linear time. In *34th International Symposium on Computational Geometry (SoCG)*, volume 99 of *LIPIcs*, pages 50:1–50:14, 2018.

5 Kolja Junginger and Evanthia Papadopoulou. Deletion in abstract Voronoi diagrams in expected linear time and related problems. *Discrete & Computational Geometry*, 69(4):1040–1078, 2023.

6 Vladimir I Levenshtein. On perfect codes in deletion and insertion metric. *Discrete Mathematics and Applications*, 2(3):241–258, 1992.

7 Raimund Seidel. Backwards analysis of randomized geometric algorithms. In *New trends in discrete and computational geometry*, pages 37–67. Springer, 1993.

Reconfiguration of plane trees in convex geometric graphs

Nicolas Bousquet[1], Lucas De Meyer[2], Théo Pierron[3], and Alexandra Wesolek[4]

1 **Université de Lyon, LIRIS, CNRS, Université Claude Bernard Lyon 1, France**
nicolas.bousquet@cnrs.fr
2 **Université de Lyon, LIRIS, CNRS, Université Claude Bernard Lyon 1, France**
lucas.de-meyer@univ-lyon1.fr
3 **Université de Lyon, LIRIS, CNRS, Université Claude Bernard Lyon 1, France**
théo.pierron@univ-lyon1.fr
4 **Technische Universität Berlin, Germany**
alexandrawesolek@gmail.com

Abstract

A non-crossing spanning tree of a set of points in the plane is a spanning tree whose edges pairwise do not cross. Avis and Fukuda in 1996 proved that there always exists a flip sequence of length at most $2n-4$ between any pair of non-crossing spanning trees (where n denotes the number of points). Two recent results of Aichholzer et al. and Bousquet et al. improved the upper bound on the length of a flip sequence to $2n-\Omega(\log n)$ and $2n-\Omega(\sqrt{n})$ when the points are in convex position.

We pursue the investigation of the convex case by improving the upper bound by a linear factor for the first time in 30 years. We prove that there always exists a flip sequence between any pair of non-crossing spanning trees T_1, T_2 of length at most cn where $c \approx 1.95$. Our result is actually stronger since we prove that, for any two trees T_1, T_2, there exists a flip sequence from T_1 to T_2 of length at most $c|T_1 \setminus T_2|$.

We give a new lower bound in terms of the symmetric difference by proving that there exists a pair of trees T_1, T_2 such that a minimal flip sequence has length $\frac{5}{3}|T_1 \setminus T_2|$. We generalize this lower bound construction to non-crossing flips (where we close the gap between upper and lower bounds) and rotations.

Related Version arXiv:2310.18518

1 Introduction

Let C be a set of n points in the plane in convex position. A *spanning tree* T on the set of points C is a subset of edges that forms a connected acyclic graph on C. A spanning tree T on C is *non-crossing* if every pair of edges of T (represented by the straight line interval between their endpoints) are pairwise non-crossing. Let us denote by $\mathcal{S}(C)$ the set of all non-crossing spanning trees on the point set C. Let $T \in \mathcal{S}(C)$. A *flip* on T consists of removing an edge e from T and adding another edge f so that the resulting graph $(T \cup f) \setminus e$ is also in $\mathcal{S}(C)$. A *flip sequence* is a sequence of non-crossing spanning trees such that consecutive spanning trees in the sequence differ by exactly one flip.

Avis and Fukuda [2] proved that there always exists a flip sequence between any pair of non-crossing spanning trees of length at most $2n-4$ by showing that there is a star S on C such that T_1 and T_2 can be turned into S with at most $n-2$ flips. In fact, they showed that this flip sequence exists even if the point set C is in general position.

Given two spanning trees T_1, T_2, the *symmetric difference* between T_1 and T_2 is denoted by $\Delta(T_1, T_2) = (T_1 \setminus T_2) \cup (T_2 \setminus T_1)$. We denote by $\delta(T_1, T_2) = |\Delta(T_1, T_2)|/2$ the number of edges in T_1 and not in T_2, which is a trivial lower bound on the length of a flip sequence from

40th European Workshop on Computational Geometry, Ioannina, Greece, March 13–15, 2024.
This is an extended abstract of a presentation given at EuroCG'24. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

T_1 to T_2. The set of spanning trees of a graph G forms a matroid, hence, for any possible pair of spanning trees T_1, T_2, there is a (non geometric) flip sequence that transforms T_1 into T_2 in exactly $\delta(T_1, T_2)$ flips. However, more flips are needed for non-crossing spanning trees. Hernando et al. [5] provided for every n, two non-crossing spanning trees T_1, T_2 on a convex set of n points whose minimal flip sequence needs $\frac{3}{2}n - 5$ flips.

During 30 years, no improvement of the lower or upper bound has been obtained until a recent result of Aichholzer et al. [1]. They showed that the upper bound of Avis and Fukuda can be improved when points are in convex position by proving that there exists a flip sequence between any pair of non-crossing spanning trees of length at most $2n - \Omega(\log n)$. Their result has been further improved by Bousquet et al. [4] who proved that $2n - \Omega(\sqrt{n})$ flips are enough. However, until now, there does not exist any general proof that there always exists a flip sequence of length at most $(2 - \epsilon)n$ for some $\epsilon > 0$. On the other side, Bousquet et al. [4] conjectured that the lower bound of Hernando et al. [5] is essentially tight:

▶ **Conjecture 1.1.** Let C be a set of n points in convex position. There exists a flip sequence between any pair of non-crossing spanning trees of length at most $\frac{3}{2}n$.

One can easily prove that there exists a flip sequence of length at most $2\delta(T_1, T_2)$ between any pair of non-crossing spanning trees in convex position. The improvement of Aichholzer et al. [1] also improves this upper bound by $\Omega(\log(\delta(T_1, T_2)))$. Since in the example of Hernando et al. the intersection is reduced to two edges, one can wonder if Conjecture 1.1 can be extended to the symmetric difference, namely:

▶ **Conjecture 1.2.** Let C be a set of n points in convex position. There exists a flip sequence between any pair of non-crossing spanning trees T_1, T_2 of length at most $\frac{3}{2}\delta(T_1, T_2)$.

Contributions Our main results first consist in (i) improving the best known upper bound to approximatively $1.95 \cdot \delta(T_1, T_2)$, breaking the linear factor 2 of the threshold on the length of a minimal flip sequence (even in terms of the symmetric difference), and (ii) disproving Conjecture 1.2 by proving that the best upper bound factor we can hope for is $\frac{5}{3}$. We complete these results by providing improved upper and lower bounds on the length of transformations in other models of flips, namely non-crossing flips and rotations. In particular, we close the gap between upper and lower bounds in the case of non-crossing flips.

Due to space constraints, we only sketch some proofs of the main results, and the full proofs can be found in a recent preprint [3].

2 Definitions

Let C be a set of points in convex position and T be a non-crossing spanning tree on C. We say two points of a convex set C are *consecutive* if they appear consecutively on the convex hull of C. A *border edge* (for T) is an edge between consecutive points. An edge of T which is not a border edge is called a *chord*. A *hole* of T is a pair of consecutive points that is not a border edge. We will say that we *fill a hole* when we apply a flip where the created edge joins the pair of points of the hole.

One can remark that, for each chord e of T, the line containing e splits the convex hull of C in two parts. A *side of a chord* e is the subset of points of C contained in one of the two closed half-planes defined by the line containing the two endpoints of e (see Figure 1 for an illustration). A *side of* T is a side of a chord e for some $e \in T$. We say an edge (or a hole) is *in a side* A if both its endpoints are in A.

In the following, for every side A of a chord, we will denote by k_A the number of holes in A, which is also the number of chords of T in A. Since T is acyclic, we also have $k_A > 0$. Note that each chord e of T defines two sides A and B whose intersection is exactly the endpoints of e. Moreover, T has exactly $k_A + k_B$ holes.

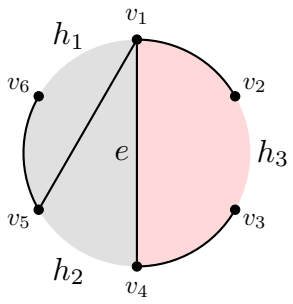


Figure 1 The side A (in grey) of the chord e is is the subset of vertex $\{v_1, v_4, v_5, v_6\}$ and the other side B (in red) of e is $\{v_1, v_4, v_2, v_3\}$. The edges of T in A are the edges v_5v_6, v_1v_5 and v_1v_4. The holes h_1 and h_2 of T are in A and h_3 is in B. So we have $k_A = 2$ and $k_B = 1$.

Let A be a side of a chord e of T. We define the *degree* of a side A in a tree T' as the number of chords of T' crossing e plus twice the number of chords of T' with both endpoints in A (see Figure 2 for an illustration). Note that, if T' has no chords with both endpoints in A, then the degree of A in T' is equal to the number of chords of T' crossing e.

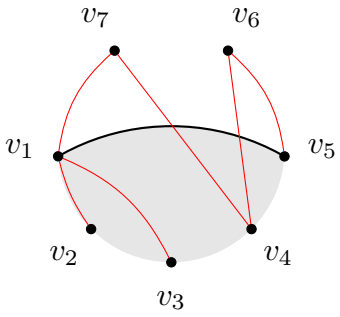


Figure 2 The side A of the edge v_1v_4 highlighted in grey contains v_1, v_2, v_3, v_4 and v_5. The degree of A in the red tree is equal to 4 : v_4v_6 and v_4v_7 cross v_1v_5, and v_1v_3 has both endpoints in A. Note that v_1v_2 is not a chord, thus it does not increase the degree of A in T'.

3 Upper bound

The first result of the paper is to improve the best upper bound of [4] by a linear factor by proving that the following holds:

▶ **Theorem 3.1.** *Let C be a set of n points in convex position. There exists a flip sequence between any pair of non-crossing spanning trees T_1 and T_2 of length at most $c \cdot \delta(T_1, T_2)$ with $c = \frac{1}{12}(22 + \sqrt{2}) \approx 1.95$.*
In particular, there exists a flip sequence of length at most $cn \approx 1.95n$ between any pair of non-crossing spanning trees.

We prove Theorem 3.1 by induction. Let T_I, T_F be two trees on a convex point set C, and assume that Theorem 3.1 holds for every pair of trees T'_I and T'_F, which are either defined on the same set of points and $\delta(T'_I, T'_F) < \delta(T_I, T_F)$ or on a smaller set of points. The goal to prove Theorem 3.1 is to match pairs of chords in T_I, T_F using few flips, i.e. less than c flips per pair of chords matched. Indeed, if we manage to apply at most ck flips on T_I and T_F to obtain T'_I and T'_F with k more edges in common, we get $\delta(T'_I, T'_F) = \delta(T_I, T_F) - k$ and we can conclude by induction.

Basic properties of T_I and T_F Since common chords and non-common border edges can be trivially reduced, we first observe that, if T_I, T_F share a common chord or do not have the same border edges, we can conclude by induction. Hence, we may assume for the rest of the proof that T_I and T_F form a *nice pair* of trees, i.e. the two trees have no common chord and have the same border edges. Note that for a nice pair of trees, every pair of consecutive points is either a common hole or a common border edge. Thus, for a nice pair of trees (T, T'), we will refer to a hole of T or T' simply as a hole.

The rest of the proof describes a transformation from the nice pair (T_I, T_F) to a pair (T_I^*, T_F^*) which matches k pairs of chords of T_I and T_F using at most $c \cdot k$ flips. The main steps of the proof are illustrated in Figure 3. We define a τ-extremal side, which is a side which always exists in a nice pair of trees (and hence which we can also find in T_I and T_F). We then transform a τ-extremal side to what we call a very good side without using too many flips. At the end we observe that in very good sides, we can match the k_A chords in the side using at most $\frac{5}{3}k_A$ flips in total. Analysing the steps then gives us the desired bound.

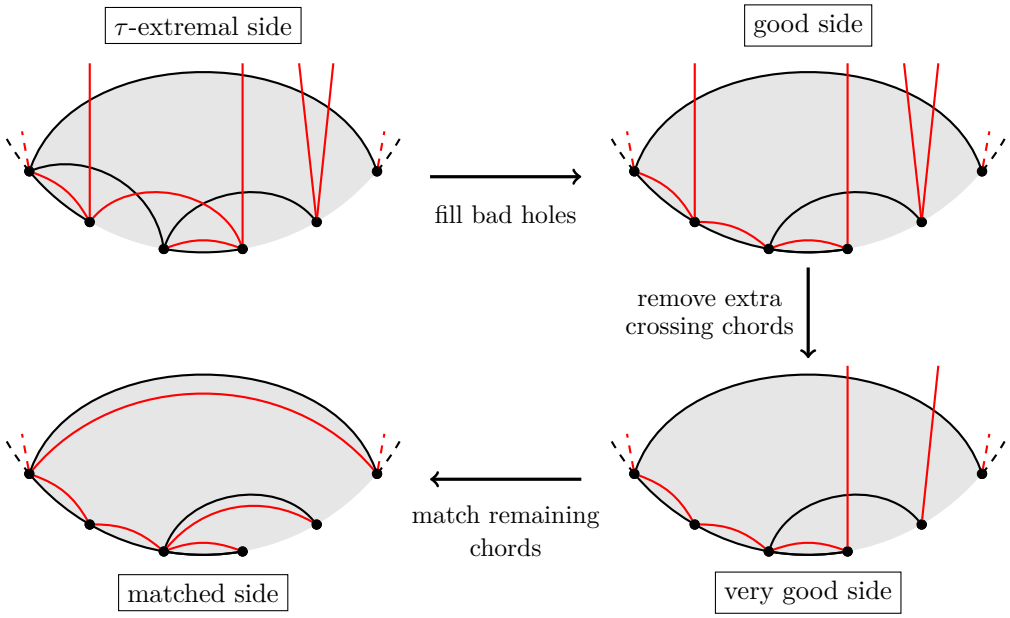


Figure 3 The main steps in the proof of Theorem 3.1. The goal is to match the chords in a side using few steps. (We define τ-extremal, bad, good and very good later).

τ-extremal side Our process starts from a τ-extremal side S of T_I. Let T, T' be a nice pair of trees and $\tau > 2$. We say a side A of a chord e of T is *τ-extremal* for a tree T' if the degree

of A in T' is at most $\tau \cdot k_A$, and, for every side $A' \subsetneq A$ of T', the degree of A' in T is more than $\tau \cdot k_{A'}$. To prove that such a side always exists in T_I or T_F (say T_I by symmetry), we start from an arbitrary side and use an iterative greedy argument until we get a τ-extremal side.

The next step is to refine S until we can show it can be matched using few flips. This refinement will start by removing *bad holes*. A hole h in a side A of T is *bad* w.r.t T' if it is also in a side $B \subsetneq A$ of T', see Figure 4. For our process to yield the desired number of flips, we first need to show that S contains few bad holes. In particular, we prove that S contains $m \leqslant \frac{2}{\tau} k_S$ bad holes w.r.t. T_F.

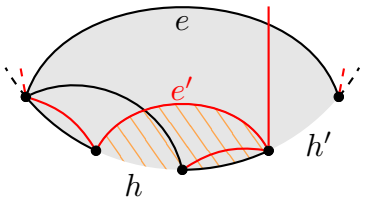


Figure 4 Let T_1 be the black tree and T_2 the red tree. The hole h is a bad hole of the side A (in grey) w.r.t T_2 since it is inside the side of e' included in A.

Refining a τ-extremal side We describe in this paragraph how we refine S into a *very good side*. Let T and T' be a pair of trees. A *good* side A of T with respect to T' is a side of T containing no chord of T' (see Figure 5 for an illustration). A *very good* side A of T (w.r.t. T') is a good side w.r.t T' whose degree in T' is at most k_A.

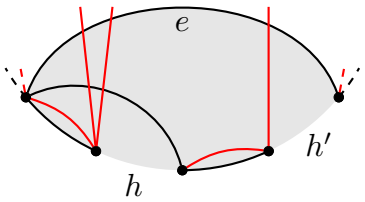


Figure 5 Let T_1 be the black tree and T_2 the red tree. The side A (in grey) of e is a good side of T_1 w.r.t. T_2 since there is no chord of T_2 inside A, but A is not very good w.r.t T_2 since the degree of A in T_2 is $3 > k_A = 2$.

As we already said, we first obtain a good side from S by filling its bad holes with chords of T_I and T_F that have both endpoints in S, therefore we perform $2m$ flips to match m pairs of chords. In the resulting pair of trees (T'_I, T'_F), the size of S is now $k'_S = k_S - m$ and its degree is $d_S - 2m$ with d_S the degree of S in T_F.

Observe that S being good but not very good simply means that there are too many chords crossing the unique chord e on the boundary of S. Hence, we now remove these $d_S - 2m - k'_S$ extra chords crossing e by matching them with a chord of T'_I on a hole which is not in S.

To refine S from a good side into a very good side, we use $2(d_S - 2m - k'_S) = 2(d_S - 2m - k'_S)$ flips to match $d_S - 2m - k'_S = d_S - 2m - k'_S$ pairs of chords.

Very good side Now that S is a very good side, we can match the k'_S chords in S with few flips using the following:

▶ **Lemma 3.2.** *Let T_1 and T_2 be a nice pair of trees, e be a chord of T_1, and A be a very good side of e (w.r.t. T_2). Then, we can match the k_A chords of T_1 in A with chords of T_2 using at most $\frac{5}{3}k_A$ flips in total.*

Bounding the number of flips Let (T_I^*, T_F^*) be the pair of trees obtained after refining S then applying Lemma 3.2 to it. We are now ready to conclude the proof of Theorem 3.1. Our transformation is as follows: first, we transform (T_I, T_F) into (T_I^*, T_F^*) by matching $m+(d_S-2m-k'_S)+k'_S = d_S-m$ pairs using $2m+2(d_S-2m-k'_S)+\frac{5}{3}k'_S = 2d_S-k_S/3-5m/3$ flips. Then, we apply induction on (T_I^*, T_F^*) and get a transformation from T_I^* to T_F^* using at most $c\delta(T_I^*, T_F^*)$ flips.

In order to conclude the proof of Theorem 3.1, we need to make sure that we save enough using Lemma 3.2 to compensate for the expensive refinement process. More precisely, we need that the total number of flips we used to get (T_I^*, T_F^*), namely $2d_S - k_S/3 - 5m/3$, is at most $c(d_S - m)$. Using that $d_S \leqslant \tau k_S$ and that $m \leqslant \frac{2}{\tau}k_S$ (since S is τ-extremal), this boils down to an inequality implying only c and τ, which is satisfied when plugging in the values $\tau = 2+\sqrt{2}$ and $c = \frac{1}{12}(22+\sqrt{2})$.

4 Lower bounds

Our second set of results consists in proving stronger lower bounds in terms of the symmetric difference of the two trees. In particular, we disprove Conjecture 1.2:

▶ **Theorem 4.1.** *For every $k > 0$, there exist two trees T_k and T'_k such that $\delta(T_k, T'_k) = 3k$ and every flip sequence between T_k and T'_k has length at least $5k = \frac{5}{3}\delta(T_k, T'_k)$.*

The proof of Theorem 4.1 consists in first providing two spanning trees T_1, T'_1 on 8 vertices for which $\delta(T_1, T'_1) = 3$ and such that the minimal flip sequence between T_1 and T'_1 needs 5 flips (see Figure 6). We can prove that if we glue many instances of (T_1, T'_1) appropriately, we can obtain a similar example with arbitrarily large value of k.

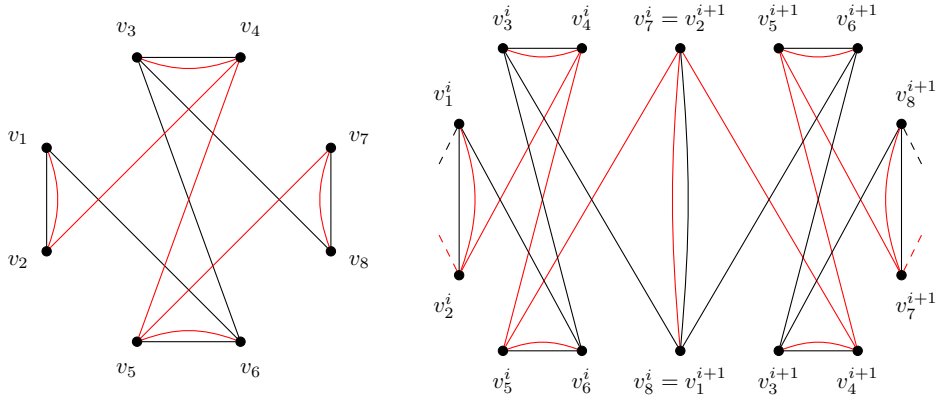


Figure 6 On the left, the tree T_1 in black and the tree T'_1 in red. On the right, an example of two copies of T_1 and T'_1 glued together.

We have not found any example for trees T_1, T_2 for which a flip sequence of length more than $\frac{5}{3}\delta(T_1, T_2)$ is necessary. We therefore leave the following as an open problem:

▶ **Question 4.2.** Let C be a set of points in convex position and T_1, T_2 two non-crossing spanning trees on C. Does there always exist a flip sequence between T_1 and T_2 of length at most $\frac{5}{3}\delta(T_1, T_2)$?

References

1 Oswin Aichholzer, Brad Ballinger, Therese Biedl, Mirela Damian, Erik D Demaine, Matias Korman, Anna Lubiw, Jayson Lynch, Josef Tkadlec, and Yushi Uno. Reconfiguration of non-crossing spanning trees. *arXiv preprint arXiv:2206.03879*, 2022.

2 David Avis and Komei Fukuda. Reverse search for enumeration. *Discrete Appl. Math.*, 65(1-3):21–46, 1996. First International Colloquium on Graphs and Optimization (GOI), 1992 (Grimentz).

3 Nicolas Bousquet, Lucas De Meyer, Théo Pierron, and Alexandra Wesolek. Reconfiguration of plane trees in convex geometric graphs. *arXiv preprint arXiv:2310.18518*, 2023.

4 Nicolas Bousquet, Valentin Gledel, Jonathan Narboni, and Théo Pierron. A note on the flip distance between non-crossing spanning trees. *arXiv preprint arXiv:2303.07710*, 2023.

5 M.C. Hernando, F. Hurtado, A. Márquez, M. Mora, and M. Noy. Geometric tree graphs of points in convex position. *Discrete Applied Mathematics*, 93(1):51–66, 1999. 13th European Workshop on Computational Geometry CG '97. URL: `https://www.sciencedirect.com/science/article/pii/S0166218X99000062`, `doi:https://doi.org/10.1016/S0166-218X(99)00006-2`.

A Note on Mixed Linear Layouts of Planar Graphs

Michael Kaufmann[1] and Maria Eleni Pavlidi[2]

1 **Department of Computer Science, University of Tübingen, Germany**
michael.kaufmann@uni-tuebingen.de
2 **Department of Mathematics, University of Ioannina, Greece**
m.e.pavlidi@uoi.gr

Abstract

In this work, we study mixed linear layouts of graphs. Our motivation stems from a result by Pupyrev [15], who disproved a conjecture by Heath and Rosenberg [14] by showing the existence of planar graphs not admitting layouts with one stack and one queue. Since stacks and queues form special cases of the recently-introduced riques, we strengthen this result by showing that there exist planar graphs that do not admit a layout with one rique and either one stack or one queue.

1 Introduction

Linear layouts of graphs [12] have a long tradition of research, e.g., in Algorithm Design, Combinatorics, Graph Theory and Graph Drawing. The ones that we consider in this paper are defined using an associated data structure [3, 4, 9, 14]. The task is to find a so-called *linear order* of the vertices of the input graph and a partition of its edges into as few parts (called *pages*) as possible, such that the edges of each part can be processed by the given data structure. Namely, assuming that the vertices are left-to-right ordered according to their linear order, each edge is added to the data structure when its left endpoint is encountered in the order and is removed from the data structure when its right endpoint is encountered.

In this context, the most prominent types of linear layouts are the *stack* [9, 17] and the *queue layouts* [11, 14] that are defined using the stack and the queue data structures, respectively. A page of the former is called *stack* and does not allow two crossing edges, while a page of the latter is called *queue* and does not allow two nesting edges; see Fig. 2. Both these layouts form special cases of the so-called *deque layouts* [3], which are defined using the *double-ended queue* (or *deque*, for short) data structure. It is well-known that a page of a deque layout, called *deque*, has the following properties: the union of (i) two stacks, or (ii) two queues or (iii) a stack and a queue forms a deque [3]. In particular, (i) and (ii) imply that the *deque-number* (i.e., the minimum required number of deques over all deque layouts) of a graph cannot be more than half its stack- or its queue-number (i.e., the corresponding required numbers of stacks and queues, respectively).

In this work, we focus on planar input graphs and mixed linear layouts consisting of two pages; one that is either a stack or a queue, and one that is *rique* [4]; such a page is defined by the *restricted-input double-ended queue* (or *rique*, for short) data structure; refer, e.g., to Fig. 1 for a sample linear layout consisting of a single rique and to Section 2 for formal definitions. The rique data structure forms a special case of the deque data structure as follows. While in a deque insertions and removals occur both at the head and at the tail of it, in a rique insertions occur only at the head (removals occur both at the head and at the tail). Our work is motivated by a result by Pupyrev [15], who disproved a conjecture by Heath and Rosenberg [14] by showing the existence of planar graphs not admitting mixed layouts with one stack and one queue. Here, we show that there exist planar graphs that do not admit mixed layouts with one rique and either one stack or one queue. In other words,

40th European Workshop on Computational Geometry, Ioannina, Greece, March 13–15, 2024.
This is an extended abstract of a presentation given at EuroCG'24. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

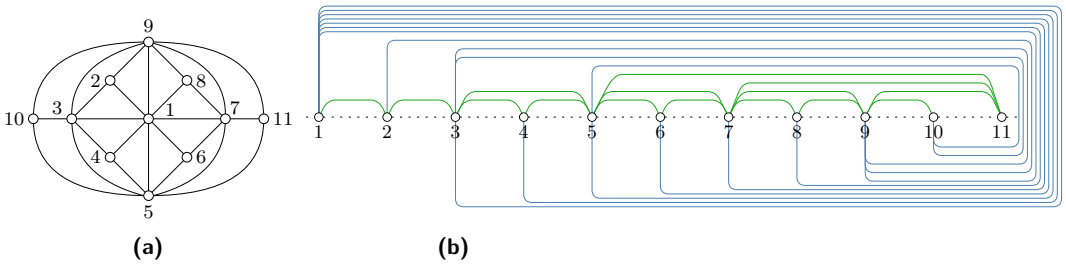


Figure 1 Illustration of: (a) the Goldner-Harary graph without the edge connecting its topmost vertex with its bottommost one, and (b) a rique layout of it with a single rique, in which the green edges are head-head, while the blue ones are head-tail.

with respect to the previously mentioned result by Pupyrev [15], our result implies that substituting one of the pages by a rique is still not enough for a positive answer to Heath and Rosenberg's conjecture.

Our work is also related to the rique-number (i.e., the minimum required number of riques over all rique layouts) of planar graphs. More precisely, since the stack-number of planar graphs is 4 [7, 17], it follows that the deque-number of planar graphs is 2, as also observed by Auer et al. [3]. So, it is natural to ask whether the rique-number of planar graphs is also 2; the obvious upper bound is 4, since a stack page is trivially a rique [4]. Unfortunately, we have not managed to completely settle this question, as our result does not close the gap on the rique-number of planar graphs (this ranges between 2 and 4, as noted). It forms, however, an indication that it might be not 2 (as observed above, both stacks and queues form special cases of riques).

2 Preliminaries

A *vertex order* $\prec$ of a graph G is a total order of its vertices, such that for any two vertices u and v of G, we write $u \prec v$ if and only if u precedes v in the order. Let F be a set of $k \geq 2$ pairwise independent edges (u_i, v_i) of G, that is, $F = \{(u_i, v_i);\ i = 1, \ldots, k\}$. If $u_1 \prec \cdots \prec u_k \prec v_k \prec \cdots \prec v_1$, then the edges of F form a *k-rainbow*, while if $u_1 \prec \cdots \prec u_k \prec v_1 \prec \cdots \prec v_k$, then the edges of F form a *k-twist*; see Fig. 2. Two edges that form a 2-twist (2-rainbow) are commonly referred to as *crossing* (*nested*). A *stack* is a set of pairwise non-crossing edges in $\prec$, while a *queue* is a set of pairwise non-nested edges in $\prec$.

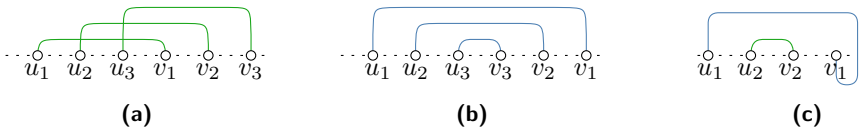


Figure 2 Illustration of: (a) a 3-twist (i.e., three pairwise crossing edges), (b) a 3-rainbow (i.e., three pairwise nesting edges), and (c) a rique page with two edges; a head-head and a head-tail.

A *rique* is a set of edges that does not contain three edges (a, a'), (b, b') and (c, c') such that $a \prec b \prec c \prec b' \prec \{a', c'\}$ in $\prec$ [4]. A more intuitive definition of a rique is the following. Assume that the vertices of the input graph are arranged on a horizontal line ℓ from left to right according to $\prec$ (say, w.l.o.g., equidistantly). Then, each edge (u, v) with $u \prec v$ can be represented either (i) as a semi-circle that is completely above ℓ connecting u and v, or (ii) as two semi-circles on opposite sides of ℓ, one that starts at u, lies above ℓ and ends at a

point p of ℓ to the right of the last vertex of $\prec$ and one that starts at p, lies below ℓ and ends at v. Then, a rique is a set of edges each of which can be represented with one of the two types (i) or (ii) that avoids crossings (such a representation is called *cylindric* in [3, 4]); see Fig. 2c. A type-(i) edge is called *head-head*, while a type-(ii) edge is called *head-tail*[1]; refer to the green and blue edges of Fig. 2c, respectively. It is not difficult to see that the subset of the head-head edges of a rique induces a stack in $\prec$, while the corresponding set of the head-tail edges of a rique induces a queue in $\prec$ [4]. Thus, in a sense, a rique is a special case of a stack and a queue; not every pair of a stack and a queue however forms a rique.

3 Our result

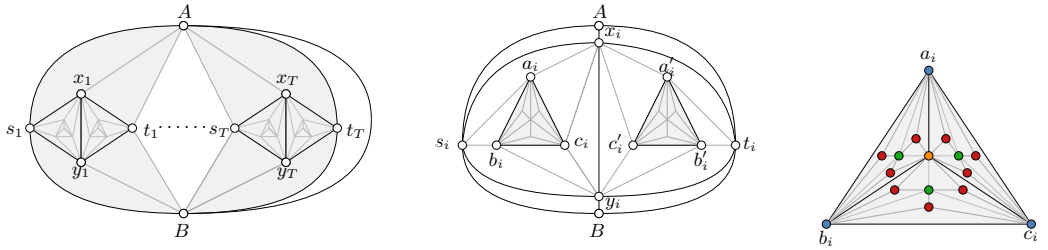
In this section, we prove that there exist planar graphs that do not admit mixed layouts with one rique and either one stack or one queue. To achieve this, we establish a recursive definition of a planar graph (Section 3.1) and we prove that every layout of it with one rique and either one stack or one queue contains at least two edges that either cross in the cylindric representation of the rique or that cross (nest) in the stack (queue). Our proof contains several combinatorial arguments (Section 3.2) but the case analysis that needs to be performed in order to obtain the desired result is deferred to the computer (Section 3.3). The reason for this is that there exist several cases that one needs to consider arising from the two different types that each edge may have; in addition to this, each edge assigned to the rique may be head-head or head-tail (Section 3.4). For the last step in the proof, we exploit a known formulation of the problem of testing whether a given (not necessarily planar) graph admits a layout with a certain number of pages (stacks, queues or riques) as a SAT instance [8]. In our approach, we use properties from our combinatorial analysis to reduce the size of the search space and to introduce several symmetry-breaking constraints in the SAT instance, which made the latter verifiable in reasonable amount of time (less than 10 minutes) using a standard SAT solver [10]. Note that, the actual implementation has become part of [5] and the corresponding code is available to the community as part of the following `GitHub` repository:

`https://github.com/linear-layouts/SAT`

3.1 The graph supporting the proof

We start with the description of the graph, which contains a set of $2T$ independent vertices s_i and t_i, with $1 \leq i \leq T$, called *terminals*. For each $i = 1, \ldots, T$, we connect each of s_i and t_i to two adjacent vertices A and B, called *poles*. Each pair of such terminals delimits a so-called *component* C_i in G_T (colored gray in Fig. 3a) as follows: For $i = 1, \ldots, T-1$, we add two vertices x_i and y_i that are connected by an edge; each of these two vertices is connected with s_i and t_i; additionally, x_i is connected with A, and y_i with B. In a second step, we construct a 3-cycle $\langle a_i, b_i, c_i \rangle$ and we connect vertex a_i with x_i and s_i, vertex b_i with s_i and y_i, and vertex c_i with x_i and y_i. Symmetrically, we construct a 3-cycle $\langle a'_i, b'_i, c'_i \rangle$ and we connect vertex a'_i with x_i and t_i, vertex b'_i with t_i and y_i, and vertex c'_i with x_i and y_i; see Fig. 3b. Aiming to introduce in G_T several subgraphs, which are neither 2-stack nor 2-queue embeddable [1, 13], the construction continues by *stellating* several already formed

[1] Note that a deque additionally supports *tail-tail edges* (semi-circles below ℓ) and *tail-head edges* (two semi-circles, one that starts at the left endpoint of the edge, lies below ℓ and ends at a point p of ℓ to the right of the last vertex of $\prec$ and one that starts at p, lies above ℓ and ends at the other endpoint).



(a) Graph G_T **(b)** Component C_i **(c)** Stellating $\langle a_i, b_i, c_i \rangle$

Figure 3 Illustrations for the construction of graph G_T: Each gray subgraph in (a) corresponds to a copy of the graph in (b); each gray subgraph in (b) corresponds to a copy of the graph in (c).

faces, where the operation of stellating a face f bounded by a cycle C introduces a vertex u in f and connects u to each of the vertices of C. In particular, we proceed by stellating the resulting faces $\langle a_i, b_i, c_i \rangle$ and $\langle a'_i, b'_i, c'_i \rangle$, introducing two new vertices d_i and d'_i, respectively (refer to the yellow vertex in Fig. 3c). Afterwards, a second round of stellations occurs involving the faces $\langle a_i, d_i, b_i \rangle$, $\langle a_i, d_i, c_i \rangle$, $\langle b_i, d_i, c_i \rangle$, $\langle a'_i, d'_i, b'_i \rangle$, $\langle a'_i, d'_i, c'_i \rangle$ and $\langle b'_i, d'_i, c'_i \rangle$ (refer to the green vertices in Fig. 3c). The final graph G_T is obtained by stellating each of the newly formed faces once more (refer to the red vertices in Fig. 3c). We refer to two vertices (edges) of two different components C_i and C_j that correspond to the same vertex (edge) in the construction above as *twin* vertices (edges), e.g., the vertices $x_1, \ldots, x_T$ are twin vertices, while the edges $(A, x_1), (A, x_2), \ldots, (A, x_T)$ are twin edges.

3.2 The combinatorial part of the proof

Assume that G_T has a mixed linear layout $\mathcal{L}$ with one rique and either one stack or one queue. By symmetry, we may assume w.l.o.g. that $A \prec B$ and $s_i \prec t_i$ holds in $\mathcal{L}$, for each $i = 1, \ldots, T$. Since each component in G_T is of fixed size, if we set T to be large enough, then we can assume by pigeonhole principle that there is a certain number, say k, of copies of components, w.l.o.g. $C_1, \ldots, C_k$, of G_T that have exactly the same layout in $\mathcal{L}$. Namely, for any two components C_i and C_j, with $1 \leq i, j \leq k$, (i) the order in which any two vertices u and v of C_i appear in $\mathcal{L}$ is the same as their twin vertices u' and v' of C_j, while (ii) any two twin edges of C_i and C_j are assigned to the same page and additionally are of the same type (e.g., both head-head or both head-tail) if assigned to the rique of $\mathcal{L}$. Using Ramsey's theory (and assuming that T is even larger), we can further guarantee that (iii) each group of twin edges form a rainbow or a twist or a necklace in the underlying linear order.

In the following, we assume that T is large enough such that we can identify $k = 4$ components C_1, C_2, C_3 and C_4 with the aforementioned properties. In this case, by symmetry, we can further assume that $t_1 \prec t_2 \prec t_3 \prec t_4$. Let w_1 be any vertex connected to t_1 that is not one of the poles A or B of G_T. Let w_2, w_3 and w_4 be the twins of w_1 in C_2, C_3 and C_4, respectively. Since the edges (t_1, w_1), (t_2, w_2), (t_3, w_3) and (t_4, w_4) are twin edges (thus, forming a rainbow or a twist or a necklace), it follows that either $w_1 \prec w_2 \prec w_3 \prec w_4$ or $w_4 \prec w_3 \prec w_2 \prec w_1$ holds in $\mathcal{L}$. Extending this argument to the neighbors of w_1, w_2, w_3 and w_4 and further, one may conclude that for every quadruple of twin vertices z_1, z_2, z_3 and z_4 in C_1, C_2, C_3 and C_4, respectively, it holds that either $z_1 \prec z_2 \prec z_3 \prec z_4$ or $z_4 \prec z_3 \prec z_2 \prec z_1$. Twin vertices satisfying this property are said to be *monotonically ordered*.

3.3 The computer-aided part of the proof

With the observations that we made in Section 3.2, we were able to prove that, for large enough values of T, graph G_T does not admit a mixed linear layout with one rique and either a stack or a queue using the SAT formulation described in [8]. More precisely, assuming to the contrary that G_T admits such a layout, the subgraph of G_T formed by the poles A and B and by the four components C_1, C_2, C_3 and C_4 that we described in Section 3.2 must also admit a corresponding layout under the following constraints:

1. Pole A precedes pole B.
2. Terminal s_i precedes terminal t_i for each $i = 1, 2, 3, 4$.
3. Every quadruple of twin edges is assigned to the same page.
4. For every quadruple of twin vertices, we require them to be (i) monotonically ordered, (ii) either all before or all after pole A and (iii) either all before or all after pole B.

Note that, by our discussion in Section 3.2, Constraints 1–4 preserve the satisfiability of the SAT instance. However, with the online implementation [6] of [8], which already provides support for encoding Constraints 1–4 in SAT, we verified that the subgraph of G_T formed by the poles A and B and by the four components C_1, C_2, C_3 and C_4 admits a mixed linear layout neither with one rique and one stack nor with one rique and one queue when Constraints 1–4 are imposed, contradicting the fact that G_T also admits such a layout. The total time needed to verify the unsatisfiability was less than 10 minutes on a single-node 4-core 3.3 GHz Intel Core i5-4590 machine with 16GM RAM. We summarize this finding in the next theorem.

▶ **Theorem 3.1.** *There exist planar graphs that do not admit mixed linear layouts with one rique and either one stack or one queue.*

3.4 Some remarks towards a purely combinatorial proof

We conclude this section by mentioning that a purely combinatorial proof is possible to be derived by further extending the arguments that we introduced in Section 3.2. As a matter of fact, the next step in the proof is to consider the six possible permutations that may arise for the poles A and B with respect to the terminals s_1, t_1, s_2, t_2, s_3, t_3, s_4 and t_4 of the components C_1, C_2, C_3 and C_4, namely: **(P.1)** $s_i \prec A \prec B \prec t_i$, **(P.2)** $A \prec s_i \prec B \prec t_i$, **(P.3)** $s_i \prec A \prec t_i \prec B$, **(P.4)** $A \prec B \prec s_i \prec t_i$, **(P.5)** $s_i \prec t_i \prec A \prec B$ and **(P.6)** $A \prec s_i \prec t_i \prec B$. Then, one has to argue on the feasible positions of the remaining (twin) vertices contained in C_1, C_2, C_3 and C_4 within each of P.1–P.6. However, these positions depend on the page that each edge is assigned (rique, stack or queue) and of its type (head-head or head-tail, if the edge is in the rique). This make the number of starting cases for the edges connecting A, B and the terminals s_1, t_1, s_2, t_2, s_3, t_3, s_4 and t_4 already very large and the resulting purely combinatorial proof very tedious.

4 Conclusions

In this work, we demonstrated planar graphs that do not admit mixed linear layouts with one rique and either one stack or one queue strengthening a corresponding result by Pupyrev [15] limited to layouts with one stack and one queue. We also made a step towards answering a question in [8] related to the rique number of planar graphs that ranges between 2 and 4; we feel that to show a lower bound of 3 is a realistic goal. Nevertheless, we consider closing this gap as an interesting open problem for future consideration.

Related to our work is also a result by Angelini et al. [2], who also provided a strengthened version of the result by Pupyrev [15] by demonstrating 2-trees that do not admit mixed linear layouts with one stack and one queue. Their result implies that 2-trees do not admit rique layouts with one rique. On the other hand, 2-trees admit stack layouts with two stacks [16], which trivially implies that they also admit mixed linear layouts with one rique and one stack. In this regard, it would be interesting to study whether this result transfers to planar 3-trees, namely, whether planar 3-trees admit mixed linear layouts with one rique and either one stack or one queue; note that planar 3-trees admit stack layouts with three stacks [13], which implies that they also admit mixed linear layouts with one deque and one stack. So, in a sense, our question is whether substituting the deque page with a rique one still suffices for such a positive result.

Acknowledgements. The authors thank M. A. Bekos for numerous fruitful discussions.

References

1 Jawaherul Md. Alam, Michael A. Bekos, Martin Gronemann, Michael Kaufmann, and Sergey Pupyrev. Queue layouts of planar 3-trees. *Algorithmica*, 82(9):2564–2585, 2020. doi:10.1007/s00453-020-00697-4.

2 Patrizio Angelini, Michael A. Bekos, Philipp Kindermann, and Tamara Mchedlidze. On mixed linear layouts of series-parallel graphs. *Theor. Comput. Sci.*, 936:129–138, 2022. URL: https://doi.org/10.1016/j.tcs.2022.09.019, doi:10.1016/J.TCS.2022.09.019.

3 Christopher Auer, Christian Bachmaier, Franz-Josef Brandenburg, Wolfgang Brunner, and Andreas Gleißner. Plane drawings of queue and deque graphs. In Ulrik Brandes and Sabine Cornelsen, editors, *Graph Drawing*, volume 6502 of *LNCS*, pages 68–79. Springer, 2010. doi:10.1007/978-3-642-18469-7_7.

4 Michael A. Bekos, Stefan Felsner, Philipp Kindermann, Stephen G. Kobourov, Jan Kratochvíl, and Ignaz Rutter. The rique-number of graphs. In Patrizio Angelini and Reinhard von Hanxleden, editors, *Graph Drawing and Network Visualization*, volume 13764 of *LNCS*, pages 371–386. Springer, 2022. doi:10.1007/978-3-031-22203-0_27.

5 Michael A. Bekos, Mirco Haug, Michael Kaufmann, and Julia Männecke. An online framework to interact and efficiently compute linear layouts of graphs. *CoRR*, abs/2003.09642, 2020. URL: https://arxiv.org/abs/2003.09642, arXiv:2003.09642.

6 Michael A. Bekos, Mirco Haug, Michael Kaufmann, and Julia Männecke. An online framework to interact and efficiently compute linear layouts of graphs. *CoRR*, abs/2003.09642, 2020. online version http://alice.math.uoi.gr/; source code available at https://github.com/linear-layouts/SAT. URL: http://arxiv.org/abs/2003.09642, arXiv:2003.09642.

7 Michael A. Bekos, Michael Kaufmann, Fabian Klute, Sergey Pupyrev, Chrysanthi N. Raftopoulou, and Torsten Ueckerdt. Four pages are indeed necessary for planar graphs. *J. Comput. Geom.*, 11(1):332–353, 2020. URL: https://journals.carleton.ca/jocg/index.php/jocg/article/view/504.

8 Michael A. Bekos, Michael Kaufmann, Maria Eleni Pavlidi, and Xenia Rieger. On the deque and rique numbers of complete and complete bipartite graphs. In Denis Pankratov, editor, *Canadian Conference on Computational Geometry*, pages 89–95, 2023. URL: https://wadscccg2023.encs.concordia.ca/assets/pdf/CCCG_2023_proc.pdf.

9 Frank Bernhart and Paul C. Kainen. The book thickness of a graph. *J. Comb. Theory, Ser. B*, 27(3):320–331, 1979. doi:10.1016/0095-8956(79)90021-2.

10 Armin Biere. Lingeling, Plingeling and Treengeling entering the SAT Competition 2013. In A. Balint, A. Belov, M. Heule, and M. Järvisalo, editors, *SAT Competition 2013*, volume B-

2013-1 of *Department of Computer Science Series of Publications B*, pages 51–52. University of Helsinki, 2013. Source code available at `https://github.com/arminbiere/lingeling`.

11 Vida Dujmovic, Louis Esperet, Cyril Gavoille, Gwenaël Joret, Piotr Micek, and Pat Morin. Adjacency labelling for planar graphs (and beyond). In *FOCS*. IEEE, 2020. `doi:10.1109/FOCS46700.2020.00060`.

12 Vida Dujmović and David R. Wood. On linear layouts of graphs. *Discrete Mathematics & Theoretical Computer Science*, 6(2):339–358, 2004. URL: `http://dmtcs.episciences.org/317`.

13 Lenwood S. Heath. Embedding planar graphs in seven pages. In *FOCS*, pages 74–83. IEEE Computer Society, 1984. `doi:10.1109/SFCS.1984.715903`.

14 Lenwood S. Heath and Arnold L. Rosenberg. Laying out graphs using queues. *SIAM J. Comput.*, 21(5):927–958, 1992. `doi:10.1137/0221055`.

15 Sergey Pupyrev. Mixed linear layouts of planar graphs. In Fabrizio Frati and Kwan-Liu Ma, editors, *Graph Drawing and Network Visualization*, volume 10692 of *LNCS*, pages 197–209. Springer, 2017. `doi:10.1007/978-3-319-73915-1\_17`.

16 S. Rengarajan and C. E. Veni Madhavan. Stack and queue number of 2-trees. In Ding-Zhu Du and Ming Li, editors, *COCOON*, volume 959 of *LNCS*, pages 203–212. Springer, 1995. `doi:10.1007/BFb0030834`.

17 Mihalis Yannakakis. Embedding planar graphs in four pages. *J. Comput. Syst. Sci.*, 38(1):36–67, 1989. `doi:10.1016/0022-0000(89)90032-9`.

Approximating the Fréchet Distance in Graphs with Low Highway Dimension*

Anne Driemel[1] and Marena Richter[1]

1 Department of Computer Science, University of Bonn, Germany

Abstract

In this paper, we study algorithms for the discrete Fréchet distance in graphs with low highway dimension. We describe a $(\frac{5}{3}+\varepsilon)$-approximation algorithm for the Fréchet distance between a shortest path P with n vertices and an arbitrary walk Q with m vertices in a graph $G=(V,E)$. The algorithm makes use of a collection of sparse shortest paths hitting sets which are precomputed for the graph G. After preprocessing, the algorithm has running time $\mathcal{O}\left(n\log D + m(h\log h\log D)^2\right)$, where h is the highway dimension and D is the diameter of G. The preprocessing for the graph is polynomial in $|G|$ and $1/\log(1+\varepsilon)$ and uses $\mathcal{O}\left(|V|\log D(1/\log(1+\varepsilon)+h\log h)\right)$ space.

1 Introduction

The notion of the Fréchet distance between polygonal curves was introduced to Computational Geometry by Alt and Godau in 1992 [5]. They also gave a $\mathcal{O}(n^2\log n)$ algorithm to compute the continuous Fréchet distance between two curves with $\mathcal{O}(n)$ vertices in Euclidean metric spaces of fixed dimension. Bringmann showed that neither the discrete nor the continuous Fréchet distance between two curves can be computed in time $\mathcal{O}(n^{2-\varepsilon})$ for any $\varepsilon>0$ unless the orthogonal vectors hypothesis fails [9] and there are known algorithms showing that this lower bound is tight in the discrete [4] and the continuous case [10]. There are various faster exact and approximation algorithms known in specialized settings for the continuous [6, 7, 11, 12, 13, 14] and discrete Fréchet distance [7, 15, 17, 19]. Some of these algorithms work with a preprocessing that stores one curve such that computing the Fréchet distance to any other curve can be done efficiently [13, 15, 17, 19]. All of the mentioned algorithms require that the curves are embedded in some sort of underlying metric space.

We consider the discrete Fréchet distance between walks in a graph with respect to the shortest path metric. This can for example be used to determine similarities between trajectories in street networks, which has been a question of interest in the past [8, 18]. Driemel, van der Hoog and Rotenberg showed that for this variant of the problem the near-quadratic conditional lower bound still holds [16]. They also study approximation algorithms for the setting that one of the walks is κ-straight, that is a near-shortest path. A path is κ-straight if any subpath between any two vertices p and q along the path has length at most κ times the shortest path distance from p to q. In particular, a shortest path is κ-straight for $\kappa=1$. They show that one can compute a $(1+\varepsilon)$-approximation of the Fréchet distance between a κ-straight path P and any walk Q in a planar graph G in $\mathcal{O}\left(|G|\log|G|/\sqrt{\varepsilon}+|P|+\frac{\kappa}{\varepsilon}|Q|\right)$ time and they give a $(\kappa+1)$-approximation algorithm, with running time in $\mathcal{O}\left((|P|+|Q|)\log^{3+o(1)}|G|\right)$, after preprocessing. For general graphs, the

* This work was funded by 390685813 (Germany's Excellence Strategy – EXC-2047/1: Hausdorff Center for Mathematics); 416767905; and funded by the Lamarr Institute for Machine Learning and Artificial Intelligence lamarr-institute.org.

40th European Workshop on Computational Geometry, Ioannina, Greece, March 13–15, 2024.
This is an extended abstract of a presentation given at EuroCG'24. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

second algorithm has running time in $\mathcal{O}((|P| + |Q|) \cdot T(G) \cdot \log D)$, where $T(G)$ denotes the time for a distance query in G.

In this paper, we focus on the case where the graph $G = (V, E)$ is not necessarily planar, but has low highway dimension, a property which has been studied in the context of road networks before [2]. We give an algorithm that computes a $(\frac{5}{3} + \varepsilon)$-approximation to the discrete Fréchet distance between a shortest path P and any walk Q in G. After preprocessing the graph, the running time of the algorithm is in $\mathcal{O}\left(|P| \log D + |Q| \left(h \log h \log D\right)^2\right)$, where h is the highway dimension of the graph and D is the diameter of G. The preprocessing is polynomial in $|V|$ and $1/\log(1+\varepsilon)$ and uses $\mathcal{O}(|V| \log D(1/\log(1+\varepsilon) + h \log h))$ space.

1.1 Highway Dimension

Abraham, Delling, Fiat, Goldberg and Werneck introduced multiple definitions of the highway dimension over the years [3, 1, 2]. We work with the latest definition from 2016 [2].

The intuition behind a low highway dimension is that there exists a small set of vertices ("hubs") such that for any point in the graph every shortest path to a destination far away visits at least one of these hubs. Abraham et al. argue that this is a realistic model of road networks [2]. A low highway dimension is especially helpful for shortest path computations. We use these hitting sets of long shortest paths for Fréchet distance queries.

Let $G = (V, E)$ be a graph with non-negative integer edge weights ℓ that satisfy the triangle inequality and can all be expressed in a word of $\Theta(\log |V|)$ bits. A *walk* in G is a sequence of vertices with an edge between any two successive vertices. A *path* is a walk where every vertex is visited at most once. We assume that shortest paths in G are unique.

Let P be a shortest path with weight $\ell(P) > r$ for some value $r > 0$. Denote by $V(P)$ the set of vertices in P. In [2], Abraham et al. call a shortest path P' an *r-witness* for a shortest path P if $\ell(P') > r$ and P is either equal to P' or it arises from P' by deleting one or both end vertices of P'. All shortest paths that have an r-witness are called *r-significant*. This means that also single vertices can be r-significant. Denote by $\mathcal{P}_r$ all r-significant paths.

▶ **Definition 1.1** (Highway dimension [2])**.** The *highway dimension* h of the graph $G = (V, E)$ is the smallest integer such that for any real value $r > 0$ and $v \in V$ there exists a set $H \subseteq V$ with $|H| \leq h$ and $H \cap V(P) \neq \emptyset$ for all r-significant paths P with an r-witness P' satisfying $\operatorname{dist}(v, P') \coloneqq \min_{w \in V(P')} \operatorname{dist}(v, w) \leq 2r$.

The sets H exist separately for every vertex and radius. Abraham et al. give a related definition of sparse hitting sets:

▶ **Definition 1.2** (Sparse Shortest Path Hitting Set (SPHS) [2])**.** For $r > 0$ an *(h, r)-SPHS* is a set $C \subseteq V(G)$ such that $|B_{2r}(v) \cap C| \leq h$ for all $v \in V(G)$ and $V(P) \cap C \neq \emptyset$ for all $P \in \mathcal{P}_r$, where $B_{2r}(v)$ is the set of vertices in G that have distance at most $2r$ to v.

It can be shown that there always exists an (h, r)-SPHS in a graph with highway dimension h [2]. Note that the set $B_{2r}(v) \cap C$ is very similar to the set H for v and r in the definition of the highway dimension but it might not hit all necessary r-significant paths even though C hits all these paths. One can extend this definition even further in the following way:

▶ **Definition 1.3** (μ-multiscale SPHS)**.** For $\mu > 1$ a *μ-multiscale* SPHS with value h of G is a collection of sets C_i for $0 \leq i \leq \left\lceil \frac{\log D}{\log \mu} \right\rceil$, where C_i is a (h, μ^{i-1})-SPHS and $D \coloneqq \max_{v,w \in V(G)} \operatorname{dist}(v, w)$ is the diameter of G.

In [2] the definition of a multiscale SPHS matches our definition of a 2-multiscale SPHS. Note that $\log D \in \mathcal{O}(\log |V|)$ because all edge weights can be expressed in a word of $\Theta(\log |V|)$

bits. Computing an (h, r)-SPHS in a graph with highway dimension h can be NP-hard. However, we can compute an approximation in polynomial time (Theorem 8.2 in [2]), which leads to the following theorem:

▶ **Theorem 1.4.** *In a graph with highway dimension h, we can compute a μ-multiscale SPHS with value $\mathcal{O}(h \log h)$ in running time polynomial in size(G) and $(\log \mu)^{-1}$.*

Using 2-multiscale SPHS, one can create a fast distance oracle in G as it is discussed in [2]:

▶ **Theorem 1.5** (see Theorem 8.3 in [2])**.** *With a polynomial-time preprocessing and $\mathcal{O}(|V| \log D \cdot h \log h)$ space we can preprocess a graph $G = (V, E)$ with highway dimension h such that a distance query between any two vertices takes $\mathcal{O}(h \log h \log D)$ time.*

1.2 Fréchet Distance

We follow [16] in our definition of the discrete Fréchet distance in a graph. The discrete Fréchet distance is a similarity measure between two walks in a graph. Assume we are given a graph G with a metric weight function w. Let $P = (p_1, \ldots, p_n)$ and $Q = (q_1, \ldots, q_m)$ be walks in G. We denote by $[n] \times [m] \subset \mathbb{N} \times \mathbb{N}$ the integer lattice of n by m integers and say that an ordered sequence F of pairs in $[n] \times [m]$ is an *xy-monotone discrete walk* if for every consecutive pair $(i, j), (k, l) \in F$, we have $k \in \{i, i + 1\}$ and $l \in \{j, j + 1\}$.

▶ **Definition 1.6** (see Section 2 in [16])**.** The *strong discrete Fréchet distance* of two walks $P = (p_1, p_2, \ldots, p_n)$ and $Q = (q_1, q_2, \ldots, q_m)$ is the minimum over the maximum pairwise distance of any xy-monotone discrete walk F from $(1, 1)$ to (n, m):

$$D_{\mathcal{F}}(P, Q) := \min_{F} \max_{(i,j) \in F} \operatorname{dist}(p_i, q_i).$$

For brevity we just call this the Fréchet distance of P and Q. One can verify that the Fréchet distance satisfies the triangle inequality.

Given two walks P, Q and some real value d, we define a $|Q| \times |P|$ matrix M which we call the *free-space matrix* M_d. The i-th column of M_d corresponds to the i-th vertex in P and the j-th row corresponds to the j-th vertex in Q. We assign to each matrix cell $M_d[i, j]$ the integer -1 if $\operatorname{dist}(p_i, q_j) \leq d$, and 0 if $\operatorname{dist}(p_i, q_j) > d$.

The Fréchet distance between two walks P and Q is at most d, iff there exists an xy-monotone discrete walk F from $(1, 1)$ to (n, m) such that $\forall (i, j) \in F$ we have $M_d[i, j] = -1$.

2 Algorithm

On a high level, our algorithm first computes a simplification of the shortest path P using a certain SPHS. It then runs a BFS on a free space matrix between the walk Q and the simplification of P to approximately determine, whether $D_{\mathcal{F}}(P, Q) \leq \delta$. In the analysis, we show that the distance between the chosen simplification and P can be bounded from above, which then bounds the approximation factor of the algorithm. Then, we prove that the BFS does not visit too many vertices making use of the fact that almost all vertices on the simplification belong to the same SPHS. This then bounds the runtime of our algorithm. In the end we choose the values for the multiscale SPHS, the simplification and the free space matrix to achieve the desired approximation factor.

Throughout the rest of the paper, let $P = \langle p_1, \ldots, p_n \rangle$ be a shortest path in G and let $Q = \langle q_1, \ldots, q_m \rangle$ be an arbitrary walk in G. We denote by $|P|$ the number of vertices in P.

First, we focus on the case, where we are already given a certain SPHS and see that we can compute an approximation of the Fréchet distance between P and Q quite fast, using only a subset of the vertices of P.

Let H_δ be a (h', δ)-SPHS for some $h' \in \mathbb{N}$ and $\delta \geq 0$. Define by P^δ the subsequence of vertices in P, where we start in p_1, then only visit the points of $P \cap H_\delta$ and end in p_n. Note that P^δ is not necessarily a walk in G. However, we treat it like a walk by adding artificial edges in between any two consecutive points of P^δ with costs equal to their distance.

▶ **Lemma 2.1.** *It holds that* $D_{\mathcal{F}}\left(P, P^\delta\right) \leq \frac{\delta}{2}$.

Proof. Denote by a_i the index of the vertex p_i in P^δ. We define an xy-monotone path through the free space matrix $M_{\delta/2}$ of P and P^δ only visiting entries with value -1 and visiting all the tuples (i, a_i) for $p_i \in P^\delta$. We start with $(1, 1)$. Let p_i and p_j be consecutive in P^δ. We now want to define an xy-monotone subpath from (i, a_i) to (j, a_j). We start with (i, a_i). If $j = i + 1$, we can take (j, a_j) as the next tuple, which is a legal step and we are done.

Now suppose that $j > i + 1$. Then, p_i and p_j are not consecutive in P. Assume $\text{dist}(p_i, p_j) > \delta$. Denote by $P[i, j]$ the subpath of P starting in p_i and ending in P_j. Observe that $P[i, j]$ is a shortest path. If we delete the two outer vertices of this subpath, we either have another subpath or a single vertex. Denote this subpath or singleton by $\tilde{P}$. The shortest path $\tilde{P}$ is δ-significant with $P[i, j]$ as a δ-witness. Hence, a vertex of $\tilde{P}$ has to be contained in H_δ because it is a hitting set for all δ-significant shortest paths. This is a contradiction to p_i and p_j being consecutive on P^δ and hence $\text{dist}(p_i, p_j) \leq \delta$ must hold.

Let i' be the largest index such that $\text{dist}(p_i, p_{i'}) \leq \frac{\delta}{2}$. This means that $\text{dist}(p_{i'+1}, p_j) \leq \frac{\delta}{2}$ because P is a shortest path. So, we define the following xy-monotone subwalk:

$$(i, a_i), (i + 1, a_i), \ldots, (i', a_i), (i' + 1, a_j), \ldots, (j - 1, a_j), (j, a_j).$$

The distance of all tuples is at most $\frac{\delta}{2}$ and hence their entries in $M_{\delta/2}$ are -1. Hence, if we combine all such subpaths, we end up with an xy-monotone walk through $M_{\delta/2}$ only visiting entries with value -1, implying that the Fréchet distance between P and P^δ is $\leq \frac{\delta}{2}$. ◀

Using Lemma 2.1 and the triangle inequality of the Fréchet distance, we get the following:

▶ **Lemma 2.2.** *If* $D_{\mathcal{F}}\left(P^\delta, Q\right) > \alpha$ *for any* $\alpha \geq 0$, *then* $D_{\mathcal{F}}\left(P, Q\right) > \alpha - \frac{\delta}{2}$.

▶ **Proposition 2.3.** *Let H_δ be a (h', δ)-SPHS of G, let P be a shortest path and let P^δ be given. Assume that a distance query in G takes time $T(G)$. Then, we can decide in time $\mathcal{O}\left(mh'T(G)\right)$ if $D_{\mathcal{F}}\left(P^\delta, Q\right) \leq \alpha$ for any $0 \leq \alpha \leq 2\delta$.*

Proof. The algorithm performs an implicit breadth first search through the non-zero entries of the free space matrix M_α between P^δ and Q and checks if (n, m) can be reached. This means that we only compute $\text{dist}(p_i, q_j)$, if the tuple (i, j) is considered in the BFS.

Note that only non-zero entries of M_α get added to the queue and every element in the queue has at most three legal successors. Hence, for the runtime it suffices to bound the number of non-zero entries in M_α. Let q be a vertex in Q. H_δ being a (h', δ)-SPHS implies $|B_\alpha(q) \cap H_\delta| \leq |B_{2\delta}(q) \cap H_\delta| \leq h'$. So, the inner vertices of P^δ with distance at most α to q all lie in this set. Adding p_1 and p_n this yields that there are at most $h' + 2$ non-zero entries in the row corresponding to q and at most $m(h' + 2)$ non-zero entries in M_α in total. For each of them we have at most three distance oracle calls. So, the BFS takes $\mathcal{O}\left(mh'T(G)\right)$. ◀

This gives us all the necessary tools to prove the following theorem:

▶ **Theorem 2.4.** *Let $G = (V, E)$ be a graph with a metric weight function and highway dimension h and let $\varepsilon > 0$. Suppose a distance query in G takes $\mathcal{O}(T(G))$ time using $\mathcal{O}(S(G))$ space. After preprocessing G in time polynomial in $|V|$ and $1/\log(1+\varepsilon)$, we can decide for any shortest path P with n vertices, any walk Q with m vertices and any $\delta > 0$, whether $D_{\mathcal{F}}(P, Q) \leq (\frac{5}{3} + \varepsilon)\delta$ or $D_{\mathcal{F}}(P, Q) > \delta$ in $\mathcal{O}(n + m(h \log h)T(G))$ time using $\mathcal{O}(|V| \log D/\log(1+\varepsilon) + S(G))$ space.*

Proof. Define $\mu := 1 + \frac{9\varepsilon}{8+3\varepsilon} > 1$. In this case, $\log(\mu)^{-1} = \Theta(\log(1+\varepsilon)^{-1})$. In the preprocessing, we compute a μ-multiscale SPHS with value $\mathcal{O}(h \log h)$ in running time polynomial in $|V|$ and $1/\log(1+\varepsilon)$ by Theorem 1.4. We save this μ-multiscale SPHS as a matrix of booleans with a row for every vertex and a column for every SPHS. This takes $\mathcal{O}(|V| \log D/\log(1+\varepsilon))$ space and ensures that we can check in constant time whether a vertex is contained in a certain SPHS.

Now we choose $\alpha = \frac{\delta}{1-\frac{\mu}{4}}$ and i such that $\mu^i < \frac{\alpha}{2} \leq \mu^{i+1}$. From the μ-multiscale SPHS we compute the set $P^{\mu^{i+1}}$ in $\mathcal{O}(|P|)$ time. Then, we compute in $\mathcal{O}(|Q|(h \log h)T(G))$ time whether $D_{\mathcal{F}}\left(P^{\mu^{i+1}}, Q\right) \leq \alpha$ using Proposition 2.3. If this is true, using Lemma 2.1 and the triangle inequality, we can derive that

$$D_{\mathcal{F}}(P, Q) \leq \alpha + \frac{\mu^{i+1}}{2} \leq (1 + \frac{\mu}{4})\alpha = \frac{1 + \frac{\mu}{4}}{1 - \frac{\mu}{4}}\delta = \left(\frac{5}{3} + \varepsilon\right)\delta.$$

In the other case, we can use Lemma 2.2 to see that

$$D_{\mathcal{F}}(P, Q) > \alpha - \frac{\mu^{i+1}}{2} > (1 - \frac{\mu}{4})\alpha = \delta.$$ ◀

Since the Fréchet distance between P and Q can be at most D, we can apply the algorithm combined with a binary search on the value of the Fréchet distance to get the following result:

▶ **Corollary 2.5.** *Let $G = (V, E)$ be a graph with a metric weight function and highway dimension h and $\varepsilon > 0$. Suppose a distance query in G takes $\mathcal{O}(T(G))$ time using $\mathcal{O}(S(G))$ space. After preprocessing G in time polynomial in $|V|$ and $1/\log(1+\varepsilon)$, we can compute for any shortest path P with n vertices and any walk Q with m vertices a $(\frac{5}{3} + \varepsilon)$-approximation of $D_{\mathcal{F}}(P, Q)$ in $\mathcal{O}(\log D(n + m(h \log h)T(G)))$ time using $\mathcal{O}(|V| \log D/\log(1+\varepsilon) + S(G))$ space.*

Using the distance oracle from Theorem 1.5, our algorithm for approximating the Fréchet distance from Theorem 2.4 can be implemented in $\mathcal{O}\left(|P| \log D + |Q|(h \log h \log D)^2\right)$ time and using $\mathcal{O}(|V| \log D(1/\log(1+\varepsilon) + h \log h))$ space.

References

1 Ittai Abraham, Daniel Delling, Amos Fiat, Andrew V. Goldberg, and Renato F. Werneck. VC-dimension and shortest path algorithms. In Luca Aceto, Monika Henzinger, and Jiří Sgall, editors, *Automata, Languages and Programming*, pages 690–699, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

2 Ittai Abraham, Daniel Delling, Amos Fiat, Andrew V. Goldberg, and Renato F. Werneck. Highway dimension and provably efficient shortest path algorithms. *J. ACM*, 63(5), 12 2016. doi:10.1145/2985473.

3 Ittai Abraham, Amos Fiat, Andrew Goldberg, and Renato Werneck. Highway dimension, shortest paths, and provably efficient algorithms. pages 782–793, 01 2010. doi:10.1137/1.9781611973075.64.

4 Pankaj K. Agarwal, Rinat Ben Avraham, Haim Kaplan, and Micha Sharir. Computing the discrete Fréchet distance in subquadratic time. *SIAM Journal on Computing*, 43(2):429–449, 2014. doi:10.1137/130920526.

5 Helmut Alt and Michael Godau. Computing the Fréchet distance between two polygonal curves. *Int. J. Comput. Geometry Appl.*, 5:75–91, 03 1995. doi:10.1142/S0218195995000064.

6 Helmut Alt, Christian Knauer, and Carola Wenk. Comparison of distance measures for planar curves. *Algorithmica*, 38:45–58, 10 2003. doi:10.1007/s00453-003-1042-5.

7 Boris Aronov, Sariel Har-Peled, Christian Knauer, Yusu Wang, and Carola Wenk. Fréchet distance for curves, revisited. pages 52–63, 09 2006. doi:10.1007/11841036_8.

8 Sotiris Brakatsoulas, Dieter Pfoser, Randall Salas, and Carola Wenk. On map-matching vehicle tracking data. volume 2, pages 853–864, 01 2005.

9 Karl Bringmann. Why walking the dog takes time: Frechet distance has no strongly subquadratic algorithms unless SETH fails. *Proceedings - Annual IEEE Symposium on Foundations of Computer Science, FOCS*, 04 2014. doi:10.1109/FOCS.2014.76.

10 Kevin Buchin, Maike Buchin, Wouter Meulemans, and Wolfgang Mulzer. Four soviets walk the dog: Improved bounds for computing the Fréchet distance. *Discrete Comput. Geom.*, 58(1):180–216, jul 2017. doi:10.1007/s00454-017-9878-7.

11 Kevin Buchin, Maike Buchin, Rolf van Leusden, Wouter Meulemans, and Wolfgang Mulzer. Computing the Fréchet distance with a retractable leash. In Hans L. Bodlaender and Giuseppe F. Italiano, editors, *Algorithms – ESA 2013*, pages 241–252, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg. doi:10.1007/s00454-016-9800-8.

12 Maike Buchin, Ivor van der Hoog, Tim Ophelders, Lena Schlipf, Rodrigo I. Silveira, and Frank Staals. Efficient Fréchet distance queries for segments, 2022. arXiv:2203.01794.

13 Anne Driemel and Sariel Har-Peled. Jaywalking your dog: Computing the fréchet distance with shortcuts. *SIAM Journal on Computing*, 42(5):1830–1866, 2013. arXiv:https://doi.org/10.1137/120865112, doi:10.1137/120865112.

14 Anne Driemel, Sariel Har-Peled, and Carola Wenk. Approximating the Fréchet distance for realistic curves in near linear time. pages 365–374, 06 2010. doi:10.1145/1810959.1811019.

15 Anne Driemel, Ioannis Psarros, and Melanie Schmidt. Sublinear data structures for short Fréchet queries, 2019. arXiv:1907.04420.

16 Anne Driemel, Ivor van der Hoog, and Eva Rotenberg. On the discrete fréchet distance in a graph. In Xavier Goaoc and Michael Kerber, editors, *Proceedings of 38th International Symposium on Computational Geometry*, Leibniz International Proceedings in Informatics, LIPIcs. Schloss Dagstuhl- Leibniz-Zentrum fur Informatik GmbH, Dagstuhl Publishing, June 2022. doi:10.4230/LIPIcs.SoCG.2022.36.

17 Arnold Filtser and Omrit Filtser. Static and streaming data structures for Fréchet distance queries. *ACM Trans. Algorithms*, 19(4), oct 2023. doi:10.1145/3610227.

18 Roniel S. De Sousa, Azzedine Boukerche, and Antonio A. F. Loureiro. Vehicle trajectory similarity: Models, methods, and applications. *ACM Comput. Surv.*, 53(5), sep 2020. doi:10.1145/3406096.

19 Ivor van der Hoog, Eva Rotenberg, and Sampson Wong. Data structures for approximate discrete Fréchet distance, 2023. arXiv:2212.07124.

GeoCluster: A latent variable generative model for continuous space geometric clustering

Minas Dioletis[1,3], Ioannis Z. Emiris[2,5], George Ioannakis[3], Evanthia Papadopoulou[6], Thomas Pappas[3], Panagiotis Repouskos[3], Panagiotis Rigas[3,4,5], and Charalambos Tzamos[3,7]

1 **School of Electrical and Computer Engineering, National Technical University of Athens, Greece**
minasdioletis@mail.ntua.gr
2 **Athena Research Center, Greece**
emiris@athenarc.gr
3 **Institute for Language and Speech Processing, Athena Research Center, Greece**
{minas.dioletis,gioannak,thomas.pappas,
panagiotis.repouskos,panagiotis.rigas,charalambos.tzamos}@athenarc.gr
4 **Archimedes, Athena Research Center, Greece**
panagiotis.rigas@athenarc.gr
5 **Department of Informatics & Telecommunication, National & Kapodistrian University of Athens, Greece**
{emiris,rigasp}@di.uoa.gr
6 **Faculty of Informatics, Università della Svizzera italiana, Switzerland**
evanthia.papadopoulou@usi.ch
7 **Czech Technical University in Prague, Czech Republic**
tzamos.charalampos@fel.cvut.cz

Abstract

Given a set of shapes realized in $\mathbb{R}^d$, an important but challenging task is, given a query point $p \in \mathbb{R}^d$, to find the nearest shape to p w.r.t. a given distance function. Finding approximate or exact nearest neighbors is a fundamental algorithmic problem, which so far has predominantly focused on point-sets. In this work, given only a point-shape distance function, we tackle the problem of approximating the nearest neighbor of a query point to a set of shapes of unknown properties. We design a shape-agnostic algorithm for partitioning the set of shapes hierarchically, and build a tree data structure for answering nearest neighbor queries. For partitioning the space in k parts, we propose a machine learning algorithm, in which the shapes are treated as high dimensional vectors. We evaluate our proposed method on an extensive set of synthetic experiments.

1 Introduction

Nearest Neighbor Search constitutes a fundamental algorithmic problem that remains an active research field due to its importance in a variety of settings. The Nearest Neighbor Search problem is defined as follows. Let $\mathcal{O}_d$ denote a finite collection of *objects* (shapes) in $\Omega \subseteq \mathbb{R}^d$ that satisfy a property $\mathcal{P}$. This property indicates the type of data that we have, namely cubes, spheres, ellipses, convex polytopes with bounded number of vertices, etc. The distance between a point $p \in \mathbb{R}^d$ and an object $O \in \mathcal{O}_d$ is denoted by $\mathcal{D}(p, O)$, and its analytical form depends on the property $\mathcal{P}$. In this work, we are interested in approximating the nearest neighbor O^* of a query point p from a collection of *objects* $\mathcal{O}_d$, such that:

$$O^* = \arg\min_{O \in \mathcal{O}_d} \mathcal{D}(p, O). \tag{1}$$

40th European Workshop on Computational Geometry, Ioannina, Greece, March 13–15, 2024.
This is an extended abstract of a presentation given at EuroCG'24. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

We assume that d a is small constant and provide experimental results for d = 2 and d = 3.

In the simple case when the *objects* represent points in $\mathbb{R}^d$, the problem of approximating the nearest neighbor has been studied extensively [12, 3, 10, 11, 16, 5].

Point-based nearest neighbor queries are essential in data analysis for a variety of applications [2]. Searching for the nearest neighbor exhaustively is infeasible because the linear, on the number of candidates, time complexity tends to be expensive in any real-world setting [21]. Hence, one turns to approximate search where the goal is to retrieve one or a set of *Approximate Nearest Neighbors (ANNs)* in a fast and effective manner, namely that achieves sublinear, logarithmic, or even constant query time.

Considering more complex than points, *objects*, in a recent work [1], nearest neighbor queries against line segments in $\mathbb{R}^d$ has been considered, for a fixed dimension d, yielding an $(1+\varepsilon)$-approximate nearest neighbor algorithm. In [7], the authors address the problem of finding the nearest neighbor to a set of ellipses in $\mathbb{R}^2$ by computing the Voronoi diagram of a set of ellipses. In [6], orthogonal polyhedra in $\mathbb{R}^2$ and $\mathbb{R}^3$ are considered, and the problem of nearest neighbor is tackled by computing the Voronoi diagram of the set of polyhedra, using the L_∞ metric. In [18], the authors propose an algorithm for computing the L_∞ Voronoi diagram of a set of non-orthogonal shapes in $\mathbb{R}^2$. To the best of our knowledge, prior work on shape-based ANN search is limited to specific *objects* families and fixed dimension.

In this paper, we consider *objects* that can be determined by a finite set of parameters, e.g. a line segment in $\mathbb{R}^2$ can be uniquely described by its two endpoints, which can be considered as a vector of four parameters. For such shape-agnostic class of *objects* in $\mathbb{R}^d$, we designed an algorithm based on a generative machine learning model that answers efficiently *ANN* queries to a given point. To demonstrate the performance and applicability of our method to different types of *objects*, we provide experimental results. Code is publicly available [1], with details on how install and train the model, coupled with a demo.

2 Method

Our method employs a latent-variable generative model for hierarchical clustering, creating a tree structure where nodes represent data space regions, not just centroids, and child nodes represent sub-regions of their parent. Such a hierarchical clustering has been used in [16]. Hierarchical clustering refers to a tree T with branching factor k, that is built by recursively partitioning the data into sub-clusters until individual or constant number of data points are reached. This deviates from traditional fixed-cardinality clustering by adapting to the data manifold's shape. Critics assess and value the edges in this tree, converting it to a weighted graph for navigational decision-making by an actor with a policy[14, 15]. Explicit data embedding in a higher-dimensional space transforms complex shapes into discrete points, enabling divergence calculations for iterative state refinement. This approach, termed GeoCluster, dynamically clusters data, facilitating efficient approximate nearest neighbor queries.

2.1 Architecture

The proposed architecture is centered around a clustering mechanism, forming the foundation of a tree. This process begins by embedding objects into the high-dimensional space. The

[1] https://github.com/PRigas96/GeoCluster

clustering approach is recursive, resulting in a tree structure where each node is not merely a centroid, but represents a specific area of the data manifold $\mathcal{M}$.

To complement the clustering process a critic that evaluates and assigns values to the edges connecting the nodes in this tree is used. This evaluation results in a weighted tree, where the edge weights reflect the critic's assessment of the relationship between nodes, given a condition query p.

For inference, the tree T is utilized, or its corresponding quantified manifold $\mathcal{S}^{'}$ obtained from $\mathcal{M}$ after training. Given a query point $q \in \mathcal{Q}$, T is inferred by the actor through its policy π. This is based on weights on nodes, that act as decision-making tools, guiding the inference process. This approach mirrors a cost-minimization strategy, akin to finding the shortest path in the tree based on cross-entropy minimization [15].

Upon reaching a leaf node, an exhaustive search is conducted among the remaining objects to find the nearest neighbor O^*. The tree's functionality is illustrated in Fig. 1, where each node, endowed with a critic function, actively contributes to the inferential process by assigning scores to its children.

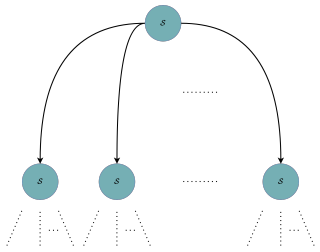


Figure 1 The tree constructed during training, and each node is associated with a critic.

2.1.1 Detailed Description

The architecture is presented at Fig. 2a, with the topology for the construction of a node in T. It consists of a latent variable $z \in \mathcal{Z}$ decoder module to produce the centroids $c \in \mathcal{S}$. A divergence Div then is calculated based on c and data $x \in \mathcal{M}$. Every f_{clk} Hz, a Gaussian sampler[4, 13, 20] is utilized and produces fuzzy centroids with radius $\zeta^{-1}Div$, scaled by a factor Sc, to alleviate the initialization problem caused by the many bad local minima, and initial centroids[17, 19]. This does not prevent clusters from being empty, which is utilized to further enable structure alignment by deleting non-participant centroids[8].

After states have been constructed, critic network C_S, a parametric function, is employed to assign low and high values to non probable and probable state transitions respectively, as shown in Fig. 2b. It is trained with data produced and labeled by an *uncertainty area sampler (UN)* module. The UN sampler consists of getting the Voronoi surfaces, sampling normal edges and points on those edges, conducting an efficient sampling in areas of uncertain prediction due to non-linear bounds between spaces.

The *clustering* network's latent variable is regularized through a one-hot encoder[9]. Representations c are regularized through $R_{rep} = \sum_i \sum_{j \neq i} d_p(c_i, c_j)^{-1}$, a repulsive loss that prevents trivial solutions and R_{proj} that enforces each representation to fall within its state.

To ensure this the states associated critic is utilized:

$$R_{proj} = \sum_{c\in\mathcal{S}}\sum_{x\in c}\sum_{u\in bbox} \text{ReLU}\left(\prod_{\forall e\in u}\frac{e-x}{|e-x|}\right)\min(||u-x||) + \sum_{S\in T}\omega_S\sum_{c\in S_\tau}\text{CE}(\text{Softmax}(C_s(c)), q_s), \quad (2)$$

where $ReLU(x) = \max(0, x)$ ensures initialization in root state, while cross entropy (CE), weighted by ω_S, that each a centroid in a path, satisfies all previous trajectory states, ensuring flow of information between different hierarchy level states. $Div = \sum_{x\in\mathcal{M}} E_w(x, \bar{z})$, where E_w is the states energy, is used to ensure clustering compactness[22].

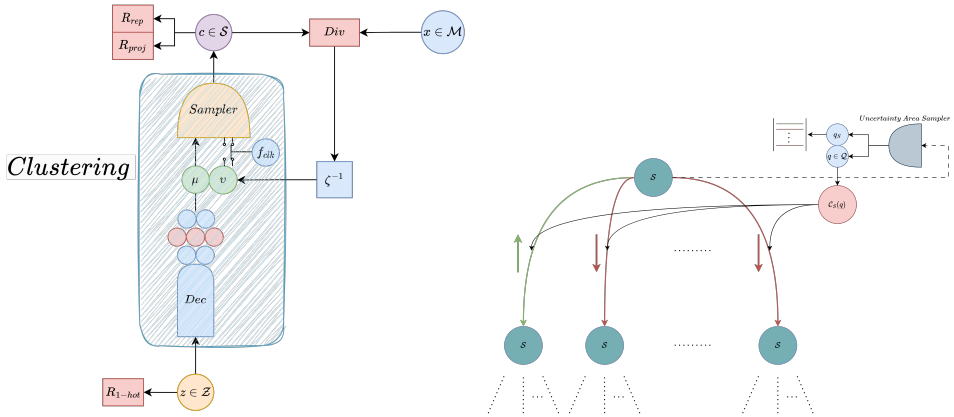


Figure 2 Proposed Network. *(a)* illustrates the network topology, for the construction of a node in the tree T. *(b)* showcases the critic network assigning weights in edges of a tree.

3 Experimental Results

In this section, we showcase the methods performance in three different aspects, the quality of produced representations and associated states, the search precision and the robustness.

3.1 Representation Evaluation

In Fig. 3 the centroids are shown for different structured data. It is clear that they capture the underlying structure effectively.

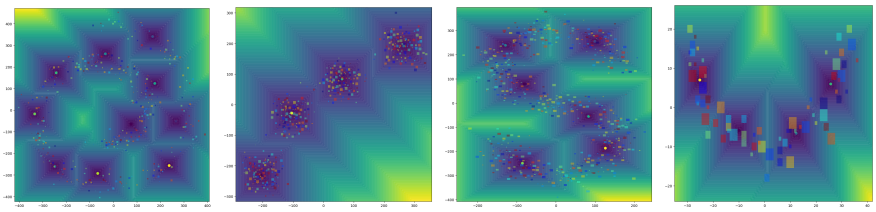


Figure 3 One-layer centroids, seen in dark blue areas for different structured 2D Data.

The resulted space for one layer of hierarchy is also shown for a 3D case in Fig. 4. Spaces can be non-linear and discontinuous.

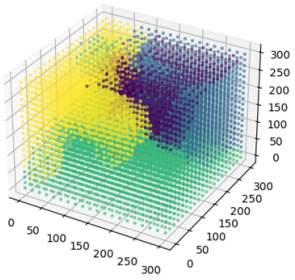


Figure 4 Quantified space $\mathcal{S}^{'}$ for 3D cuboids. Each color corresponds to a different cluster.

3.2 Search Precision and Parallelization

In this section, we delve into the model's search precision, specifically its efficacy in identifying complex geometric forms, as explicated in Tables 1 and 2. Table 1 provides insights the model's adeptness across various geometric shapes, demonstrating its commendable generalization capabilities. Notably, the model exhibits similar layer-wise accuracies, with final $acc = \prod_{i \in N} acc_i$, where N is the number of layers, being stable.

	Layer 1-2	Layer 2-3	Layer 3-4	Layer 4-5	Layer 5-6	Dimensions	Metric
Squares	82.8 ± 2.0	97.7 ± 0.7	99.8 ± 0.2	-	-	$\mathbb{R}^2$	L_∞
Cuboids	91.3 ± 0.8	87.9 ± 1.4	90.5 ± 2.4	93 ± 2.2	99.3 ± 0.7	$\mathbb{R}^3$	L_∞
Ellipses	95.7 ± 0.6	95.2 ± 1.1	97.7 ± 0.9	-	-	$\mathbb{R}^2$	L_2

Table 1 Table of Layer-wise Accuracies for Each Shape in the Dataset of 10,000 elements. The table displays mean accuracy and standard deviation for each shape. "Accuracy" is defined as the model's ability to precisely locate the nearest neighbor between layers, tested with randomly generated query points within the data's bounding area. Final accuracies can be obtained as a dot product of each layer.

The effectiveness of the model in accurately identifying and approximating the nearest neighbor is demonstrated in the results presented in Table 2. These results affirm the model's proficiency across various shapes, underscoring its strong generalization and scalability. Further, the data shown in Fig. 5 reinforces our assertion that the model efficiently captures the k-nearest neighbors, even when k is small, showcasing its robustness in neighbor detection.

	$k=1$	$k=2$	$k=5$	$k=10$	$k=1\%$	$k=5\%$	$k=10\%$	Layers
Squares	77.3 ± 2.3	79.1 ± 1.8	81.7 ± 2.4	83.4 ± 2.1	90.9 ± 2.0	99.7 ± 0.3	99.9 ± 0.1	5
Cuboids	61.9 ± 2.0	73.3 ± 1.6	85.3 ± 2.0	91.0 ± 1.2	99.0 ± 0.7	100	100	7
Ellipses	76.9 ± 1.6	79.3 ± 1.7	81.5 ± 1.8	83.8 ± 2.1	91.0 ± 0.9	98.4 ± 0.1	99.6 ± 0.4	5

Table 2 This table shows the accuracy of leaf nodes in the hierarchical structure for each shape in the dataset, with each shape having 10k instances. Here, "k" represents the number of top results considered for a successful search, indicating accurate neighbor detection among the first k results.

The results are also showcased for a soft-accuracy in non-randomly placed 2D data. The distance $\mathcal{D}(q, O^*)$, given a query q and its nearest neighbor O^* is measured. Then a relaxed criterion, that the nearest neighbor we find $\bar{O}$ is in a radius of O^*, is calculated. As shown

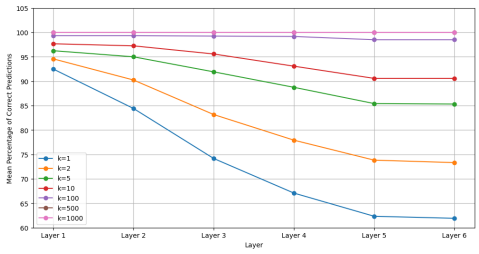


Figure 5 Cuboids: Mean Percentage of Correct Predictions per Layer for Different k-nearest neighbor Values.

in Fig. 6, a twofold increase in ball radius ration the accuracy converges, with a low mean bound of 95%.

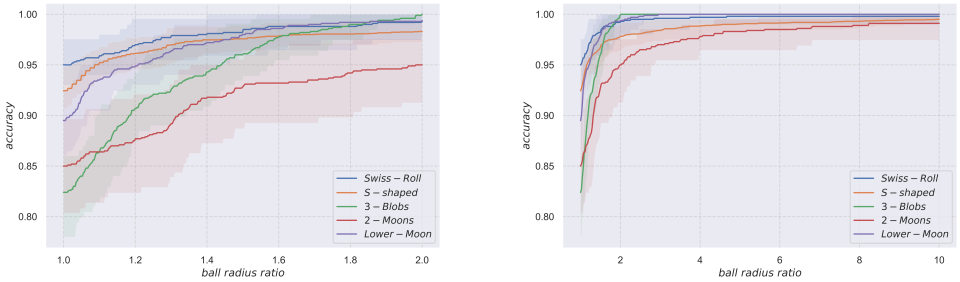


Figure 6 Soft-Accuracy tests different kind of structured 2D data, 1000 in number with 4 layers and width factor k=3. As ball radius ratio $\mathcal{D}(q,\bar{O})/\mathcal{D}(q,O^*)$, where $\bar{O}$ is the nearest neighbor found by our method and O^* is the true nearest neighbor, increases, accuracy converges.

Fig. 7 illustrates the model's superior performance in comparison to traditional serial search methods. Specifically, Fig. 7*a* highlights the model's adeptness at parallelization, a direct benefit of utilizing batching techniques. For a single node, as the quantity of query points escalates, our model consistently operates averagely at just $p = 1/100$th of the time required by linear search methods, denoted as $N \cdot \tau_1$. This efficiency underscores the stark contrast in performance scalability, particularly in handling large volumes of queries. Furthermore, Fig. 7*b* reveals that expanding the width factor k—which could potentially complicate the search process—does not detrimentally affect the model's performance. This observation confirms the model's robust scalability concerning data size, effectively demonstrating that its empirical average computational complexity remains practical and manageable, best captured by the expression $O(\lceil \frac{|Q|}{p} \rceil \log_k n)$, where Q is a set of queries, n the data objects and p an observed random value characterizing parallelization due to neural networks, shown in Fig. 7*a*. This notation adeptly reflects the combined influence of batching, hardware acceleration, and the model's algorithmic design on enhancing processing efficiency beyond simple linear expectations. Worst case complexity still remains $O(|Q| \log_k n)$, when no parallelization takes place.

3.3 Robustness

The robustness of our method is evaluated using various structured 2D data sets, as illustrated in Fig. 8. Although accuracy initially drops with high variance when the test set deviates from

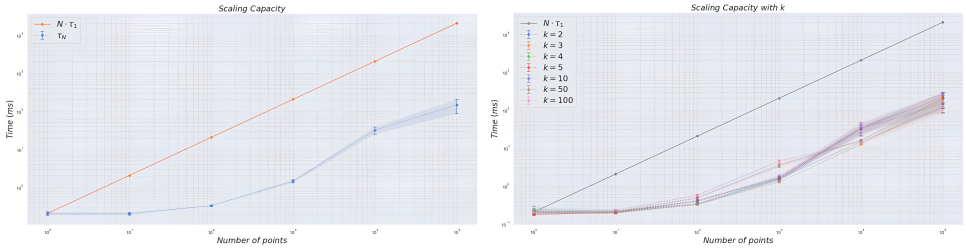


Figure 7 Performance (in seconds), in log scale, for a number of query points and one layer. τ_1 denotes a query pass and thus $N\tau_1$ can be regarded as $\mathcal{O}(n)$. *(a)* shows the difference between $\mathcal{O}(n)$ and our method, while *(b)* shows for different width factors (k).

the trained area, it quickly converges, demonstrating the network's effective extrapolation capabilities. Additionally, Table 1 further corroborates the method's robustness, highlighting its consistent performance across a diverse range of shapes and dimensions.

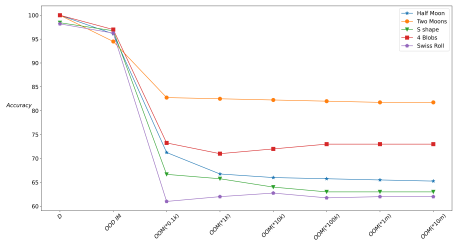


Figure 8 Robustness tests for different structured 2D data in one layer hierarchy. Eight columns show different areas, D for Data-trained area, *OOD IM* for data inside data manifold with trained area and *OOM*a* for outside manifold area scaled with a. Accuracy converges outside manifold rapidly but displays high variance for different structure

4 Conclusions

In this study, we leverage neural networks, specifically a latent variable generative model, to construct a tree. This is coupled with a critic mechanism to weight the edges, streamlining the nearest neighbor search across various domains and metrics. Addressing the challenge of point-to-shape nearest neighbor in 3D, even for simple shapes, our model stands out for its domain-agnostic nature. It requires only the definition of an embedding and a metric for divergence, demonstrating its versatility and effectiveness in simplifying complex geometric computations, while its inherent design makes it scalable.

5 Acknowledgment

This work has been partially co-financed by the European Union and the "Greece 2.0" national recovery and resilience plan, under the call "RESEARCH-CREATE-INNOVATE" (Code: TAEΔK-06168).

References

1. Ahmed Abdelkader and David M Mount. Approximate nearest-neighbor search for line segments. In *37th International Symposium on Computational Geometry*, 2021.
2. Yannis Avrithis, Yannis Kalantidis, Evangelos Anagnostopoulos, and Ioannis Z Emiris. Web-scale image clustering revisited. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1502–1510, 2015.
3. Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.
4. Carl Doersch. Tutorial on variational autoencoders. *arXiv preprint arXiv:1606.05908*, 2016.
5. Wei Dong, Charikar Moses, and Kai Li. Efficient k-nearest neighbor graph construction for generic similarity measures. In *Proceedings of the 20th international conference on World wide web*, pages 577–586, 2011.
6. Ioannis Z Emiris and Christina Katsamaki. Voronoi diagram of orthogonal polyhedra in two and three dimensions. In *International Symposium on Experimental Algorithms*, pages 1–20. Springer, 2019.
7. Ioannis Z Emiris, Elias P Tsigaridas, and George M Tzoumas. The predicates for the exact voronoi diagram of ellipses under the euclidiean metric. *International Journal of Computational Geometry & Applications*, 18(06):567–597, 2008.
8. Greg Hamerly and Charles Elkan. Learning the k in k-means. *Advances in neural information processing systems*, 16, 2003.
9. John T Hancock and Taghi M Khoshgoftaar. Survey on categorical data for neural networks. *Journal of Big Data*, 7(1):1–41, 2020.
10. Sariel Har-Peled. A replacement for voronoi diagrams of near linear size. In *Proceedings 42nd IEEE Symposium on Foundations of Computer Science*, pages 94–103. IEEE, 2001.
11. Sariel Har-Peled and Nirman Kumar. Approximating minimization diagrams and generalized proximity search. *SIAM Journal on Computing*, 44(4):944–974, 2015.
12. Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 604–613, 1998.
13. Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.
14. Vijay Konda and John Tsitsiklis. Actor-critic algorithms. *Advances in neural information processing systems*, 12, 1999.
15. Yann LeCun. A path towards autonomous machine intelligence version 0.9. 2, 2022-06-27. *Open Review*, 62(1), 2022.
16. Marius Muja and David G Lowe. Scalable nearest neighbor algorithms for high dimensional data. *IEEE transactions on pattern analysis and machine intelligence*, 36(11):2227–2240, 2014.
17. Shi Na, Liu Xumin, and Guan Yong. Research on k-means clustering algorithm: An improved k-means clustering algorithm. In *2010 Third International Symposium on intelligent information technology and security informatics*, pages 63–67. Ieee, 2010.
18. E. Papadopoulou and D. T. Lee. The L_∞ Voronoi diagram of segments and VLSI applications. *International Journal of Computational Geometry and Applications*, 11(5):503–528, 2001. doi:10.1142/S0218195901000626.
19. Kristina P Sinaga and Miin-Shen Yang. Unsupervised k-means clustering algorithm. *IEEE access*, 8:80716–80727, 2020.
20. Arash Vahdat and Jan Kautz. Nvae: A deep hierarchical variational autoencoder. *Advances in neural information processing systems*, 33:19667–19679, 2020.
21. Jun Wang, Wei Liu, Sanjiv Kumar, and Shih-Fu Chang. Learning to hash for indexing big data - A survey. *Proc. IEEE*, 104(1):34–57, 2016. doi:10.1109/JPROC.2015.2487976.

22 Jyoti Yadav and Monika Sharma. A review of k-mean algorithm. *Int. J. Eng. Trends Technol*, 4(7):2972–2976, 2013.

Oriented dilation of undirected graphs

Kevin Buchin[1], Antonia Kalb[1], Carolin Rehs[1], and Andre Schulz[2]

1 **Technical University of Dortmund**
`kevin.buchin, antonia.kalb, carolin.rehs@tu-dortmund.de`
2 **FernUniversität in Hagen**
`andre.schulz@fernuni-hagen.de`

Abstract

Given an oriented graph $\overrightarrow{G}$ on a set of points P in the Euclidean plane, the oriented dilation of $p, p' \in P$ is the ratio of the length of the shortest cycle in $\overrightarrow{G}$ through p and p' to the perimeter of the smallest triangle in P containing p and p'. The oriented dilation of $\overrightarrow{G}$ is maximum oriented dilation over all pair of points. We show that given an undirected graph G on P, it is NP-hard to decide whether the edges can be oriented in way that the oriented dilation of the resulting graph is below a given threshold. For the case that G is complete, it is known that there is always an orientation of the edges with oriented dilation at most 2. As a first step towards improving this bound, we show that for $|P| = 4$ there is always a tournament, i.e., an oriented complete graph, with oriented dilation at most 1.5. This holds not only in the Euclidean but more generally in the metric plane. In the latter the bound is tight.

1 Introduction

Geometric spanners have may applications like wireless ad-hoc networks [4, 10], robot motion planning [5] and the analysis of road networks [1, 6]. The need to orient edges naturally arise since edges might only support one-way communication/traffic. Thus, in such applications it may be necessary to find an orientation of the edges that still provides relatively short paths between vertices. While undirected spanners are a widely researched topic during the last decades (see [2, 9] for a survey), oriented spanners have been only introduced recently [3].

Given a point set in the Euclidean plane and a parameter t, an oriented t-spanner $\overrightarrow{G}$ is an oriented subgraph of the complete bi-directed graph, such that for every pair of points, the shortest cycle in $\overrightarrow{G}$ containing those points is at most a factor t longer than their smallest triangle in the complete graph. Formally, given a point set $P \subset \mathbb{R}^d$ and a parameter $t \in \mathbb{R}^+$, an oriented graph $\overrightarrow{G} = (P, \overrightarrow{E})$ (thus a graph where $(u, v) \in \overrightarrow{E}$ implies $(v, u) \notin \overrightarrow{E}$) is called oriented t-spanner if for every two points $p, p' \in P$ the oriented dilation $\mathrm{odil}(p, p') = \frac{|C_{\overrightarrow{G}}(p,p')|}{|\Delta(p,p')|} \leq t$. Here, $C_{\overrightarrow{G}}(p, p')$ denotes the shortest oriented cycle containing p and p' in $\overrightarrow{G}$ and $\Delta(p, p')$ is the triangle $\Delta pp'p''$ with $p'' = \arg\min_{p^* \in P} |p - p^*| + |p^* - p'|$.

The problem of finding an oriented t-spanner with at most some fixed number m of edges is NP-hard [3], thus there is little hope to compute minimum oriented spanners efficiently. A natural approach for nonetheless computing an oriented spanner is to first compute a suitable undirected graph and then orienting it. For convex point sets, for instance, one can obtain an $\mathcal{O}(1)$-spanner by orienting a greedy triangulation [3]. However, no constructions are known to compute oriented spanners of small size for general point sets.

Here, we show that finding an orientation of an undirected graph such that the oriented dilation is minimal, is NP-hard even on Euclidean graphs. As our NP-hardness construction does not hold for complete graphs, we look into the oriented dilation of tournaments. As first step and potential building block for larger point sets, we show that for every point set

40th European Workshop on Computational Geometry, Ioannina, Greece, March 13–15, 2024.
This is an extended abstract of a presentation given at EuroCG'24. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

P with $|P| = 4$ even in a metric plane there is a tournament $\overrightarrow{K}(P)$ such that the oriented dilation of $\overrightarrow{K}(P)$ is at most 1.5. We further prove this bound to be tight.

2 Hardness

▶ **Theorem 2.1.** *Given an undirected geometric graph G and a parameter t', it is NP-hard to decide if there is an orientation $\overrightarrow{G}$ of G with oriented dilation* $\mathrm{odil}(\overrightarrow{G}) \leq t'$.

We will give a proof idea which is mainly described graphically here. A detailed proof with an explanation for the coordinates of every point can be found in the full version.

Proof sketch. We reduce from the NP-complete problem planar 3-SAT [8]. We start with a planar Boolean formula φ in conjunctive normal form with an incidence graph G_φ that can be embedded on a polynomial-size 1×1-grid [7, 8] as illustrated in Figure 1. We give a construction for a graph G based on G_φ such that there is an orientation $\overrightarrow{G}$ of G with dilation $\mathrm{odil}(\overrightarrow{G}) \leq t'$ with $t' := 1.043$ if and only if φ is satisfiable.

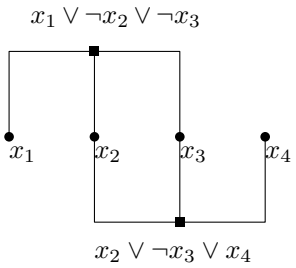


Figure 1 Example: Incidence graph of a planar 3-SAT formula embedded on a square grid

In the following, every point $p = (x, y)$ on the grid will be replaced by a so-called *oriented point*, which is a pair of points $P = \{t(p), b(p)\}$ with *top* $t(p) = (x, y + \frac{\varepsilon}{2})$ and *bottom* $b(p) = (x, y - \frac{\varepsilon}{2})$, where $\varepsilon \geq 0$ is a small constant. We will present the proof with $\varepsilon = 0$, i.e., $t(p)$ and $b(p)$ are two points with the same coordinates, while using a small positive ε in all figures for illustration purposes. This choice of ε simplifies the proof. However, the proof stays valid for a sufficiently small $\varepsilon > 0$.

We add an edge between $t(p)$ and $b(p)$, its orientation encodes whether this points represent "true" or "false". W.l.o.g. we assume that an oriented edge from $b(p)$ to $t(p)$, thus an *upwards* edge, represents "true" and a *downwards* edge represents "false". When this is not the case, we can achieve this by flipping the orientation of all edges.

Edges in the plane embedding of our formula graph G_φ will be replaced by *wire* gadgets. First, we add (a polynomial number of) grid points on the edge such that all edges have length 1. Then, we create a wire as in Figure 2. Note that wires propagate the orientation of oriented points - if two points next to each other on a wire have different orientations, their dilation would be significantly larger than $t' := 1.043$, since the shortest oriented cycle needs to go through an additional oriented point. If they have the same orientation, their dilation is 1 (since $\varepsilon = 0$; otherwise slightly larger). To switch a signal (for a negated variable in a formula), we start the wire as in Figure 3.

To ensure that all clause gadgets encode the same orientation of oriented points as "true", we add a *tree of knowledge*. This is a tree with vertices on the 1×1-grid shifted by $(0.5, 0.5)$ relative to the grid of G_φ and with wires as edges. The tree will have two leaves per clause

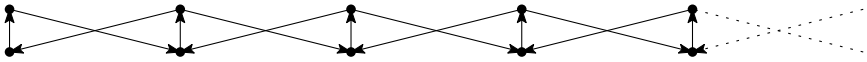


Figure 2 A wire where oriented points are oriented upwards

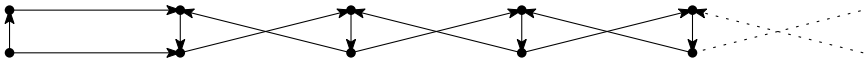


Figure 3 A wire where the orientation is switched to negate the signal

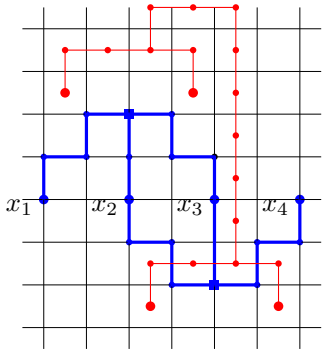


Figure 4 G_φ (blue) and its underlying grid together with a tree of knowledge (red)

(see Figure 4). W.l.o.g we assume that all oriented points of the tree of knowledge are oriented upwards (thus "true").

All oriented points, which are not direct neighbours of a wire, are linked by a $K_{2,2}$ (compare to Figures 5 and 6). This ensures dilation 1 between those points.

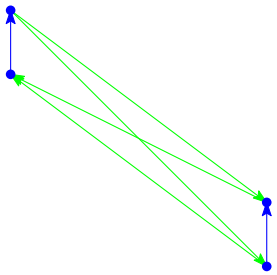


Figure 5 $K_{2,2}$ between oriented points with same orientation

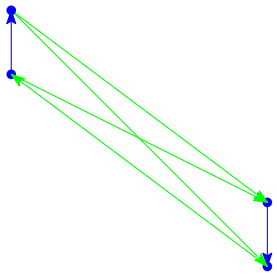


Figure 6 $K_{2,2}$ between oriented points with different orientation

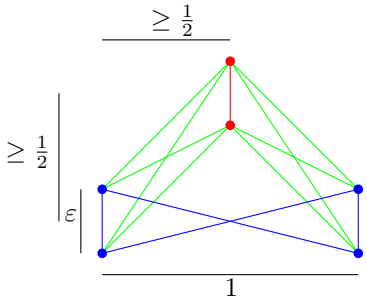


Figure 7 A wire (blue) can not be shortcut by $K_{2,2}$s (green)

Let p and p' be direct neighbours and p^* a third non-neighbouring point. Since p^* has at least distance $(0.5, 0.5)$ to p and p', the $K_{2,2}$s between p and p^* and p' and p^* do not affect that the wire between p and p' ensures equal orientation of the neighbours (compare to Figure 7).

The dilation of $t(p)$ and $b(p)$ is bounded by the dilation of p and its closest point p'. That is 1, both if p, p' are direct neighbours and not.

The two leaves of the tree of knowledge for every clause are not linked by a $K_{2,2}$. Figure 8 shows the two leaves of the tree at a clause, and G_φ at the clause. We can assume that G_φ is embedded as shown, in particular leaving the area directly above the clause empty. The two leaves are now linked by a *clause gadget*. We show how such a gadget looks like in Figure 9, more detailed in Figure 10.

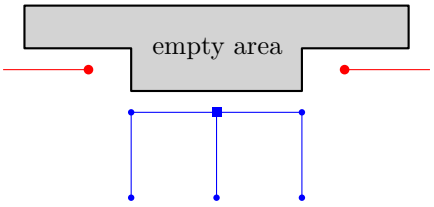


Figure 8 Embedded clause

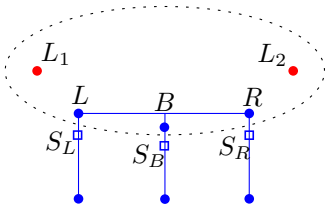


Figure 9 Clause gadget

The oriented points L (left), R (right) and B (bottom) are the ends of the variable wires of a clause. They are placed such that they lie just inside an ellipse with the locations of the leaves L_1 and L_2 as foci, and without any other points in the ellipse (compare to Figures 8 and 9). Stated differently, the triangles with endpoints L_1, L_2 and one of these three points, have nearly the same size, and any triangle with L_1, L_2, and a different oriented point has a larger perimeter. Adding edges as shown in Figure 10 guarantees that to obtain a short cycle through L_1, L_2 and one of these points, the orientation of that point has to be the same as of L_1 and L_2 (thus, the literal is "true".)

For each of $\{L, R, B\}$ there exists a *satellite* point, which is an oriented point on the variable wire, which is close but outside the ellipse. Its purpose is to make sure that the oriented dilation of L_1 (and likewise L_2) with L, B and R is stays below t' if if the corresponding literal does not satisfy the clause. We omitted all $K_{2,2}$ in the drawing. As described before, a $K_{2,2}$ exists between all unrelated oriented points, thus between all oriented points where there are no edges drawn in the figure.

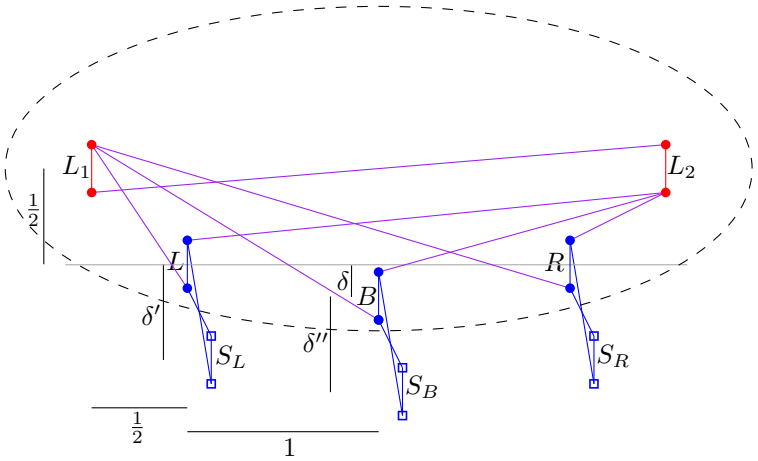


Figure 10 Detailed clause gadget

By setting $\delta = 0.1335$, $\delta' = 0.0303$ and $\delta'' = 0.35$, we obtain the following properties:

- The dilation between one of the points L_1, L_2 and one of the points L, B, R is lower or equal t', as a smallest cycle containing those points can use the related satellite point.
- If one of the oriented points L, B, R is oriented upwards, the dilation between L_1 and L_2 is smaller than $t' := 1.043$
- If none of the oriented points L, B, R is oriented upwards, the smallest cycle containing L_1 and L_2 either leaves the ellipse or takes at least two points from $\{L, B, R\}$ and thus their dilation is greater than t'.

Thus, formula φ is satisfiable if and only if there exists an orientation of our constructed graph with dilation at most 1.043. ◀

Following the construction in the proof of Theorem 2.1, Figure 11 illustrates the graph for the formula $\varphi = (x_1 \vee \neg x_2 \vee x_3) \wedge (x_2 \vee \neg x_3 \vee x_4)$ (see also Figure 4).

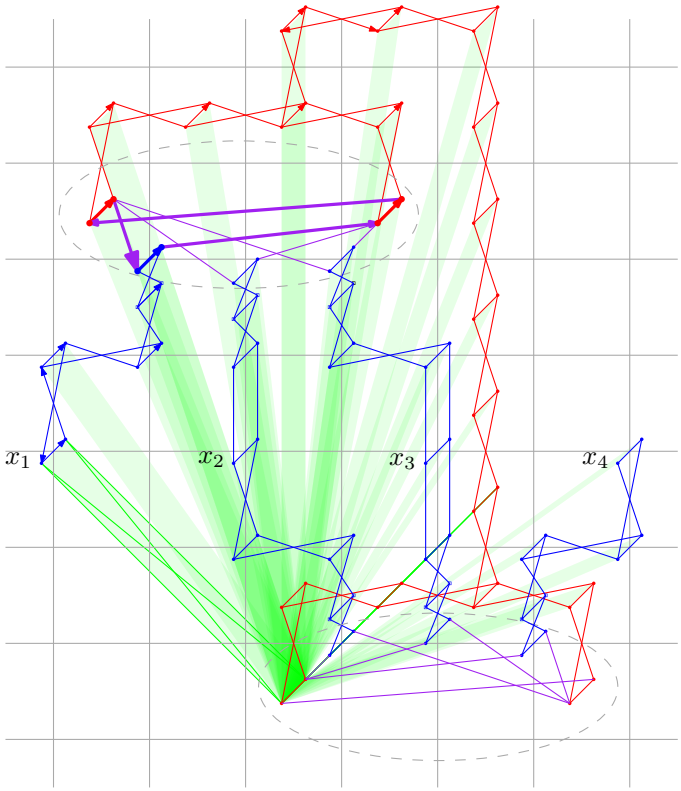


Figure 11 Graph constructed for $\varphi = (x_1 \vee \neg x_2 \vee x_3) \wedge (x_2 \vee \neg x_3 \vee x_4)$ (compare to Figures 1 and 4): For visibility, oriented points are placed diagonally instead of vertically. Only for one point, $K_{2,2}$s are indicated by green parallelograms. If the oriented point at the end of the wire from x_1 (blue) is –as indicated– oriented the same way as the tree of knowledge (red), this corresponds to setting it to true, resulting in an oriented cycle in the clause gadget (purple) that gives a dilation smaller than 1.043.

3 Bounding the dilation of tournaments

Buchin et al. [3] showed by example that there are (Euclidean) point sets for which no oriented t-spanner exists for $t < 2\sqrt{3} - 2 \approx 1.46$. For every (metric) point set P, they give an algorithm that returns a tournament $\overrightarrow{K}(P)$ on P with dilation $\mathrm{odil}(\overrightarrow{K}(P)) \leq 2$.

Our goal is to improve these bounds on the worst-case dilation $2\sqrt{3} - 2 \leq t \leq 2$ of the minimum dilation tournament. As a first step, we show a tight bound for sets of four points.

The complete graph on four points and its tournaments satisfy the following properties:

▶ Observation 3.1. For every undirected complete graph K_4 holds:

- K_4 contains $\binom{4}{3} = 4$ triangles.
- Every pair of these triangles shares exactly one edge.

- Every strongly connected tournament $\overrightarrow{K}_4$ contains exactly two *consistently* oriented triangles. This means the triangle is confined by an oriented cycle.

The following theorem gives a tight bound on the dilation of minimum dilation tournament on any metric point set of size four:

▶ **Theorem 3.2.** *For every point set P of size $|P| = 4$ embedded in a metric plane there is a tournament $\overrightarrow{K}(P)$ with dilation $\operatorname{odil}(\overrightarrow{K}(P)) \leq \frac{3}{2}$. This bound is tight.*

Proof. We prove that the following algorithm computes an tournament $\overrightarrow{K}(P)$ with dilation $\operatorname{odil}(\overrightarrow{K}(P)) \leq \frac{3}{2}$ for a point set $P = \{p_1, p_2, p_3, p_4\}$ embedded in a metric plane:

1. Let $\Delta_{p_1p_2p_3}$ be the shortest and $\Delta_{p_1p_2p_4}$ the second shortest triangle of the four triangles in K_4. Orient $\Delta_{p_1p_2p_3}$ and $\Delta_{p_1p_2p_4}$ consistently. That is always possible (compare to observation 3.1).
2. Orient the remaining edge between p_3 and p_4 such that the shortest oriented cycle $C_{\overrightarrow{K}(P)}(p_3, p_4)$ containing p_3 and p_4 is minimised.

By $d(p, p')$ we denote the weight of the edge between p and p'. Note that the weights satisfy triangle inequality.

We distinct cases by the orientation of the edge between p_3 and p_4, meaning

- $|C_{\overrightarrow{K}(P)}(p_3, p_4)| = d(p_1, p_2) + d(p_2, p_3) + d(p_3, p_4) + d(p_1, p_4)$ if

$$d(p_1, p_3) + d(p_2, p_4) \leq d(p_2, p_3) + d(p_1, p_4), \text{ or} \tag{1}$$

- $|C_{\overrightarrow{K}(P)}(p_3, p_4)| = d(p_1, p_2) + d(p_2, p_4) + d(p_3, p_4) + d(p_1, p_3)$ if

$$d(p_1, p_3) + d(p_2, p_4) > d(p_2, p_3) + d(p_1, p_4). \tag{2}$$

We show case (1), the other case can be proven analogously.

Since $\Delta_{p_1p_2p_3}$ and $\Delta_{p_1p_2p_4}$ are the shortest triangles and they are oriented consistently, the dilation of every pair of points is 1, except the pair p_3, p_4. So, we want to prove

$$t = \operatorname{odil}(p_3, p_4) = \frac{d(p_1, p_2) + d(p_2, p_3) + d(p_3, p_4) + d(p_1, p_4)}{\min\{|\Delta_{p_3p_4p_1}|, |\Delta_{p_3p_4p_2}|\}} \leq \frac{3}{2}.$$

Assume $|\Delta_{p_3p_4p_2}| \leq |\Delta_{p_3p_4p_1}|$ otherwise the names of the points belonging to the shortest and second shortest triangle can be swapped. Since $\Delta_{p_1p_2p_3}$ and $\Delta_{p_1p_2p_4}$ are the two shortest triangles, it holds

$$d(p_1, p_2) + d(p_1, p_4) \leq d(p_3, p_4) + d(p_2, p_3), \text{ and} \tag{3}$$

$$d(p_1, p_2) + d(p_1, p_3) \leq d(p_3, p_4) + d(p_2, p_4). \tag{4}$$

Summing up the inequalities 1, 3 and 4 we achieve

$$\begin{aligned}
& 2d(p_1, p_3) + 2d(p_1, p_2) + d(p_2, p_4) && \leq 2d(p_3, p_4) + 2d(p_2, p_3) + d(p_2, p_4) \\
\Leftrightarrow\quad & 2\left(d(p_1, p_3) + d(p_1, p_2) + d(p_2, p_4) + d(p_3, p_4)\right) && \leq 4d(p_3, p_4) + 2\left(d(p_2, p_3) + d(p_2, p_4)\right) \\
& && \overset{\Delta\text{-ineq.}}{\leq} 3\left(d(p_3, p_4) + d(p_2, p_3) + d(p_2, p_4)\right) \\
\Leftrightarrow\quad & \operatorname{odil}(p_3, p_4) = \tfrac{d(p_3,p_4)+d(p_1,p_3)+d(p_1,p_2)+d(p_2,p_4)}{d(p_3,p_4)+d(p_2,p_3)+d(p_2,p_4)} && \leq \frac{3}{2}.
\end{aligned}$$

For tightness, we show there is a point set P with $|P| = 4$, such that every strongly connected tournament on P has dilation $t = \frac{3}{2}$. The following metric give such an point set: $d(p_1, p_3) = d(p_2, p_3) = d(p_3, p_4) = 1$ and $d(p_1, p_2) = d(p_1, p_4) = d(p_2, p_4) = 2$. Taking into account mirroring and rotation, Figure 12 lists all strongly connected tournaments on P. We see that every tournament is a 1.5-spanner. ◀

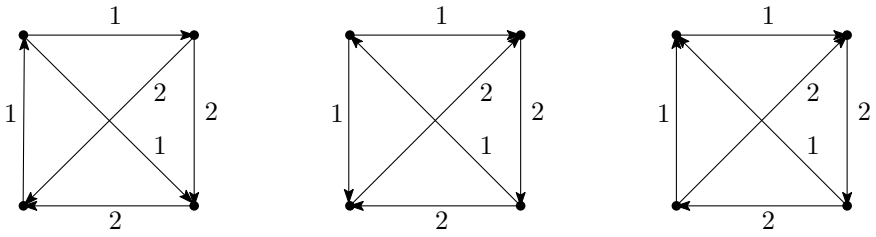


Figure 12 A metric point set where every connected tournament is a 1.5-spanner

4 Conclusion

We have shown that orienting a given geometric graph to minimise the oriented dilation is NP-hard. The complexity of this problem when restricting the graph class remains open. In particular: Is the problem NP-hard for planar graphs, or for complete graphs?

In the second part of the paper we studied the oriented dilation of metric point sets of size 4, i.e., with the K_4 as underlying graph. We proved that the oriented dilation is at most 1.5, while there are instances where it is tight. We know that in general the oriented dilation of K_n on metric instances can be upper-bounded by 2. Is it strictly less than 2 also for $n > 4$? Even for Euclidean instances this is open.

As noted in [3], in many applications some bi-directed edges might be allowed. This opens up a whole new set of questions on the trade-off between dilation and the number of bi-directed edges. Since this is a generalisation of the oriented case, our hardness proof also applies to such models.

Acknowledgements. We thank Guangping Li and Marco Ricci for helpful discussions.

References

1 Boris Aronov, Kevin Buchin, Maike Buchin, Bart Jansen, Tom De Jong, Marc van Kreveld, Maarten Löffler, Jun Luo, Bettina Speckmann, and Rodrigo Ignacio Silveira. Connect the dot: Computing feed-links for network extension. *J. Spatial Information Science*, 3:3–31, 2011. doi:10.5311/JOSIS.2011.3.47.

2 Prosenjit Bose and Michiel Smid. On plane geometric spanners: A survey and open problems. *Comput. Geom. Theory Appl.*, 46(7):818–830, 2013. doi:10.1016/j.comgeo.2013.04.002.

3 Kevin Buchin, Joachim Gudmundsson, Antonia Kalb, Aleksandr Popov, Carolin Rehs, André van Renssen, and Sampson Wong. Oriented spanners. In Inge Li Gørtz, Martin Farach-Colton, Simon J. Puglisi, and Grzegorz Herman, editors, *31st Annual European Symposium on Algorithms, ESA 2023, September 4-6, 2023, Amsterdam, The Netherlands*, volume 274 of *LIPIcs*, pages 26:1–26:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPIcs.ESA.2023.26.

4 Martin Burkhart, Pascal Von Rickenbach, Rogert Wattenhofer, and Aaron Zollinger. Does topology control reduce interference? In *Proc. 5th ACM Internat. Sympos. Mobile Ad Hoc Networking and Computing*, pages 9–19, 2004. doi:10.1145/989459.989462.

5 Andrew Dobson and Kostas E. Bekris. Sparse roadmap spanners for asymptotically near-optimal motion planning. *Internat. J. Robotics Research*, 33(1):18–47, 2014. doi:10.1177/0278364913498.

6 David Eppstein. Spanning trees and spanners. In *Handbook of Computational Geometry*, pages 425–461. Elsevier, 2000. doi:h10.1016/B978-044482537-7/50010-3.

7 Michael Godau. *On the complexity of measuring the similarity between geometric objects in higher dimensions*. Dissertation, Free University Berlin, 1999. doi:10.17169/refubium-7780.

8 David Lichtenstein. Planar formulae and their uses. *SIAM Journal on Computing*, 11(2):329–343, 1982. doi:10.1137/0211025.

9 Giri Narasimhan and Michiel Smid. *Geometric Spanner Networks*. Cambridge University Press, 2007. doi:10.1017/CBO9780511546884.

10 Christian Schindelhauer, Klaus Volbert, and Martin Ziegler. Geometric spanners with applications in wireless networks. *Comput. Geom. Theory Appl.*, 36(3):197–214, 2007. doi:10.1016/j.comgeo.2006.02.001.

Pairwise Triangles Intersections in a Query Rectangle

Waseem Akram[1] **and Sanjeev Saxena** [2]

1 **Indian Institute of Technology, Kanpur(India)**
akram@iitk.ac.in

2 **Indian Institute of Technology, Kanpur(India)**
ssax@iitk.ac.in

Abstract

The range searching problem is one of the most studied problems in computational geometry. In this paper, we study the following range-searching problem. Given a set of n homothetic right-angled triangles in the plane, we want to compute the pairs of triangles intersecting inside an axis-aligned query rectangle. A triangle T is said to be homothetic to another triangle T' if T can be obtained from T' using scaling and translation operations. We show that after preprocessing the given set in $O(n \log n)$-space and time, each subsequent query can be answered in $O(\log n + k)$-time, where k is the number of reported pairs.

1 Introduction

Range searching problems are fundamental in computational geometry and find applications in numerous domains, including motion planning, robotics, and spatial databases [2, 4, 5]. The range searching problems involving orthogonal objects (points, line segments, rectangles) have been well explored [3, 4, 7]. In this paper, we study a range-searching problem that considers a class of non-orthogonal objects (homothetic triangles). A triangle T is said to be homothetic to another triangle T' if T can be obtained from T' using scaling and translation operations. The problem is defined as follows.

> Given a set of n homothetic right-angled triangles with perpendicular sides parallel to the coordinate axes, we want to preprocess the set so that, given an axis-aligned query rectangle Q, all the pair of triangles (T_i, T_j) intersecting inside the rectangle Q (i.e. $T_i \cap T_j \cap Q \neq \emptyset$) can be reported efficiently.

This problem is a generalization of the problem studied in [4, 5], which asks to compute all pairs of rectangles intersecting inside a query rectangle. Mark de Berg et al. [4] solved the problem using $O(n \log n)$ space and $O(\log n \log^* n + k \log n)$ query time, where k is the output size. Oh and Ahn [5] improved the query time to $O(\log n + k)$ without aggravating the space bound. The problem finds application in motion planning. A common application scenario for such problems is described next. We can think that geometric objects are the trajectories of moving objects (drones/airplanes in the sky or robots in a factory), and we would like to know parts of trajectories where two entities may collide so that one can take appropriate measures to avoid collisions. Furthermore, we likely want to ask this question for a particular small region only.

Our Result: We show that after preprocessing in $O(n \log n)$-space and time, each query can be answered in $O(\log n + k)$-time, where k is the number of reported pairs of triangles.

40th European Workshop on Computational Geometry, Ioannina, Greece, March 13–15, 2024.
This is an extended abstract of a presentation given at EuroCG'24. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

Definitions and Notations

Let $S = \{T_1, T_2, \ldots, T_n\}$ be a set of n homothetic right-angled triangles in the plane such that their orthogonal sides are parallel to the coordinate axes. Without loss of generality, we assume that all the triangles are present in the first quadrant and their right-angled vertices are at the bottom-left position (i.e., minimum x and minimum y coordinates). Let m be the slope of the hypotenuses of the triangles in S. The horizontal, vertical, and hypotenuse sides of a triangle T_i are denoted by h_i, v_i, and hy_i respectively.

Let l_i be the line containing the segment hy_i, and let l_0 be the line passing through the origin $(0, 0)$ with slope m. For each $i \in \{1, 2, \ldots, n\}$, we define the *distance* of the segment hy_i from the line l_0, denoted by d_i, as the Euclidean distance between the parallel lines l_0 and l_i. We say that $T_i \preceq T_j$ if $d_i \leq d_j$.

Let ab be a side of a triangle $T \in S$ (with a and b as its endpoints). The *stretch* of ab, denoted by $a'b'$, is the smallest segment of ab which contains all the intersection points of the side ab with sides of other triangles of S. If ab has no intersection point, then *stretch* of ab is undefined. Thus, a triangle in S can have at most three stretches, one for each side. Let P' be the set of their endpoints. Note that $|P'| \leq 6n$. We say that a line segment s crosses a rectangle Q if $s \cap Q \neq \emptyset$ and each endpoint lies outside Q (see Figure 1).

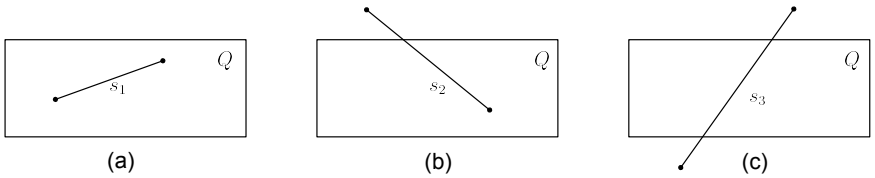


Figure 1 Segments s_1 and s_2 do not cross Q, while the segment s_3 does.

2 Proposed Solution

We design a space-efficient data structure that supports queries in optimal time. Our approach is similar to the one used in [5]. For a given query rectangle Q, we need to compute all the pairs (T_i, T_j) of S such that T_i and T_j intersect each other inside Q. Observe that the intersection of T_i and T_j, denoted by $I(i, j)$, is also a triangle homothetic to the triangles in S. We identify a few configurations for a given Q, which will be used to compute the output set $U(Q)$. The size of the set $U(Q)$ is denoted by $k(Q)$. For any pair (T_i, T_j) of triangles of S with $I(i, j) \cap Q \neq \emptyset$, it can be seen that at least one of the following conditions holds. The configurations have been depicted in Figure 2.

- C1: T_i contains Q and T_j intersects Q or vice-versa.
- C2: An endpoint of a stretch l of T_i (or T_j) lies in Q, and T_j (or T_i) intersects $l \cap Q$.
- C3: A stretch l_i of T_i and a stretch l_j of T_j cross Q and intersect each other inside Q.
- C4: $I(i, j)$ contains a vertex of Q.
- C5: The vertical sides of Q intersect the horizontal and hypotenuse sides of $I(i, j)$, or the horizontal sides of Q intersect the vertical and hypotenuse sides of $I(i, j)$.

2.1 Data structures

We now build a set of data structures that will be used to compute the pairs of the configurations mentioned above.

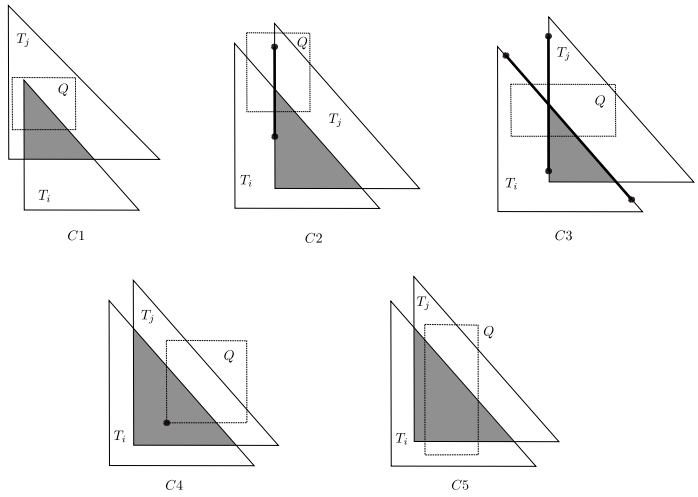


Figure 2 The gray region denotes the intersection $I(i, j)$ of T_i and T_j in each configuration. The bold segment of a side denotes its stretch.

- *Enclosure Searching Problem*: Given a set of geometric objects, the problem is to find the objects o enclosing a query object q (i.e., $q \in o$). We preprocess the set S of triangles for enclosure searching queries as described in [1]. Let D_{enc} denotes the built data structure. The structure D_{enc} supports enclosure searching queries for points, line segments, and trapezoids in $O(\log n + t)$-time, where t is the number of reported triangles.
- *The Orthogonal Range Searching Problem*: Given a set of points in the d-dimensional space and an orthogonal rectangle, the problem is to find all the points lying inside the rectangle. Chazelle [3] gave an optimal solution to the problem in the 2-dimensional space, which takes $O(\log n + \#output)$-query time and $O(n \log n / \log \log n)$-space. We build an orthogonal range reporting data structure for the set P' of stretch endpoints using Chazelle's method [3] and denote it by D'_r.
- *The Segment Intersection Problem*: Given a set of line segments and a line segment q, the segment intersection problem is to find the segments s intersected by the segment q (i.e., $s \cap q \neq \emptyset$). The following result is due to Chazelle [3].

  > **Theorem 1 in [3]**: It is possible to preprocess n segments in $O(n \log n)$ time and $O(n)$-space so that computing their intersections with a query segment that either has a fixed slope or has its supporting line passing through a fixed point can be done in $O(k + \log n)$ time, where k is the number of intersections to be reported. It is assumed that the interior of the segments are pairwise disjoint.

  We now describe an overview of the Chazelle's solution. Given a set of pairwise (interior) disjoint line segments, a planar subdivision is built. The subdivision is then preprocessed for *point location queries*, which, given a point in the plane, finds the face of the subdivision containing the point. Given a query segment, the face containing an endpoint of the segment is located and one moves towards the other endpoint along the query segment. The segments encountered on the way are exactly those which are intersected by the query segment. The query takes $O(\log n + t)$-time, where t is the output size: $O(\log n)$ time for locating the face and $O(t)$ time for reporting the segments. Note that if we know the face containing an endpoint of the query segment, then query time would be $O(t)$. For more details, please see [3].
  Let H' be the set of all horizontal stretches. We build two data structures D_h^v and D_h^{hy}

over the set H' using Chazelle's method [3]: D_h^v supports queries with vertical segments, and D_h^{hy} supports queries with segments along the hypotenuse-sides. Similarly, we build the structures D_v^h and D_v^{hy} over the set V' of all vertical stretches, and D_{hy}^v and D_{hy}^h over the set H'_y of all stretches with slope m. We keep two pointers with each stretch-endpoint $a' \in P'$ to save time while answering a query. If a' is an endpoint of a vertical stretch, we store two pointers pointing to the faces of D_h^v and D_{hy}^v containing the endpoint a'. Analogously, we store pointers for the endpoints of the other types of stretches.

2.2 Query Algorithms

Let Q be the query rectangle. A pair (T_i, T_j) with $I(i,j) \cap Q \neq \emptyset$ may belong to more than one configurations. We denote by k_i the number of C_i-pairs not belonging to any other configuration C_j with $j < i$.

Reporting $C1$***-pairs***: We find the set E of the triangles enclosing Q by querying the data structure D_{enc} in $O(\log n + |E|)$-time. For each triangle $T_i \in E$, a triangle $T_j \in S$ intersecting Q would form a $C1$-pair with T_i. If the set E is empty, then no $C1$-pair exists. The triangles $T_i \in S$ intersecting Q can be computed in $O(\log n + \#output)$-time due to the Lemma 2.1.

▶ **Lemma 2.1.** *We can preprocess the set S in $O(n \log n)$-time and space so that given a query rectangle Q, the triangles $T \in S$ with $T \cap Q \neq \emptyset$ can be computed in $O(\log n + \#output)$-time.*

▶ **Corollary 2.2.** *Given a query rectangle Q, we can compute the $C1$-pairs for Q in $O(\log n + k_1)$-time and $O(n \log n)$-space.*

Reporting $C2$***-pairs***: We find the stretches with an endpoint inside Q by querying the structure D'_r. For each reported stretch l, we compute the triangles intersecting $l \cap Q$ using the segment intersection structures. If l is the stretch of the vertical side of a triangle T_i, we can compute the horizontal and hypotenuse sides intersecting $l \cap Q$ using D_h^v and D_{hy}^v, respectively. The details are omitted due to the space limitations.

▶ **Lemma 2.3.** *We can compute all $C2$-pairs for Q in $O(\log n + k_2)$-time.*

Reporting $C3$***-pairs***: We first consider the simpler case of orthogonal stretches: a vertical stretch l_i and a horizontal stretch l_j crossing Q. We can compute such pairs of stretches in $O(\log n + t)$-time using 3-d range reporting queries [4, 6], where t is the number of reported pairs. The space used is $O(n \log n / \log \log n)$.

We now consider the case of computing all pairs $(s_1, s_2), s_1 \in H'$ and $s_2 \in H'_y$, such that both stretches cross Q and intersect each other inside Q; the symmetric case of vertical and hypotenuse stretches is analogous. We design a segment tree-based data structure and give an algorithm that takes $O(\log n + \#output)$-time to find all such pairs. The space used by the structure is $O(n \log n)$.

▶ **Lemma 2.4.** *All the $C3$-pairs for Q can be computed in $O(\log n + k_3)$-time. The space used is $O(n \log n)$.*

Reporting $C4$***-pairs***: For each vertex of Q, we find the set of the triangles enclosing the vertex using the structure D_{enc}. If the set size is less than two, no pair of triangles enclosing the vertex exists. Otherwise, every possible pair of triangles would be a $C4$-pair. Of course, we make sure that each pair is reported once.

▶ **Lemma 2.5.** *One can compute all $C4$-pairs for Q in $O(\log n + k_4)$-time.*

Reporting $C5$***-pairs***: We show how to find the $C5$-pairs (T_i, T_j) not belonging to any other configuration such that the non-vertical sides of $I(i,j)$ are intersected by the vertical sides of Q. The $C5$-pairs of the other type can be computed analogously.

We build a segment tree $\mathcal{T}$ over S along the x-axis. Each node $v \in \mathcal{T}$ stores a vertical slab $SL(v) = [x, x'] \times \mathbb{R}^2$ such that the vertical slabs corresponding to the nodes of a particular level form a partition of $\mathbb{R}^2$. We say that a triangle $T \in S$ spans a slab $SL(v)$ if $T \cap SL(v) \neq \emptyset$ and none of its vertex lies in $SL(v)$. We store two sets $S_c(v)$ and $S_b(v)$ at node v. The set $S_c(v)$ consists of the triangles $T \in S$ such that T spans the slab $SL(v)$ but not the slab of v's parent node. The set $S_b(v)$ consists of the triangles $T \in S$ such that one or both endpoints of its horizontal side lies in $SL(v)$. We denote by $S(v)$ the union of the sets $S_c(v)$ and $S_b(v)$. The segment tree $\mathcal{T}$ can be built in $O(n \log n)$-time and space.

Consider a pair (T_i, T_j) of triangles of S with $T_i \cap T_j \cap Q \neq \emptyset$. A node v in the tree $\mathcal{T}$ is said to be a canonical node of (i,j,Q) if the left side l_Q of the rectangle Q lies in the interior or on the left side of $SL(v)$, and $T_i, T_j \in S(v)$ such that $T_i \in S_c(v)$ or $T_j \in S_c(v)$. We have the following results related to the canonical nodes.

▶ **Lemma 2.6.** *For each $C5$-pair (T_i, T_j) of Q such that the non-vertical sides of $I(i,j)$ intersect the vertical sides of Q, there is a canonical node of (i,j,Q) in $\mathcal{T}$.*

Proof. The proof is analogous to that of Lemma 3 in [5]. ◀

▶ **Lemma 2.7.** *For any rectangle Q and any pair (T_i, T_j) of rectangles of S with $I(i,j) \cap Q \neq \emptyset$, there is at most one canonical node of (i,j,Q) in $\mathcal{T}$.*

Proof. The proof is analogous to that of Lemma 4 in [5]. ◀

We have the following corollary from Lemma 2.6 and Lemma 2.7.

▶ **Corollary 2.8.** *The total number of canonical nodes for a query rectangle Q is $O(k(Q))$.*

▶ Remark. A node in $\mathcal{T}$ could be the canonical node for more than one pair $(T_i, T_j) \in U(Q)$.

Given a rectangle Q, we compute a set of nodes of $\mathcal{T}$ that includes the unique canonical node of (i,j,Q) for each $C5$-pair (T_i, T_j) not belonging to any other configuration such that the non-vertical sides of $T_i \cap T_j$ intersect the vertical sides of Q. For each such node v, we find all $C5$-pairs such that v is the canonical node of (i,j,Q).

Finding all canonical nodes for $C5$-pairs: We compute a set $\mathcal{V}_Q$ of canonical nodes which contains the canonical node of (i,j,Q) for every $C5$-pair (T_i, T_j). For computing such a set, we define the *trimmed polygon* for a pair (T, v), $T \in S(v)$, as the smallest simple polygon containing the region $T \cap SL(v) \cap U(v)$. Here, $U(v)$ is the union of all the triangles in $S_c(v)$ except T if $T \in S_c(v)$. See Figure 3 for examples. The trimmed polygon for (T, v) has at most two sides with slope m; the number of horizontal and vertical sides are bounded above by 2 and 3, respectively. We compute the required set $\mathcal{V}_Q$ by finding the sides of all trimmed polygons intersected by the left side Q (the details are omitted).

▶ **Lemma 2.9.** *Given a query rectangle Q, we can find a set of at most k nodes containing all canonical nodes of $C5$-pairs not belonging to any other configuration in $O(\log n + k)$ time.*

Handling each canonical node to find all $C5$-pairs: Let $\mathcal{V}_Q$ be the set of canonical nodes for Q computed due to Lemma 2.9. We now compute all $C5$-pairs present at each node $v \in \mathcal{V}_Q$. For this, we have the following lemma.

▶ **Lemma 2.10.** *For each $v \in \mathcal{V}_Q$, we can compute all $C5$-pairs (T_i, T_j) such that v is the canonical node of (i,j,Q) in $O(k_5)$-time. The preprocessing takes $O(n \log n)$-time and space.*

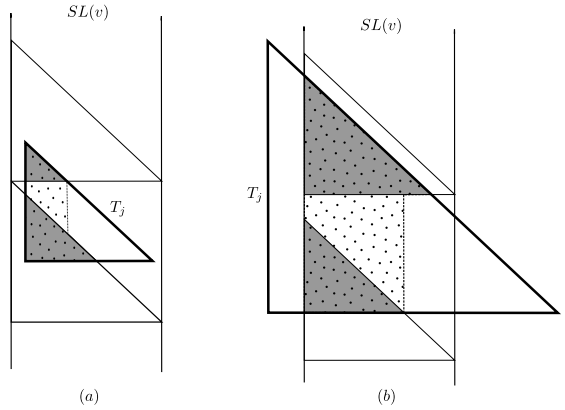


Figure 3 The trimmed polygon (dotted region) for (T_j, v) when (a) $T_j \in S_b(v)$ and (b) $T_j \in S_c(v)$.

We finally put together the results for all configurations and got the following theorem.

▶ **Theorem 2.11.** *Given a set of n right-angled homothetic triangles, one can preprocess so that all pairs of triangles intersecting inside a query rectangle can be computed in $O(\log n + \#output)$-time. The space complexity is $O(n \log n)$.*

References

1 Waseem Akram and Sanjeev Saxena. Dominance for containment problems. *arXiv preprint arXiv:2212.10247*, 2022.

2 Kreveld Overmars Berg, Cheong. *More Geometric Data Structures*, pages 219–241. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008. doi:10.1007/978-3-540-77974-2_10.

3 Bernard Chazelle. Filtering search: A new approach to query-answering. *SIAM Journal on Computing*, 15(3):703–724, 1986. doi:10.1137/0215051.

4 Mark de Berg, Joachim Gudmundsson, and Ali D. Mehrabi. Finding pairwise intersections inside a query range. *Algorithmica*, 80(11):3253–3269, Nov 2018. doi:10.1007/s00453-017-0384-3.

5 Eunjin Oh and Hee-Kap Ahn. Finding pairwise intersections of rectangles in a query rectangle. *Computational Geometry*, 85:101576, 2019. doi:10.1016/j.comgeo.2019.101576.

6 Kalina Petrova and Robert Tarjan. A dynamic data structure for segment in-tersection queries. URL: http://www.moi.math.bas.bg/moiuser/~ACCT2016/a41.pdf.

7 Saladi Rahul, Ananda Swarup Das, K. S. Rajan, and Kannan Srinathan. Range-aggregate queries involving geometric aggregation operations. In *WALCOM: Algorithms and Computation*, pages 122–133, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

On Orbital Labeling with Circular Contours*

Annika Bonerath[1], Martin Nöllenburg[2], Soeren Terziadis[3], Markus Wallinger[2], and Jules Wulms[3]

1 **University of Bonn**
`bonerath@igg.uni-bonn.de`
2 **Algorithms and Complexity Group, TU Wien**
`{noellenburg, mwallinger}@ac.tuwien.ac.at`
3 **TU Eindhoven**
`{s.d.terziadis, j.j.h.m.wulms}@tue.nl`

Abstract

Schematic depictions in text books and maps often need to label specific point features with a text label. We investigate one variant of such a labeling, where the image contour is a circle and the labels are placed as circular arcs along the circumference of this circle. To map the labels to the feature points, we use orbital-radial leaders, which consist of a circular arc concentric with the image contour circle and a radial line to the contour. In this paper, we provide a framework, which captures various dimensions of the problem space as well as several polynomial time algorithms and complexity results for some problem variants.

1 Introduction

Map labeling is an extensively studied topic in computational geometry [1,7,12] that typically involves annotating feature points with names or additional descriptions, ensuring non-overlapping annotations. While traditional maps often use *internal* label positions next to the feature points [11], *external* labeling models [4] place labels remotely along the contour of a bounding shape and connect them to their feature points by crossing-free leaders. This model is frequently used in applications, where feature points are dense, the details of a map or an illustration should not be obscured by labels, or labels are relatively large, e.g., in

* AB was partially funded by German Research Foundation under Germany's Excellence Strategy, EXC-2070 - 390732324 - PhenoRob. ST and MW were funded by the Vienna Science and Technology Fund (WWTF) [10.47379/ICT19035]. ST was funded by the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement Grant Agreement No 101034253.

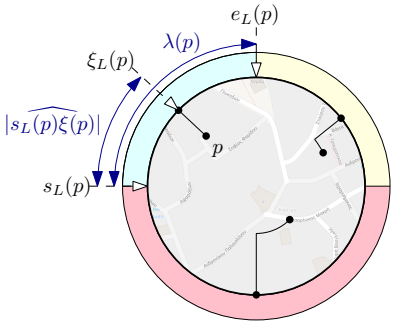


Figure 1 An orbital labeling on a map for illustrating our notation.

40th European Workshop on Computational Geometry, Ioannina, Greece, March 13–15, 2024.
This is an extended abstract of a presentation given at EuroCG'24. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

anatomy atlases or assembly drawings. In this paper we study a novel variant of external labeling with a circular bounding shape, e.g., for displays of smartwatches; see Figure 1. The circular map is displayed in the center of the display and each label is bent and turned into a segment of the circular boundary of the map; we call these labels *orbital* labels. This is a special case of external and boundary labeling [3]. We assume that the lengths of the orbital labels are normalized and sum up to the perimeter of the boundary of the map. Previous research on circular map display considered either multirow circular labels where the sum of label lengths does not equal the map's boundary length [8], radial labels [2,6], or horizontal labels [6,9,10]. The latter two settings are relevant for circular maps on rectangular displays but not suitable for circular displays with a narrow annulus for labels.

Formally, we assume that we are given a disk D in the plane $\mathbb{R}^2$. The disk contains n points $P = \{p_1, \ldots, p_n\}$. We call the set P of points *features* and we refer to the boundary of the disk as *the boundary* B. Every feature $p \in P$ has an associated *label* which represents additional information that is to be placed along a circular arc on the boundary starting at a point $b_1 \in B$ and ending at a point $b_2 \in B$. The circular arc along B is denoted as $\overset{\frown}{b_1 b_2}_B$. Usually, the start and endpoint of the label are not fixed in the input, however, the length of the arc is part of the input. We represent the associated label simply as a number $\lambda(p)$, which indicates the length of the associated label. We assume that $\sum_{i=1}^{n} \lambda(p_i)$ is equal to the circumference of D, i.e., if all labels are placed non-overlapping then there are no gaps between the arcs on B.

In a *labeling* L, every feature $p \in P$ is assigned a label with starting point $s_L(p) \in B$ and an endpoint $e_L(p) \in B$, s.t., $|\overset{\frown}{s_L(p) e_L(p)}| = \lambda(p)$. We assume that all labels are pairwise non-overlapping. Additionally, every feature p is connected to its label via a *leader*. In this paper, we consider *orbital-radial* leaders, which consist of two parts: (1) starting at the feature p with a (possibly empty) orbital circular arc that ends at a bend point q, and (2) a radial segment that connects q to the boundary B; see Figure 1. We call the leader endpoint, i.e., the point where the leader B the *port* $\xi_L(p)$ of the leader starting at p. Note that q has the same distance to the circle center as p since the first part of the leader is an orbital-radial arc. We denote the length of the leader of feature p by $l(p)$. Let the *port ratio* $\rho_L(p) = \frac{|\overset{\frown}{s_L(p) \xi(p)}|}{\lambda(p)}$ be the ratio of the arc from the starting point to the port and the arc from the start-point to the end-point. Now, we define the generic orbital labeling problem.

▸ **Problem 1.** *Given a disk D, containing n feature points P compute a labeling L, in which all leaders are pairwise non-intersecting and the sum of leader lengths is minimal.*

We discuss different variants of the problem and give an overview of the obtained running times. The paper provides detailed explanations for a subset of the results. The remaining results and proofs of statements marked with a star (*) will be presented in the full version.

2 Problem Space

In the following, we discuss the dimensions of our problem space. For the different dimensions, we use the notation based on the *COSA*-Orbital Boundary Labeling scheme and use each letter to describe the variants for the respective dimension.

- [C] **Candidate port positions on the boundary.** If we are given a set C of candidate positions on B and in any valid labeling L we require that for any port $\xi \in \Xi_L$ we have $\xi \in C$, we say *the port candidates are locked* (and use the symbol $C^{\text{🔒}}$) otherwise they are *free* ($C^{\text{🔓}}$).

Table 1 A tabular overview of the problem space and our results. Empty cells remain open. Results marked with † will be presented in the full version.

			$A_{\equiv}^{\text{🔒}}$	$A_{\equiv}^{\text{🔑}}$	$A_{\not\equiv}^{\text{🔒}}$	$A_{\not\equiv}^{\text{🔑}}$
$C^{\text{🔒}}$	$O^{\text{🔒}}$	$S_{\equiv}$	$O(n^2\lvert C\rvert)$ †		$O(n^2\lvert C\rvert)$ †	
		$S_{\not\equiv}$	$O(n^2\lvert C\rvert)$ †		$O(n^2\lvert C\rvert)$ †	
	$O^{\text{🔑}}$	$S_{\equiv}$	$O(n\lvert C\rvert^3)$ †			
		$S_{\not\equiv}$				
$C^{\text{🔑}}$	$O^{\text{🔒}}$	$S_{\equiv}$	$O(n^2)$ [Sec. 4.1]		$O(n^2)$ [Sec. 4.1]	
		$S_{\not\equiv}$	$O(n^2)$ [Sec. 4.1]		$O(n^2)$ [Sec. 4.1]	
	$O^{\text{🔑}}$	$S_{\equiv}$	$O(n^5)$ [Sec. 4.2]			
		$S_{\not\equiv}$	NP-c [Sec. 4.2.1]	NP-c [Sec. 4.2.1]	NP-c [Sec. 4.2.1]	

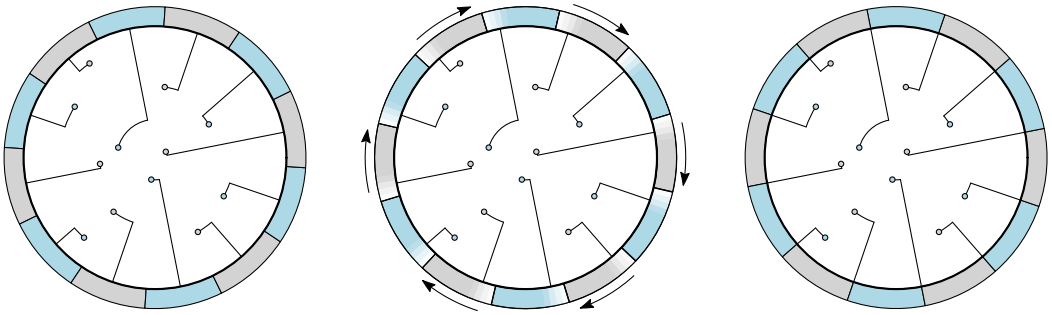
- [O] **Order.** Next, we consider the cyclic order of labels around B. If a certain label order is pre-specified we say the label order is locked ($O^{\text{🔒}}$); otherwise, for the unconstrained setting, we say the label order is free ($O^{\text{🔑}}$).
- [S] **Size of labels.** Then, we distinguish the setting where $\forall p \in P : \lambda(p) = 1$, in which case we say that the label size is *uniform* ($S_{\equiv}$), otherwise the label size is *non-uniform* ($S_{\not\equiv}$).
- [A] **Port position on labels.** Lastly, we distinguish different positions of the ports on the labels. We differentiate between *uniform* port ratios, where $\forall i, j\ \rho_L(p_i) = \rho_L(p_j)$, and *non-uniform* port ratios. We also distinguish between the ratios being predefined as part of the input, in which case we call the ratios *locked*, or not, in which case we call them *free*. We obtain the following four settings:
  - Ratios are uniform and locked to a value $k \in [0, 1]$ given in the input ($A_{\equiv}^{\text{🔒}}$).
  - Ratios are uniform and free, i.e., we have to find a value $k \in [0, 1]$ for the ratios ($A_{\equiv}^{\text{🔑}}$).
  - Ratios are non-uniform and locked, meaning, we are given a set $K = \{k_1, \ldots, k_n\}$ of ratios, s.t., in a valid labeling L, we have $\rho_L(p_i) = k_i$ ($A_{\not\equiv}^{\text{🔒}}$).
  - Ratios are non-uniform and free, i.e., ports can be chosen freely and independently ($A_{\not\equiv}^{\text{🔑}}$).

For our problem variants, we use the notation based on the $COSA$-Orbital Boundary Labeling scheme where we substitute C, O, S and A with $C^{\text{🔑}}/C^{\text{🔒}}$, $O^{\text{🔑}}/O^{\text{🔒}}$, $S_{\equiv}/S_{\not\equiv}$ and $A_{\not\equiv}^{\text{🔑}}/A_{\equiv}^{\text{🔑}}/A_{\equiv}^{\text{🔒}}/A_{\not\equiv}^{\text{🔒}}$, respectively. An overview of all variants and our results can be seen in Table 1. Whenever a statement applies to all variants along a certain dimension of the problem space, we drop the sub- or superscript of C, O, S, or A. For example, $C^{\text{🔑}}O^{\text{🔒}}SA_{\equiv}^{\text{🔒}}$ refers to the variants where the port candidates are free ($C^{\text{🔑}}$), the order is locked ($O^{\text{🔒}}$), the label sizes could be fixed to be uniform or they could be non-uniform (S) and all port ratios are fixed to a given value ($A_{\equiv}^{\text{🔒}}$). Therefore, $C^{\text{🔑}}O^{\text{🔒}}SA_{\equiv}^{\text{🔒}}$ covers a set of two problem variants.

3 Uniformly Spaced Ports

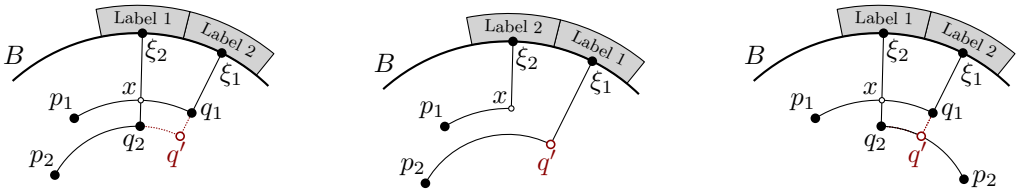
Using a simple argument about shifting labels illustrated in Figure 2, we can show the following equivalence.

▸ **Observation 1.** *All problems in $COS_{\equiv}A_{\equiv}^{\text{🔒}}$ are equivalent over all $k \in [0, 1]$. Similarly all problems in $COS_{\equiv}A_{\equiv}^{\text{🔑}}$ are equivalent over all $k \in [0, 1]$.*



(a) Uniform ratio of 0.5. **(b)** Clockwise rotation. **(c)** Uniform ratio of 0.

Figure 2 Any solution with uniform label sizes and a uniform ratio (e.g., 0.5) (a) can be rotated (b) to obtain a solution of any other ratio, e.g., 0 (c).



(a) Case 1 **(b)** Rerouted leaders **(c)** Case 2

Figure 3 Given a free label order O^f we can reroute the leaders to arrive at a crossing-free solution with a shorter total leader length.

This equivalence is based on the fact that the ports in these problems are necessarily equally spaced, which is only the case if both the label size and the port ratio are uniform. Based on the same property, we make the following statement, visualized in Figure 3.

▸ **Lemma 3.1 (*).** *Given an instance of a problem variant in $CO^{\bullet}SA_{\equiv}$ any leader-length minimal labeling L is crossing-free, assuming that all feature points in P lie on circles of different radii concentric with D.*

4 Free Candidates

In this section, we consider the problem set $C^{\bullet}OSA$. Intuitively, these are problem sets, where solutions can be continuously rotated around B. Let $g : P \times [0, 2\pi] \to \mathbb{R}$ be a function which maps a feature $p \in P$ and an angle θ to the length of a leader that connects p and a port on B, s.t., the orbital segment of the leader spans the angle θ. Let r be the radius of the circle containing p concentric with D. If D has a circumference of $\mathcal{C}$ it has a radius of $\frac{\mathcal{C}}{2\pi}$. Then $g(p, \theta) = \frac{\mathcal{C}}{2\pi} - r + r\theta$; see Figure 4.

▸ **Observation 2.** *The function $g(p, \theta)$ is linear in θ.*

The total leader length of a labeling L can obtained as $h(L) = \sum_{i=1}^{n} g(p_i, \theta(p_i))$, where $\theta(p_i)$ is the angle spanned by the orbital segment of the leader connected to p_i.

Note that by fixing a port ξ on the boundary for a feature p, there are two orbital-radial leaders by which we could choose to connect them (with a clockwise or a counter-clockwise orbital segment). We call these *clockwise* and *counter-clockwise* leader, respectively.

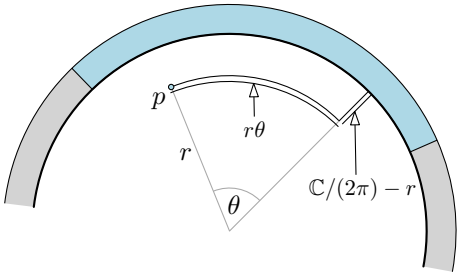


Figure 4 Determining the leader length by the length of the orbital and radial segment.

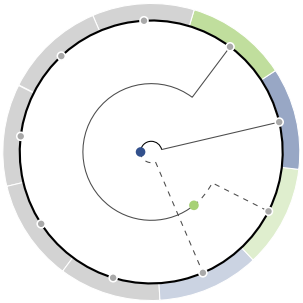


Figure 5 The port position of the inner-most feature (blue) determines labeling of other features: Depending on the labeling of the inner-most feature, the green feature point has access to different candidate ports (dark labels and solid leaders vs. light labels and dashed leaders).

▸ **Observation 3.** *The inner-most feature, i.e., the feature which lies on a circle concentric with D whose radius is smallest among all features can always be labeled with a clockwise or a counterclockwise leader.*

Observe that the leader of the inner-most feature p uses a radial line segment s starting on a circle concentric with D, which does not contain any other feature of P. Consider any other feature p' and any other point ξ' on B, then, p' and ξ' can be connected either with a clockwise or a counter-clockwise leader; see Figure 5. The orbital segments of the clockwise and the counter-clockwise leader of p' together form an entire circle concentric with D containing p and, hence, one of them has to intersect s.

▸ **Observation 4.** *A leader of the inner-most feature determines for every other leader γ connecting a feature and a point on B if γ is a clockwise or counter-clockwise leader.*

4.1 Locked Order.

Next, we consider problems in $C^{\text{🔑}}O^{\text{🔒}}SA$.

▸ **Lemma 4.1.** *For $C^{\text{🔑}}O^{\text{🔒}}S_{\equiv}A^{\text{🔒}}_{\equiv}$, $C^{\text{🔑}}O^{\text{🔒}}S_{\equiv}A^{\text{🔒}}_{\equiv}$ and by extension $C^{\text{🔑}}O^{\text{🔒}}S_{\equiv}A^{\text{🔑}}_{\equiv}$, the choice of a port point on B for the inner-most feature determines all other label placements including their port positions as well as their leaders. This also includes the length of their leaders and the angle that is spanned by the orbital segment of these leaders.*

Proof. This lemma directly follows from Observation 4 and the key point that the placement of one label not only determines the placement of others but also their port positions. ◂

With this, we state a method of solving the four problems $C^{\text{🔑}}O^{\text{🔒}}SA^{\text{🔒}}$ and by extension $C^{\text{🔑}}O^{\text{🔒}}S_{\equiv}A^{\text{🔑}}_{\equiv}$ (recall Observation 1). By Lemma 4.1 the exact position of the port of the

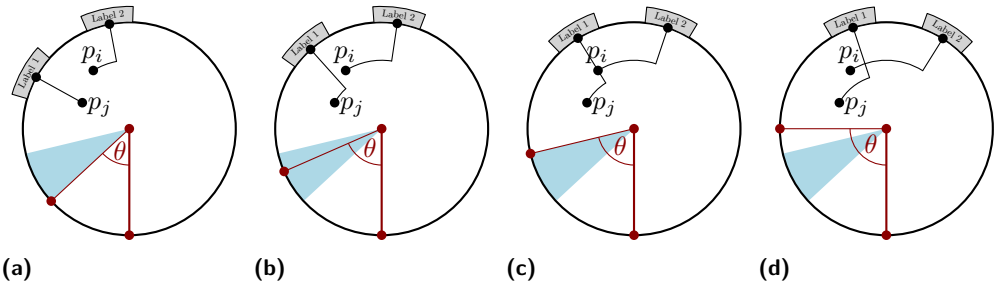


Figure 6 Two clockwise leaders whose ports are rotated, s.t., (a) the length of the orbital segment of p_j is 0, (b) the leaders are non intersecting, (c), the radial segment of p_j contains p_i and (d) the leaders intersect. The admissible range of θ is shown in blue.

inner-most feature p_1 is the only degree of freedom when choosing a labeling. By Lemma 4.1, we immediately obtain the angle θ_1 spanned by its orbital segment. By Lemma 4.1, we likewise obtain all angles $\theta_2, \ldots, \theta_n$ of all the other leaders. Therefore we can express the functions $g(p_2, \theta_2), \ldots, g(p_n, \theta_n)$ all as piecewise linear functions of θ_1, which consist of exactly two linear pieces. The sum over all of these functions is therefore a piece-wise linear uni-variate function and we can find the minimum of it in $O(n)$ time.

To guarantee that a solution (if it exists) found in this way is crossing free we compute an *admissible range* $I_{i,j}$ for θ_1, s.t., if $\theta_1 \in I_{i,j}$ the leaders of p_i and p_j are crossing free; see Figure 6. $I_{i,j}$ is one continuous interval and therefore we can in $O(n^2)$ time determine all ranges as well as their (also continuous) intersection if it exists. Then we either restrict our search for a minimum to this intersection or – if the intersection is empty – know that no solution exists.

4.2 Free Order.

These are the problems in $C^{\not}O^{\not}SA$. We will use a reduction of some of these problems to a non-circular variant called Boundary Labeling [5].

▸ **Lemma 4.2.** *In any (crossing-free) labeling of an instance of a problem in $C^{\not}O^{\not}SA$, there exists a point $b \in B$, s.t., db does not intersect any leader, where d is the center of D.*

Proof. Let x be the smallest angle between two points, two ports, or a point and a port in an optimal labeling L (measured with 0 as the center). Consider the radial segment of the leader of the inner-most feature p_1, and assume w.l.o.g. that the orbital segment is clockwise. Set b' to be $\xi(p_1)$ but rotate it clockwise by $x/2$. Since the leader of p_1 does not intersect any other leader and the next feature or port is at least at an angle x in clockwise position from $\xi(p_1)$, the segment db' must now be crossing free. ◂

The previous lemma argues the existence of this *splitting line* in any labeling. Next, we state that we only need to consider $O(n^2)$ possibilities for such a line.

▸ **Lemma 4.3 (*).** *For any problem in $C^{\not}O^{\not}S_{\equiv}A_{\equiv}$ there are only n^2 possibilities for the port of the inner-most feature.*

We obtain an algorithm for the problems $C^{\not}O^{\not}S_{\equiv}A_{\equiv}$ by creating an instance of Boundary Labeling for every possible port of the inner-most feature and using the $O(n^3)$ algorithm [5] to obtain a labeling in a total time of $O(n^5)$; see algorithm in the full version.

▸ **Theorem 4.4.** *Any problem in $C^{\not}O^{\not}S_{\equiv}A_{\equiv}$ can be solved exactly in $O(n^5)$.*

4.2.1 Non-uniform label sizes are NP-hard.

Finally, we investigate problems without candidate ports, a free order on the labels and non-uniform label sizes. For $C^{\bullet}O^{\bullet}S_{\equiv}A_{\equiv}$, we show NP-hardness and the hardness of $C^{\bullet}O^{\bullet}S_{\equiv}A_{\equiv}^{\bullet}$ extends to $C^{\bullet}O^{\bullet}S_{\equiv}A_{\equiv}^{\bullet}$. $C^{\bullet}O^{\bullet}S_{\equiv}A_{\equiv}^{\bullet}$ remains open.

▸ **Theorem 4.5 (*).** *Given an instance of $C^{\bullet}O^{\bullet}S_{\equiv}A_{\equiv}^{\bullet}$, $C^{\bullet}O^{\bullet}S_{\equiv}A_{\equiv}^{\bullet}$ or $C^{\bullet}O^{\bullet}S_{\equiv}A_{\equiv}^{\bullet}$ together with $k \in \mathbb{R}$ it is (weakly)* NP*-hard to decide if there exists a labeling L with a total leader length of less than k.*

References

1. Pankaj K. Agarwal, Marc van Kreveld, and Subhash Suri. Label placement by maximum independent set in rectangles. *Computational Geometry: Theory and Applications*, 11(3-4):209–218, 1998. doi:10.1016/S0925-7721(98)00028-5.
2. Sarah E. Battersby, John E. Stewart, Ana Lopez-De Fede, Kevin C. Remington, and Kathy Mayfield-Smith. Ring maps for spatial visualization of multivariate epidemiological data. *Journal of Maps*, 7(1):564–572, 2011. doi:10.4113/jom.2011.1182.
3. Michael A Bekos, Benjamin Niedermann, and Martin Nöllenburg. External labeling techniques: A taxonomy and survey. In *Computer Graphics Forum*, volume 38, pages 833–860. Wiley Online Library, 2019.
4. Michael A. Bekos, Benjamin Niedermann, and Martin Nöllenburg. *External Labeling: Fundamental Concepts and Algorithmic Techniques*. Synthesis Lectures on Visualization. Morgan & Claypool Publishers, 2021. doi:10.2200/S01115ED1V01Y202107VIS014.
5. Marc Benkert, Herman J Haverkort, Moritz Kroll, and Martin Nöllenburg. Algorithms for multi-criteria boundary labeling. *Journal of Graph Algorithms and Applications*, 13(3):289–317, 2009.
6. Martin Fink, Jan-Henrik Haunert, André Schulz, Joachim Spoerhase, and Alexander Wolff. Algorithms for labeling focus regions. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2583–2592, 2012.
7. Michael Formann and Frank Wagner. A packing problem with applications to lettering of maps. In *Proc. 7th Symposium on Computational Geometry (SoCG)*, pages 281–288. ACM, 1991. doi:10.1145/109648.109680.
8. Andreas Gemsa, Jan-Henrik Haunert, and Martin Nöllenburg. Multirow boundary-labeling algorithms for panorama images. *ACM Transactions on Spatial Algorithms and Systems (TSAS)*, 1(1):1–30, 2015.
9. Jan-Henrik Haunert and Tobias Hermes. Labeling circular focus regions based on a tractable case of maximum weight independent set of rectangles. In *Proc. 2nd ACM SIGSPATIAL International Workshop on Interacting with Maps (MapInteract@GIS)*, pages 15–21, 2014.
10. Benjamin Niedermann and Jan-Henrik Haunert. Focus+ context map labeling with optimized clutter reduction. *International Journal of Cartography*, 5(2-3):158–177, 2019.
11. Marc van Kreveld, Tycho Strijk, and Alexander Wolff. Point labeling with sliding labels. *Computational Geometry: Theory and Applications*, 13(1):21–47, 1999. doi:10.1016/S0925-7721(99)00005-X.
12. Alexander Wolff. Graph drawing and cartography. In Roberto Tamassia, editor, *Handbook on Graph Drawing and Visualization*, pages 697–736. Chapman and Hall/CRC, 2013.