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.

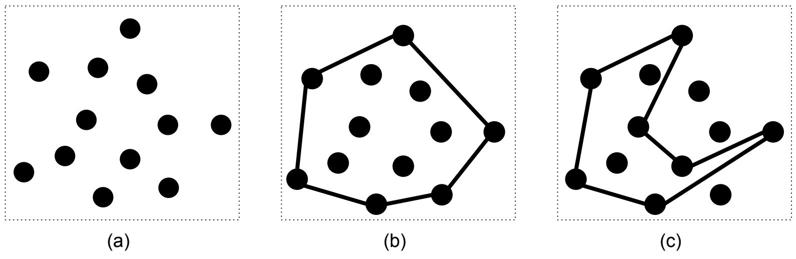

Figure 1 (a) A given point set S and (b), (c) two at most 3-out polygons of S.

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.

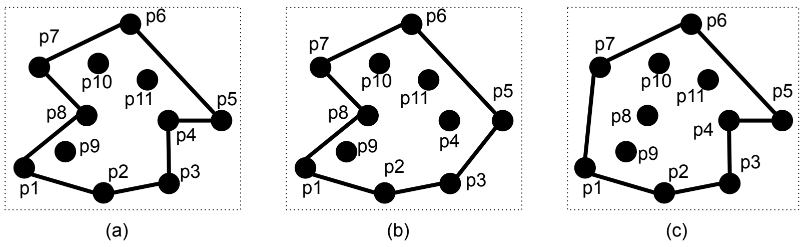

Figure 2 (a) A simple polygon of a point set $S = \{p_1, p_2, \ldots, p_{11}\}$, where p_4 and p_8 are embeddable vertices. (b) The simple polygon of S obtained from the polygon of (a) by embedding p_4. (c) The simple polygon of S obtained from the polygon of (a) by embedding p_8.

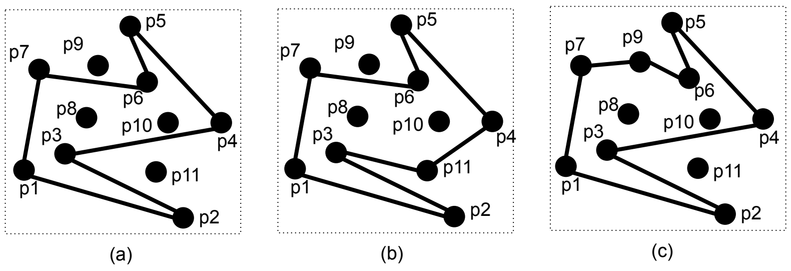

Figure 3 (a) An at most 2-out polygon of a point set with no embeddable vertices and $\mathsf{out}(P) = \{p_9, p_{11}\}$. (b) The polygon obtained from the polygon of (a) by inserting p_{11} to (p_3, p_4). (c) The polygon obtained from the polygon of (a) by inserting p_9 to (p_6, p_7).

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

$$\mathsf{dist}(\mathsf{cloe}(P, p), p) = \min_{q \in \mathsf{iout}(P)} \{\mathsf{dist}(\mathsf{cloe}(P, q), q)\}$$

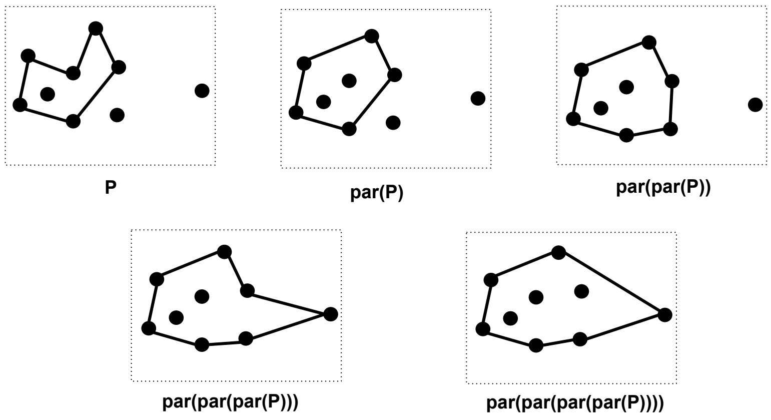

Figure 4 A parent sequence of at most 2-out polygon P.

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

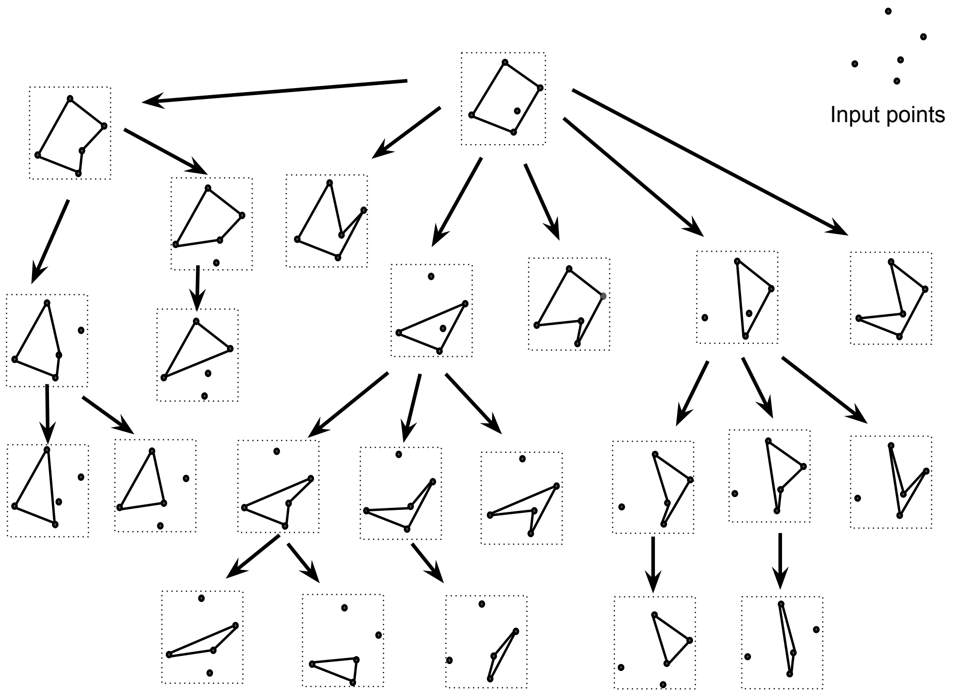

Figure 5 An example of a family tree

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 $0 \le k \le n-3$. One can enumerate all the at most k-out polygons in $\mathcal{S}_{\le k}(S)$ in $\mathcal{O}(n^3 \log n |\mathcal{S}_{\le k}(S)|)$ 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 $0 \le k \le n-3$. One can enumerate all the at most k-out polygons in $\mathcal{S}_{\le k}(S)$ 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 $\mathcal{S}_{\le k}(S)$ 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.