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

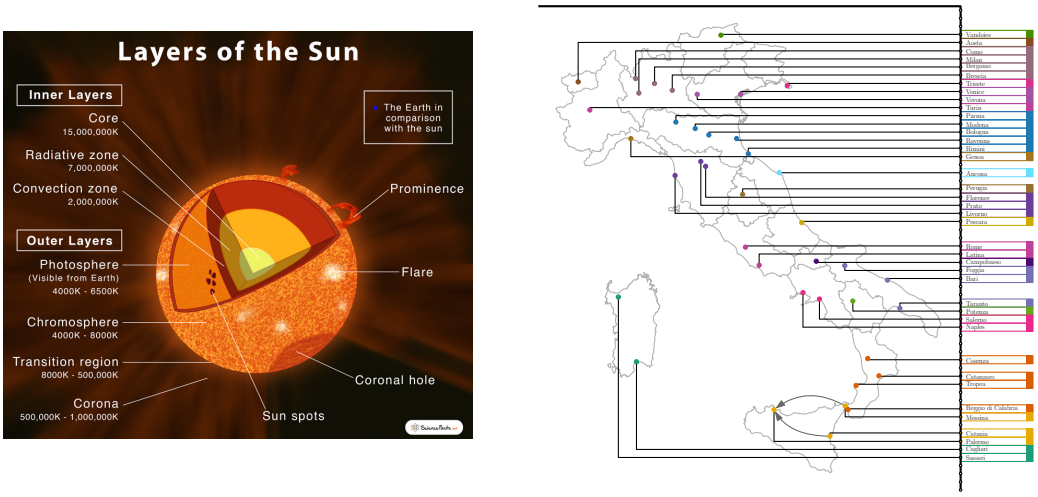
1 Introduction

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.

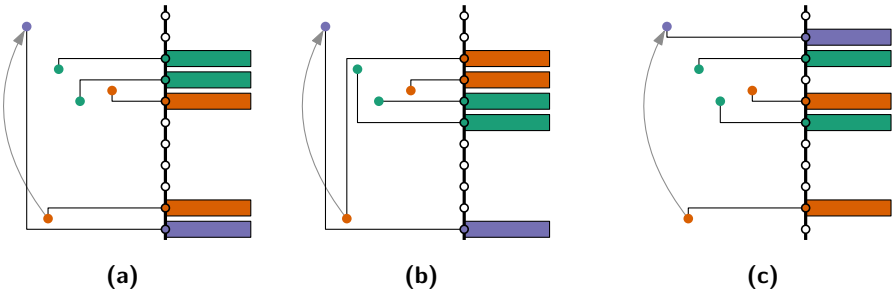
40th European Workshop on Computational Geometry, Ioannina, Greece, March 13–15, 2024.
This is an extended abstract of a presentation given at EuroCG'24. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work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



(a) **(b)**

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



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

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.

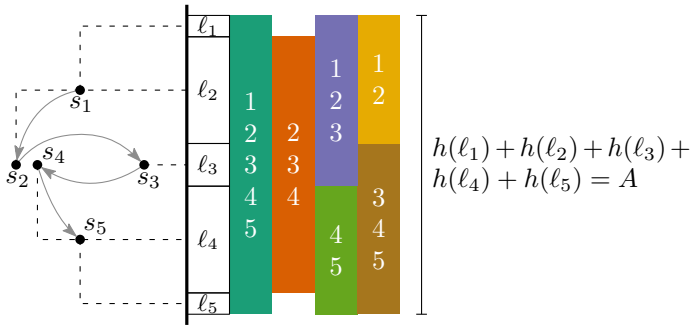


Figure 3 Creating an obstacle using grouping constraints as indicated by colored bands. An alternative way of forcing the same structure with only ordering constraints is shown with the arrows.

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.

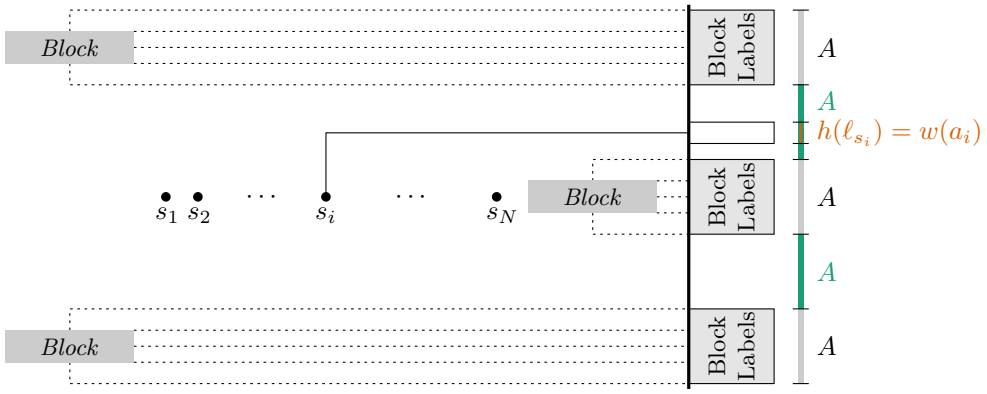


Figure 4 The placement of the sites in the reduction that shows weakly NP-hardness for 1-CBL.

▶ **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].

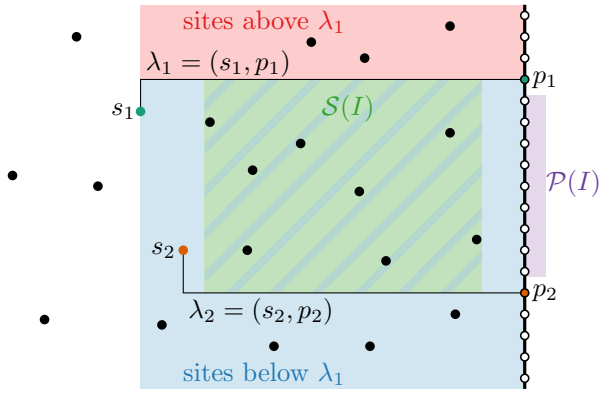


Figure 5 A sub-instance $I = (s_1, p_1, s_2, p_2)$ of our DP-algorithm and the used notation.

▶ **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.

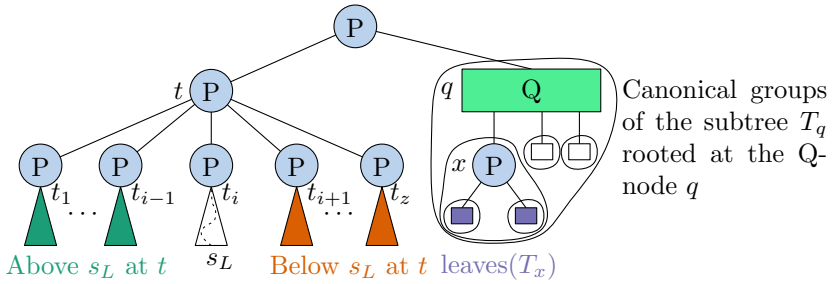


Figure 6 A sample PQ-A-Graph together with the used terminology.

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), ..., 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}), ..., 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.

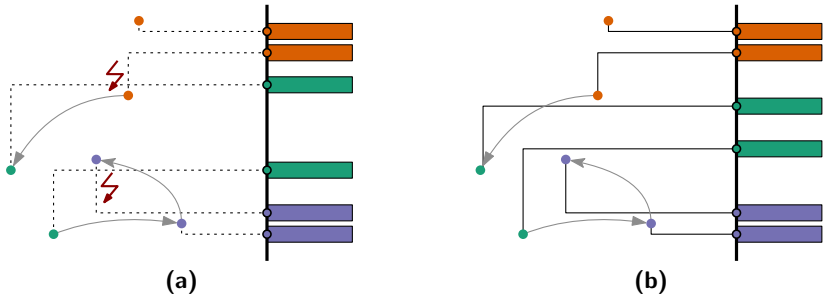


Figure 7 An instance whose admissibility depends on the position of the ports.

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.*

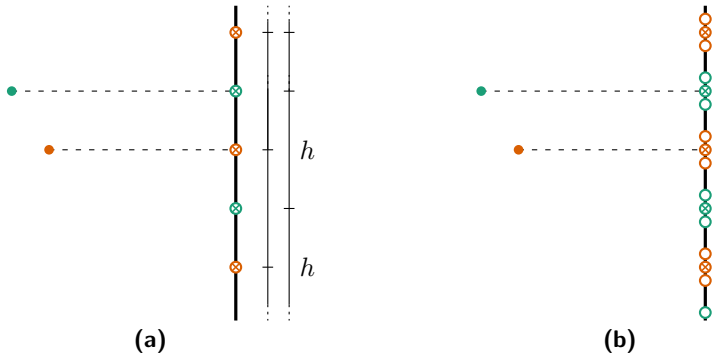


Figure 8 Set of ports **(a)** induced by the sites as in [11] and **(b)** extended to our canonical ports.

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.