Journal of Graph Algorithms and Applications
http://jgaa.info/ vol. 19, no. 3, pp. 779–816 (2015)
DOI: 10.7155/jgaa.00379

Many-to-One Boundary Labeling with Backbones

Michael A. Bekos[1] *Sabine Cornelsen*[2] *Martin Fink*[3]
Seok-Hee Hong[4] *Michael Kaufmann*[1] *Martin Nöllenburg*[5]
Ignaz Rutter[6] *Antonios Symvonis*[7]

[1]Institute for Informatics, University of Tübingen, Germany
[2]Department of Computer and Information Science, University of Konstanz, Germany
[3]Department of Computer Science, UC Santa Barbara, USA
[4]School of Information Technologies, University of Sydney, Australia
[5]Algorithms and Complexity Group, TU Wien, Vienna, Austria
[6]Institute of Theoretical Informatics, KIT, Karlsruhe, Germany
[7]School of Applied Mathematics and Physical Sciences, NTUA, Greece

Abstract

We study a boundary labeling problem, where multiple points may connect to the same label. In this new many-to-one model, a horizontal backbone reaches out of each label into the feature-enclosing rectangle. Feature points that need to be connected to this label are linked via vertical line segments to the backbone. We present dynamic programming algorithms for minimizing the total number of label occurrences and for minimizing the total leader length of crossing-free backbone labelings. When crossings are allowed, we aim at obtaining solutions with the minimum number of crossings. This can be achieved efficiently in the case of fixed label order; however, in the case of flexible label order we show that minimizing the number of leader crossings is NP-hard.

Submitted:	Reviewed:	Revised:	Accepted:	Final:
August 2014	May 2015	October 2015	November 2015	December 2015

Published:
December 2015

Article type:	Communicated by:
Regular paper	H. Meijer

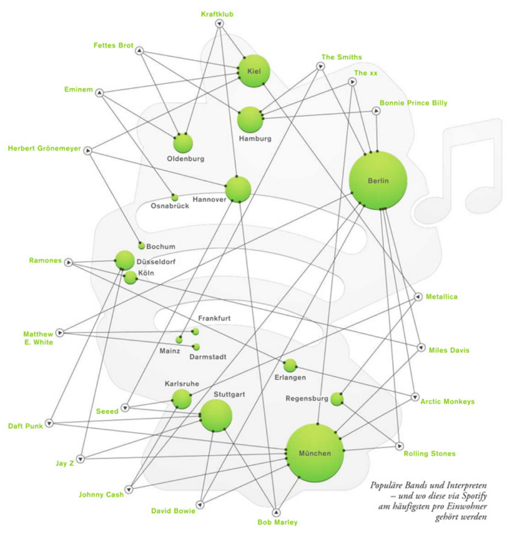
E-mail addresses: bekos@informatik.uni-tuebingen.de (Michael A. Bekos) sabine.cornelsen@uni-konstanz.de (Sabine Cornelsen) fink@cs.ucsb.edu (Martin Fink) shhong@it.usyd.edu.au (Seok-Hee Hong) mk@informatik.uni-tuebingen.de (Michael Kaufmann) noellenburg@ac.tuwien.ac.at (Martin Nöllenburg) rutter@kit.edu (Ignaz Rutter) symvonis@math.ntua.gr (Antonios Symvonis)

1 Introduction

The process of annotating images by placing labels in the image that contain short texts or icons describing specific features of interest is referred to as *labeling*. Typically, a label should not occlude features of the image and it should not overlap with other labels. In map labeling (or *internal labeling*) the goal is usually to place relatively small labels (often consisting of a single word/name) directly on the map so that they are located in the immediate vicinity of the features they describe. In addition to research in cartography and geographic information science (GIS), map labeling has been studied in computer science for more than two decades [7]. A survey on map labeling algorithms and an extensive bibliography are given by Neyer [17] and Wolff and Strijk [19], respectively.

However, internal labeling is not feasible anymore when large labels are employed, a typical situation that arises in technical drawings and medical atlases. Instead, graphic designers often resort to external labels that are placed along the image boundary and connect points and labels by crossing-free arcs referred to as *leaders*. The point, where a leader attaches to its label is called a *label port*. *Boundary labeling* (or *external labeling*) was formally introduced as an algorithmic optimization problem by Bekos et al. [4]. It stimulated a lot of follow-up work, discussed, for example, in Kaufmann's 2009 survey [11]. Initially, most work concentrated on the case that each label is associated with a single feature point (or *site*). However, the case where each label is associated with more than one site (the topic of this paper) is also common in applications, even though algorithmic solutions are rare. For instance, when showing shops or restaurants in a city unknown to a user, the user will probably be more interested in knowing the type of store (or the cuisine of a restaurant) than in its name. By placing just one label for a group of sites of the same type, and connecting all these sites to the label, it is made very clear to the user that these sites are of the same type. An example of a map-based infographics linking point triples to the same label is given in Figure 1a. Similarly, but in more static applications, associating several points to the same label is useful when showing similar components (e.g., types of screws) in assembly instructions for furniture or technical devices. Another well-known example are anatomical drawings that show different types of muscles or bones within the human body. Figure 1b gives an example of the human neck labeled using several many-to-one leaders. In all these cases, we can conceptually think of groups of sites sharing the same label as having the same color. Then, we need to connect these identically colored sites via leaders to a label of the same color.

Different boundary labeling approaches can be distinguished by the leader shapes that are used. Polygonal leaders may consist of a single straight-line segment (denoted as type-s leaders) or a polygonal path with multiple segments. In the latter case, the leader shape may be described by a word over the alphabet $\{o, p, d\}$, where o, p, or d, respectively, denote a segment orthogonal, parallel, or diagonal to the side of the rectangle where the label is placed. Each describing word, for example, *opo* or *do*, encodes the leader shape read from the


(a)

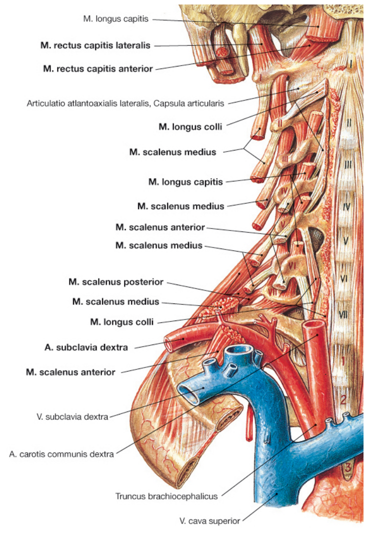

(b)

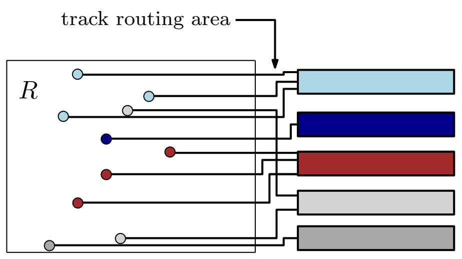
Figure 1: Examples of many-to-one labeling. (a) Infographics relating popular bands on Spotify to German cities with most listeners using many-to-one labeling. Image courtesy of Jörg Block (`www.joergblock.de`). (b) Anatomical drawing of the human neck with several many-to-one leaders. Image source: Waschke, Paulsen, Sobotta Atlas der Anatomie des Menschen, 23rd edition 2010 ©Elsevier GmbH, Urban & Fischer, München.

feature point towards the label. Further, one can distinguish boundary labeling algorithms by the different sides of the image rectangle, where labels are placed and by the objective, for example, minimization of the total leader length or the number of bends. Finally, additional restrictions on the labels may be relevant, such as uniform or non-uniform label sizes or different label port positions.

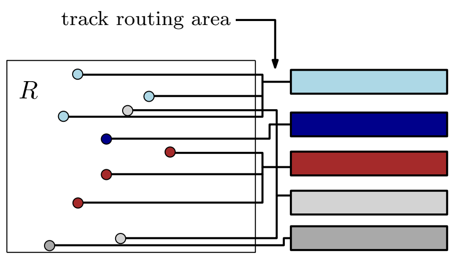
Related Work. Bekos et al. [4] presented efficient labeling algorithms in the one-, two-, and four-sided model using type-s, type-*po* and type-*opo* leaders. They were mainly interested in minimizing the total leader length, but also presented an algorithm for minimizing the number of bends in one-sided *opo*-labeling. Benkert et al. [5] studied algorithms for one- and two-sided *po*- and *do*-labeling with arbitrary leader-dependent cost functions (including total leader length and number of bends). Nöllenburg et al. [18] presented an $O(n \log n)$-time algorithm for one-sided minimum-length *po*-labeling using labels that can slide along the boundary. Huang et al. [10], too, considered the problem of computing boundary labelings with sliding label positions. They gave several polynomial-time algorithms for computing one- and two-sided type-*opo* and type-*po* boundary labelings with minimum leader length or minimum total number of bends. Kindermann et al. [12] studied *po*-labelings, where the labels are placed on two adjacent sides of the image rectangle. In the same paper the authors also extend the algorithm such that it can handle three and four-sided labelings. Bekos et al. [2] presented efficient algorithms for uniform labels and NP-hardness results for non-uniform labels using combinations of more general *octilinear* leaders of types *do*, *od*, and *pd* and labels on one, two, and four rectangle sides. Bekos et al. [3] gave algorithms for label size maximization in a one-sided model with two or three parallel columns of labels on a vertical rectangle side and type-*opo* leaders. Gemsa et al. [8] presented several algorithms for one-sided labeling of wide panorama images using type-*o* leaders and variable-width labels in multiple parallel rows above the image. Recently, Barth et al. [1] conducted a user study on the readability of different leader types, which, based on the evaluation of task accuracy and response time, recommends to use *po*- or *do*-leaders.

Many-to-one boundary labeling was formally introduced by Lin et al. [15]. In their initial definition of many-to-one labeling each label had one port for each connecting site, that is, each point uses an individual leader (see Figure 2a). This inevitably lead to (i) tall labels, (ii) a wide track-routing area between the labels and the enclosing rectangle (since leaders are not allowed to overlap), and (iii) leader crossings in the track routing area. Lin et al. [15] examined one and two-sided boundary labeling using type-*opo* leaders; see Figure 2a. They showed that several crossing minimization problems are NP-complete and, subsequently, developed approximation and heuristic algorithms. In a variant of this model, referred to as *boundary labeling with hyperleaders*, Lin [14] resolved the multiple port issue by joining together all leaders attached to a common label with a vertical line segment in the track-routing area; see Figure 2b. At the cost of label duplications, leader crossings could be eliminated; see Figure 2d. Bruckdorfer et al. [6] recently studied a related set visualization problem for colored points using so-called *buses*, where a bus is a horizontal line segment to which points of the same color (label) are connected by vertical line segments and all line segments must be crossing-free. The main difference to many-to-one boundary labeling is that the buses do not extend to the bounding box of the input point set and thus cannot be used to attach an actual label.

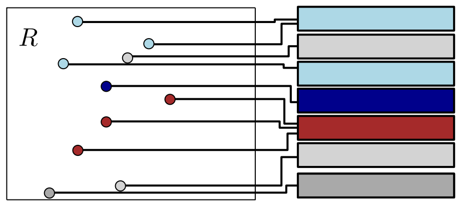
Our Contribution. We study *many-to-one boundary labeling with backbone leaders* (for short, *backbone labeling*). In this new model, a horizontal backbone reaches out of each label into the site-enclosing rectangle. Sites connected to a label are linked via vertical line segments to the label's backbone (see Figure 3a). The backbone model does not need a track routing area and thus overcomes several disadvantages of previous many-to-one labeling models, in particular the issues (ii) and (iii) mentioned above. As Figure 3 shows, backbone labelings also require much less "ink" in the image than the previous methods and thus are expected to be less disturbing for the viewer. Note that crossing-free labelings may make it necessary to use multiple labels for the same color, as we did in Figures 3c and 3d. We note that backbone labeling can be seen as a variation of Lin's *opo*-hyperleaders. Lin [14] posed it as an open problem to study *po*-hyperleaders (which is his terminology for backbones), in particular to minimize



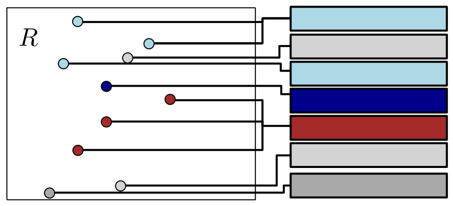
(a) Individual leaders [15].



(b) Hyperleaders [14].

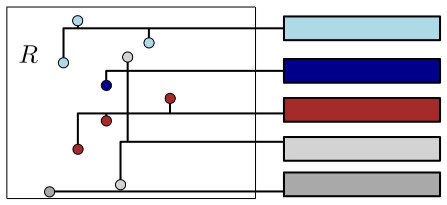


(c) Crossing-free individual leaders.

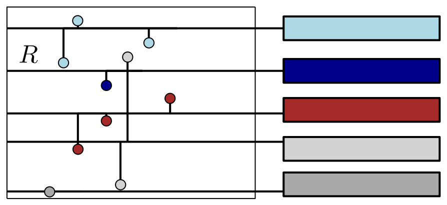


(d) Crossing-free hyperleaders.

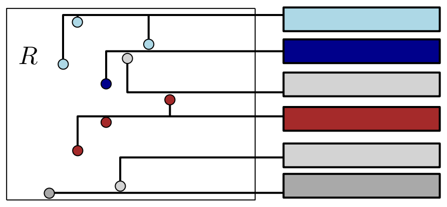
Figure 2: Different types of many-to-one *opo* boundary labelings.



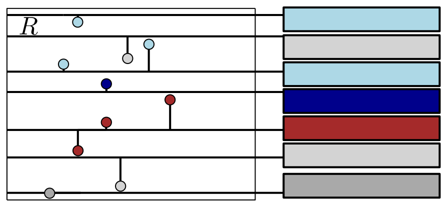
(a) One-sided backbones.



(b) Two-sided backbones.



(c) Crossing-free one-sided backbones.



(d) Crossing-free two-sided backbones.

Figure 3: Different types of many-to-one labelings with backbone leaders.

Table 1: Overview of our results. Runtimes of our algorithms are shown, where n is the number of input points, C is the color set, and α is either the number K of globally allowed labels or the product $\prod_{c \in C} k_c$ of the individual label bounds in the color vector $\vec{k} = (k_1, \ldots, k_{|C|})$, depending on the problem setting.

problem		two-sided backbones		one-sided backbones	
minimize	constraints	result	Thm	result	Thm
backbones		$O(n)$	1	$O(n^4\|C\|^2)$	2
backbones	minimum gap Δ	—		$O(n^7\|C\|^2)$	3
length		$O(n^2)$	5	$O(n^4\|C\|^2)$	6
length	bounded # backbones	$O(n^2\alpha)$	4+5	$O(n^4\|C\|^2\alpha^2)$	7
length	minimum gap Δ	—		$O(n^7\|C\|^2\alpha^2)$	8
crossings	fixed order	$O(n\|C\|)$	9	$O(n\|C\|)$	10
crossings	flex. order, fixed pos.	$O(n + \|C\|^3)$	11	—	
crossings	flex. order	—		NP-complete	12

the number of duplicate labels in a crossing-free labeling.

We study three problem variants for backbone labeling, *label number minimization* (Section 3), *total leader length minimization* (Section 4), and *crossing minimization* (Section 5). The first two variants require crossing-free leaders. We consider both *one-sided backbones* (see Figure 3a) and *two-sided backbones* (see Figure 3b). One-sided backbones extend horizontally from the label to the furthest point connected to the backbone, whereas two-sided backbones span the whole width of the rectangle (thus one could use duplicate labels on both sides). Furthermore, our algorithms vary depending on whether the order of the labels is fixed or flexible and whether more than one label per color class can be used.

For crossing-free backbone labeling we derive efficient algorithms based on dynamic programming to minimize label number and total leader length (Section 3 and 4), which solves the open problem of Lin [14]. The main idea is that backbones can be used to split an instance into two independent subinstances. For two-sided leaders faster algorithms are possible since each backbone generates two independent instances; for one-sided backbones the algorithms require more effort since a backbone does not split the whole point set and thus the outermost point connected to each backbone must be considered. For the case where crossings are allowed, we present an efficient algorithm for crossing minimization with fixed label order and show NP-completeness for flexible label order (Section 5). Table 1 summarizes our results.

2 Problem Definition

In backbone labeling, we are given a set P of n points in an axis-aligned rectangle R. There is a set C of colors (or categories) for the points, and each point $p \in P$ is assigned a color $c(p)$ from the set C. Our goal is to place colored labels on the boundary of R and to assign each point $p \in P$ to a label $l(p)$ of color $c(p)$.

All points assigned to the same label will be connected to the label through a single backbone leader. A *backbone leader* consists of a horizontal *backbone* attached to the left or right side of the enclosing rectangle R and vertical line segments that connect the points to the backbone.

Only a single backbone leader can be attached to a label. Hence, we can use the terms *label* and *backbone* interchangeably. Since the backbones are horizontal, we consider labels to be fully described by the y-coordinate of their backbone. Note that, at first sight, this may imply that labels are of infinitely small height. However, by imposing a minimum separation distance between backbones, we can also accommodate labels of fixed height.

Let $\mathcal{L}$ be a set of colored labels and consider label $l \in \mathcal{L}$. By $c(l)$, $y(l)$, and $P(l)$ we denote the color of label l, the y-coordinate of the backbone of label l on the boundary of R and the set of points that are connected/associated to label l, respectively.

A *backbone (boundary) labeling* for a set of colored points P in a rectangle R is a set $\mathcal{L}$ of colored labels together with a mapping of each point $p \in P$ to some $c(p)$-colored label in $\mathcal{L}$. The drawing can be easily produced since the backbone leader for label l is fully specified by $y(l)$ and $P(l)$. A backbone labeling is called *legal* if and only if (i) each point is connected to a label of the same color, and (ii) there are no backbone leader overlaps (though crossings are allowed in some cases).

Several restrictions on the number of labels of a specific color may be imposed: The number of labels may be unlimited, effectively allowing us to assign each point to a distinct label. Alternatively, the number of labels may be bounded by $K \geq |C|$. If $K = |C|$, all points of the same color have to be assigned to a single label. We may also restrict the maximum number of allowed labels for each color in C separately by specifying a *color vector* $\vec{k} = (k_1, \ldots, k_{|C|})$. A legal backbone labeling that satisfies all of the imposed restrictions on the number of labels is called *feasible*. Our goal in this paper is to find feasible backbone labelings that optimize different quality criteria.

A backbone labeling without leader crossings is called *crossing-free*. An interesting variation of backbone labeling concerns the size of the backbone. A *one-sided backbone* attached to a label at, say, the right side of R extends up to the leftmost point that is assigned to it. A *two-sided backbone* spans the whole width of R; see Figure 3 for examples of both types of backbones. Note that, in the case of crossing-free labelings, two-sided backbones may result in labelings with a larger number of labels and increased total leader length.

From now on we denote the points of P as $\{p_1, p_2, \ldots, p_n\}$ and we assume that no two points share the same x- or y-coordinate. For simplicity, we consider the points to be sorted in decreasing order of y-coordinates, with p_1 being the topmost point in all of our relevant drawings.

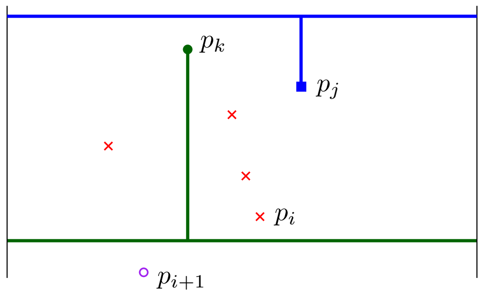


Figure 4: Point p_i cannot be labeled.

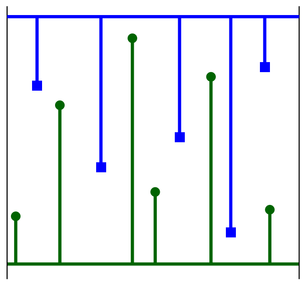


Figure 5: Labeling a 2-colored instance with one backbone per color.

3 Minimizing the Total Number of Labels

In this section we study the problem of finding a crossing-free many-to-one boundary labeling that minimizes the total number of labels. We can, therefore, set $K = n$ so that there is effectively no upper bound on the number of labels.

3.1 Two-sided Backbones

We first investigate the case of two-sided backbones. As, in this setting, a backbone cuts the whole instance into two parts, it is clear that the points enclosed by two consecutive backbones can only have the colors of these backbones. Similarly, we can make an important observation on the structure of crossing-free labelings between two consecutive points.

Lemma 1 *Let p_i and p_{i+1} be two points that are vertically consecutive. Let p_j (with $j < i$) be the first point above p_i with $c(p_j) \neq c(p_i)$, and let $p_{j'}$ (with $j' > i+1$) be the first point below p_{i+1} with $c(p_{j'}) \neq c(p_{i+1})$ if such points exist. In any crossing-free backbone labeling p_i and p_{i+1} are vertically separated by at most two backbones. Furthermore, any separating backbone has color $c(p_i)$, $c(p_{i+1})$, $c(p_j)$, or $c(p_{j'})$.*

Proof: Suppose that there are three separating backbones. Then the middle one could not be connected to any point. Now, suppose that a separating backbone is connected to a point p_k above p_i and has color $c(p_k) \notin \{c(p_j), c(p_i)\}$. Then $k < j < i$. The backbone for p_j has to be above p_k. Hence, point p_i is lying between two backbones of other colors; see Figure 4. Its own backbone cannot be placed there without crossing a vertical segment connecting p_k or p_j to their corresponding backbones. Symmetrically, we see that a backbone separating p_i and p_{i+1} that is connected to a point below p_{i+1} can only have color $c(p_{i+1})$ or $c(p_{j'})$. □

Clearly, if all points have the same color, one label always suffices. Even in an instance with two colors, one label per color is enough: We place the backbone of one color above all points, and the backbone of the second color below all points; see Figure 5. However, if a third color is involved, then many labels may be required.

We denote the number of labels of an optimal crossing-free solution of P by $NL(P)$. In the general case of the problem, P may contain several consecutive points of the same color. We proceed by constructing a simplified instance $C(P)$ based on the instance P; in $C(P)$, there are no two consecutive points of the same color. To do so, we identify each maximal set of identically-colored consecutive points of P and represent it by the topmost point of the set; all other points of the set are removed from the simplified instance. Note that in order to achieve this, a simple top-to-bottom sweep is enough. Let $C(P) = \{p'_1, p'_2, \ldots, p'_k\}$ be the *clustered point set*, which we just constructed. For the sake of simplicity, we assume that $f\colon P \to C(P)$ is a function that maps each point of P to its representative point in the simplified instance $C(P)$.

Lemma 2 *The number of labels needed in an optimal crossing-free labeling of P with two-sided backbones is equal to the number of labels needed in an optimal crossing-free solution of $C(P)$, that is, $NL(P) = NL(C(P))$.*

Proof: Since $C(P) \subseteq P$, it trivially follows that $NL(C(P)) \leq NL(P)$. So, in order to complete the proof it remains to show that $NL(P) \leq NL(C(P))$. Let $S(C(P))$ be an optimal solution of $C(P)$ with $NL(C(P))$ labels. If we manage to construct a solution of P that has exactly the same number of labels as $S(C(P))$, then obviously $NL(P) \leq NL(C(P))$.

Let p'_i with $1 \leq i \leq k$, be an arbitrary point in $C(P)$ and let $f^{-1}(p'_i) = \{p_j, p_{j+1}, \ldots, p_{j+m}\}$ be the maximal set of consecutive, identically-colored points of P that have p'_i as their representative in $C(P)$. Let $H(p'_i)$ be the horizontal strip that is defined by the two horizontal lines through p_j and p_{j+m}, respectively. Any two consecutive strips $H(p'_i)$ and $H(p'_{i+1})$ are separated by an empty strip of positive height, which contains no point of P. We argue that we can modify $S(C(P))$ by shifting all backbones into these empty strips such that no strip $H(p'_i)$ for any $p'_i \in C(P)$ contains a backbone and the labeling remains legal and crossing-free. Let $H(p'_i)$ be a horizontal strip that contains a backbone. By Lemma 1 there can be at most two such backbones. Since $C(P)$ does not contain any point between p'_i and p'_{i+1}, shifting these backbones (and keeping their relative order) into the empty strip between p_{j+m} and p'_{i+1} cannot create any backbone crossings. If we perform this step for all strips $H(p')$ with $p' \in C(P)$ the resulting solution $S'(C(P))$ is again an optimal solution for $C(P)$ and no cluster of consecutive equally colored points in P is separated by a backbone of $S'(C(P))$. Now observe that each point $p \in P$ can be attached by a crossing-free vertical line segment to the backbone of $f(p)$ in $S'(C(P))$. This yields a crossing-free legal solution for P and thus $NL(P) \leq NL(C(P))$. $\square$

With the help of the previous lemmas, we are ready to present a linear-time algorithm for minimizing the number of two-sided backbones.

Theorem 1 *Let $P = \{p_1, p_2, \ldots, p_n\}$ be an input point set consisting of n points sorted from top to bottom. Then, a crossing-free labeling of P with the minimum number of two-sided backbones can be computed in $O(n)$ time.*

Proof: In order to simplify the proof, we assume that no two consecutive points have the same color, with the help of Lemma 2. If this is not already the case, we can first replace P by the simplified instance $C(P)$. After finding a solution for the simplified instance, we can transform this solution into a solution for P as we did in the proof of Lemma 2. Note that both transformations can be done in $O(n)$ time.

We will use dynamic programming on simplified instances. For $i = 1, 2, \ldots, n$, $c_{\text{bak}}, c_{\text{free}} \in C \cup \{\emptyset\}$, and $\text{cur} \in \{\texttt{true}, \texttt{false}\}$, let $nl[i, \text{cur}, c_{\text{bak}}, c_{\text{free}}]$ be the optimal number of backbones above or at p_i in the case where:

- The lowest backbone has color c_{bak}.
- If $\text{cur} = \texttt{true}$, the lowest backbone coincides with p_i; hence, it is $c(p_i)$-colored and $c_{\text{bak}} = c(p_i)$. Otherwise, if $\text{cur} = \texttt{false}$, the lowest backbone is above p_i. Note that in the latter case p_i might be unlabeled (for instance if the color of the lowest backbone is not $c(p_i)$, that is, $c_{\text{bak}} \neq c(p_i)$). We use the notation $c_{\text{bak}} = \emptyset$ for the case that no backbone is placed above or at point p_i.
- In general, we allow unlabeled points between the lowest backbone and point p_i (or, above p_i, if no backbone exists above or at p_i). However, we know that between two consecutive backbones any point must have the color of one of the two backbones. Therefore, we only allow one additional color c_{free} for unlabeled points between the lowest backbone and p_i, that is, any such unlabeled point has color c_{bak} or color c_{free}. Obviously, in the case where $\text{cur} = \texttt{true}$ (that is, the lowest backbone coincides with p_i) such an unlabeled point does not exist. So, in general, if no unlabeled point above p_i exists, we use the notation $c_{\text{free}} = \emptyset$.

Obviously, $nl[1, \texttt{true}, c(p_1), \emptyset] = 1$ for the case that a backbone is placed through p_1 and $nl[1, \texttt{false}, \emptyset, c(p_1)] = 0$ for the case that the lowest backbone is below p_1. Now assume that we have computed all entries of table nl that correspond to different labelings induced by the point p_i. In order to compute the corresponding table entries for the next point p_{i+1}, we distinguish two cases:

1. *The lowest backbone coincides with p_{i+1}:* In this case, the lowest backbone should be $c(p_{i+1})$-colored, $\text{cur} = \texttt{true}$, and obviously there is no unlabeled point between the backbone through p_{i+1} and the point p_{i+1}, that is, $c_{\text{free}} = \emptyset$. Hence, we need to compute entry $nl[i+1, \texttt{true}, c(p_{i+1}), \emptyset]$. To do so, we distinguish the following subcases with respect to the color of the lowest backbone b above or at point p_i.

   1.1 *b is above or at point p_i and $c(p_i)$-colored.* If b is at point p_i (see Figure 6a), then trivially there is no unlabeled point below it. Hence, a feasible solution can be derived from $nl[i, \texttt{true}, c(p_i), \emptyset]$ by adding a new backbone, namely the one incident to p_{i+1}.

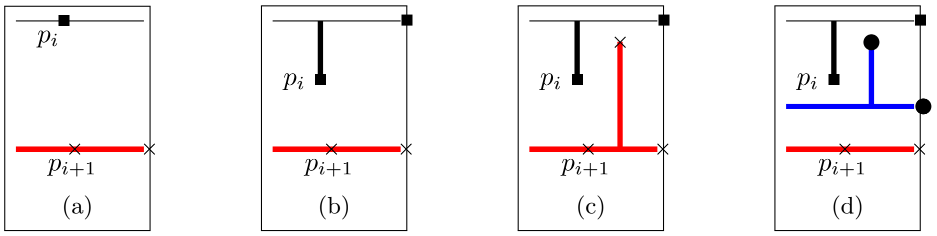


Figure 6: Different configurations that arise in case 1.1. Bold backbone segments are newly added in this step.

If b is above point p_i, then we distinguish two subcases.

(a) If there is no unlabeled point below b (see Figure 6b), or each unlabeled point has color $c(p_i)$, then a feasible solution can, again, be derived from $nl[i, \texttt{false}, c(p_i), \emptyset]$ by adding a new backbone, namely the one incident to p_{i+1}.

(b) On the other hand, if there are unlabeled points of color $\neq c(p_i)$ below b, then we need to distinguish two subcases based on the color of these points.

(b.1) If the unlabeled points are colored $c(p_{i+1})$ (see Figure 6c), then a single additional backbone incident to p_{i+1} suffices. The corresponding solution can be derived using the entry $nl[i, \texttt{false}, c(p_i), c(p_{i+1})]$.

(b.2) However, in the case where unlabeled points are c-colored and $c \notin \{c(p_i), c(p_{i+1})\}$ (see Figure 6d), two backbones are required and the corresponding feasible solution is derived from $nl[i, \texttt{false}, c(p_i), c]$ with $c \notin \{c(p_i), c(p_{i+1})\}$. Note that the case where all unlabeled points below b are of color $c(p_i)$ cannot occur, since we have assumed that consecutive points are not of the same color.

1.2 *b is above p_i and $c(p_{i+1})$-colored.* Again, we distinguish two subcases.

(a) If there is no unlabeled point below b (see Figure 7a), then a feasible solution can be derived from $nl[i, \texttt{false}, c(p_{i+1}), \emptyset]$ by adding two new backbones, that is, the one incident to p_i and the one incident to p_{i+1}.

(b) If there are unlabeled points below b (see Figure 7b), then they may only have colors $c(p_{i+1})$ and $c(p_i)$. (Otherwise, a point of different color cannot connect to any backbone.) Again two backbones are required, that is, the one incident to p_i and the one incident to p_{i+1}. The corresponding solution is derived from $nl[i, \texttt{false}, c(p_{i+1}), c(p_i)]$.

1.3 *b is above p_i and c-colored, where $c \neq c(p_i)$ and $c \neq c(p_{i+1})$.* In this case, either there is no unlabeled point below b (see Figure 7c) or all

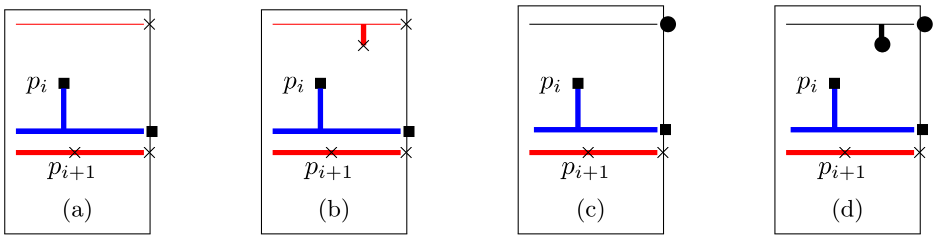


Figure 7: Different configurations that arise in cases 1.2 and 1.3.

such points have color c or $c(p_i)$ (see Figure 7d). In both cases, two backbones have to be placed: one incident to p_i and one incident to p_{i+1}. In the former case, the corresponding feasible solution is derived from $nl[i, \texttt{false}, c, \emptyset]$ with $c \notin \{c(p_i), c(p_{i+1})\}$, while in the latter it is derived from $nl[i, \texttt{false}, c, c(p_i)]$ with, $c \notin \{c(p_i), c(p_{i+1})\}$.

From the above cases, it follows:

$$nl[i+1, \texttt{true}, c(p_{i+1}), \emptyset] = \min \begin{cases} nl[i, \texttt{true}, c(p_i), \emptyset] + 1 \\ nl[i, \texttt{false}, c(p_i), \emptyset] + 1 \\ nl[i, \texttt{false}, c(p_i), c(p_{i+1})] + 1 \\ nl[i, \texttt{false}, c(p_i), c] + 2, \ \ c \notin \{c(p_i), c(p_{i+1})\} \\ nl[i, \texttt{false}, c(p_{i+1}), \emptyset] + 2 \\ nl[i, \texttt{false}, c(p_{i+1}), c(p_i)] + 2 \\ nl[i, \texttt{false}, c, \emptyset] + 2, \ \ c \notin \{c(p_i), c(p_{i+1})\} \\ nl[i, \texttt{false}, c, c(p_i)] + 2, \ \ c \notin \{c(p_i), c(p_{i+1})\} \end{cases}$$

2. *The lowest backbone is above p_{i+1}:* Again, we distinguish subcases with respect to the color of the lowest backbone b above or at point p_i:

2.1 *b is above or at point p_i and $c(p_i)$-colored.* If b is at point p_i (see Figure 8a) or b is above point p_i and either there is no unlabeled point below b (see Figure 8b) or all unlabeled points below b have color $c(p_{i+1})$ or $c(p_i)$ (see Figure 8c), then no additional backbone is required. Then, the corresponding feasible solutions are as follows:

$$nl[i+1, \texttt{false}, c(p_i), \emptyset] = \min\{nl[i, \texttt{true}, c(p_i), \emptyset], \\ nl[i, \texttt{false}, c(p_i), \emptyset]\}$$

$$nl[i+1, \texttt{false}, c(p_i), c(p_{i+1})] = nl[i, \texttt{false}, c(p_i), c(p_{i+1})]$$

However, in the case where there is an unlabeled point below b which is c-colored with $c \notin \{c(p_i), c(p_{i+1})\}$, a new backbone is required (see Figure 8d). Hence, the corresponding feasible solution can be derived

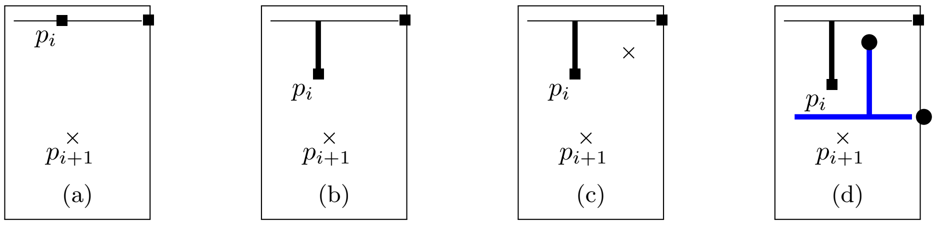


Figure 8: Different configurations that arise in case 2.1.

as

$$nl[i+1, \texttt{false}, c, \emptyset] = nl[i, \texttt{false}, c(p_i), c] + 1 \text{ for } c \notin \{c(p_i), c(p_{i+1})\}.$$

2.2 *b is above p_i and $c(p_{i+1})$-colored.* In this case, either there is no unlabeled point below b (see Figure 9a) or all such unlabeled points have color $c(p_{i+1})$ or $c(p_i)$ (see Figure 9b). In both cases no backbone is required. Hence, the corresponding feasible solutions can be derived as follows:

$$nl[i+1, \texttt{false}, c(p_{i+1}), c(p_i)] = nl[i, \texttt{false}, c(p_{i+1}), \emptyset]$$

$$nl[i+1, \texttt{false}, c(p_{i+1}), c(p_i)] = nl[i, \texttt{false}, c(p_{i+1}), c(p_i)]$$

2.3 *b is above p_i and c-colored, where $c \neq c(p_i)$ and $c \neq c(p_{i+1})$.* In this case, if there is no unlabeled point below b (see Figure 9c) or all such points have color c or $c(p_i)$ (see Figure 9d), then one backbone is required for p_i. The corresponding feasible solution can be derived as follows:

$$\begin{aligned} &nl[i+1, \texttt{false}, c(p_i), \emptyset] \\ &= \min\{nl[i, \texttt{false}, c, \emptyset] + 1, nl[i, \texttt{false}, c, c(p_i)]\} + 1 \\ &\qquad \text{with } c \notin \{c(p_i), c(p_{i+1})\} \end{aligned}$$

Finally, we consider the case that a forth color is involved, say $c' \notin \{c(p_i), c(p_{i+1}), c\}$. In this case, either the c'-colored points are labeled and p_i remains unlabeled (see Figure 9e), or the c'-colored points remain unlabeled and p_i is labeled (see Figure 9f). Note that a feasible solution to the latter case only exists if the unlabeled points of colors c and c' are vertically separated. However, this is not a problem: In both cases, the additional backbone is placed above point p_i, a contradiction to the choice of backbone b as the lowest backbone above p_i. Hence, we do not have to take these cases into account.

Having computed table nl, the number of labels of the optimal solution of P equals the minimum entry of the form $nl[n, \texttt{false}, \cdot, \emptyset]$. Since the algorithm maintains an $n \times 2 \times |C| \times |C|$ table and each entry is computed in constant time,

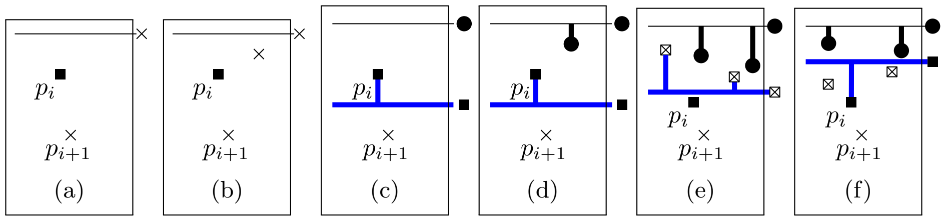


Figure 9: Different configurations that arise in cases 2.2 and 2.3.

the time complexity of our algorithm is $O(n|C|^2)$. However, it can be reduced to $O(n)$. In all computations for the cases that we distinguished, there was at most one color c with $c \neq c(p_i)$ and $c \neq c(p_{i+1})$ involved. By Lemma 1, we know that there are only two possibilities for color c: It could be the first or the second color different from $c(p_i)$ that appears above p_i. Hence, for any point p_{i+1}, only a constant number of colors have to be considered for computing all relevant entries $nl[i+1, \cdot, \cdot, \cdot]$. By a simple sweep from top to bottom over the points, in which we keep track of the last four different colors of points, we can determine all relevant combinations of entries in linear time. Afterwards, each entry of the table can be computed in constant time. A solution with the minimum number of labels can be found by backtracking in the dynamic program. $\square$

3.2 One-sided Backbones

We now consider minimizing the total number of labels for one-sided backbones. First, note that we can always slightly shift the backbones in a given solution so that backbones are placed only in gaps between points. We number the gaps from 0 to n where gap 0 is above point p_1, gap n is below point p_n, and gap i is between point p_i and point p_{i+1} for $1 \leq i < n$.

Suppose that a point p_ℓ lies between a backbone of color c in gap g and a backbone of color c' in gap g' with $0 \leq g < \ell \leq g' \leq n$ such that both backbones horizontally extend to at least the x-coordinate of p_ℓ; see Figure 10. Suppose that all points except the ones in the rectangle $R(g, g', \ell)$, spanned by the gaps g and g' and limited by p_ℓ to the left and by the boundary to the right, are already labeled. An optimum solution for connecting the points in $R(g, g', \ell)$ cannot reuse any backbone except for the two backbones in gaps g and g'; hence, such a partial solution is independent of the rest of the solution.

We use this observation for minimizing the number of backbones by a dynamic program. For $0 \leq g \leq g' \leq n$, $\ell \in \{g+1, \ldots, g'\} \cup \{\emptyset\}$, and two colors c and c' let $T[g, c, g', c', \ell]$ be the minimum number of additional labels that are needed for labeling all points in the rectangle $R(g, g', \ell)$ under the assumption that there is a backbone of color c in gap g, a backbone of color c' in gap g', between these two backbones there is no backbone placed yet, and both backbones extend to the left of p_ℓ as in Figure 10. Note that for $\ell = \emptyset$ the rectangle is empty and $T[g, c, g', c', \emptyset] = 0$. Furthermore, also the case $g' = g$ can occur; in this case, as there is no point inside a gap, the relevant entry of table T is

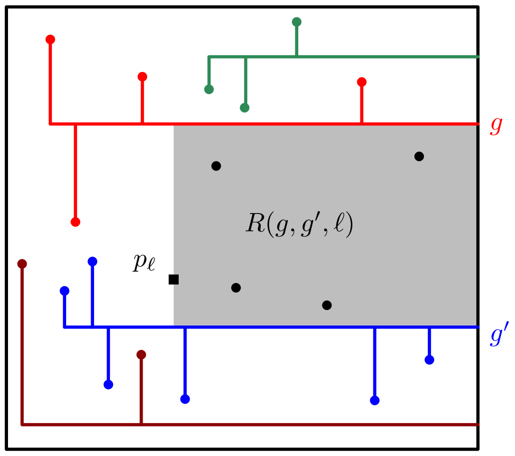


Figure 10: A partial instance in the rectangle $R(g, g', \ell)$ bounded by the two backbones in gaps g and g' and the leftmost point p_ℓ.

$T[g, c, g, c', \emptyset] = 0$.

We distinguish cases based on the connection of point p_ℓ. First, if $c(p_\ell) = c$ or $c(p_\ell) = c'$, it is always optimal to connect p_ℓ to the top or bottom backbone, respectively, as all remaining points will be to the right of the new vertical segment. Hence, in this case,

$$T[g, c, g', c', \ell] = T[g, c, g', c', \mathit{left}(g, g', \ell)],$$

where $\mathit{left}(g, g', \ell)$ is the index of the leftmost point in the interior of $R(g, g', \ell)$ or $\mathit{left}(g, g', \ell) = \emptyset$ if no such point exists.

Otherwise, suppose that $c(p_\ell) \notin \{c, c'\}$. For connecting p_ℓ, we need to place a new backbone of color $c(p_\ell)$; this is possible in any gap $\tilde{g}$ with $g \leq \tilde{g} \leq g'$. Note that reusing gap g or g' is allowed. The backbone splits the instance into two parts, one between gaps g and $\tilde{g}$ and one between gaps $\tilde{g}$ and g'; see Figure 11. Hence, we obtain the recursion

$$\begin{aligned} T[g, c, g', c', \ell] = \min_{g \leq \tilde{g} \leq g'} \Big(& T[g, c, \tilde{g}, c(p_\ell), \mathit{left}(g, \tilde{g}, \ell)] \\ & + T[\tilde{g}, c(p_\ell), g', c', \mathit{left}(\tilde{g}, g', \ell)] \Big) + 1. \end{aligned}$$

Finally, let $\bar{c} \notin C$ be a dummy color, and let $p_{\bar{\ell}} \in P$ be the leftmost point. Then the value $T[0, \bar{c}, n, \bar{c}, \bar{\ell}]$ obtained using dummy backbones above and below all points yields the minimum number of labels needed for labeling all points. We can compute each of the $(n+1) \times |C| \times (n+1) \times |C| \times (n+1)$ entries of table T in $O(n)$ time. Note that all $\mathit{left}(\cdot, \cdot, \cdot)$-values can easily be precomputed in $O(n^3)$ total time by first sorting the points from left to right and then, for each pair of gaps g and g' with $g < g'$, sweeping once over the points $\{p_{g+1}, \dots, p_{g'}\}$ in this direction. Summing up, we get the following result.

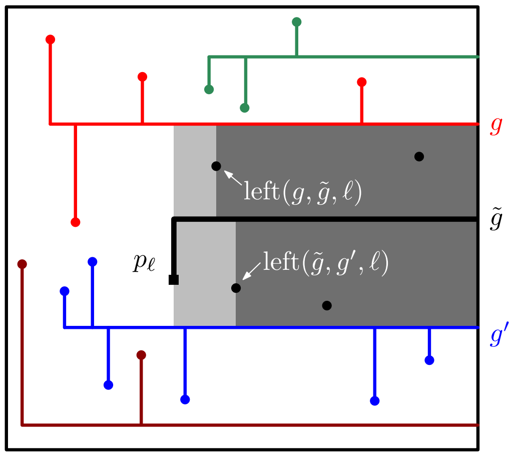


Figure 11: The partial instance in the rectangle $R(g, g', \ell)$ is split by a new backbone for p_ℓ into two partial instances in rectangles $R(g, \tilde{g}, \mathit{left}(g, \tilde{g}, \ell))$ and $R(\tilde{g}, g', \mathit{left}(\tilde{g}, g', \ell))$.

Theorem 2 *Given a set P of n colored points and a color set C, we can compute a feasible labeling of P with the minimum number of one-sided backbones in $O(n^4|C|^2)$ time.*

Minimum Distances. Our algorithm might place many labels inside a gap, which can result in a solution with very small distances between backbones. In practice, we may want to ensure a minimum distance of Δ between backbones, and between a backbone and a point not connected to this backbone. To this end, in any gap, we insert as many candidate positions for backbones as possible for the chosen Δ and the height of that gap (but no more than n). Now, instead of using gaps in table T, we use these candidate positions; a position must never be used twice. The number of entries of the table increases by a factor of $O(n^2)$, since we now have $O(n^2)$ instead of $n+1$ candidate positions for each of the gaps g and g' (but not for ℓ); hence, the table now has $O(n^5|C|^2)$ entries. We need $O(n^2)$ time for computing an entry because there are $O(n^2)$ possible choices for the gap $\tilde{g}$. Therefore, the total running time increases to $O(n^7|C|^2)$.

Theorem 3 *Given a set P of n colored points, a color set C and a minimum distance $\Delta > 0$, we can compute a feasible labeling of P with the minimum number of one-sided backbones of minimum pairwise distance Δ in $O(n^7|C|^2)$ time.*

4 Length Minimization

In the previous section, we have presented algorithms for finding backbone labelings with the minimum number of labels. However, even two labelings with the same number of labels can look quite different. For making the leaders easy to follow, it is important that they have small length. Hence, the objective is that the total length of leader segments is minimum. In order to avoid that the number of labels is very large in solutions with this new objective, we allow to specify an upper bound for the number of labels as part of the input.

In this section we minimize the total length of all leaders in a crossing-free solution, either including or excluding the horizontal lengths of the backbones. We distinguish between a global bound K on the number of labels or a vector $\vec{k}$ of individual bounds per color.

4.1 Two-sided Backbones

First, we consider labelings with two-sided backbones. We use a parameter λ to weight the number of backbones and the vertical leader length in the objective function. The parameter λ allows a fine-grained control over the problem; in particular, setting $\lambda = 0$ minimizes the lengths of the vertical segments, setting λ to the width of the rectangle R minimizes the total leader length and setting λ to n times the height of R minimizes the number of leaders.

Single Color. As a first simple case, we assume that all points have the same color. In this case, we have to choose a set S of at most K y-coordinates where we draw the backbones and connect each point to its nearest backbone; this does, of course, not lead to crossings. Hence, we must solve the following problem: Given n points with y-coordinates $y(p_1) > \cdots > y(p_n)$, find a set S of at most K y-coordinates that minimizes

$$\lambda \cdot |S| + \sum_{i=1}^{n} \min_{y \in S} |y - p_i|. \tag{1}$$

We can optimize the value in Equation (1) by choosing $S \subseteq \{y(p_1), \ldots, y(p_n)\}$, that is, by selecting only backbones that pass through input points: For a backbone position $y \in S \setminus \{y(p_1), \ldots, y(p_n)\}$ let $\{p_i, \ldots, p_j\}$ be the set of points that we would connect to the backbone through y. Let $y(p_i) > \cdots > y(p_{i'}) > y > y(p_{i'+1}) > \cdots > y(p_j)$. If $i' - i + 1 \geq j - i'$, that is, if the majority of sites connected to the backbone at position y lie above the backbone, replace y by $y(p_{i'})$. Otherwise replace y by $y(p_{i'+1})$. Then the objective value in Equation (1) cannot get worse. Hence, the problem can be solved in $O(Kn)$ time if the points are sorted according to their y-coordinates using the algorithm of Hassin and Tamir [9]. Note that the problem corresponds to the 1-dimensional K-median problem if $\lambda = 0$.

Multiple Colors. If the n points have different colors, we can no longer assume that all backbones go through one of the given n points since we have to avoid crossings. However, by Lemma 1, it suffices to add between any pair of vertically consecutive points two additional candidates for backbone positions,

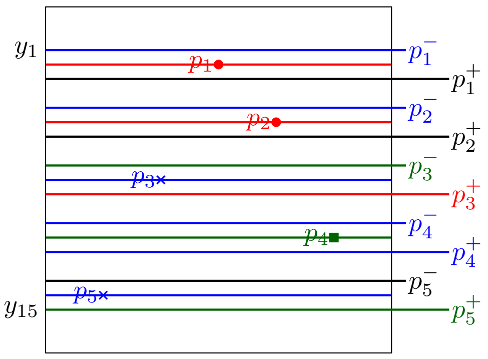


Figure 12: Candidates for five points. Red points are circles, blue points are crosses, and the green point is a square. Candidates through a point have the same color as the point. Candidate $p_i^\pm$ has the same color as the first point with a different color as p_i that is met when walking from $p_i^\pm$ over p_i. Candidates p_1^+, p_2^+, and p_5^- will not be used and have no color.

plus one additional candidate above all points and one below all points. Hence, we have a set of $3n$ candidate lines at y-coordinates

$$p_1^- > y(p_1) > p_1^+ > p_2^- > y(p_2) > p_2^+ > \cdots > p_n^- > y(p_n) > p_n^+, \qquad (2)$$

where for each i the values p_i^- and p_i^+ are as close to $y(p_i)$ as the label heights allow. Clearly, a backbone through p_i can only be connected to points with color $c(p_i)$. If we use a backbone through p_i^- (or p_i^+, respectively), it will have the same color as the first point below p_i (or above p_i, respectively) that has a different color than p_i; compare Lemma 1. For example, in Figure 12, p_1^- is colored blue since p_3 is the first point below p_1 that has a different color than red, namely blue. Hence, the colors of all candidates are fixed or the candidate will never be used as a backbone. For an easier notation, we denote the y-coordinate of the ith point in Equation (2) by y_i and its color by $c(y_i)$.

We minimize the total leader length by dynamic programming. For each $i = 1, \ldots, 3n$, and for each vector $\vec{k}' = (k'_1, \ldots, k'_{|C|})$ with $k'_1 \leq k_1, \ldots, k'_{|C|} \leq k_{|C|}$, let $L[i, \vec{k}']$ denote the minimum length of a feasible backbone labeling of $p_1, \ldots, p_{\lfloor \frac{i+1}{3} \rfloor}$ using k'_c two-sided backbones of color c for $c = 1, \ldots, |C|$ such that the bottommost backbone is at position y_i, if such a labeling exists. Otherwise $L[i, \vec{k}'] = \infty$. In the following, we describe, how to compute the values $L[i, \vec{k}']$.

Assume that we want to place a new backbone at position y_i and that the previous backbone was at position y_j with $j < i$. Then, we have to connect each point p_x with $(j+2)/3 \leq x \leq i/3$ to one of the backbones through y_i or y_j as these points are enclosed between the two backbones. Let $\mathit{link}(j, i)$ denote the minimum total length of the vertical segments linking these points to their respective backbone. We set $\mathit{link}(j, i) = \infty$ if there is a point p_x between y_i and y_j with $c(p_x) \notin \{c(y_i), c(y_j)\}$ because p_x cannot be connected to the surrounding backbones. Otherwise, we have

$$link(j,i) = \begin{cases} \sum\limits_{\frac{j+2}{3} \le x \le \frac{i}{3}} \min\big(y_j - y(p_x), y(p_x) - y_i\big) & \text{if } c(y_i) = c(y_j) \\ \sum\limits_{\substack{\frac{j+2}{3} \le x \le \frac{i}{3} \\ c(p_x) = c(y_j)}} (y_j - y(p_x)) + \sum\limits_{\substack{\frac{j+2}{3} \le x \le \frac{i}{3} \\ c(p_x) = c(y_i)}} (y(p_x) - y_i) & \text{if } c(y_i) \ne c(y_j) \end{cases} \tag{3}$$

The base cases are $L[i, \vec{0}] = \infty$,

$$L[i, (0, \ldots, 0, k'_{c(y_i)} = 1, 0, \ldots, 0)] = \lambda + \sum_{0 < x \le i/3} (y_i - p_x)$$

if all points above y_i have color $c(y_i)$ and $L[i, (0, \ldots, 0, k'_{c(y_i)} = 1, 0, \ldots, 0)] = \infty$ otherwise.

For computing an entry $L[i, (k'_1, \ldots, k'_{|C|})]$ we test all candidate positions $y_j > y_i$ for the previous backbone; to the length of the corresponding solution we have to add the connection cost $link(j,i)$ as well as λ for the new backbone at position y_i. Hence, we get the following recursion:

$$L[i, (k'_1, \ldots, k'_{|C|})] = \lambda + \min_{j<i} \Big(L[j, (k'_1, \ldots, k'_{c(y_i)} - 1, \ldots, k'_{|C|})] + link(j,i) \Big) \tag{4}$$

Note that we need to interpret any entry of table L for which a color bound is negative as ∞.

In order to see that each entry of table L can be computed in $O(n)$ time, we have to show, that, for a fixed index i, all values $link(j,i)$ with $j < i$ can be computed in $O(n)$ time. Let c' be the first color of a point above y_i that is different from $c(y_i)$. For a fixed i, starting from $j = i - 1$, we scan the candidates twice in decreasing order of their indices until we find the first point that is neither colored c' nor $c(y_i)$.

For color $c \in \{c(y_i), c'\}$, we traverse the points above y_i from bottom to top. For any point $p_{x'}$ that we see, we store two values: the number $n_c(x')$ of points of color c that we have seen so far and the sum of distances of these c-colored points to y_i, that is,

$$l_c(x') = \sum_{x' \le x \le \frac{i}{3}, c(p_x) = c} (y(p_x) - y_i).$$

Note that we can easily compute $l_c(x'-1)$ in constant time from $l_c(x')$: If $p_{x'-1}$ is not c-colored, then $l_c(x'-1) = l_c(x')$; if $c(p_{x'-1}) = c$, then we have to connect the point $p_{x'-1}$ to y_i and, hence, $l_c(x'-1) = l_c(x') + (y(p_{x'-1}) - y_i)$.

With these values we can compute any value $link(j,i)$ as follows. First, suppose that $c(y_i) \ne c(y_j)$ as in the second case of Equation (3). Let $p_{x'}$ be the point immediately below y_j. Then

$$\sum_{\substack{\frac{j+2}{3} \le x \le \frac{i}{3} \\ c(p_x) = c(y_i)}} (y(p_x) - y_i) = l_{c(y_i)}(x').$$

Furthermore, we can also compute the length needed for connecting the points of color $c(y_j)$ to the backbone at position y_j since we know their number $n_{c(y_j)}(p_{x'})$ and $y_j - y = (y_j - y_i) - (y - y_i)$ for $y_j \geq y \geq y_i$:

$$\sum_{\substack{\frac{j+2}{3} \leq x \leq \frac{i}{3} \\ c(p_x)=c(y_j)}} (y_j - y(p_x)) = \sum_{\substack{x' \leq x \leq \frac{i}{3} \\ c(p_x)=c(y_j)}} \big((y_j - y_i) - (y(p_x) - y_i)\big)$$
$$= n_{c(y_j)}(x') \cdot (y_j - y_i) - l_{c(y_j)}(x')$$

Hence, we can compute all values $link(j,i)$ with $c(y_j) \neq c(y_i)$ in $O(n)$ total time for fixed i.

Now, assume that $c(y_i) = c(y_j)$ and let again be $p_{x'}$ the point immediately below y_j. If $n_{c'}(x') > 0$ there is a point of color $c' \neq c(y_i)$ between the two backbones; as this point cannot be connected, $link(j,i) = \infty$. If no such point exists, every point connects to the closer backbone, either y_j or y_i. Hence, the points are split into two subsets, where $p_{x''}$ is the topmost point that connects down to y_i and all points $p_{x'}, \ldots, p_{x''-1}$ connect to y_j. Similar to the previous computation, we get that

$$\begin{aligned} link(j,i) &= \sum_{x' \leq x \leq \frac{i}{3}} \min\big(y_j - y(p_x), y(p_x) - y_i\big) \\ &= \sum_{x' \leq x < x''} (y_j - y(p_x)) + \sum_{x'' \leq x \leq \frac{i}{3}} (y(p_x) - y_i) \\ &= \big(n_{c(y_i)}(x') - n_{c(y_i)}(x'')\big) \cdot (y_j - y_i) \\ &\qquad - \big(l_{c(y_i)}(x') - l_{c(y_i)}(x'')\big) + l_{c(y_i)}(x''). \end{aligned}$$

This can be computed in constant time. Note that by simply sweeping once over the backbone positions y_j and the points from y_i to the top in parallel, we can easily find the right x'' for each y_j in $O(n)$ total time.

We have now seen that we can compute all values $link(\cdot, i)$ in $O(n)$ total time. As a consequence, we know that we can compute any entry of table L in $O(n)$ time. For computing all entries of the table, we need, hence, $O\big(n^2 \prod_{c \in C} k_c\big)$ time.

Let S be the set of candidates y_i such that all points below y_i have the same color as y_i. Any solution with y_i as the lowest backbone is a candidate for the optimum solution; we do, however, have to consider the cost of connecting the points below y_i to the backbone through y_i. Note that $y_{3n-1} = y(p_n)$ and y_{3n} are always included in the set S. Summing up, we can compute the minimum total length of a backbone labeling of $p_1, \ldots, p_n$ with at most k_c labels per color $1 \leq c \leq |C|$ as

$$\min_{y_i \in S, k'_1 \leq k_1, \ldots, k'_{|C|} \leq k_{|C|}} \left(L[i, (k'_1, \ldots, k'_{|C|})] + \sum_{\frac{i+2}{3} \leq x \leq n} (y_i - p_x) \right).$$

Hence, we get the following theorem.

Theorem 4 *A minimum length backbone labeling with two-sided backbones for n points with $|C|$ colors can be computed in $O\big(n^2 \cdot \prod_{c\in C} k_c\big)$ time if a color vector $\vec{k}$ bounds the allowed number of labels for each color.*

If we globally bound the total number of labels by K, we can use a similar dynamic program; in the table L, we replace the color vector $\vec{k}$ with the global bound K, that is, we recursively compute values

$$L[i,k] = \lambda + \min_{j<i} \left(L[j,k-1] + \mathit{link}(j,i)\right) \tag{5}$$

with $1 \le i \le 3n$ and $k \le K$. The only difference in the dynamic program is, hence, that we always use the global bound instead of the specific bounds for colors.

Similarly, we can use dynamic programming to compute length-minimal labelings without bounding the number of labels. Here, we recursively compute values

$$L[i] = \lambda + \min_{j<i} \left(L[j] + \mathit{link}(j,i)\right) \tag{6}$$

for $1 \le i \le 3n$. We get the following result.

Theorem 5 *A minimum length backbone labeling with two-sided backbones for n points with $|C|$ colors can be computed in $O(n^2K)$ time if up to K labels in total are allowed and in $O(n^2)$ time if the number of labels is not restricted.*

Note that our dynamic program can also be used for deciding whether a feasible crossing-free solution subject to the bounds on the numbers of labels exists. If no feasible solution exists, the reported minimum length will be ∞.

4.2 One-sided Backbones

We now turn to leader length minimization for labeling with one-sided backbones. Here, the length of a backbone segment may differ heavily; hence, we do not use a parameter λ as we did for two-sided backbones in Section 4.1, but we always count both horizontal and vertical lengths. Recall that we solved the minimization of the number of backbones with the help of a dynamic program based on rectangular subinstances bounded by two backbones and the leftmost point; see Section 3.2. We modify this dynamic program for minimizing the total leader length.

As a first obvious change, we store in the dynamic programming table T the additional length of segments and backbones needed for labeling the points of the subinstance. However, we have to adjust more details. By the case of a single point connected to a backbone, we see that we have to allow backbones passing through input points of the same color for length minimization. Additionally, for computing the vertical length needed for connecting to a backbone placed in a gap, we need to know its actual y-coordinate.

Suppose that there is a set B of backbones that all lie in the same gap between points p_i and p_{i+1}. Let $b^\star$ be the longest of these backbones; see

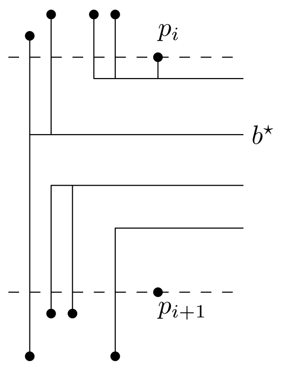


Figure 13: The longest backbone $b^\star$ splits the backbones between p_i and p_{i+1}.

Figure 13. The backbone $b^\star$ vertically splits the set B; any backbone $b' \in B$ above $b^\star$ can only connect to points above itself and any backbone $b'' \in B$ below $b^\star$ can only connect to points below itself. By moving b' to the top and b'' to the bottom as far as possible the total leader length decreases. Hence, in any optimum solution, the backbones above $b^\star$ will be arbitrarily close to $y(p_i)$ and the backbones below $b^\star$ will be arbitrarily close to $y(p_{i+1})$. Furthermore, depending on the numbers of points connected to $b^\star$ from above and from below, by moving $b^\star$ either to the top or to the bottom we will find a solution that is not worse, and in which any backbone of B is arbitrarily close to p_i or to p_{i+1}.

If, for now, we allow backbones to be infinitely close to points or other backbones, we can use backbone positions p_i^- and p_i^+ that lie infinitely close above and below p_i, respectively, and share its y-coordinate. Each of these positions may be used for an arbitrary number of backbones. We will see how to maintain a minimum separation between backbones later.

Now, in the case distinction, we have to be a bit more careful. When the leftmost point p_ℓ in the subinstance bounded by backbones of color c to the top and of color c' to the bottom, respectively, has color c or c', we can no longer always connect p_ℓ to the existing backbones. Although such a connection is always possible, opening a new backbone may save some leader length in this step or in later steps. Hence, we have to additionally test all positions for placing a new backbone in the same way as we do in the case that p_ℓ has a different color. Note that this does not increase the runtime.

With the new positions as well as the input points as possible label positions and the updated case analysis, we can then find a solution with minimum total leader length in $O(n^4|C|^2)$ time, if the number of labels is not bounded, by adding the length of the newly placed segments in any calculation.

Theorem 6 *Given a set P of n colored points, and a color set C, we can compute a feasible labeling of P with one-sided backbones that minimizes the total leader length in $O\big(n^4|C|^2\big)$ time.*

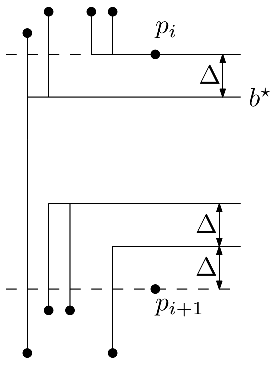
Bounded Numbers of Labels. If we want to integrate an upper bound K on the total number of labels, or a color vector $\vec{k}$ of individual bounds for the colors, into the dynamic program—as we did for two-sided backbones—, we need an additional dimension for the remaining number of backbones that we can use in the subinstance (or a dimension for each color $c \in C$ for the remaining number of backbones of that color); that is, we now use table entries of the form $T[i, c, i', c', \ell, K']$ (or $T[i, c, i', c', \ell, \vec{k}']$) where

- y_i is the position of the upper backbone and $y_{i'}$ is the position of the lower backbone, with $i < i'$.
- the upper backbone has color c and the lower backbone has color c'.
- p_ℓ with $(i+2)/3 \le \ell \le i'/3$ is the leftmost point of the subinstance; if the subinstance is empty, we use the notation $\ell = \emptyset$.
- K' is the maximum number of labels that we allow for the subinstance. If we bound the number of labels for the colors individually, we use a vector $\vec{k}'$ of bounds instead.

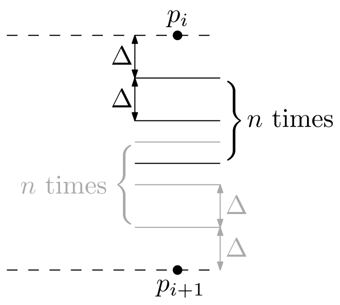
Additionally, when splitting the instance into two parts, we have to consider not only the position of the splitting backbone of color $c(p_\ell)$, but also the different combinations of distributing the allowed numbers of backbones among the subinstances. For a global bound K, we need, hence, $O(nK)$ time for computing an entry of the table. If we have a color vector $\vec{k}$ of individual bounds, we need $O\big(n \prod_{c \in C} k_c\big)$ time. Together with the additional dimension(s) of the table, we can minimize the total leader length in $O\big(n^4 |C|^2 K^2\big)$ time if we have a global bound K, and in $O\Big(n^4 |C|^2 \big(\prod_{c \in C} k_c\big)^2\Big)$ time if we have a color vector $\vec{k}$. Note that we can easily detect cases where we have to add a backbone of color $c(p_\ell)$ but the current bound $k_{c(p_\ell)} = 0$ (or $K = 0$) in the subinstance. In such a case, we report $+\infty$ as the total leader length, indicating that no feasible solution with the given bounds exists.

Theorem 7 *Given a set P of n colored points, a color set C, and a label bound K (or a vector $\vec{k}$ of bounds per color), we can compute a feasible labeling of P with one-sided backbones that minimizes the total leader length in $O\big(n^4 |C|^2 K^2\big)$ time (or in $O\Big(n^4 |C|^2 \big(\prod_{c \in C} k_c\big)^2\Big)$ time).*

Minimum Distances. So far, we allowed backbones to be infinitely close to unconnected points and to other backbones, which will, in practice, lead to undesirable overlaps. One would rather enforce a small distance between two backbones or a backbone and a point, even if this increases the total leader length a bit. Let $\Delta > 0$ be the minimum allowed distance, which depends, for example, on the font size used for the labels. In an optimum solution, there will be two sequences of backbones on the top and on the bottom of a gap between p_i and p_{i+1}, such that inside a sequence consecutive backbones have distance Δ; see Figure 14a. We get all possible backbone positions inside the gap by taking all y-coordinates inside whose y-distance to either p_i or p_{i+1} is an integer multiple of Δ; see Figure 14b. Note that n positions of each type suffice in a gap; if the gap is too small, there are fewer such positions. The two



(a) Backbones placed with the minimum leader length.



(b) Candidate positions for backbones inside the gap.

Figure 14: Situation between two consecutive points for one-sided backbones.

sequences can overlap. In this case, we have to check that we do not combine two positions with a distance smaller than Δ in the dynamic program.

Together with the input points, we get a set of $O(n^2)$ candidate positions for backbones, each of which can be used at most once. This increases the number of entries of table T by a factor of $O(n^2)$, and the running time of computing a single entry by a factor of $O(n)$. The resulting running time of our dynamic program is $O\big(n^7|C|^2\big)$ if we do not bound the number of labels, $O\big(n^7|C|^2K^2\big)$ if we have a global bound K on the number of labels, and $O\Big(n^7|C|^2\big(\prod_{c\in C} k_c\big)^2\Big)$ if we have a color vector $\vec{k}$ of individual bounds per color.

Theorem 8 *Given a set P of n colored points, a color set C, a label bound K (or a vector $\vec{k}$ of bounds per color), and a minimum distance $\Delta > 0$, we can compute a feasible labeling of P with one-sided backbones of minimum pairwise distance Δ that minimizes the total leader length in $O\big(n^7|C|^2K^2\big)$ time (or in $O\Big(n^7|C|^2\big(\prod_{c\in C} k_c\big)^2\Big)$ time).*

Note that, as in the case of two-sided backbones, also for one-sided backbones we can use the dynamic program for deciding whether a feasible solution for the given bounds on the numbers of labels exists: If no such solution exists, the reported total leader length will be ∞.

5 Crossing Minimization

In this section we allow crossings between backbone leaders, which generally allows us to use fewer labels. More precisely, if crossings are allowed, it is trivially possible to label all points using just one label per color. Such a solution may, however, lead to many crossings between backbones and vertical leader segments. Therefore, we are interested in minimizing the number of such crossings. We concentrate on the case that $K = |C|$, that is, only one label per color is allowed. We will first consider the case that the relative order of labels for the colors from top to bottom is prescribed. For this case we will present efficient algorithms for minimizing the number of crossings. Then, we will see that without this restriction the problem is NP-hard, at least for one-sided backbones.

5.1 Fixed y-Order of Labels

We first assume that the color set C is ordered and we require that for each pair of colors $i < j$ the label of color i is above the label of color j. We will develop a fast algorithm for crossing minimization with two-sided backbones. Then, we will show how this algorithm can be modified for the case of one-sided backbones.

5.1.1 Two-sided Backbones

Since the order of the labels is fixed, the order in which the backbones appear from top-to-bottom is also fixed. This implies that the i-th backbone in the given y-ordering from top to bottom is connected to the points of color i.

Observe that it is always possible to slightly shift the backbones of a solution without increasing the number of crossings such that no backbone contains a point. Thus, the backbones can be assumed to be positioned in the gaps between vertically adjacent points; we number the gaps from 0 to n as in Section 3.2.

Suppose that we fix the position of the i-th backbone to gap g. For $1 \leq i \leq |C|$ and $0 \leq g \leq n$, let $\mathit{cross}(i, g)$ be the number of crossings of the vertical segments of the non-i-colored points when the color-i backbone is placed at gap g. Note that this number depends only on the y-ordering of the backbones, which is fixed, and not on their actual positions. So, we can precompute the table *cross*, using dynamic programming, as follows.

All table entries of the form $\mathit{cross}(\cdot, 0)$ can clearly be computed in $O(n|C|)$ total time because, for color i, $\mathit{cross}(i, 0)$ is equal to number of points having some color $j < i$. Then, $\mathit{cross}(i, g) = \mathit{cross}(i, g-1) + 1$, if the point p_g between gaps $g-1$ and g has color j with $j > i$. In the case where p_g has color j with $j < i$, $\mathit{cross}(i, g) = \mathit{cross}(i, g-1) - 1$. If p_g has color i, then $\mathit{cross}(i, g) = \mathit{cross}(i, g-1)$. From the above, it follows that the computation of the table *cross* takes $O(n|C|)$ time.

Now, we use another dynamic program for computing the minimum number of crossings. Let $T[i, g]$ denote the minimum number of crossings on the backbones $1, \ldots, i$ in any solution subject to the condition that the backbones are placed in the given ordering and backbone i is positioned in gap g. Clearly, $T[0, g] = 0$ for $g = 0, \ldots, n$. For computing an entry $T[i, g]$ with $i > 0$, we test all positions for the previous backbone $i - 1$ in a gap g' above (and including) gap g. In addition to the number of crossings from the entry $T[i-1, g']$, we also have to take the number $\mathit{cross}(i, g)$ of crossings of the new backbone into account. Hence, we have

$$T[i, g] = cross(i, g) + \min_{g' \leq g} T[i-1, g'].$$

Having precomputed the table *cross* and assuming that for each entry $T[i, g]$, we also store the smallest entry $T[i, g']$ with $g' \leq g$, each entry of table T can be computed in constant time. Hence, table T can be filled in time $O(n|C|)$. Then, the minimum crossing number is $\min_{0 \leq g \leq n} T[|C|, g]$. A corresponding solution can be found by backtracking in the dynamic program.

Theorem 9 *Given a set P of n colored points and an ordered color set C, a backbone labeling with one label per color, labels in the given color order, two-sided backbones, and minimum number of crossings can be computed in $O(n|C|)$ time.*

5.1.2 One-sided Backbones

We can easily modify the approach used for two-sided backbones for minimizing the number of crossings for one-sided backbones, if the y-order of labels is fixed, as the following theorem shows.

Theorem 10 *Given a set P of n colored points and an ordered color set C, a backbone labeling with one label per color, labels in the given order, one-sided backbones, and minimum number of crossings can be computed in $O(n|C|)$ time.*

Proof: We develop a dynamic program very similar to the one presented for two-sided backbones. The only part that we have to change is that the computation of the number of crossings when fixing a backbone at a certain position should take into consideration that the backbones are of finite length. Recall that the dynamic program could precompute these crossings, by maintaining an $n \times |C|$ table *cross*, in which each entry $cross(i, g)$ corresponds to the number of crossings of the non-i-colored points when the color-i-backbone is placed at gap g, for $1 \leq i \leq |C|$ and $0 \leq g \leq n$. For one-sided backbones, $cross(i, g) = cross(i, g-1) + 1$, if the point between gaps $g-1$ and g is right of the leftmost i-colored point and has color j with $j > i$. In the case, where the point p_g between gaps $g-1$ and g is right of the leftmost i-colored point and has color j with $j < i$, $cross(i, g) = cross(i, g-1) - 1$. Otherwise, $cross(i, g) = cross(i, g-1)$. Again, all table entries of the form $cross(\cdot, 0)$ can clearly be computed in $O(n)$ time. The remainder of the dynamic program works as before. □

5.2 Flexible y-Order of Labels

We now no longer assume that the order of labels is prescribed, that is, we need to minimize the number of crossings over all label orders. While there is an efficient algorithm for a restricted variant of the problem with two-sided backbones, the problem is NP-complete for one-sided backbones.

5.2.1 Two-sided Backbones

We give an efficient algorithm for the case that there are $K = |C|$ fixed label positions $y_1, \ldots, y_K$ on the right boundary of R, for instance, uniformly distributed.

Theorem 11 *Given a set P of n colored points, a color set C, and a set of $|C|$ fixed label positions, we can compute a feasible backbone labeling with two-sided backbones that minimizes the number of crossings in $O(n + |C|^3)$ time.*

Proof: First observe that if the backbone of color k with $1 \leq k \leq |C|$ is placed at position y_i with $1 \leq i \leq |C|$, then the number of crossings created by the vertical segments leading to this backbone is fixed, since all label positions will be occupied by a two-sided backbone. Let n_k be the number of points of color k. The crossing number $\mathrm{cr}(k, i)$ can be determined in $O(n_k + |C|)$ time. In fact, by a sweep from top to bottom, we can even determine all crossing numbers $\mathrm{cr}(k, \cdot)$ for backbone k with $1 \leq k \leq |C|$ in $O(n_k + |C|)$ time. Now, we construct an instance of a weighted bipartite matching problem, where for each position y_i with $1 \leq k \leq |C|$ and each backbone k with $1 \leq k \leq |C|$, we establish an edge $\{k, i\}$ of weight $\mathrm{cr}(k, i)$. In total, this takes $O(n + |C|^2)$ time. A minimum-cost weighted bipartite perfect matching can be computed in $O(|C|^3)$ time using the Hungarian method [13] and this yields a backbone labeling with the minimum number of crossings. □

Note that the previous approach does not work for one-sided backbones. In contrast to two-sided backbones a crossing of a vertical segment for some color with a backbone depends on the horizontal length and, hence, on the color of this backbone. Therefore, it is not possible to simply calculate a number $\mathrm{cr}(k, i)$ of crossings for the placement of backbone k on position y_i.

5.2.2 One-sided Backbones

Next, we consider the variant with one-sided backbones and prove that it is NP-hard to minimize the number of crossings. Here, we do not restrict ourselves to candidate positions for backbones. For simplicity, we allow points that share the same x- or y-coordinates. This can be remedied by a slight perturbation. Our arguments do not make use of this special situation and, hence, carry over to the perturbed constructions. We first introduce a number of gadgets that are required for our proof and explain their properties, before describing the hardness reduction.

The first gadget is the *range restrictor gadget.* Its construction consists of a middle backbone, whose position will be restricted to a given vertical range R, and an upper and a lower *guard gadget* that ensure that positioning the middle backbone outside range R creates many crossings; see Figure 15. We assume that the middle backbone is connected to at least one point further to the left such that it extends beyond all points of the guard gadgets. Additionally, the middle backbone is connected to two *range points* whose y-coordinates are the

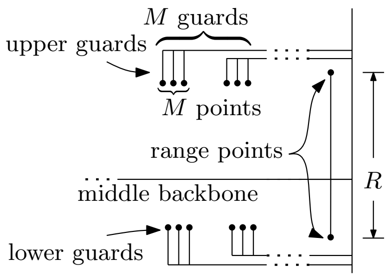


Figure 15: The range restrictor gadget.

upper and lower boundary of the range R. Their x-coordinates are such that they are on the right of the points of the guard gadgets. A *guard* consists of a backbone that connects to a set of M points, where $M > 1$ is an arbitrary number. The M points of a guard lie left of the range points. The upper guard points are horizontally aligned and lie slightly below the upper bound of range R. The lower guard points are horizontally aligned and are placed such that they are slightly above the lower bound of range R. We place M upper and M lower guards such that the guards form pairs for which the guard points overlap horizontally. The upper (respectively lower) guard gadget is formed by the set of upper (respectively lower) guards. We call M the *size* of the guard gadgets. The next lemma shows the important properties of the range restrictor gadget.

Lemma 3 *The backbones of the range restrictor gadget can be positioned such that there are no crossings. If the middle backbone is positioned outside the range R, there are at least $M - 1$ crossings.*

Proof: The first statement is illustrated by the drawing in Figure 15. Suppose for a contradiction to the second statement that the middle backbone is positioned outside range R and that there are fewer than $M - 1$ crossings. Assume, without loss of generality, that the middle backbone is embedded below range R; the other case is symmetric.

First, observe that all backbones of guards must be positioned above the middle backbone, as a guard backbone below the middle backbone would create M crossings, namely between the middle backbone and the segments connecting the points of the guard to its backbone. Hence the middle backbone is the lowest. Now observe that any guard that is positioned below the upper range point crosses the segment that connects this range point to the middle backbone. To avoid having $M - 1$ crossings, it follows that at least $M + 1$ guards (both upper and lower) must be positioned above range R. Hence, there is at least one pair consisting of an upper and a lower guard that are both positioned above the range R. This, however, independent of their ordering, creates at least $M - 1$ crossings; see Figure 16, where the two alternatives for the lower guard are drawn in black and bold gray, respectively. This contradicts our assumption. $\square$

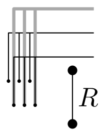


Figure 16: Crossings caused by a pair of an upper and a lower guard that are positioned on the same side outside range R.

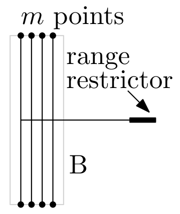


Figure 17: The blocker gadget.

Let B be an axis-aligned rectangular box and let R be a small interval that is contained in the range of y-coordinates spanned by B. A *blocker gadget* of *width* m consists of a backbone that connects to $2m$ points, half of which are positioned on the top and on the bottom side of B, respectively. Moreover, a range restrictor gadget is used to restrict the backbone of the blocker to the range R. Figure 17 shows an example. Note that, due to the range restrictor, this drawing is essentially fixed. We say that a backbone *crosses* the blocker gadget if its backbone crosses the box B. It is easy to see that any backbone that crosses a blocker gadget creates m crossings, where m is the width of the blocker.

We are now ready to show that the crossing minimization problem with flexible y-order of the labels is NP-complete.

Theorem 12 *Given a set P of input points in $k = |C|$ different colors and an integer Y it is* NP*-complete to decide whether a backbone labeling with one label per color and at most Y leader crossings exists.*

Proof: The proof of NP-hardness is by reduction from the NP-complete Fixed Linear Crossing Number problem [16], which is defined as follows. Given a graph $G = (V, E)$, a bijective function $f : V \to \{1, \ldots, |V|\}$, and an integer Z, one has to decide whether there is a drawing of G with the vertices placed on a horizontal line (the *spine*) in the order specified by f and the edges drawn as semi-circles above or below the spine so that there are at most Z edge crossings. Masuda et al. [16] showed that the problem is NP-complete even if G is a matching.

Let G be a matching. Then the number of vertices is even and we can assume that the vertices $V = \{v_1, \ldots, v_{2n}\}$ are indexed in the order specified by f, that is, $f(v_i) = i$ for $1 \leq i \leq 2n$. Furthermore, we direct every edge $\{v_i, v_j\}$ with $i < j$ from v_i to v_j. Let $\{u_1, \ldots, u_n\}$ be the ordered source vertices and let $\{w_1, \ldots, w_n\}$ be the ordered sink vertices. Figure 18 shows an example graph G drawn on a spine in the specified order.

In our reduction we will create an *edge gadget* for every edge of G. The gadget consists of five blocker gadgets and one *side selector gadget.* Each of the six sub-gadgets uses its own color and thus defines one middle backbone. The edge gadgets are ordered from left to right according to the sequence of source vertices $(u_1, \dots, u_n)$. Figure 19 shows a sketch of the instance I_G created for the matching G with four edges shown in Figure 18.

The edge gadgets are placed symmetrically with respect to the x-axis. We create $2n + 1$ special rows below the x-axis and $2n + 1$ special rows above, indexed by $-(2n+1), -2n, \dots, 0, \dots, 2n, 2n+1$. The gadget for an edge (v_i, v_j) uses five blocker gadgets (denoted as *central, upper, lower, upper gap*, and *lower gap* blockers) in two different columns to create two small gaps in rows j and $-j$, see the hatched blocks in the same color in Figure 19. The upper and lower blockers extend vertically to rows $2n+1$ and $-2n-1$, respectively. The gaps are intended to create two alternatives for routing the backbone of the side selector. Every backbone that starts left of the two gap blockers is forced to cross at least one of these five blocker gadgets as long as it is vertically placed between rows $2n + 1$ and $-2n - 1$. The blockers have width $m = 8n^2$. Their backbones are fixed to lie between rows 0 and -1 for the central blocker, between rows $2n$ and $2n + 1$ (respectively $-2n$ and $-2n - 1$) for the upper (respectively lower) blocker, and between rows j and $j + 1$ (respectively $-j$ and $-j - 1$) for the upper (respectively lower) gap blocker; recall that this can easily be done by placing the range restrictor gadget of the blocker at the respective position.

The *side selector* consists of two horizontally spaced *selector points* $s_1^{(i)}$ and $s_2^{(i)}$ in rows i and $-i$ located between the left and right blocker columns. They have the same color and thus define one joint backbone, the *selector backbone*, which is supposed to pass through one of the two gaps in an optimal solution. The n edge gadgets are placed from left to right in the order of their source vertices; see Figure 19.

The backbone of every selector gadget is vertically restricted to the range between rows $2n + 1$ and $-2n - 1$ in any optimal solution by augmenting each selector gadget with a range restrictor gadget. This means that we add two more points for each selector to the right of all edge gadgets, one in row $2n + 1$ and the other in row $-2n - 1$. They are connected to the selector backbone. In combination with a corresponding upper and lower guard gadget of size $M = \Omega(n^4)$ between the two selector points $s_1^{(i)}$ and $s_2^{(i)}$ this achieves the range restriction according to Lemma 3.

This completes the backbone labeling instance. We will now show two important properties of optimal solutions for the constructed instance. We first show that the selector backbones do indeed pass through one of their two gaps.

Claim 1 *In a crossing-minimal labeling the backbone of the selector gadget for every edge (v_i, v_j) passes through one of its two gaps in rows j or $-j$.*

Proof: There are basically three different options for placing a selector backbone: (a) outside its range restriction, that is, above row $2n + 1$ or below row $-2n - 1$, (b) between rows $2n + 1$ and $-2n - 1$, but not in one of the two gaps, and (c) in rows j or $-j$, that is, inside one of the gaps. In case (a) we get at least $M = \Omega(n^4)$ crossings by Lemma 3. So we may assume that case (a) never occurs for any selector gadget; we will see that in this case there are only $O(n^4)$

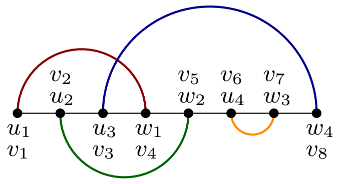


Figure 18: An instance of Fixed Linear Crossing Number with four edges.

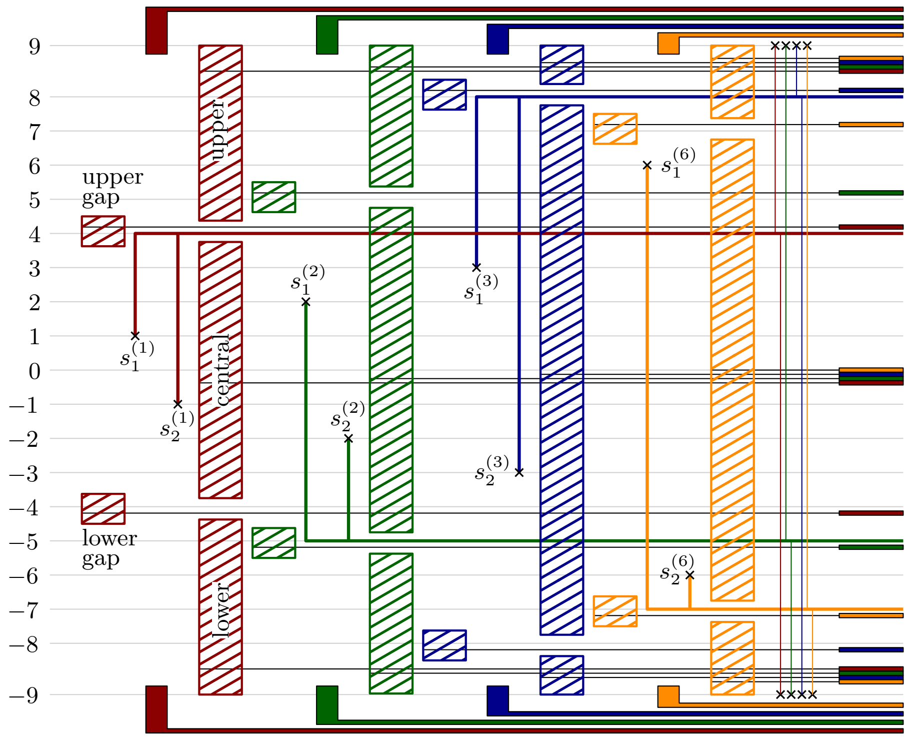


Figure 19: Sketch of the reduction of the graph of Figure 18 to a backbone labeling instance. Hatched rectangles represent blockers, thick segments represent side selectors, and filled shapes represent guard gadgets or range restrictor gadgets.

crossing in total for the selector gadgets. In cases (b) and (c) we note that the backbone will cross one blocker for each edge whose source vertex is right of v_i in the order $(u_1, \dots, u_n)$. Let k be the number of these edges. Additionally, in case (b), the backbone crosses one of its own blockers. In cases (b) and (c) the two vertical segments of the range restrictor of edge (v_i, v_j) cross every selector and blocker backbone regardless of the position of its own backbone, which yields $6n-1$ crossings. Thus, case (b) causes at least $(k+1) \cdot m + 6n - 1$ crossings.

For giving an upper bound on the number of crossings in case (c) we note that the backbone can cross at most three vertical segments of any other selector gadget: the two segments connected to its selector points and one segment connected to a point in either row $2n+1$ or row $-2n-1$, which is part of the range restrictor gadget. The two vertical segments connected to points $s_1^{(i)}$ and $s_2^{(i)}$ together will cross the backbone of each central blocker at most once, the backbones of each pair of upper/lower gap blockers at most twice, and each selector backbone at most twice. Backbones of upper and lower blockers are never crossed in case (c). So in case (c) the segments of the selector gadget cross at most $km + 8n - 1$ segments, which is less than the lower bound of $(k+1)m + 6n - 1$ in case (b). We conclude that each backbone indeed passes through one of the gaps in an optimal solution. Any violation of this rule would create at least m additional crossings, which is more than what an arbitrary assignment of selector backbones to gaps yields. □

Next, we show how the number of crossings in the backbone labeling instance relates to the number of crossings in the instance of the Fixed Linear Crossing Number problem. There is a large number of unavoidable crossings regardless of the backbone positions of the selector gadgets. By Property 1 and the fact that violating any range restriction immediately causes $M = \Omega(n^2)$ crossings, we can assume that every backbone adheres to the rules, that is, stays within its range as defined by the range restriction gadgets or passes through one of its two gaps, in the case of selector backbones.

Claim 2 *An optimal solution of the backbone labeling instance I_G created for a matching G with n edges has $X+2Z$ crossings, where X is a constant depending on G and Z is the minimum number of crossings of G in the* Fixed Linear Crossing Number *instance.*

Proof: Aside from guard backbones, which never have crossings, there are two types of backbones in our construction, blocker and selector backbones. We argue separately for all four possible types of crossings and distinguish fixed crossings that must occur and variable crossings that depend on the placement of the selector backbones. The types of crossings are

(i) crossings between blocker backbones and vertical blocker segments,

(ii) crossings between blocker backbones and vertical selector segments,

(iii) crossings between selector backbones and vertical blocker segments, and

(iv) crossings between selector backbones and vertical selector segments.

We will analyze the numbers of crossings for these types individually.

Case i: By construction each blocker backbone must intersect exactly one blocker gadget of width m for each edge gadget to its right. Thus we obtain

$$X_1 = 5m \sum_{i=1}^{n-1} i = 5m \cdot \frac{n^2 - n}{2}$$

fixed crossings in total from Case i.

Case ii: Each blocker backbone crosses, for each edge, exactly one vertical selector segment that is part of the range restrictor gadget on the right-hand side of our construction. Each central blocker backbone additionally crosses for each edge gadget to its right one vertical segment incident to one of the selector points, regardless of the selector position. The two gap blocker backbones for gaps in rows j and $-j$ together cause two additional crossings for each edge gadget to its right whose target vertex v_k satisfies $k > j$. To see this we need to distinguish two cases. Let $e = (v_i, v_k)$ be the edge of an edge gadget with $k > j$. If $i < j$, then both vertical selector segments either cross the lower gap blocker backbone or they both cross the upper gap blocker backbone (see edges (v_1, v_4) and (v_2, v_5) in Figure 19). If $i > j$, then one of the two vertical selector segments crosses both gap blocker backbones, and the other one crosses none (see edges (v_1, v_4) and (v_6, v_7) in Figure 19). The backbones of the upper and lower blockers do not cross any other vertical selector segment.

Let $\kappa = |\{\{(v_i, v_j), (v_k, v_l)\} \in E^2 \mid i < k \text{ and } j < l\}| = O(n^2)$ be the number of pairs of edges causing crossings with gap blocker backbones. Then we obtain

$$X_2 = 5n^2 + \frac{n^2 - n}{2} + 2\kappa$$

fixed crossings from Case ii.

Case iii: Each selector backbone that passes through one of its gaps crosses exactly one blocker gadget for each edge gadget to its right. Thus, we obtain

$$X_3 = m \sum_{i=1}^{n-1} i = m \cdot \frac{n^2 - n}{2}$$

fixed crossings in Case iii.

Case iv: Let $e = (v_i, v_j)$ and $f = (v_k, v_l)$ be two edges in G, and let $i < k$. Then there are three sub-cases: (a) e and f are *sequential*, that is, $i < j < k < l$, (b) e and f are *nested*, that is, $i < k < l < j$, or (c) e and f are *interlaced*, that is, $i < k < j < l$. For every pair of sequential edges there is exactly one crossing, regardless of the gap assignments (see edges (v_1, v_4) and (v_6, v_7) in Figure 19). For every pair of nested edges there is no crossing, regardless of the gap assignments (see edges (v_3, v_8) and (v_6, v_7) in Figure 19). Finally, for every pair of interlaced edges there are no crossings if the respective side selector backbones are assigned to opposite sides of the x-axis or two crossings if they are assigned to the same side. Therefore, pairs of interlaced edges do not contribute to the number of fixed crossings. Let $\tau = \big|\big\{\{(v_i, v_j), (v_k, v_l)\} \in E^2 \mid i < j < k < l\big\}\big| = O(n^2)$ be the number of pairs of sequential edges. Then we obtain

$$X_4 = \tau$$

fixed crossings from Case iv.

From the discussion of the four cases we can immediately see that all crossings are fixed, except for those related to pairs of interlaced edges (see, for example, edges (v_1, v_4) and (v_3, v_8) or (v_2, v_5) in Figure 19). These are exactly the edge pairs that create crossings in the Fixed Linear Crossing Number problem if assigned to the same side of the spine. As discussed in Case iv the selector gadgets of two interlaced edges create two extra crossings if and only if they are assigned to gaps on the same side of the x-axis. If we create a bijection that maps a selector backbone placed in the upper gap to an edge drawn above the spine, and a selector backbone in the lower gap to an edge drawn below the spine, we see that an edge crossing on the same side of the spine in a drawing of G corresponds to two extra crossings in a labeling of I_G and vice versa. So, if Z is the minimum number of crossings in a spine drawing of G, then $2Z$ is the minimum number of variable crossings in a labeling of I_G. By setting $X = X_1 + X_2 + X_3 + X_4$ this proves Claim 2. □

From Claim 2 it follows immediately that crossing minimization with one-sided backbones is NP-hard since the size of the instance I_G is polynomial in n (more precisely, we need only $O(n^5)$ points for I_G).

Furthermore, we can guess an order of the backbones and then, using the algorithm of Theorem 10, compute the minimum crossing number for this order. This shows that crossing minimization is contained in NP. Hence, the problem is NP-complete. □

6 Conclusion

We have introduced the new model of many-to-one boundary labeling with backbones; this model generalizes the *po*-leader model of classical one-to-one boundary labeling. For our two crossing-free settings, one-sided and two-sided backbones, we have seen that minimizing the total number of labels as well as minimizing the total length of leaders can be achieved in polynomial time by dynamic programming. On the other hand, only very restricted versions of crossing minimization can be solved efficiently. In general, crossing minimization with a bounded number of labels per color is NP-hard for one-sided backbones.

Open Problems. In the setting of crossing minimization, we have just seen hardness for one-sided backbones. In our hardness proof it was essential that backbones do not extend infinitely to the left; hence, the hardness proof we gave cannot simply be modified for showing hardness for two-sided backbones. It is, therefore, an open problem whether the simpler structure of many-to-one labelings with two-sided backbones (and no predefined backbone slots) allows for efficiently minimizing the number of crossings or whether the problem is also NP-hard.

The other optimization criteria, that is, minimizing the total number of labels or the total leader length can be solved optimally. We did, however, only consider the case where just one side of the focus region is used for placing the labels. For two-sided backbones, using both the left and the right boundary does not make a difference. However, we get much more flexibility in the case of one-sided backbones. An open question is, hence, whether the 2-sided problem variant is still solvable in polynomial time.

In the setting with limited numbers of labels, either in total or per color, it is possible that no feasible labeling for all sites exists; our algorithms were able to detect such situations, reporting a cost of $+\infty$ for labeling all sites. However, one may still want to find a labeling for a subset of the sites subject to the bound on the number of labels. This gives rise to the new objective of finding a feasible labeling for a maximum number of sites. Both for one-sided and two-sided backbones it should be possible to integrate this new objective into the respective dynamic programs presented for leader length minimization in Section 4. To this end, entries of the tables must represent the maximum weight of sites that can be labeled in subinstances and minimization must be changed to maximization. Some further modifications will be necessary, but should not be very difficult to realize.

In a slight generalization of labeling by one-sided backbones, one could allow labels to be placed on either side of the enclosing rectangle, left or right. This increased freedom can allow labelings with smaller leader length and can even allow a feasible labeling where no one-sided labeling with all labels on the same side exists. Note that in this generalized one-sided version, even a mixed model with both one-sided and two-sided backbones can make sense: Since a two-sided backbone allows us to attach two labels, one on either side, the distance between a point and its label may be decreased, which can improve the readability. Hence, a possible optimization could aim at maximizing the number of points that are connected to two-sided backbones. Apart from crossing minimization, we expect that also these generalized models can be solved optimally—with increased time complexity, however—by using more complex dynamic programs.

Acknowledgements. This work was started at the Bertinoro Workshop on Graph Drawing 2013. M. Nöllenburg received financial support by the Concept for the Future of KIT. The work of M. A. Bekos and A. Symvonis is implemented within the framework of the Action "Supporting Postdoctoral Researchers" of the Operational Program "Education and Lifelong Learning" (Action's Beneficiary: General Secretariat for Research and Technology), and is co-financed by the European Social Fund (ESF) and the Greek State. We also acknowledge partial support by GRADR – EUROGIGA project no. 10-EuroGIGA-OP-003.

References

[1] L. Barth, A. Gemsa, B. Niedermann, and M. Nöllenburg. On the readability of boundary labeling. In E. Di Giacomo and A. Lubiw, editors, *Graph Drawing (GD'15)*, LNCS. Springer, 2015. To appear. URL: http://arxiv.org/abs/1509.00379.

[2] M. A. Bekos, M. Kaufmann, M. Nöllenburg, and A. Symvonis. Boundary labeling with octilinear leaders. *Algorithmica*, 57(3):436–461, 2010. doi:10.1007/s00453-009-9283-6.

[3] M. A. Bekos, M. Kaufmann, K. Potika, and A. Symvonis. Multi-stack boundary labeling problems. In S. Arun-Kumar and N. Garg, editors, *Foundations of Software Technology and Theoretical Computer Science (FSTTCS'06)*, volume 4337 of *LNCS*, pages 81–92. Springer, 2006. doi:10.1007/11944836_10.

[4] M. A. Bekos, M. Kaufmann, A. Symvonis, and A. 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.

[5] M. Benkert, H. Haverkort, M. Kroll, and M. Nöllenburg. Algorithms for multi-criteria boundary labeling. *J. Graph Algorithms Appl.*, 13(3):289–317, 2009. doi:10.7155/jgaa.00189.

[6] T. Bruckdorfer, M. Kaufmann, S. G. Kobourov, and S. Pupyrev. On embeddability of buses in point sets. In E. Di Giacomo and A. Lubiw, editors, *Graph Drawing (GD'15)*, LNCS. Springer, 2015. To appear. URL: http://arxiv.org/abs/1508.06760.

[7] M. Formann and F. Wagner. A packing problem with applications to lettering of maps. In *Proc. Symp. Comput. Geom. (SCG'91)*, pages 281–288. ACM, 1991. doi:10.1145/109648.109680.

[8] A. Gemsa, J.-H. Haunert, and M. Nöllenburg. Multi-row boundary-labeling algorithms for panorama images. *ACM Trans. Spatial Algorithms and Systems*, 1(1):1:1–1:30, 2015. doi:10.1145/2794299.

[9] R. Hassin and A. Tamir. Improved complexity bounds for location problems on the real line. *Operations Research Letters*, 10(7):395–402, 1991. doi:10.1016/0167-6377(91)90041-M.

[10] Z.-D. Huang, S.-H. Poon, and C.-C. Lin. Boundary labeling with flexible label positions. In S. P. Pal and K. Sadakane, editors, *Algorithms and Computation (WALCOM'14)*, volume 8344 of *LNCS*, pages 44–55. Springer, 2014. doi:10.1007/978-3-319-04657-0_7.

[11] M. Kaufmann. On map labeling with leaders. In S. Albers, H. Alt, and S. Näher, editors, *Festschrift Mehlhorn*, volume 5760 of *LNCS*, pages 290–304. Springer, 2009. doi:10.1007/978-3-642-03456-5_20.

[12] P. Kindermann, B. Niedermann, I. Rutter, M. Schaefer, A. Schulz, and A. Wolff. Two-sided boundary labeling with adjacent sides. In F. Dehne, R. Solis-Oba, and J.-R. Sack, editors, *Algorithms and Data Structures (WADS'13)*, volume 8037 of *LNCS*, pages 463–474. Springer, 2013. doi:10.1007/978-3-642-40104-6_40.

[13] H. W. Kuhn. The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1–2):83–97, 1955. doi:10.1002/nav.3800020109.

[14] C.-C. Lin. Crossing-free many-to-one boundary labeling with hyperleaders. In *Proc. IEEE Pacific Visualization Symp. (PacificVis'10)*, pages 185–192. IEEE, 2010. doi:10.1109/PACIFICVIS.2010.5429592.

[15] C.-C. Lin, H.-J. Kao, and H.-C. Yen. Many-to-one boundary labeling. *Journal of Graph Algorithms and Applications*, 12(3):319–356, 2008. doi:10.7155/jgaa.00169.

[16] S. Masuda, K. Nakajima, T. Kashiwabara, and T. Fujisawa. Crossing minimization in linear embeddings of graphs. *IEEE Trans. Computers*, 39(1):124–127, 1990. doi:10.1109/12.46286.

[17] G. Neyer. Map labeling with application to graph drawing. In M. Kaufmann and D. Wagner, editors, *Drawing Graphs*, volume 2025 of *LNCS*, pages 247–273. Springer, 2001. doi:10.1007/3-540-44969-8_10.

[18] M. Nöllenburg, V. Polishchuk, and M. Sysikaski. Dynamic one-sided boundary labeling. In *Advances in Geographic Information Systems (SIGSPATIAL'10)*, pages 310–319. ACM, 2010. doi:10.1145/1869790.1869834.

[19] A. Wolff and T. Strijk. The Map-Labeling Bibliography, 1996. URL: http://i11www.ira.uka.de/map-labeling/bibliography.