

# Many-to-One Boundary Labeling with Backbones

Michael A. Bekos<sup>1</sup>, Sabine Cornelsen<sup>2</sup>, Martin Fink<sup>3</sup>, Seokhee Hong<sup>4</sup>,  
Michael Kaufmann<sup>5</sup>, Martin Nöllenburg<sup>6</sup>, Ignaz Rutter<sup>6</sup>, and Antonios Symvonis<sup>1</sup>

<sup>1</sup> National Technical University of Athens

<sup>2</sup> Universität Konstanz

<sup>3</sup> Universität Würzburg

<sup>4</sup> University of Sydney

<sup>5</sup> Universität Tübingen

<sup>6</sup> Karlsruhe Institute of Technology

**Abstract.** In this paper we introduce and study *many-to-one boundary labeling with backbone leaders*. 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 label number and total leader length minimization of crossing-free backbone labelings. When crossings are allowed, we aim to obtain 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.

## 1 Introduction

The process of annotating images with text in order to fully describe 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, due to the small size of labels (usually a single word/name) and our ability to control the feature density, we usually manage to place the labels on the map so they are in the immediate vicinity of the feature they describe. Map labeling has been studied in computer science for more than two decades [2]. Survey on algorithmic map labeling and an extensive bibliography are given by Neyer [9] and Wolff and Strijk [10], respectively. However, *internal labeling* is not feasible when large labels are employed, a typical situation in technical drawings and medical atlases. *Boundary labeling* was developed by Bekos et al. [1] as a framework and an algorithmic response to the poor quality (feature occlusion, label overlap) of specific labeling applications. In boundary labeling, labels are placed at the boundary of a rectangle and are connected to their associated features via arcs referred to as *leaders*. Leaders attach to labels at *label-ports*. A survey article by Kaufmann [4] presents the different boundary labeling models that have been studied in the literature.

As in map labeling, most work on boundary labeling has been devoted to the case where each label is associated with a single feature point. However, the case where each label is associated to more than one feature point (the topic of this paper) is also common in applications. We can think of groups of features sharing common properties (e.g., identical components of technical devices or locations of plants/animals of the

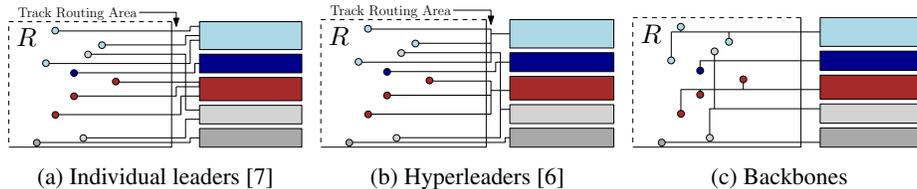


Fig. 1: Different types of many-to-one labelings.

same species in a map), which we express as having the same color. Then, we need to connect via leaders these identically colored feature points to a label of the same color. *Many-to-one boundary labeling* was formally introduced by Lin et al. [7]. In their initial definition of many-to-one labeling each label had one port for each connecting feature point, i.e., each point uses an individual leader (see Fig. 1a). 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. [7] examined one and two-sided boundary labeling using so-called *opo*-leaders [1]. 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 [6] 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 Fig. 1b). At the cost of label duplications, leader crossings could be eliminated.

We introduce and study *many-to-one boundary labeling with backbone leaders* (for short, *backbone labeling*). 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 a label are linked via vertical line segments to the label’s backbone (see Fig. 1c). 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 Fig. 1 shows, backbone labelings also require much less “ink” in the image than the previous methods and thus is expected to be less disturbing for the viewer. We note that backbone labeling can be seen as a variation of Lin’s *opo*-hyperleaders. Lin [6] posed it as an open problem to study *po*-hyperleaders (which is his terminology for backbones), in particular to minimize the number of duplicate labels in a crossing-free labeling.

**Contribution.** Our paper studies three aspects of backbone labeling, *label number minimization* (Section 3), *total leader length minimization* (Section 4), and *crossing minimization* (Section 5). The first two aspects require crossing-free leaders. We consider both *finite backbones* and *infinite backbones*. Finite backbones extend horizontally from the label to the furthest point connected to the backbone, whereas infinite 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 (Sections 3 and 4),

which solves the open problem of Lin [6]. The main idea is that backbones can be used to split an instance into two independent subinstances. For infinite leaders faster algorithms are possible since each backbone generates two independent instances; for finite 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. If crossings are allowed, we give an efficient algorithm for crossing minimization with fixed label order and show NP-completeness for flexible label order (Section 5).

## 2 Problem Definition

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

All points assigned to the same label will be connected to it 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. Since only a single backbone attaches to a label, we can use the terms *label* and *backbone* interchangeably. In addition, 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 finite 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  $l$ , its exact position 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*  $\mathbf{k} = (k_1, \dots, k_{|C|})$ . 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 referred to as *crossing-free*. An interesting variation of the backbone labeling concerns the size of the backbone. A *finite backbone* attached to a label at, say, the right side of  $R$  extends up to the leftmost point that is assigned to it. An *infinite backbone* spans the whole width of  $R$ . Note that,

in the case of crossing-free labelings, infinite backbones may result in labelings with a larger number of labels and increased total leader length.

In the rest of the paper, we denote the points of  $P$  as  $\{p_1, p_2, \dots, p_n\}$  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.

### 3 Minimizing the Total Number of Labels

In this section we minimize the total number of labels in a crossing-free solution, i.e., we set  $K = n$  so that there is effectively no upper bound on the number of labels.

**Infinite Backbones.** In a crossing-free solution any infinite backbone splits the drawing into two independent subinstances above and below the backbone. Clearly, a backbone traversing a point has to be of the same color. On the other hand, we can check that a backbone lying between two points  $p_i$  and  $p_{i+1}$  can only have color  $c(p_i), c(p_{i+1})$ , or the color of the next point of distinct color above  $p_i$  or of the one below  $p_{i+1}$ ; see Lemma 2 in Appendix A.

We sketch how to find an optimum solution in  $O(n)$  time; a detailed description is found in Appendix A. First, we replace any maximal set of identically colored consecutive points by the topmost point in the set as representative. The optimum solutions of the reduced instance, in which two consecutive points never have the same color, are equivalent to optimum solutions of the original instance; see Lemma 3 in Appendix A.

We can solve the reduced instance using dynamic programming. We store a table  $nl$  of values  $nl(i, cur, c_{bak}, c_{free})$  representing the minimum number of backbones needed above or at point  $p_i$  such that the lowest backbone is  $c_{bak}$ -colored, the lowest backbone goes through  $p_i$  if the flag  $cur = true$  and lies above  $p_i$  otherwise, and the (single) point between  $p_i$  and the lowest backbone (if  $cur = false$ ) has color  $c_{free}$ .

By careful case analysis, we can see that any entry of the table can be computed in constant time. (Details in Appendix A)

**Theorem 1.** *Let  $P = \{p_1, p_2, \dots, p_n\}$  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 infinite backbones can be computed in  $O(n)$  time.*

**Finite Backbones.** We now consider finite 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 and gap  $n$  is below all the points, respectively.

Suppose a point  $p_l$  lies between a backbone of color  $c$  in gap  $g$  and a backbone of color  $c'$  in gap  $g'$  with  $0 \leq g < l \leq g' \leq n$  such that both backbones horizontally extend to at least the  $x$ -coordinate of  $p_l$ ; see Fig. 2. Suppose all points except the ones in the rectangle  $R(g, g', l)$ , 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  $R$  cannot reuse any backbone except for the two backbones in gaps  $g$  and  $g'$ ; hence, it is independent of the rest of the solution.

We use this observation for solving the problem by a dynamic program. For  $0 \leq g \leq g' \leq n, l \in \{g, \dots, g'\} \cup \{\emptyset\}$ , and two colors  $c$  and  $c'$  let  $T[g, c, g', c', l]$  be the minimum number of additional labels that are needed for labeling all points in the rectangle  $R(g, g', l)$  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 they both extend to the left of  $p_l$ . Note that for  $l = \emptyset$  the rectangle is empty and  $T[g, c, g', c', \emptyset] = 0$ .

We distinguish cases based on the connection of  $p_l$ . First, if  $c(p_l) = c$  or  $c(p_l) = c'$ , 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,  $T[g, c, g', c', l] = T[g, c, g', c', \text{left}(g, g', l)]$  where  $\text{left}(g, g', l)$  is the index of the leftmost point in the interior of  $R(g, g', l)$  or  $\text{left}(g, g', l) = \emptyset$  if no such point exists.

Otherwise suppose  $c(p_l) \notin \{c, c'\}$ . For connecting  $p_l$  we need to place a new backbone of color  $c(p_l)$ , which is possible in any gap  $\tilde{g}$  with  $g \leq \tilde{g} \leq g'$ . The backbone splits the instance into two parts, between gaps  $g$  and  $\tilde{g}$  and between gaps  $\tilde{g}$  and  $g'$ . Hence, we obtain the recursion  $T[g, c, g', c', l] = 1 + \min_{g \leq \tilde{g} \leq g'} (T[g, c, \tilde{g}, c(p_l), \text{left}(g, \tilde{g}, l)] + T[\tilde{g}, c(p_l), g', c', \text{left}(\tilde{g}, g', l)])$ .

Finally, let  $\bar{c} \notin C$  be a dummy color, and let  $p_{\bar{l}}$  be the leftmost point. Then the value  $T[0, \bar{c}, n, \bar{c}, \bar{l}]$  is 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. All  $\text{left}(\cdot, \cdot, \cdot)$ -values can easily be precomputed in  $O(n^3)$  total time.

**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 finite backbones in  $O(n^4|C|^2)$  time.*

## 4 Length Minimization

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 color vector  $\mathbf{k}$  of individual bounds per color.

**Infinite Backbones.** We use a parameter  $\lambda$  to distinguish the two minimization goals, i.e., we set  $\lambda = 0$ , if we want to minimize only the sum of the length of all vertical segments and we set  $\lambda$  to be the width of the rectangle  $R$  if we also take the length of the backbones into account. In this section, we assume that  $p_1 > \dots > p_n$  are the y-coordinates of the input points.

*Single Color.* If all points have the same color, 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. Hence, we must solve the following problem: Given  $n$  points with y-coordinates

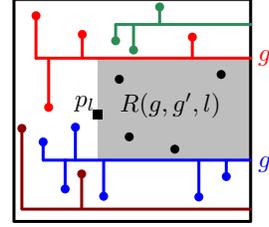


Fig. 2: A partial instance bounded by two backbones and the leftmost point  $p_l$ .

$p_1 > \dots > p_n$ , find a set  $S$  of at most  $K$   $y$ -coordinates that minimizes

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

Note that we can optimize the value in Eq. (1) by choosing  $S \subseteq \{p_1, \dots, p_n\}$ . 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 [3]. Note that the problem corresponds to the  $K$ -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. However, by Lemma 2, it suffices to add between any pair of vertically consecutive points two additional candidates for backbone positions, plus one additional candidate above all points and one below all points. Hence, we have a set of  $3n$  candidate lines at  $y$ -coordinates

$$p_1^- > p_1 > p_1^+ > p_2^- > p_2 > p_2^+ > \dots > p_n^- > p_n > p_n^+ \quad (2)$$

where for each  $i$  the values  $p_i^-$  and  $p_i^+$  are as close to  $p_i$  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  $p_i$ . 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  $i$ th point in Eq. (2) by  $y_i$  and its color by  $c(y_i)$ .

We solve the problem using dynamic programming. For each  $i = 1, \dots, 3n$ , and for each vector  $\mathbf{k}' = (k'_1, \dots, k'_{|C|})$  with  $k'_1 \leq k_1, \dots, k'_{|C|} \leq k_{|C|}$ , let  $L(i, \mathbf{k}')$  denote the minimum length of a feasible backbone labeling of  $p_1, \dots, p_{\lfloor \frac{i+1}{3} \rfloor}$  using  $k'_c$  infinite backbones of color  $c$  for  $c = 1, \dots, |C|$  such that the bottommost backbone is at position  $y_i$ , if such a labeling exists. Otherwise  $L(i, \mathbf{k}') = \infty$ . In Appendix B.1, we describe, how the values  $L(i, \mathbf{k}')$  can be computed recursively in  $O(n^2 \prod_{i=1}^{|C|} k_i)$  time in total.

Let  $S$  be the set of candidates  $y_i$  such that all points below  $y_i$  have the same color as  $y_i$ . Then we can compute the minimum total length of a backbone labeling of  $p_1, \dots, p_n$  with at most  $k_c, c = 1, \dots, |C|$  labels per color  $c$  as

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

If we globally bound the total number of labels by  $K$ , we obtain a similar dynamic program with the corresponding values  $L(i, k), i = 1, \dots, 3n, k < K$ . We summarize the results in the following theorem.

**Theorem 3.** *A minimum length backbone labeling with infinite backbones for  $n$  points with  $|C|$  colors can be computed in  $O(n^2 \cdot \prod_{i=1}^{|C|} k_i)$  time if at most  $k_i$  labels are allowed for color  $i, i = 1, \dots, |C|$  and in  $O(n^2 \cdot K)$  time if in total at most  $K$  labels are allowed.*

**Finite Backbones.** We sketch how to modify the dynamic program for minimizing the total number of labels (in Section 3) to minimize the total leader length and refer to Appendix B.2 for details.

We change the meaning of an entry in the table  $T$  to denote the additional length of segments and backbones needed for labeling the points of the subinstance. Moreover, the precise positions of backbones matter for length minimization. A clear candidate set is the y-coordinates of input points which may be used by a backbone of the same color. We can also identify candidates for backbones inside a gap between points  $p_i$  and  $p_{i+1}$ . We observe that the longest backbone  $b^*$  inside the gap splits all other backbones lying between  $p_i$  and  $p_{i+1}$ ; see Fig. 8 in Appendix B.2. The backbones above  $b^*$  connect only to points above and, hence, must be placed as close to  $p_i$  as possible for length minimization. Symmetrically, the backbones below  $b^*$  connect only to the bottom and must be placed as close to  $p_{i+1}$  as possible.

For avoiding overlaps and to accommodate labels with fixed heights, we enforce a minimum distance  $\Delta > 0$  between pairs of backbones, as well as backbones and differently colored points. Then, for the labels close to  $p_i$ , we get a sequence of consecutive candidate positions with distance  $\Delta$  below  $p_i$ ; see Fig. 9 and 10 in Appendix B.2. Symmetrically, there is such a sequence above  $p_{i+1}$ . Any such sequence contains up to  $n$  points (less if the gap is too small). Note that the two sequences might overlap; we can, however, easily ensure that no two backbones with distance less than  $\Delta$  are used in the dynamic program. To address entries in  $T$  we use the  $O(n^2)$  candidate positions (input points and positions in gaps) instead of the gaps; no position can be used twice.

As a final step, we integrate the global value  $K$  or the color vector  $\mathbf{k}$  as a bound on the allowed numbers of labels. To this end, we add additional dimensions for  $K$  or for  $k_c, c \in C$  to the table that specify the remaining available numbers of labels in the subinstance.

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

## 5 Crossing Minimization

In this section we allow crossings between backbone leaders, which generally allows us to use fewer labels. We concentrate on minimizing the number of crossings for the case  $K = |C|$ , i.e., one label per color, and distinguish fixed and flexible label orders.

### 5.1 Fixed y-Order of Labels

In this part we assume that the color set  $C$  is ordered and we require that for each pair of colors  $i < j$  the label (and backbone) in color  $i$  is above the label in color  $j$ .

**Infinite Backbones.** 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.

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  $\text{cross}(i, g)$  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 pre-compute table  $\text{cross}$ , using dynamic programming, as follows. All table entries of the form  $\text{cross}(\cdot, 0)$  can be clearly computed in  $O(n)$  time. Then,  $\text{cross}(i, g) = \text{cross}(i, g - 1) + 1$ , if the point between gaps  $g - 1$  and  $g$  has color  $j$  and  $j > i$ . In the case where the point between gaps  $g - 1$  and  $g$  has color  $j$  and  $j < i$ ,  $\text{cross}(i, g) = \text{cross}(i, g - 1) - 1$ . If it has color  $i$ , then  $\text{cross}(i, g) = \text{cross}(i, g - 1)$ . From the above, it follows that the computation of table  $\text{cross}$  takes  $O(n|C|)$  time.

Now, we use another dynamic program to compute the minimum number of crossings. Let  $T[i, g]$  denote the minimum number of crossings on the backbones  $1, \dots, i$  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  $g = 0, \dots, n$ . Moreover, we have  $T[i, g] = \min_{g' \leq g} T[i - 1, g'] + \text{cross}(i, g)$ . Having pre-computed table  $\text{cross}$  and assuming that for each entry  $T[i, g]$ , we also store the smallest entry of row  $T[i, \cdot]$  to the left of  $g$ , 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  $\min_g T[|C|, g]$ . A corresponding solution can be found by backtracking in the dynamic program.

**Theorem 5.** *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, infinite backbones, and minimum number of crossings can be computed in  $O(n|C|)$  time.*

**Finite Backbones.** We can easily modify the approach used for infinite backbones to minimize the number of crossings for finite backbones, if the y-order of labels is fixed; see Theorem 8 in Appendix C.1. We just have to take into account that a finite backbone can only cross vertical segments that lie within its horizontal range when computing the table  $\text{cross}$ . Hence, we can again minimize crossings in  $O(n|C|)$  time.

## 5.2 Flexible y-Order of Labels

In this part the order of labels is no longer given and we need to minimize the number of crossings over all label orders. While there is an efficient algorithm for infinite backbones, the problem is NP-complete for finite backbones.

**Infinite Backbones.** We give an efficient algorithm for the case that there are  $K = |C|$  fixed label positions  $y_1, \dots, y_K$  on the right boundary of  $R$ , e.g., uniformly distributed.

First observe that if the backbone of color  $k$ ,  $1 \leq k \leq |C|$  is placed at position  $y_i$ ,  $1 \leq i \leq |C|$ , then the number of crossings created by the vertical segments leading to this backbone is fixed, since all label positions will be occupied by an infinite backbone. This crossing number  $\text{cr}(k, i)$  can be determined in  $O(n_k)$  time, where  $n_k$  is the number of points of color  $k$ . By a sweep from top to bottom, we can even determine all crossing numbers  $\text{cr}(k, \cdot)$  for backbone  $k$ ,  $1 \leq k \leq |C|$  in time  $O(n_k + |C|)$ . Now, we construct an instance of a weighted bipartite matching problem, where for each position  $y_i$ ,  $1 \leq i \leq |C|$  and each backbone  $k$ ,  $1 \leq k \leq |C|$ , we establish an edge  $\{k, i\}$  of

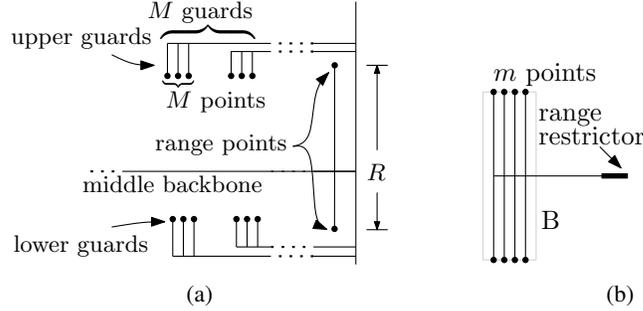


Fig. 3: The range restrictor gadget (a), and a blocker gadget (b).

weight  $cr(k, i)$ . In total, this takes  $O(n + |C|^2)$  time. The minimum-cost weighted bipartite matching problem can be computed in time  $O(|C|^3)$  with the Hungarian method [5] and yields a backbone labeling with the minimal number of crossings.

**Theorem 6.** *Given a set  $P$  of  $n$  colored points, a color set  $C$ , and a set of  $|C|$  fixed label positions, we can compute in  $O(n + |C|^3)$  time a feasible backbone labeling with infinite backbones that minimizes the number of crossings.*

**Finite Backbones.** Next, we consider the variant with finite backbones and prove that it is NP-hard to minimize the number of crossings. 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 sketch their properties, before describing the hardness reduction. Some details of the descriptions and proofs are deferred to Appendix C.2.

Our first step is the construction of a *range restrictor gadget* that allows a certain backbone to be placed only in a specific interval of  $y$ -coordinates, i.e., unless it creates many crossings. We describe the properties of the construction illustrated in Fig. 3a; Appendix C.2 contains a more detailed description. The range restrictor consists of a middle backbone, two range points that span the vertical *range* of the gadget, and a set of  $M$  upper and  $M$  lower guards for some integer  $M > 1$ , which is called the *size* of the guards. The next lemma shows the properties of the range restrictor.

**Lemma 1.** *The backbones of the range restrictor 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.*

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 bottom side of  $B$ , respectively. Moreover, a range restrictor gadget is used to restrict the backbone of the blocker to the range  $R$ . Figure 3b 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-hard.

**Theorem 7.** *Given a set of  $n$  input points in  $k$  different colors and an integer  $Y$  it is NP-complete to decide whether a backbone labeling with one label per color and flexible  $y$ -order of the labels that has at most  $Y$  leader crossings exists.*

*Proof.* The proof is by reduction from the NP-complete Fixed Linear Crossing Number problem [8], which is defined as follows. Given a graph  $G = (V, E)$ , a bijective function  $f: V \rightarrow \{1, \dots, |V|\}$ , and an integer  $Z$ , is there 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

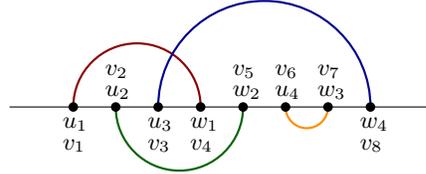


Fig. 4: An input instance with four edges.

above or below the spine so that there are at most  $Z$  edge crossings? Masuda et al. [8] showed that the problem remains 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  $V = \{v_1, \dots, v_{2n}\}$  are indexed in the order specified by  $f$ , i.e.,  $f(v_i) = i$  for all  $i$ . Furthermore, we direct every edge  $\{v_i, v_j\}$  with  $i < j$  from  $v_i$  to  $v_j$ . Let  $\{u_1, \dots, u_n\}$  be the ordered source vertices and let  $\{w_1, \dots, w_n\}$  be the ordered sink vertices. Figure 4 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 in  $G$ . 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 backbone. The edge gadgets are ordered from left to right according to the sequence of source vertices  $(u_1, \dots, u_n)$ . Figure 5 shows a sketch of the instance  $I_G$  created for a matching  $G$  with four edges.

The edge gadgets are placed symmetrically with respect to the  $x$ -axis. We create  $2n+1$  special rows above the  $x$ -axis and  $2n+1$  special rows below, 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 Fig. 5. The upper and lower blockers extend vertically to rows  $2n+1$  and  $-2n-1$ . 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  $2n+1$  ( $-2n$  and  $-2n-1$ ) for the upper (resp. lower) blocker, and between rows  $j$  and  $j+1$  ( $-j$  and  $-j-1$ ) for the upper (resp. lower) gap blocker.

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 that is supposed to pass through one of the

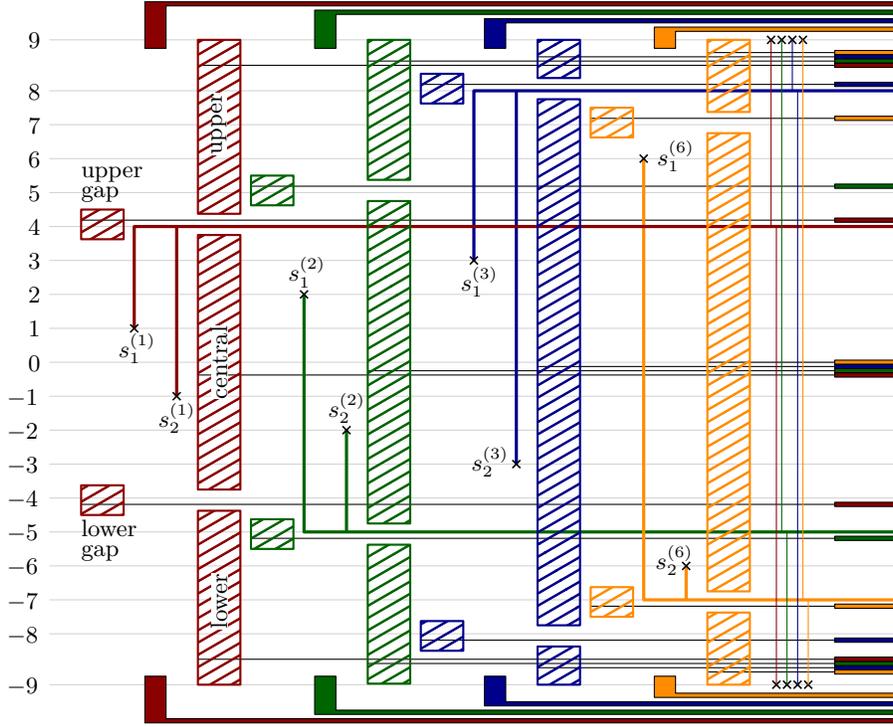


Fig. 5: Sketch of the reduction of the graph of Fig. 4 into a backbone labeling instance. Hatched rectangles represent blockers, thick segments represent side selectors, and filled shapes represent guard gadgets or range restrictor gadgets.

two gaps in an optimal solution. The  $n$  edge gadgets are placed from left to right in the order of their source vertices, see Fig. 5 for an example.

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 1.

In Appendix C.2 we show two claims that are central to our reduction.

*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  $-j$ .

Next, we show how the number of crossings in the backbone labeling instance relates to the number of crossings in the Fixed Linear Crossing Number problem. There is a large number of unavoidable crossings regardless of the backbone positions of the selector gadgets. By Claim 1 and the fact that violating any range restriction immediately causes  $M$  crossings, we can assume that every backbone adheres to the rules, i.e.,

stays within its range as defined by the range restriction gadgets or passes through one of its two gaps.

*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 problem.

The proof of this claim is based on carefully counting crossings in four different cases, depending on which types of backbones and vertical segments intersect. It turns out that almost all crossings are fixed (yielding the number  $X$ ), except for those of selector backbones with vertical selector segments for which the two underlying edges  $(v_i, v_j)$  and  $(v_k, v_l)$  with  $i < k$  are *interlaced*, i.e.,  $i < k < j < l$  holds. Each pair of interlaced edges creates two crossings in the reduction if they are assigned to the same side of row 0 and no crossing otherwise (see for example edges  $(v_1, v_4)$  and  $(v_3, v_8)$  or  $(v_2, v_5)$  in Fig. 5). This adds up to at least  $2Z$  crossings, which shows the claim and concludes the hardness reduction. Furthermore, we can guess an order of the backbones and then apply Theorem 8 to compute the minimum crossing number for this order. This shows NP-completeness.  $\square$

*Acknowledgements.* This work was started at the Bertinoro Workshop on Graph Drawing 2013. M.N. received financial support by the Concept for the Future of KIT.

## References

1. 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.
2. 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.
3. R. Hassin and A. Tamir. Improved complexity bounds for location problems on the real line. *Operations Research Letters*, 10(7):395 – 402, 1991.
4. 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-Verlag, 2009.
5. H. W. Kuhn. The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1–2):83–97, 1955.
6. 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.
7. C.-C. Lin, H.-J. Kao, and H.-C. Yen. Many-to-one boundary labeling. *J. Graph Algorithms Appl.*, 12(3):319–356, 2008.
8. S. Masuda, K. Nakajima, T. Kashiwabara, and T. Fujisawa. Crossing minimization in linear embeddings of graphs. *IEEE Trans. Computers*, 39(1):124–127, 1990.
9. 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.
10. A. Wolff and T. Strijk. The Map-Labeling Bibliography (1996). On line: <http://i11www.ira.uka.de/map-labeling/bibliography>.

# Appendix

## A Minimizing the Total Number of Infinite Backbones

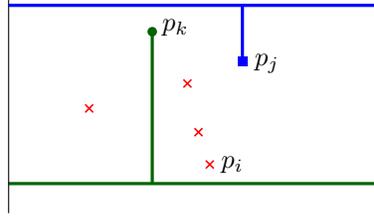


Fig. 6: Point  $p_i$  cannot be labeled.

We start with an important observation on the structure of crossing-free labelings with infinite backbones.

**Lemma 2.** *Let  $p_i, p_{i+1}$  be two points that are vertically consecutive. Let  $p_j$  ( $j < i$ ) be the first point above  $p_i$  with  $c(p_j) \neq c(p_i)$ , and let  $p_{j'}$  ( $j' > i + 1$ ) 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 2 backbones. Furthermore, any separating backbone has color  $c(p_i), c(p_{i+1}), c(p_j)$ , or  $c(p_{j'})$ .*

*Proof.* Suppose there are three separating backbones. Then the middle one could not be connected to any point. Now, suppose 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  $p_k$ . Point  $p_i$  is lying between two backbones of other colors; see Fig. 6. Its own backbone cannot be placed there without crossing a vertical segment connecting  $p_k$  or  $p_j$  to their corresponding backbone. 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  $c(p_{j'})$ .  $\square$

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. However, if a third color is involved, then many labels may be required.

We denote by  $NL(P)$  the number of labels of an optimal crossing-free solution of  $P$ . In the general case of the problem,  $P$  may contain several consecutive points of the same color. We proceed to construct a new instance  $C(P)$  based on instance  $P$ , in which no two consecutive points are of the same color. To do so, we identify each maximal set of same-colored consecutive points of  $P$  and we replace them by a single point of the

same color that lies in the position of the topmost point of this set. Note that in order to achieve this, a simple top-to-bottom sweep is enough. Let  $C(P) = \{p'_1, p'_2, \dots, p'_k\}$  be the *clustered point set*, that we just constructed. For the sake of simplicity, we assume that  $f: P \rightarrow C(P)$  is a function which computes the representative in  $C(P)$  for a given a point of  $P$ , and, with a slight abuse of notation let  $f^{-1}: P \rightarrow \mathcal{P}(C(P))$  be its “inverse function”.

**Lemma 3.** *The number of labels needed in an optimal crossing-free labeling of  $P$  with infinite backbones is equal to the number of labels needed in an optimal crossing-free solution of  $C(P)$ , i.e.,  $NL(P) = NL(C(P))$ .*

*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 the optimal solution of  $C(P)$ , then obviously  $NL(P) \leq NL(C(P))$ .

Let  $p'_i, i = 1, 2, \dots, k$ , be an arbitrary point of  $C(P)$  and let  $\{p_j, p_{j+1}, \dots, p_{j+m}\}$  be the maximal set of consecutive, same-colored points of  $P$  that has  $p'_i$  as its representative in  $C(P)$ , i.e.,  $f^{-1}(p'_i) = \{p_j, p_{j+1}, \dots, p_{j+m}\}$ . Let  $S(p'_i)$  be the horizontal strip that is defined by the two horizontal lines through  $p_j$  and  $p_{j+m}$ , respectively. Clearly, in a legal solution of  $P$ ,  $S(p'_i)$  can accommodate at most one backbone, i.e., the one of  $\{p_j, p_{j+1}, \dots, p_{j+m}\}$ , as we look for crossing-free solutions. Now, observe that  $S(p'_1), S(p'_2), \dots, S(p'_k)$  do not overlap with each other, since we have assumed that our point set  $P$  is in general position, and subsequently, all maximal sets of consecutive, same-colored points of  $P$  are well separated. We proceed to derive a first solution  $S(P)$  of  $P$  from  $S(C(P))$  as follows: We connect each point  $p_i$  to the backbone of its representative  $f(p_i)$  in  $S(C(P))$ . Clearly,  $S(P)$  is not necessarily crossing-free. However, all potential crossings should appear in horizontal strips  $S(p'_1), S(p'_2), \dots, S(p'_k)$ ; otherwise  $S(C(P))$  is not crossing-free as well. Let  $S(p'_i), i = 1, 2, \dots, k$ , be a horizontal strip which contains crossings. As already stated,  $S(p'_i)$  can accommodate at most one backbone, i.e., the one of  $\{p_j, p_{j+1}, \dots, p_{j+m}\}$ . We proceed to move all backbones in  $S(p'_i)$  that are above (below, resp.) the one of  $\{p_j, p_{j+1}, \dots, p_{j+m}\}$  on to the top of (below, resp.)  $S(p'_i)$ , without changing their relative order and without influencing the strips above and below  $S(p'_i)$  (recall that  $S(p'_1), S(p'_2), \dots, S(p'_k)$  do not overlap with each other, which suggests that this is always possible). From the above it follows that the constructed solution is crossing-free and has the same number of labels as the one of  $C(P)$ , which completes the proof of this lemma.  $\square$

We now present a linear-time algorithm for minimizing the number of infinite backbones.

**Theorem 1.** *Let  $P = \{p_1, p_2, \dots, p_n\}$  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 infinite 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 aid of Lemma 3. For  $i = 1, 2, \dots, n$ ,  $\{c_{\text{bak}}, c_{\text{free}}\} \subseteq C$ , and,  $\text{cur} \in \{\text{true}, \text{false}\}$ , 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:

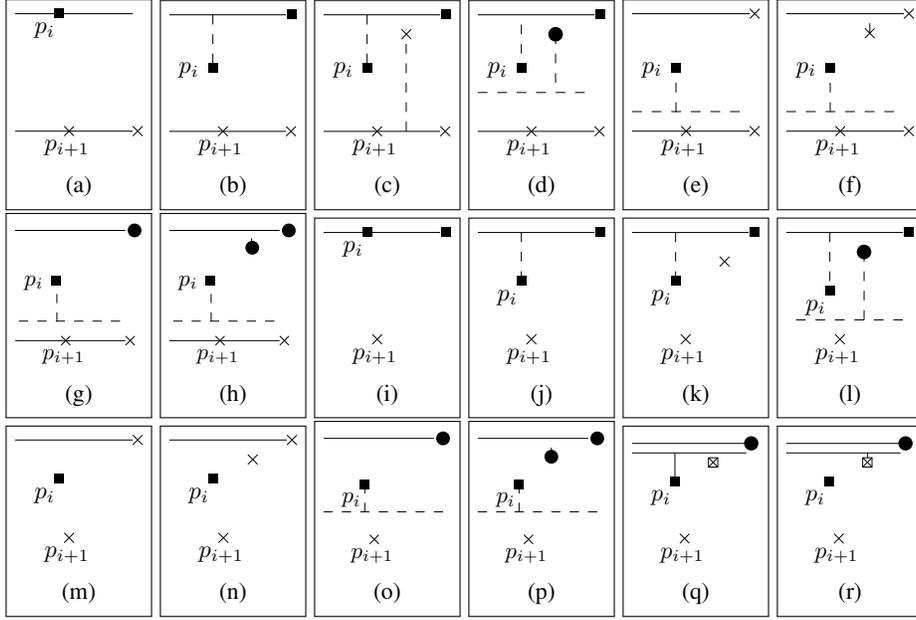


Fig. 7: Different configurations that arise in proof of Theorem 1.

- The lowest backbone has color  $c_{\text{bak}}$ .
- If  $\text{cur}$  is true, the lowest backbone coincides with  $p_i$ . Hence, it is  $c(p_i)$ -colored, i.e.,  $c_{\text{bak}} = c(p_i)$ ; otherwise the lowest backbone is above  $p_i$ . Note that in the latter case  $p_i$  might be unlabeled (e.g., the color of the lowest backbone is not  $c(p_i)$ ), or equivalently,  $c_{\text{bak}} \neq c(p_i)$ .
- The point that, by Lemma 2, may exist between  $p_i$  and the lowest backbone has color  $c_{\text{free}}$ . Obviously, in the case where  $\text{cur} = \text{true}$  (i.e., the lowest backbone coincides with  $p_i$ ) this point does not exist. So, in general, if this point does not exist, we assume that  $c_{\text{free}} = \emptyset$ .

Obviously,  $nl(1, \text{true}, c(p_1), \emptyset) = 1$  and  $nl(1, \text{false}, \emptyset, c(p_1)) = 0$ . Now assume that we have computed all entries of table  $nl$  that correspond to different labelings induced by 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,  $\text{cur} = \text{true}$  and obviously there is no unlabeled point between the backbone through  $p_{i+1}$  and  $p_{i+1}$ , i.e.,  $c_{\text{free}} = \emptyset$ . Hence, we need to compute entry  $nl(i+1, \text{true}, c(p_{i+1}), \emptyset)$ . To do so, we distinguish the following subcases with respect to the color of the lowest backbone, say  $bb$ , above or at point  $p_i$ :
  - 1.1  *$bb$  is above or at point  $p_i$  and  $c(p_i)$ -colored.* If  $bb$  is at point  $p_i$  (see Fig. 7a), then trivially there is no unlabeled point below it. Hence, a feasible solution can be derived from  $nl(i, \text{true}, c(p_i), \emptyset)$  by adding a new backbone, i.e., the

one incident to  $p_{i+1}$ . If  $bb$  is above point  $p_i$ , then we distinguish two subcases. (a) If there is no unlabeled point below it (see Fig. 7b), a feasible solution can be derived from  $nl(i, false, c(p_i), \emptyset)$  again by adding a new backbone, i.e., the one incident to  $p_{i+1}$ . (b) On the other hand, if there is an unlabeled point below  $bb$ , then again we need to distinguish two subcases. (b.1) If the unlabeled point is colored  $c(p_{i+1})$  (see Fig. 7c), then a single backbone, i.e., the one incident to  $p_{i+1}$  is enough. The corresponding solution is derived from  $nl(i, false, c(p_i), c(p_{i+1}))$ . (b.2) However, in the case where the unlabeled point is colored  $c$  and  $c \notin \{c(p_i), c(p_{i+1})\}$  (see Fig. 7d), two backbones are required and the corresponding feasible solution is derived from  $nl(i, false, c(p_i), c), c \notin \{c(p_i), c(p_{i+1})\}$ . Note that the case where the unlabeled point below  $bb$  is of color  $c(p_i)$  cannot occur, since we have assumed that consecutive points are not of the same color.

- 1.2 *bb is above  $p_i$  and  $c(p_{i+1})$ -colored.* Again, we distinguish two subcases. (a) If there is no unlabeled point below it (see Fig. 7e), then a feasible solution can be derived from  $nl(i, false, c(p_{i+1}), \emptyset)$  by adding two new backbones, i.e., the one incident to  $p_i$  and the one incident to  $p_{i+1}$ . (b) If there is an unlabeled point below it (see Fig. 7f), then its color should be  $c(p_{i+1})$ . If this is not the case, it is easy to see that the backbone above  $p_i$  is not  $c(p_{i+1})$ -colored. Again two backbones are required, i.e., the one incident to  $p_i$  and the one incident to  $p_{i+1}$ . The corresponding solution is derived from  $nl(i, false, c(p_{i+1}), c(p_{i+1}))$ .
- 1.3 *bb 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  $bb$  (see Fig. 7g) or there is one which is  $c$ -colored (see Fig. 7h). 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  $nl(i, false, c, \emptyset), c \notin \{c(p_i), c(p_{i+1})\}$ , while in the latter it is derived from  $nl(i, false, c, c), c \notin \{c(p_i), c(p_{i+1})\}$ .

From the above cases, it follows:

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

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

- 2.1 *bb is above or at point  $p_i$  and  $c(p_i)$ -colored.* If  $bb$  is at point  $p_i$  (see Fig. 7i) or  $bb$  is above point  $p_i$  and either there is no unlabeled point below it (see Fig. 7j) or the unlabeled point below it is colored  $c(p_{i+1})$  (see Fig. 7k), then no backbone is required. Then, the corresponding feasible solutions are as follows:

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

$$nl(i + 1, true, c(p_i), c(p_{i+1})) = nl(i, false, c(p_i), c(p_{i+1}))$$

However, in the case where the unlabeled point below  $bb$  is  $c$ -colored, where  $c \neq c(p_i)$  and  $c \neq c(p_{i+1})$ , then a backbone is required (see Fig. 7l). Hence, the corresponding feasible solution can be derived as follows:

$$nl(i + 1, true, c(p_i), c) = nl(i, false, c(p_i), c) + 1, c \notin \{c(p_i), c(p_{i+1})\}$$

2.2  $bb$  is above  $p_i$  and  $c(p_{i+1})$ -colored. In this case, either there is no unlabeled point below it (see Fig. 7m) or there is one which is  $c(p_{i+1})$ -colored (see Fig. 7n). In both cases no backbone is required. Hence, the corresponding feasible solutions can be derived as follows:

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

$$nl(i + 1, false, c(p_{i+1}), c(p_{i+1})) = nl(i, false, c(p_{i+1}), c(p_{i+1}))$$

2.3  $bb$  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 it (see Fig. 7o) or there is one which is  $c$ -colored (see Fig. 7p), then one backbone is required for  $p_i$ . The corresponding feasible solution can be derived as follows:

$$nl(i + 1, false, c(p_i), \emptyset) = \min\{nl(i, false, c, \emptyset), \\ nl(i, false, c, c)\} + 1, c \notin \{c(p_i), c(p_{i+1})\}$$

The most interesting case of our case analysis arises when a forth color is involved, say  $c' \notin \{c(p_i), c(p_{i+1}), c\}$ . In this case, either the  $c'$ -colored point remains unlabeled and  $p_i$  is labeled (see Fig. 7q), or, the  $c'$ -colored point is labeled and  $p_i$  remains unlabeled (see Fig. 7r). The corresponding feasible solutions can be described as follows.

$$nl(i + 1, false, c(p_i), c') \\ = nl(i, false, c, c') + 1, c \notin \{c(p_i), c(p_{i+1})\}, c' \notin \{c(p_i), c(p_{i+1}), c\}$$

$$nl(i + 1, false, c', c(p_i)) \\ = nl(i, false, c, c') + 1, c \notin \{c(p_i), c(p_{i+1})\}, c' \notin \{c(p_i), c(p_{i+1}), c\}$$

Having computed table  $nl$ , the number of labels of the optimal solution of  $P$  equals to the minimum entry of the form  $nl(n, false, \cdot, \emptyset)$ . Since the algorithm maintains an  $n \times 2 \times |C| \times |C|$  table and each entry is computed in constant time, the time complexity of our algorithm is  $O(n|C|^2)$ . However, with the aid of Lemma 3, it can be reduced to  $O(n)$  (since there is a constant number of colors that “surround” each point). A corresponding solution can be found by backtracking in the dynamic program.  $\square$

## B Length Minimization

### B.1 Length Minimization with Infinite Backbones

For each candidate  $y_i, i = 1, \dots, 3n$ , and for each vector  $\mathbf{k}' = (k'_1, \dots, k'_{|C|})$  with  $k'_1 \leq k_1, \dots, k'_{|C|} \leq k_{|C|}$ , if there is a feasible backbone labeling of  $p_1, \dots, p_{\lfloor \frac{i+1}{3} \rfloor}$  using  $k'_c$  infinite backbones of color  $c$  for  $c = 1, \dots, |C|$  such that the bottommost backbone is at position  $y_i$  we defined  $L(i, \mathbf{k}')$  to be the minimum length of such a labeling. Otherwise  $L(i, \mathbf{k}') = \infty$ .

We now show how to recursively compute  $L(i, \mathbf{k}')$ . Assume that we want to place a new backbone at  $y_i$  and that the previous backbone was at  $y_j, j < i$ . Then we have to connect the points  $p_x, (j+2)/3 \leq x \leq i/3$  between  $y_i$  and  $y_j$  to one of the backbones through  $y_i$  or  $y_j$ . Let  $\text{LINK}(j, i)$  denote the minimum total length of the vertical segments linking these points to their respective backbone. If there is a point  $p_x$  between  $y_i$  and  $y_j$  with  $c(p_x) \notin \{c(y_i), c(y_j)\}$ , we set  $\text{LINK}(j, i) = \infty$ . Otherwise, we have

$$\text{LINK}(j, i) = \begin{cases} \sum_{\frac{j+2}{3} \leq x \leq \frac{i}{3}} \min\{y_j - y_x, y_x - y_i\} & \text{if } c(y_i) = c(y_j) \\ \sum_{\substack{\frac{j+2}{3} \leq x \leq \frac{i}{3} \\ c(p_x) = c(y_j)}} (y_j - y_x) + \sum_{\substack{\frac{j+2}{3} \leq x \leq \frac{i}{3} \\ c(p_x) = c(y_i)}} (y_x - y_i) & \text{if } c(y_i) \neq c(y_j) \end{cases} \quad (3)$$

With the base cases  $L(i, 0, \dots, 0, k'_i = 1, 0, \dots, 0) = \sum_{0 < x \leq i/3} (y_i - p_x)$  if all points above  $y_i$  have the same color as  $y_i$  and  $\infty$  otherwise, as well as  $L(i, \mathbf{k}') = \infty$  if  $k'_i = 0$ , we get the following recursion

$$L(i, k'_1, \dots, k'_{|C|}) = \lambda + \min_{j \leq i} \left( L(j, k'_1, \dots, k'_{c(y_i)} - 1, \dots, k'_{|C|}) + \text{LINK}(j, i) \right). \quad (4)$$

It remains to show that for a fixed index  $i$ , all values  $\text{LINK}(j, i), j < i$  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 in decreasing order of their indices until we find the first point that is neither colored  $c'$  nor  $c(y_i)$ . For each  $j$ , let  $P'_i(j) = \{p_x; \frac{j+2}{3} \leq x \leq \frac{i}{3}, c(p_x) = c'\}$ ,  $P_i(j) = \{p_x; \frac{j+2}{3} \leq x \leq \frac{i}{3}, c(p_x) = c(y_i)\}$ ,  $P_i^{\text{up}}(j) = \{p_x; \frac{j+2}{3} \leq x \leq \frac{i}{3}, c(p_x) = c(y_i), y_j - p_x \leq p_x - y_i\}$ . In each step, we can update in constant time the values  $|P'_i(j)|$ ,  $\text{SECONDLLENGTH}(j) = \sum_{p \in P'_i(j)} (y_j - p) = \text{SECONDLLENGTH}(j-1) + |P'_i(j)| \cdot (y_j - y_{j-1})$ , and  $\text{FIRSTLENGTH}(j) = \sum_{p \in P_i(j)} (p - y_i)$ . Further, we can update in amortized constant time the index of the lowest element in  $P_i^{\text{up}}(j)$  as well as the values  $|P_i^{\text{up}}(j)|$ ,  $\text{FIRSTDOWNLENGTH}(j) = \sum_{p \in P_i(j) \setminus P_i^{\text{up}}(j)} (p - y_i)$ , and  $\text{FIRSTUPLLENGTH}(j) = \sum_{p \in P_i^{\text{up}}(j)} (y_j - p)$ . With these values we can directly compute  $\text{LINK}(j, i)$  according to the case distinction in Eq. 3. Hence, we can compute a value  $L(i, k'_1, \dots, k'_{|C|})$  according to the recursion in Eq. (4) in  $O(n)$  time. Therefore, all values  $L(i, \mathbf{k}')$  can be computed in  $\mathcal{O}(n^2 \prod_{i=1}^{|C|} k_i)$  time in total.

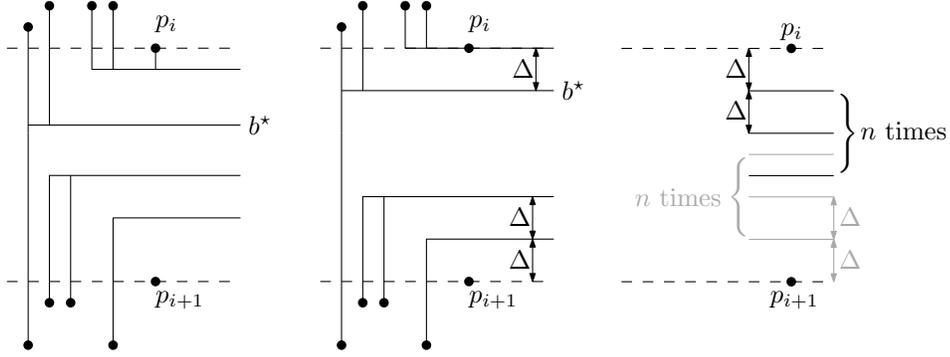


Fig. 8: Longest backbone  $b^*$  splitting the backbones between  $p_i$  and  $p_{i+1}$ . Fig. 9: Backbones placed with the minimum leader length. Fig. 10: Candidate positions for backbones inside the gap.

## B.2 Length Minimization with Finite Backbones

We modify the dynamic program used for minimizing the total number of labels (in Section 3) for minimizing the total leader length. First, we now denote by the  $T$ -values the additional length of segments and backbones needed for labeling the points of the subinstance. We do, however, have to apply more changes. 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. 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 there is a set  $B$  of backbones that all lie in the same gap between points  $p_i$  and  $p_{i+1}$ . Let  $b^*$  be the longest of those; see Fig. 8.  $b^*$  vertically splits the set  $B$ ; any backbone  $b' \in B$  above  $b^*$  connects only to points above it and any backbone  $b'' \in B$  below  $b^*$  connects only to points below it. 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  $b^*$  will be very close to  $y(p_i)$  and the backbones below  $b^*$  will be very close to  $y(p_{i+1})$ . Furthermore, depending on the numbers of connected points above and below, by either moving  $b^*$  to the top or to the bottom we will find a solution that is not worse, and in which any backbone of  $B$  is close to  $p_i$  or  $p_{i+1}$ .

If 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. With these positions as well as the input points as possible label positions, 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.

*Bounded numbers of labels.* If we want to integrate an upper bound  $K$  on the total number of labels, or, for each color  $c \in C$ , an upper bound  $k_c$  on the number of labels of color  $c$ , into the dynamic program, 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). Additionally, when splitting the instance into two parts, we have to consider not only the position of the splitting backbone of color  $c$ , but also the different combinations of distributing the allowed number(s) of backbones among the subinstances. For a global bound  $K$ , we need  $O(nK)$  time for computing an entry of the table. If we have individual bounds  $k_c$  for  $c \in C$ , we need  $O(n \prod_{c \in C} k_c)$  time. Together with the additional dimension(s) of the table, we can minimize the total leader length in  $O(n^4 |C|^2 K^2)$  time if we have a global bound  $K$ , and in  $O\left(n^4 |C|^2 \left(\prod_{c \in C} k_c\right)^2\right)$  time if we have an individual bound  $k_c$  for each color  $c \in C$ .

*Minimum distances.* So far, we allowed backbones to be infinitely close to unconnected points and other backbones, which will, in practice, lead to 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. 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  $\Delta$ ; see Fig. 9. 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  $\Delta$ ; see Fig. 10. Note that  $n$  positions of each type suffice in a gap; if the gap is too small, there might even be less positions. The two sequences can overlap. In this case, we have to check that we do not combine two positions with a distance smaller than  $\Delta$  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  $O(n^7 |C|^2)$  if we do not bound the number of labels,  $O(n^7 |C|^2 K^2)$  if we have a global bound  $K$  on the number of labels, and  $O(n^7 |C|^2 (\prod_{c \in C} k_c)^2)$  if we have an individual bound  $k_c$  for each color  $c \in C$ .

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

## C Crossing Minimization

### C.1 Crossing Minimization with fixed $y$ -Order of Labels for Finite Backbones

**Theorem 8.** *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, finite backbones, and minimum number of crossings can be computed in  $O(n|C|)$  time.*

*Proof.* We adopt the scenario of Section 5.1, assuming that each backbone is of finite length. In particular, we seek for crossing-minimal labelings, in which all points of the same color are routed to the same label and the order of the labels is fixed, i.e., the label of the  $i$ -th colored points is above the label of the  $j$ -th colored points, when

$i < j$ . Then, the dynamic program presented in the proof of Theorem 5 can also be exploited. However, the computation of the number of crossings when fixing a backbone at a certain position should take into consideration that the backbones are not of infinite length. Recall that the dynamic program could precompute these crossings, by maintaining an  $n \times |C|$  table  $\text{cross}$ , in which each entry  $\text{cross}(i, g)$  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  $0 \leq g \leq n$ . In our case,  $\text{cross}(i, g) = \text{cross}(i, g - 1) + 1$ , if the point between gaps  $g - 1$  and  $g$  is to the left of the  $i$ -colored points and has color  $j$  s.t.  $j > i$ . In the case, where the point between gaps  $g - 1$  and  $g$  is to the left of the  $i$ -colored points and has color  $j$  and  $j < i$ ,  $\text{cross}(i, g) = \text{cross}(i, g - 1) - 1$ . Otherwise,  $\text{cross}(i, g) = \text{cross}(i, g - 1)$ . Again, all table entries of the form  $\text{cross}(\cdot, 0)$  can be clearly computed in  $O(n)$  time.  $\square$

## C.2 Flexible y-Order of Labels for Finite Backbones

Here we give a more detailed description of the range restrictor gadget shown in Fig. 3a.

The construction consists of the middle backbone, whose position will be restricted to a given range  $R$ , and an upper and a lower *guard gadget* that ensure that positioning the middle backbone outside range  $R$  creates many crossings. 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. The middle backbone is connected to two *range points* whose y-coordinates are the 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 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 (resp. lower) guard gadget is formed by the set of upper (resp. lower) guards. We call  $M$  the *size* of the guard gadgets.

**Lemma 1.** *The backbones of the range restrictor 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 Fig. 3a. It remains to show that positioning the middle backbone outside range  $R$  results in at least  $M - 1$  crossings. Suppose for a contradiction 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 guards must be positioned above the middle backbone, as a guard 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 Fig. 11, where the two alternatives for the lower guard are drawn in black and bold gray, respectively. This contradicts our assumption.  $\square$

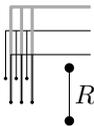


Fig. 11: Crossings caused by a pair of an upper and a lower guard that are positioned on the same side outside range  $R$ .

*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, i.e., above row  $2n + 1$  or below row  $-2n - 1$ , (b) between rows  $2n + 1$  and  $-2n - 1$ , but outside one of the two gaps, and (c) in rows  $j$  or  $-j$ , i.e., inside one of the gaps. In case (a) we get at least  $M = \Omega(n^4)$  crossings by Lemma 1. So we may assume that case (a) never occurs for any selector gadget; we will see that in this case there are only  $O(n^4)$  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.

To give 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  $-2n - 1$ , 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.  $\square$

*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 problem.

*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.

- (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.

**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  $X_1 = 5m \sum_{i=1}^{n-1} i = 5m(n^2 - n)/2$  fixed crossings.

**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 Fig. 5). 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 Fig. 5). The backbones of the upper and lower blockers do not cross any other vertical selector segments.

Let  $\kappa = |\{(v_i, v_j), (v_k, v_l)\} \in E^2 \mid i < k \text{ and } j < l\}| = O(n^2)$ . Then we obtain  $X_2 = 5n^2 + (n^2 - n)/2 + 2\kappa$  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  $X_3 = m(n^2 - n)/2$  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*, i.e.,  $i < j < k < l$ , (b)  $e$  and  $f$  are *nested*, i.e.,  $i < k < l < j$ , or (c)  $e$  and  $f$  are *interlaced*, i.e.,  $i < k < j < l$ . 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 Fig. 5). 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 Fig. 5). Finally, for every pair of interlaced edges there are no crossings if they are assigned to opposite sides of the x-axis or two crossings if they are assigned to the same side. So interlaced edges do not contribute to the number of fixed crossings. Let  $\tau = |\{(v_i, v_j), (v_k, v_l)\} \in E^2 \mid i < j < k < l\}| = O(n^2)$  be the number of sequential edge pairs. 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 interlaced edge pairs (see for example edges  $(v_1, v_4)$

and  $(v_3, v_8)$  or  $(v_2, v_5)$  in Fig. 5). But 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  $I_G$ . Setting  $X = X_1 + X_2 + X_3 + X_4$  this proves Claim 2.  $\square$