Available online at www.sciencedirect.com
ScienceDirect
Computational Geometry 36 (2007) 215–236
Computational Geometry
Theory and Applications
www.elsevier.com/locate/comgeo

Boundary labeling: Models and efficient algorithms for rectangular maps ☆

Michael A. Bekos [a,*], Michael Kaufmann [b], Antonios Symvonis [a], Alexander Wolff [c]

[a] *National Technical University of Athens, Department of Mathematics, Athens, Greece*
[b] *University of Tübingen, Institute for Informatics, Sand 13, 72076 Tübingen, Germany*
[c] *Fakultät für Informatik, Universität Karlsruhe, P.O. Box 6980, 76128 Karlsruhe, Germany*

Received 18 October 2005; received in revised form 1 May 2006; accepted 3 May 2006

Available online 27 June 2006

Communicated by P. Agarwal

Abstract

We introduce *boundary labeling*, a new model for labeling point sites with large labels. According to the boundary-labeling model, labels are placed around an axis-parallel rectangle that contains the point sites, each label is connected to its corresponding site through a polygonal line called *leader*, and no two leaders intersect. Although boundary labeling is commonly used, e.g., for technical drawings and illustrations in medical atlases, this problem has not yet been studied in the literature. The problem is interesting in that it is a mixture of a label-placement and a graph-drawing problem.

In this paper we investigate several variants of the boundary-labeling problem. We consider labels of identical or different size, straight-line or rectilinear leaders, fixed or sliding ports for attaching leaders to sites and attaching labels to one, two or all four sides of the bounding rectangle. For any variant of the boundary labeling model, we aim at highly esthetical placements of labels and leaders. We present simple and efficient algorithms that minimize the total leader length or, in the case of rectilinear leaders, the total number of bends.

Keywords: Automated label placement; Boundary labeling; Straight-line leaders; Rectilinear leaders

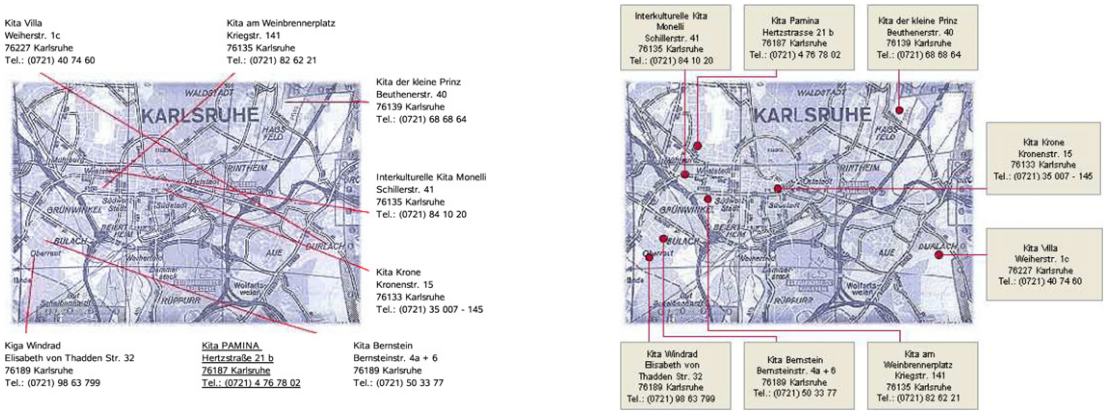
1. Introduction

Label placement is one of the key tasks in the process of information visualization. In diagrams, maps, technical or graph drawings, features like points, lines, and polygons must be labeled to convey information. The interest

☆ This article is based on the preliminary version [M.A. Bekos, M. Kaufmann, A. Symvonis, A. Wolff, Boundary labeling: Models and efficient algorithms for rectangular maps, in: Proc. 12th Int. Symposium on Graph Drawing (GD'04), in: Lecture Notes in Computer Science, Springer-Verlag, Berlin, 2005]. Our work has been supported by grants Ka 512/8-2 and WO 758/4-1 of the German Research Foundation (DFG), by the German–Greek cooperation program GRC 01/048, and by the program "Pythagoras" which is co-funded by the European Social Fund (75%) and Greek National Resources (25%).

* Corresponding author.
E-mail addresses: mikebekos@math.ntua.gr (M.A. Bekos), mk@informatik.uni-tuebingen.de (M. Kaufmann), symvonis@math.ntua.gr (A. Symvonis).
URL: http://i11www.ira.uka.de/people/awolff.

0925-7721/$ – see front matter
doi:10.1016/j.comgeo.2006.05.003



(a) Original map taken from the Internet service of the Arbeiterwohlfahrt Karlsruhe-Stadt e.V. [3].

(b) Label-leader placement by our algorithm using rectilinear (type-*opo*) leaders of minimum total length.

Fig. 1. Map with kindergartens in Karlsruhe, Germany.

in algorithms that automate this task has increased with the advance in type-setting technology and the amount of information to be visualized. Due to the computational complexity of the label-placement problem, which is NP-hard in general [9], cartographers, graph drawers, and computational geometers have suggested numerous approaches, such as expert systems [2], zero-one integer programming [25], approximation algorithms [9], simulated annealing [26] and force-driven algorithms [12] to name only a few. Wolff and Strijk [24] maintain an extensive bibliography about label placement. The ACM Computational Geometry Impact Task Force report [6] denotes label placement as an important research area. Manually labeling a map is a tedious task that is estimated to take 50% of total map production time [21].

In this paper, we deal with labeling dense point sets with comparatively large labels. This is a common problem, e.g., in medical atlases, where certain features of a drawing or photo are explained by blocks of text that are arranged around the drawing. The same problem occurs when several locations on a map are to be labeled with large labels that must not occlude the map; see Fig. 1(a). Our model is as follows: we assume that we are given a set $P = \{p_1, \ldots, p_n\}$ of points and an axis-parallel rectangle R that contains P. Each point, or *site*, p_i is associated with an axis-parallel rectangular open label. The labels have to be placed and connected to their corresponding sites by polygonal lines, so-called leaders, such that (a) no two labels intersect, (b) no two leaders intersect, and (c) the labels lie outside R but touch R. We investigate various constraints concerning the location of the labels and the type of leaders. More specifically, we allow labels to be attached to one side, two sides, or all four sides of R, and we either use straight-line or rectilinear leaders. We refer to straight-line leaders as type-*s* leaders, and we consider two types of rectilinear leaders, namely type-*po* and type-*opo* leaders that consists of two and three axis-parallel segments, respectively. We also consider two ways to attach leaders to labels: using fixed and sliding ports. For details refer to Section 2. We propose efficient algorithms that find *some* non-intersecting (i.e. feasible) leader-label placement, but we also consider two natural objectives: minimize the total length of the leaders (see e.g., Fig. 1(b)) and, if leaders are not straight-line segments, minimize the total number of bends over all leaders. Table 1 gives an overview over our results.

These new problems are combinations of label-placement and graph-drawing problems. They are somewhat related to graph-drawing problems arising in the automated layout of UML class diagrams where sometimes boxes with notes have to be attached to class nodes [5]. Similar layout problems arise when labels are placed *after* the layout of the graph structure [18]. The reason might be that the layout algorithm does not support immediate labeling or the size of the labels is not known during the layout process but changing interactively. Due to the complexity of either step there are still very few publications that combine graph drawing and label placement. Klau and Mutzel [16] have coined the term *graph labeling* for this discipline and have given a mixed-integer program for computing orthogonal graph layouts with node labels. Their approach has recently been extended by Binucci et al. [4] who additionally allow edge labels.

Leaders have so far only been used by Zoraster [26] and Freeman et al. [7]. Zoraster [26] investigates the labeling of seismic survey maps. Such maps are special in that the sites typically lie on a few seismic lines, which also must be

Table 1
Running times of our algorithms (in big-Oh-Notation) for various versions of boundary labeling, where ε is an arbitrarily small positive constant

Leader type	Number of rectangle sides with labels	Feasible solution		Bend-minimal solution	Length-minimal solution		
					Fixed ports	Sliding ports	Theorem
s	1	$n \log n$	Theorem 9	N/A	$n^{2+\varepsilon}$	n^3	10
s	4	$n \log n$	Theorem 11	N/A	$n^{2+\varepsilon}$	n^3	12
po	1			open	n^2	n^2	7
po	2			open	n^2	n^2	8
opo	1	$[n \log n]$	Lemma 1	$[n^2]$	$n \log n$ (Remark 1)	$[n^2]$	1
opo	2			open	n^2 $[nH^2]^\star$	n^2	3
opo	4	$n \log n$	Theorem 2	open	$n^2 \log^3 n$	n^3	4

The time bound in square parentheses refers to the case of non-uniform labels. The problem marked by * is NP-hard. Our pseudo-polynomial solution (see Theorem 5) assumes that label heights and the height H of the bounding rectangle are integers. N/A stands for non-applicable. Entries in column "Feasible solution" are filled only if we can compute a feasible solution asymptotically faster than a bend- or length-optimal solution.

labeled, and in that a site-label is placed orthogonally to the line that contains the corresponding site. In order to cope with the density of the site sets, Zoraster used 24 instead of the usual four label positions and connected each label to its site by a leader. He uses simulated annealing to minimize an objective function that takes into account (a) the number of objects that receive a label and (b) the position of labels. The objective function favors labels that are close to the object they annotate.

Freeman et al. [7] present ALPS, a software system for automated labeling of soil survey maps. If a soil polygon is too small to accommodate its own label, the system tries to place the polygon's label into an adjacent polygon. If this is possible, a straight-line leader is used to connect label and polygon, otherwise the polygon remains unlabeled. Label positions are determined by an iterative raster-based method.

An example of interactive label placement can be found in the widely used infotip mechanism which supplies the user with additional information about screen objects whenever the mouse pointer rests a certain amount of time in their vicinity. Fekete and Plaisant [8] extend the infotip paradigm to cope with labeling of dense maps interactively. They draw a circle of fixed radius around the current cursor position, the so-called *focus circle*, and label the sites that fall into the circle by axis-parallel rectangles that contain the names associated with the sites. Labels are left-aligned and placed in two stacks to the left and the right of the circle. If the cursor is too close to the left or right border of the screen, only one stack is employed. The labels in the right (left) stack correspond to those sites whose projection is on the right (left) half of the focus circle. The order of the labels corresponds to that of the projected sites. To connect a site with its label, Fekete and Plaisant use a non-orthogonal leader that consists of two line segments: one radially from the site to its projection on the focus circle and one from there to the midpoint of the left edge of the corresponding label. For labels on the left side, sometimes a third segment is used. This approach guarantees that no two leaders cross. In the worst case they may overlap within the focus circle. Fekete and Plaisant do not specify any asymptotic running time, but it is obvious that their algorithm runs in $O(|S| \log |S|)$ time once the set S of sites in the focus circle has been determined.

Iturriaga and Lubiw [14] give an $O(n^4)$-time decision algorithm for attaching *elastic* labels to n sites on the perimeter of a rectangle. An elastic label models a block of text of fixed area, but varying width and height. Iturriaga and Lubiw place their labels *inside* the rectangle. The problem is motivated by labeling shops on maps of the downtown areas of North American cities such that the text labels are placed within the rectangles defined by the surrounding streets.

Iturriaga [15] also briefly investigates the inverse problem, where elastic labels must be attached to their sites *outside* the given rectangle R. She presents an algorithm that finds a label placement that uses the minimum-width strip around R. If n sites are given in order on the boundary of R, the algorithm takes $O(n)$ time.

This paper is structured as follows: In Section 2, we model and define the boundary labeling problem. In Section 3, we are concerned with rectilinear leaders. We present algorithms for leader-bend minimization, legal leader-label placement, and leader-length minimization. Straight-line leaders are considered in Section 4. In Section 5, we give example layouts produced by our algorithms. We conclude in Section 6 with open problems and directions for future work.

2. Defining and modeling the problem

We consider the following problem. Given an axis-parallel rectangle $R = [l_R, r_R] \times [b_R, t_R]$ of width $W = r_R - l_R$ and height $H = t_R - b_R$, and a set $P \subset R$ of n sites $p_i = (x_i, y_i)$, each associated with an axis-parallel rectangular open label l_i of width w_i and height h_i, our task is to find a *legal* or an *optimal* leader-label placement. Our criteria for a legal leader-label placement are the following:

1. Labels have to be disjoint.
2. Labels have to lie outside R, close to the boundary of R.
3. Leader c_i connects site p_i with label l_i for $1 \leqslant i \leqslant n$.
4. Intersections of leaders with other leaders, sites or labels are not allowed.
5. The *ports* where leaders touch labels may be *fixed* (the center of a label edge, say) or may be arbitrary (*sliding ports*).

In this paper we present algorithms that compute legal leader-label placements (for brevity, simply referred to as *labelings*) for various types of leaders defined below, but we also approach optimal placements according to the following two objective functions:

- short leaders (minimum total length) and
- simple leader layout (minimum number of bends).

These criteria have been adopted from the area of graph drawing since leaders do not play a significant role in the label-placement literature. In Zoraster's work [26], the leader length is only indirectly minimized by ranking the above-mentioned 24 label positions such that positions closer to the site are favored. We will evaluate the two criteria under two models for drawing leaders. In the first model we require that each leader is rectilinear, i.e., a connected sequence of orthogonal line segments. In the second model each leader is drawn as a straight-line segment.

A rectilinear leader consists of a sequence of axis-parallel segments that connects a site with its label. These segments are either parallel (p) or orthogonal (o) to the side of the bounding rectangle R to which the label is attached. This notation yields a classification scheme for rectilinear leaders: let a *type* be an alternating string over the alphabet $\{p, o\}$. Then a leader of type $t = t_1 \dots t_k$ consists of an x- and y-monotone connected sequence $(e_1, \dots, e_k)$ of segments from site to label, where each segment e_i has the direction that the letter t_i prescribes. In this paper we focus on leaders of the types *po* (see Fig. 2) and *opo* (see Figs. 1(b) and 3). We consider type-o leaders to be of type *opo* and of type *po* as well. We extend this notation by referring to straight-line leaders as type-s leaders; see Fig. 4.

For each type-*opo* leader we further insist that the central p-segment is immediately outside the bounding rectangle R and is routed in a so-called *track-routing area*. We assume that the width of the track-routing area is fixed and large enough to accommodate all leaders with a sufficient distance. Due to this assumption the total length of the o-segments of all leaders is identical in all label-leader placements. Thus we are left with optimizing the length of the p-segments. Minimizing the width of the track-routing area for a given minimum leader distance is an interesting problem in itself, which is not the topic of this work.

We start with a negative result. Assume that the labels must be attached either to the right or to the left side of the rectangle R and that their heights are not equal. Furthermore, assume that the label heights sum up to twice the height of R. Clearly, the task of assigning the labels to the two sides corresponds to the well known problem PARTITION, which is weakly NP-complete [11]. Because of the NP-completeness of the general problem, it is reasonable to study

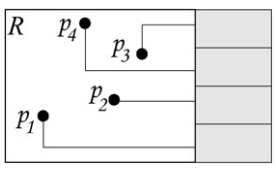


Fig. 2. Type-*po* leaders.

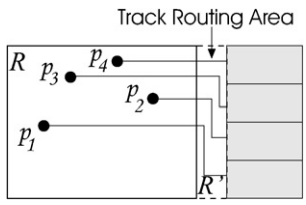


Fig. 3. Type-*opo* leaders.

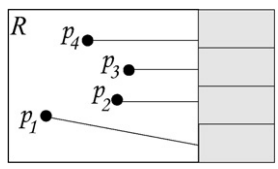


Fig. 4. Type-*s* leaders.

the case of uniform labels, i.e., labels that have the same size. Throughout this paper we assume that input sites are in *general position*, i.e., no three sites lie on a line and no two sites have the same x- or y-coordinate.

3. Rectilinear leaders

In this section, we investigate different ways of drawing rectilinear leaders. We present algorithms for leader-bend minimization, legal leader-label placement, and leader-length minimization. We consider attaching labels to one, two, and four sides of the rectangle R and connecting sites to their labels with leaders of type *po* and *opo*; see Figs. 2 and 3, respectively.

In the description of our algorithms for type-*opo* leaders we focus on label placement and ignore the leader routing. For the routing we assume that there is a rectangle $R' \supset R$ broader than R and use $R' \setminus R$ as a fixed-width track-routing area, i.e., we place all leader segments which are parallel to the corresponding side of R in $R' \setminus R$. If the track-routing area is not of fixed width, our algorithms become more complicated and slower by a factor of $\Omega(n)$—note that the worst-case track width is $\Theta(n)$.

3.1. Leader-bend minimization

In this subsection we consider the problem of attaching labels of variable height to one, say the right, side of the rectangle R. We use type-*opo* leaders and sliding ports, i.e., every leader simply has to touch some point on the perimeter of the corresponding label. We assume that the sum of the label heights is at most the height of R and that the sites are sorted according to increasing y-coordinate.

Observe that, in any legal one-side labeling with type-*opo* leaders, the vertical order of the sites is identical to the vertical order of their corresponding labels. This, together with the assumption that no two sites share the same y-coordinate, guarantees that leaders do not intersect. We summarize this observation in the following lemma.

Lemma 1. *Given a rectangle R of sufficient size, a side s of R, a set $P \subset R$ of n sites in general position and a rectangular label for each site, there is an $O(n \log n)$-time algorithm that attaches labels to s and connects the sites with non-intersecting type-opo leaders to the corresponding leaders using sliding ports.*

Proof. Without loss of generality, assume that s is the right side of rectangle R. We first stack the labels (in increasing order of the y-coordinate of the corresponding sites) immediately to the right of the track-routing area (i.e., to the right of rectangle R') and on top of each other such that the bottom side of label l_1 has the same y-coordinate with the bottom side of rectangle R. Then, we connect each site p_i by a horizontal segment $y_i \times [x_i, r_R]$ to the right side of rectangle R. Finally we use the track-routing area to lay out the remaining parts of the leaders from the right side of R to the, say, midpoints of the left label sides; see Fig. 3. We note that there are several efficient ways to determine the x-coordinate of the vertical leader segments, which are placed in the track-routing area. The simplest one uses, as offset from the boundary, the index of the sites in a bottom-to-top ordering. More sophisticated linear-time methods can be based of computing the number of vertically overlapping p-segments. The time complexity of our algorithm is due to the sorting of the sites. □

Remark 1. For the case of uniform labels of maximum size (or, in general of fixed size and location) and fixed ports, there only exists a single legal labeling. Thus, the algorithm described in the proof of Lemma 1 also yields a labeling that minimizes the total leader length (the topic of Section 3.3).

Clearly, this approach is not optimal in terms of the total number of leader bends. Each type-*opo* leader contributes up to two bends. By sliding the labels along the side of R (without changing the order of the labels) and by allowing sliding label ports, it is possible to connect some of the sites to their corresponding labels by leaders that consist of a single straight-line segment and contribute zero bends to the total number of bends. Thus, minimizing the total number of bends is equivalent to placing the labels to appropriate locations so that the number of straight-line (zero-bend) leaders is maximized. This is a one-dimensional label-placement problem. There has been work on similar problems where labels are not restricted to a constant number of positions, but can slide. Kim et al. [17] have observed that it is trivial to decide whether a set of points on a line can be labeled with (unit) intervals such that each interval

contains the point it labels. In the same paper, however, they also considered the problem of finding the maximum interval length that allows to label all points. By a clever geometric transformation they managed to solve the problem in linear time if the points are given in order. Garrido et al. [10] have investigated the problem of deciding whether a set of points on a line can be labeled with labels of given lengths. They showed that the problem becomes NP-hard if labels can be placed both above and below the line.

Our problem is new in that labels do not necessarily have to contain the point they label, but even if they do not (and thus contribute to the objective function in a negative way), they must be placed within an interval whose length is restricted (by the height of R). We now give an algorithm for placing labels that uses as many horizontal (i.e., zero-bend) leaders as possible. We have the following result:

Theorem 1. *Given a rectangle R of sufficient size, a side s of R, a set $P \subset R$ of n sites in general position and a rectangular label for each site, there is an $O(n^2)$-time algorithm that attaches the labels to s and connects the sites with non-intersecting type-opo leaders to the corresponding leaders using sliding ports such that the total number of leader bends is minimized.*

Proof. Without loss of generality, we assume that s is the right side of rectangle R. We also assume that the sum of the label heights is at most the height of R and that the sites are sorted according to increasing y-coordinate. Recall that by $p_i = (x_i, y_i)$ we denote the ith site, by h_i we denote the height of the ith label, $1 \leqslant i \leqslant n$, and by b_R and t_R we denote the y-coordinate of the bottom right and top right corner of R, respectively.

Our dynamic programming algorithm employs a table T of size $(n+1) \times (n+1)$. For $0 \leqslant k \leqslant i \leqslant n$ the entry $T[i,k]$ will contain the minimum y-coordinate that is needed to accommodate the lowest i labels such that *at least* k of them use horizontal leaders. If it is impossible to connect k out of the lowest i labels with horizontal leaders, we set $T[i,k]$ to ∞. As usual, the table entries are computed in a bottom-up fashion. By definition of $T[i,k]$, it always holds that:

$$T[i,k] \leqslant T[i,k+1]$$

For computing $T[i,k]$ we only need to know the entries $T[i-1,k-1]$ and $T[i-1,k]$. We distinguish two cases based on whether $y_i \leqslant T[i-1,k-1]$.

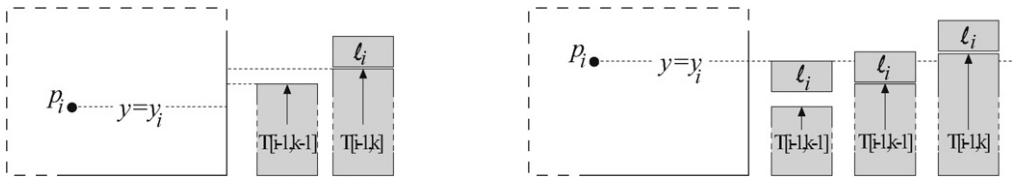
Case 1: $y_i \leqslant T[i-1,k-1]$.

Refer to Fig. 5(a). In the case where $y_i \leqslant T[i-1,k-1]$ it is obvious that p_i cannot be connected to label l_i with a horizontal leader and, thus, the leader out of p_i cannot be the kth horizontal leader. $T[i,k]$ can have a finite value only if $T[i-1,k]$ is finite. In this subcase, we stack label l_i on top of the $i-1$ already placed labels, and obtain a placement with k horizontal leaders and height $T[i-1,k]+h_i$. If $T[i-1,k]=\infty$, no solution with k horizontal leaders out of the lowest i sites exists and, thus, $T[i-1,k]=\infty$. Since we assume that $\infty + h_i = \infty$, both subcases can be described by the equation:

$$T[i,k] = T[i-1,k] + h_i$$

Case 2: $y_i > T[i-1,k-1]$.

Refer to Fig. 5(b). Consider first the subcase where $y_i \leqslant T[i-1,k-1]+h_i$. If we place label l_i on top of the already placed labels then it will be "hit" by the horizontal leader out of site p_i. In the subcase where $y_i > T[i-1,k-1]+h_i$, we can place label l_i (above the already placed labels) so that its top side lies on line $y = y_i$. From these two subcases, we conclude that if site p_i is connected to its corresponding label by a horizontal leader, then $T[i,k] = \max(y_i, T[i-1,k-1]+h_i)$.



(a) $y_i \leqslant T[i-1,k-1]$ (b) $y_i > T[i-1,k-1]$

Fig. 5. Label placements that the dynamic programming algorithm takes into account when computing $T[i,k]$.

Input: a set P of n point sites with y-coordinates $y_1 < \cdots < y_n$, an axis-parallel rectangle $R = [l_R, r_R] \times [b_R, t_R]$ that contains P, and n labels of heights $h_1, \ldots, h_n$.
Output: the maximum possible number of horizontal leaders.

1 {Fill dynamic programming table T}
$T[0,0] = b_R$
for $i = 1$ **to** n **do**
  $T[i,0] = T[i-1,0] + h_i$
  $T[i-1,i] = \infty$
  for $k = 1$ **to** i **do**
    if $T[i-1,k-1] \geqslant y_i$ **then**
      $T[i,k] = T[i-1,k] + h_i$
    else
      $T[i,k] = \min(T[i-1,k] + h_i, \max(y_i, T[i-1,k-1] + h_i))$
    end
  end
end

2 {Determine best placement below t_R}
$j = 0$
while $T[n,j] \leqslant t_R$ **do**
  $j = j + 1$
end
return $j - 1$

Algorithm 1. 1SIDEROUTEOPO.

If $T[i-1,k]$ is finite then a different solution, which is obtained by stacking label l_i on top of the already placed labels, is also possible. The height of this solution is $T[i-1,k] + h_i$.

The above subcases can be expressed by the equation:

$$T[i,k] = \min\bigl(T[i-1,k] + h_i, \max\bigl(y_i, T[i-1,k-1] + h_i\bigr)\bigr)$$

Based on the above cases, we conclude that $T[i,k]$ can be computed by using the following recurrence relation:

$$T[i,k] = \begin{cases} T[i-1,k] + h_i & \text{if } y_i \leqslant T[i-1,k-1] \\ \min(T[i-1,k] + h_i, \max(y_i, T[i-1,k-1] + h_i)) & \text{if } y_i > T[i-1,k-1] \end{cases}$$

Algorithm 1 computes the maximum possible number of horizontal leaders. A placement with the maximum number of horizontal leaders has the minimum number of bends. The algorithm is directly based on the recurrence relation computed above (see block 1 of the algorithm). Block 1 of Algorithm 1 computes the maximum possible number of zero-bend leaders by identifying the largest j with $0 \leqslant j \leqslant n$ such that $T[n,j] \leqslant t_R$, that is, such that all labels fit on the side of rectangle R.

It is obvious that Algorithm 1 runs in $O(n^2)$ time and uses $O(n^2)$ space. The algorithm can easily be modified such that it also computes the label and leader positions in an optimal solution. In that case the algorithm needs an extra table of the same size as T. □

3.2. Legal leader-label placement

In this subsection we investigate the problem of attaching labels to all sides of the rectangle R. Since optimizing leader length or bend number is difficult in this setting we content ourselves with a legal placement. Our basic idea is simple: we partition R into four disjoint regions such that the algorithms for type-*opo* leaders from the previous subsection can be applied to each region separately. For the sake of brevity our discussion ignores the problem of how to distribute the boundary of the areas between them.

We have two requirements for a region A in the partition of R: (a) A must be adjacent to a specific side s_A of R and (b) each site in A must see the point with the same x- or y-coordinate on s_A. Requirement (b) is a consequence of using type-*opo* leaders. We observe that a partition of R into four regions such that each region A contains the

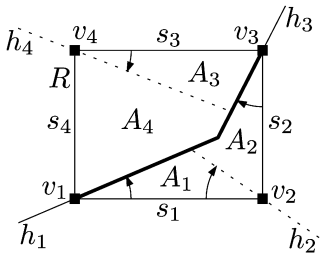


Fig. 6. Partition into monotone regions.

side s_A of R and A is monotone in the direction of s_A satisfies both requirements (for example; see the partition of the rectangle in Fig. 6). As a consequence, we focus on the identification of such a partition.

We introduce some notation; see Fig. 6. Let $v_1, \ldots, v_4$ be the corners of R from the lower left corner in counterclockwise order, and let $s_1 = \overline{v_1 v_2}, \ldots, s_4 = \overline{v_4 v_1}$ be the sides of R. For the sake of brevity we view the sides of R as line segments that do not contain their endpoints. We assume that the corners of R do not lie on any line determined by a pair of sites.

To avoid the NP-hard problem PARTITION as discussed in Section 2 we assume that we know how many labels have to be attached to which side of R. In case we want to attach labels to non-parallel sides of R this assumption makes good sense if we have uniform square labels. Let $n_1, \ldots, n_4$ be the number of labels that have to be attached to the sides $s_1, \ldots, s_4$, respectively, and let $n = n_1 + \cdots + n_4$.

We construct the partition of the rectangle R as follows:

1. We first partition R into two regions A_{12} and A_{34} such that A_{12} contains $n_1 + n_2$ sites as well as the sides s_1 and s_2. We proceed as follows. Let h_1 be the halfplane below the horizontal line through v_1 and let h_3 be the halfplane to the right of the vertical line through v_3. Now we turn h_1 around v_1 in counterclockwise direction and h_3 around v_3 in clockwise direction until $A_{12} = R \cap (h_1 \cup h_3)$ contains exactly $n_1 + n_2$ sites. Due to our assumption concerning the general position of the sites and the corners of R this is always possible. The two resulting regions are both x- and y-monotone; one is a convex, one a non-convex quadrilateral; see the bold solid line segments in Fig. 6.
2. Now we split A_{12} into two regions A_1 and A_2 such that for $i \in \{1, 2\}$, A_i contains side s_i and n_i sites. Let h_2 be the halfplane below the horizontal line through v_2. We turn h_2 in clockwise direction around v_2 until $A_1 = h_2 \cap A_{12}$ contains n_1 sites. Again this is possible due to our assumption regarding the general position of sites and corners. Clearly, $A_2 = A_{12} \setminus A_1$ contains side s_2 and the remaining n_2 sites. In the same fashion we split A_{34} into A_3 and A_4. All four resulting regions are x- and y-monotone; see the dotted lines in Fig. 6.

Turning the halfplanes can be implemented by sorting the sites according to the angles they enclose with the horizontal or vertical lines through the appropriate corners of R. Using the $O(n \log n)$-time algorithm of Lemma 1 we have the following result:

Theorem 2. *Given a rectangle R of sufficient size, a set $P \subset R$ of n sites in general position, square uniform labels, one per site, and numbers $n_1, \ldots, n_4$ that express how many labels are to be attached to which side of R, there is an $O(n \log n)$-time algorithm that attaches the labels to R and connects them to the corresponding sites with non-intersecting type-opo leaders.*

Recall that our objective in this subsection is to obtain a legal label placement. This is what the $O(n \log n)$-time algorithm of Lemma 1 yields. In general, we will obtain a drawing with fewer bends by investing a running time of $O(n^2)$ and using Algorithm 1. However, in order to obtain a routing with the *minimum* number of bends with type-*opo* leaders in the four-side case, we would have to go through all combinatorially different ways of partitioning the set of sites into four subsets with the cardinality and monotonicity constraints listed above.

A related problem has been considered by Iturriaga and Lubiw [14]. Given a set of n points on the boundary of a rectangle and for each point an elastic label of a certain area, they want to decide whether it is possible to attach these labels to their points *inside* the rectangle. To solve this problem they observe that in any solution the rectangle

can be split by a so-called *corridor partition* into at most four *corner blocks* and a *corridor* such that each label lies completely within one of these areas. For the two types of areas they use different label-placement algorithms. They state that the number of combinatorially different corridor partitions is $O(n^6)$. It seems that such an approach that enumerates all possible partitions of R into four areas cannot be used to obtain an efficient algorithm for bend minimization. The problem is that there are site sets that cause an exponential number of such partitions—even in the two-side case. For example, if n is even, there are $\binom{n}{n/2}$ different ways to split a rectangle containing the sites $(1, 1), (2, 2), \ldots, (n, n)$ such that half of the sites lie in the area incident to s_1 and s_4, respectively. It is an interesting open question how a minimum-bend type-*opo* routing can be found in the two- or four-side case.

3.3. Leader-length minimization

In this section we focus on computing label placements of minimum total leader length. We present a variety of algorithms that attach labels to one side (right), two opposite sites (left and right) and four sides of rectangle R, examine uniform and non-uniform labels, and fixed or sliding ports.

3.3.1. Two-side labeling with type-opo leaders and uniform labels

Labels are placed on opposite sides of the rectangle, say s_{left} and s_{right}, $n/2$ labels on each side. The labels are assumed to be uniform in the sense that they all are of identical height (or width, if they are placed on the top and the bottom sides of the rectangle). The $n/2$ labels are of maximum height, covering the full length of the side of the rectangle they reside at, and hence their position at each side is determined. We are given n sites $p_i = (x_i, y_i)$, $i = 1, 2, \ldots, n$, which have to be connected with leaders to labels on s_{left} and s_{right} so that the total leader length is minimized.

We consider type-*opo* leaders which may connect to the labels with fixed or sliding ports. The ith site p which is assigned to s_{left} is connected to the ith label of s_{left} with a type-*opo* leader. Since the location of each label is fixed, the length of the leader to the ith label of s_{left} is determined. In the case of fixed ports we define $\text{Left}(p, i)$ to be the distance from site p to the (closest) port of the ith label of s_{left}, while in the case of sliding ports $\text{Left}(p, i)$ is defined as the distance from site p to the closest *point* of the ith label of s_{left}. $\text{Right}(p, i)$ is defined similarly. We obtain the following result:

Theorem 3. *Given a rectangle R with $n/2$ uniform labels of maximum height on its left and right side, and a set $P \subset R$ of n sites in general position, a non-crossing minimum-length type-opo leader placement can be computed in $O(n^2)$ time for both fixed and sliding ports.*

Proof. To compute a label placement of minimum total leader length, we use dynamic programming (see Algorithm 2). We first assume that n is even. Later we describe how to deal with the case that n is odd. The algorithm maintains a table $T[0 : n/2, 0 : n/2]$. Entry $T[l, r]$ contains the minimum total leader length for the $l + r$ lowest sites under the condition that l are connected to labels on s_{left} and the remaining r to labels on s_{right}.

It can easily be proven by induction that $T[l, r]$ satisfies the following recurrence relation for $0 \leqslant l, r \leqslant n/2$:

Input: a set P of n sites $p_1(x_1, y_1), \ldots, p_n(x_n, y_n)$, sorted in order of increasing y-coordinate.
Output: the minimum total leader length

$T[0, 0] = 0$
for $i = 1$ **to** n **do**
  $T[i, -1] = T[-1, i] = \infty$
  for $l = 0$ **to** $\min\{i, n/2\}$ **do**
    $r = i - l$
    $T[l, r] = \min\{T[l, r-1] + \text{Right}(p_i, r),\ T[l-1, r] + \text{Left}(p_i, l)\}$
  end
end
Return $T[n/2, n/2]$

Algorithm 2. UNIFORMLABEL2SIDEROUTEOPO.

$$T[0,0] = 0 \tag{1}$$

$$T[0,r] = T[0,r-1] + \text{Right}(p_r, r) \tag{2}$$

$$T[l,0] = T[l-1,0] + \text{Left}(p_l, l) \tag{3}$$

$$T[l,r] = \min\{T[l,r-1] + \text{Right}(p_{l+r}, r), T[l-1,r] + \text{Left}(p_{l+r}, l)\} \tag{4}$$

Having computed table T, entry $T[n/2, n/2]$ corresponds to a label placement of minimum total leader length. In order to compute the actual placement and not only its cost, we need to maintain an additional table T'. For $0 \leqslant l$, $r \leqslant n/2$ entry $T'[l,r]$ stores the side of the rectangle to which the label of site p_{l+r} is attached. This is determined by the term that minimizes (4).

Since the algorithm maintains an $(n/2+1) \times (n/2+1)$ table and each entry is computed in constant time, the time complexity of our algorithm is $O(n^2)$.

To complete the proof of the theorem, we have to show how to deal with the case that n is odd. Assume that $n = 2k-1$ for some $k > 0$. In this case, we will attach k labels to one side of the rectangle and $k-1$ to the other. Since we are using uniform labels, the side which receives $k-1$ labels can be considered to have an unoccupied label slot. Note that the label slot can be on either the left or the right side. The revised dynamic programming algorithm maintains a table $T[0 : n/2, 0 : n/2, 0 : 1, 0 : 1]$ such that $T[l,r,a,b]$ gives the minimum total leader length for the $l+r$ lowest sites where l of them have labels on s_{left}, r on s_{right} and so far we have used a empty label slots on the left side and b empty label slots on the right, with $a, b \in \{0,1\}$ satisfying $a+b \leqslant 1$. The recurrence relation which must be satisfied by T can be easily obtained as an extension of the one for the case of an even number of sites. Finally, observe that the size of the table remains $O(n^2)$, leaving time and space complexity unchanged. □

3.3.2. Four-side labeling with type-opo leaders

We present a polynomial-time algorithm that computes type-*opo* leaders of minimum total length and places labels on all four sides of the boundary of the rectangle R. We assume labels of uniform size and sliding ports.

Before we proceed with the description of our algorithm, we make some observations regarding *opo*-labeling (which might contain crossings) of minimum total leader length for the case of four-side labeling with labels of uniform size and sliding ports. Consider an *opo*-leader c which originates from point p and is connected to a label on side AB of the rectangle at port q (see Fig. 7). The line containing the segment of the leader which is incident to site p (and is orthogonal to side AB) divides the plane into two half-planes. We say that leader c *is oriented towards* corner A of the rectangle if port q and corner A are on the same half-plane, otherwise, we say that leader c *is oriented away from* corner A. In the case where the *opo*-leader consists of only one segment, i.e., the port lies on the line which defines the two half-planes, we consider the leader to be oriented towards corner A (and also towards corner B).

Lemma 2. *Consider four-side labeling with labels of uniform size and sliding ports and let L be an opo-labeling (which might contain crossings) of minimum total leader length. Let c_i and c_j be two leaders that connect sites p_i and p_j to labels l_i and l_j, respectively. If c_i and c_j cross, the following statements hold*:

(i) *Labels l_i and l_j lie on adjacent sides of the rectangle.*
(ii) *Leaders c_i and c_j are oriented towards the same corner of the rectangle.*
(iii) *Leaders c_i and c_j can be rerouted so that they do not cross each other and the sum of their leader lengths remains unchanged.*

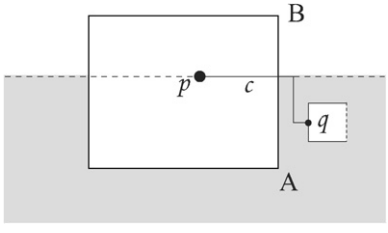


Fig. 7. Leader c is oriented towards corner A and away from corner B.

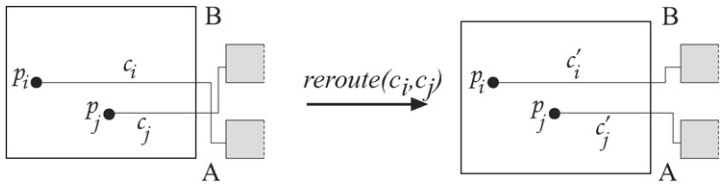


Fig. 8. Rerouting used to prove that in an *opo*-labeling (where crossings are allowed) of minimum total leader length, the labels associated with two crossing leaders do not lie on the same side of the rectangle.

Proof. For proving statement (i), we show labels l_i and l_j cannot both lie on the same side or on opposite sides of the rectangle. For the sake of contradiction, assume first that the labels lie on the same side, say AB, of the rectangle. Then the segments of the leaders which are incident to the sites are parallel to each other. Since the sites have distinct x- and y-coordinates, these segments do not overlap each other, and thus, the intersection of the two leaders takes place outside the rectangle, i.e., in the track-routing area. This implies that, along the direction of side AB, the order of the sites is the reverse of the order of their associated labels. However, by swapping the labels, we can reduce the total leader length (and also eliminate a crossing). This is a contradiction since we assumed that the total leader length of the labeling is minimum (see Fig. 8).

Consider now the case where, for the sake of contradiction, the labels lie on opposite sides of the rectangle. Then, since the leaders intersect each other, the segments of the leaders which are inside the rectangle (and incident to the sites) have to intersect. However, since these segments are parallel to each other, they have to overlap, and thus have the same x- or y-coordinates. This is the desired contradiction since we assume that the sites are in general position. Having eliminated the cases that the labels lie on the same or on opposite sides of the rectangle implies that labels must lie on adjacent sides of the rectangle if their leaders intersect.

Let A be the corner which is incident to the two sides of the rectangle containing the labels associated with leaders c_i and c_j. In order to prove statement (ii) of the lemma, it is enough to show that (in a labeling of minimum total leader length) it is impossible to have one or both leaders oriented away from corner A. We consider these two cases.

Case 1: Exactly one leader, say c_i, is oriented away from corner A.

This case is depicted in the left-hand side of Fig. 9(a). Rerouting the leaders as described in Fig. 9(a) results in a reduction of the total leader length, a contradiction since we assumed that the total leader length of the labeling is minimum. Note that in the figure we only show the sub-case where site p_j is below the horizontal line passing through port q_i. When p_j is on or above the horizontal line passing through port q_i, rerouting again results in a reduction of the total leader length. Thus, a labeling of minimum total leader length does not contain two crossing leaders where one of them is oriented away from the corner A incident to the sides containing their associated labels.

Case 2: Both leaders c_i and c_j are oriented away from corner A.

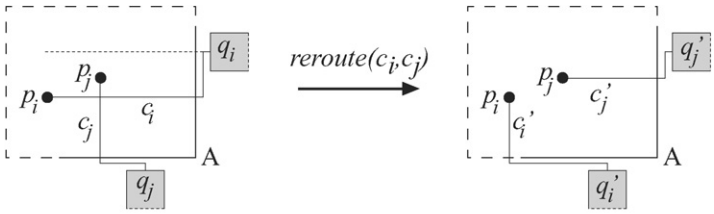
The rerouting of the leaders is shown in Fig. 9(b). Again, only one of the four possible sub-cases based on whether site p_i (p_j) is to the right (below) the vertical (horizontal) line passing through port q_j (q_i) is shown. Given that rerouting results in a reduction of the total leader length, we conclude that a labeling of minimum total leader length does not contain two crossing leaders where both of them are oriented away from the corner A incident to the sides containing their associated labels.

Having eliminated the cases where one or both crossing leaders are oriented away from corner A, implies that they are both oriented towards corner A(assuming that we can identify two crossing leaders).

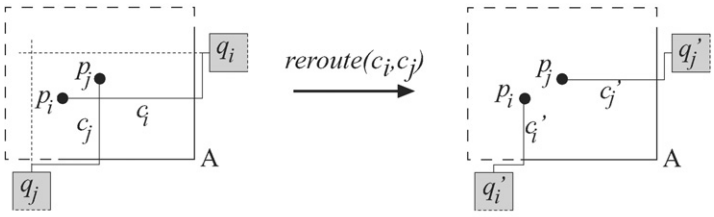
It remains to show statement (iii) of the lemma, namely that leaders c_i and c_j can be rerouted so that they do not cross each other and the sum of their leader lengths remains unchanged. In the rerouting described in Fig. 9(c), we use the crossing point O to partition the first segment of each leader c_i and c_j into two sub-segments. Then, leaders c'_i and c'_j can be obtained by a parallel translation of the (sub)segments of leaders c_i and c_j. This does not change the sum of the leader lengths.

To complete the proof of the lemma, we note that whenever we perform a rerouting, we never change the position of a port. Since the used port would also be available in the case of sliding ports, the lemma applies to sliding ports, as stated. □

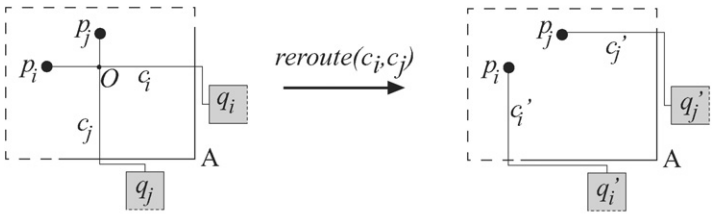
Lemma 3. *Given a set P of n sites and an opo-labeling L of P with uniform labels and sliding ports that has minimum total leader length, there is a crossing-free opo-labeling L' whose total leader length equals that of L. Moreover, labeling L' can be obtained from L in* $O(n \log n)$ *time.*



(a) Leader c_i is oriented away from corner A.



(b) Both leaders c_i and c_j are oriented away from corner A.



(c) Both leaders c_i and c_j are oriented towards corner A.

Fig. 9. In an *opo*-labeling of minimum total leader length, two crossing leaders are always oriented towards the corner that is incident to the rectangle sides with the associated labels. The crossing can be eliminated without changing the sum of the leader lengths.

Proof. We will show how to eliminate all crossings in L by rerouting the intersecting leaders. Our method performs two passes over the sites, one from left-to-right and one from right-to-left.

We first do the left-to-right pass. Consider all sites with labels on the right side of the rectangle which are incident to crossing leaders. Let p be the leftmost such site and let c be the leader that connects p to its corresponding label on the right side of the rectangle (see Fig. 10). Given that L is an *opo*-labeling of minimum total leader length, Lemma 2(i) implies that leader c intersects only leaders that are connected to labels on the top and bottom sides of the rectangle. Without loss of generality, assume that c is oriented towards the bottom-right corner of the rectangle, say A. Then all leaders that intersect c have their labels on the bottom of the rectangle and are also oriented towards A (Lemma 2(ii)). Let c_i be the leftmost leader that intersects c, and let p_i be its incident site. According to Lemma 2(iii), we can reroute leaders c and c_i so that the total leader length remains unchanged (Fig. 10). Observe that the rerouting possibly eliminates more than one crossing (e.g., the crossings between leader c and leaders c_j and c_k) but, in general, it might also introduce new crossings (e.g., the crossings between leaders c_i' and c_k). However, the leftmost site that is (a) incident to an intersecting leader and (b) connected to a label on the right side of the rectangle, now lies to the right of site p. Continuing in this manner, the leftmost site which participates in a crossing (in the left-to-right pass) is pushed to the right, which guarantees that all "left-to-right" crossings are eventually eliminated.

The left-to-right pass eliminates all crossings involving leaders attached to labels on the right side of the rectangle. Now we want to show that during the left-to-right pass we do not introduce any crossings that involve leaders attached to labels on the left side of the rectangle (and have to be examined during a right-to-left pass). This will guarantee that only two passes are required to resolve all crossings. To see this, assume that such a crossing was introduced and that it involves leader c' and the leader c_l which connects site p_l to a label on the left side of the rectangle (Fig. 11). Given that the rerouting does not increase the total leader length, the labeling resulting after all rerouting is still one of minimum total leader length. Then, according to Lemma 2(i), both leaders c' and c_l must be oriented towards corner D, a contradiction since leader c' is oriented away from corner D (and towards corner A).

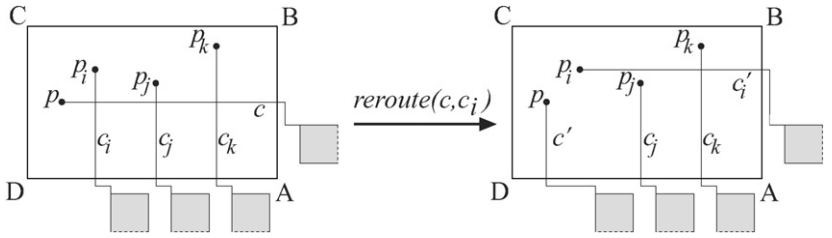


Fig. 10. Rerouting eliminates crossings in an *opo*-labeling of minimum total leader length.

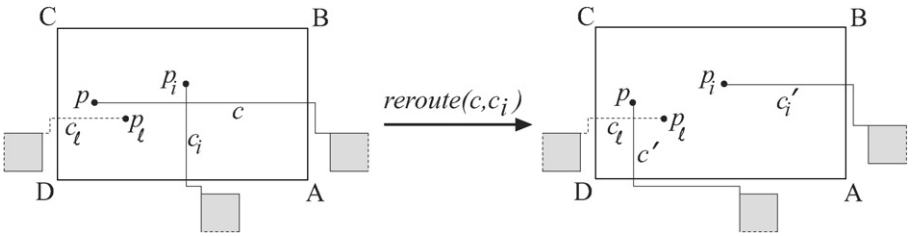


Fig. 11. No right-to-left crossing is introduced during the left-to-right pass described in the proof of Lemma 3.

From the above discussion, it follows that a left-to-right pass eliminating crossings involving leaders with their associated labels on the right side of the rectangle, followed by a similar right-to-left pass, results in a labeling L' without any crossings and of total leader length equal to that of L, i.e., minimum.

To complete the proof of the lemma, it remains to explain how to obtain in $O(n \log n)$ time the new labeling L', given labeling L of minimum total leader length. Consider the left-to-right pass. The analysis for the right-to-left pass is symmetric. During the pass, we process the sites with labels on the right side of the enclosing rectangle in order of increasing x-coordinate. Sorting the sites in increasing order with respect to their x-coordinate can be done in $O(n \log n)$ time.

In order to process site $p = (x_p, y_p)$ and to eliminate the crossings (if any) involving its leader c, we have to identify the leftmost site p_i such that its corresponding leader (say c_i) intersects leader c. Of course, the intersection involves the first segment of leader c_i that is parallel to the y-axis. The processing of the sites during the left-to-right pass can be accomplished by employing a data structure that stores vertical line segments and supports *visibility queries* of the form: given a query point $p_0 = (x_0, y_0)$ return the first line segment to the right of p_0 that is intersected by line $y = y_0$. The same data structure supports *insert* (for initialization) and *delete* operations. For the case of vertical line segments of *finite size*, the visibility query can be answered in $O(\log^2 n)$ time by employing a combination of interval trees and priority search trees [20, p. 211]. This results in $O(n \log^2 n)$ time for the left-to-right pass and, consequently, for the elimination of all crossings. However, as we will show next, the time needed to eliminate all crossings can be further reduced to $O(n \log n)$ if we take into account the fact that all vertical segments considered during the left-to-right pass have one of their endpoints on the bottom or the top side of the enclosing rectangle.

Without loss of generality, assume that leader c is oriented towards the bottom-right corner of the enclosing rectangle. (The case where it is oriented towards the top-right corner can be handled in a symmetric manner.) Then, according to Lemma 2(ii) all leaders intersecting leader c are also oriented towards the bottom-left corner and, thus, their associated labels are placed on the bottom side of the enclosing rectangle. Therefore, leader c can only intersect vertical line segments which have one of their end-points on the bottom side of the enclosing rectangle.

When we have to solve a visibility query on the set of line segments having one of their end-points on the bottom side of the enclosing rectangle, we can relax the restriction that the segments are of finite size and assume that they are semi-infinite rays having their associated site as their higher endpoint. This is due to the fact that all leader intersections take place inside the enclosing rectangle. Recall that r_R denotes the y-coordinate of the right side of the enclosing rectangle R. In the case of semi-infinite segments, the visibility query (with $p_0 = (x_0, y_0)$ as the query point) on set of vertical line segments reduces to finding the site of smallest x-coordinate in the semi-infinite vertical strip defined by $x > x_0$, $y \leqslant y_0$, and $x < r_R$. The query just described can be answered in time $O(\log n)$ by employing a dynamic priority search tree based on half-balanced trees [20, p. 209]. Insertions and deletions are also supported in $O(\log n)$ time.

Thus, identifying the (at most n) pairs of leaders to be rerouted during the left-to-right pass takes only $O(n \log n)$ time, resulting in a total time complexity of $O(n \log n)$ for computing the crossing-free boundary labeling L'. □

Now we are ready to present the main theorem of the section:

Theorem 4. *Consider four-side opo-labeling of n sites with uniform labels. A crossing-free solution of minimum total length can be computed in* $O(n^2 \log^3 n)$ *time in the case of fixed ports and in* $O(n^3)$ *time in the case of sliding ports.*

Proof. For the sake of simplicity let us assume that we attach labels to the right side of R. To compute an assignment that is minimum in terms of total leader length for fixed ports we compute a Manhattan minimum-cost bipartite matching between sites and ports using Vaidya's algorithm [22]. It runs in $O(n^2 \log^3 n)$ time and finds a matching that minimizes the total Manhattan distance of the matched pairs.

Note that this approach fails for sliding ports, although there is a small set of candidate positions for ports, namely the bottom- and topmost points of each left label edge and the horizontal projections of the sites to the corresponding label edges. In a Manhattan minimum-cost bipartite matching each site would be matched to its horizontal projection—even if several such candidate ports were lying on the same left label edge. Instead we compute the complete bipartite graph between sites and labels where the weight of an edge is the Manhattan distance of the site to the closest point on the corresponding label. Computing a minimum-cost bipartite matching in this graph takes $O(n^3)$ time [19].

The leaders induced by this solution may overlap (in the track-routing area). However, by Lemma 3 we can obtain a crossing-free solution in $O(n \log n)$ additional time. □

3.3.3. Type-opo leaders and non-uniform labels

We focus on two-side label placement of type-*opo* leaders. We are given n sites $p_i = (x_i, y_i)$, $i = 1, 2, \ldots, n$, each associated with a label l_i of height h_i which can be placed on either the left side (s_{left}) or the right side (s_{right}) of rectangle R. We assume that the heights of the rectangle and the labels are all integers. Observe that the height of rectangle R must be large enough to accommodate the labels. In the event that the height of rectangle R is equal to half the sum of the label heights, just placing the labels amounts to solving the NP-hard problem PARTITION. Therefore, we cannot expect an algorithm whose running time is polynomial just in n, the number of sites. Instead we present an algorithm whose running time is a polynomial in n and the height H of the rectangle R. This algorithm can be considered the counterpart of the pseudo-polynomial solution for PARTITION.

Here we again ignore the routing of the type-*opo* leaders and assume the existence of a slightly wider rectangle R'. We obtain the following theorem:

Theorem 5. *Given a rectangle R of integral height H, a set* $P \subset R$ *of n sites in general position, where site* p_i *is associated with label* l_i *of integral height* h_i*, there is an* $O(nH^2)$*-time algorithm that places the labels to the right and left side of the rectangle and attaches the corresponding sites with non-intersecting type-opo leaders such that the total leader length is minimized.*

Proof. We say that label l is *placed at height* h if its bottom edge has y-coordinate h. If the ith site p_i is connected to s_{left} and its label l_i is placed at height y then the length of the leader from p_i to l_i leftward is well defined. Call this length $\text{Left}(p_i, y)$ and call the analogously defined right leader length $\text{Right}(p_i, y)$.

We denote by $T[i, \lambda, \rho]$ the total length of the type-*opo* leaders of the i lowest sites, where the left side of the rectangle is occupied up to λ and the right side is occupied up to ρ. By $L[i, \lambda, \rho]$ we denote the total leader length for the case where the ith site has its label on the left side, the left side of the rectangle R is occupied up to y-coordinate λ (including label l_i) and the right side of R is occupied up to ρ. Similarly we define $R[i, \lambda, \rho]$. Then, by induction we can show that the following recurrence relations hold. (We omit the boundary conditions.)

$$T[i, \lambda, \rho] = \min\{L[i, \lambda, \rho], R[i, \lambda, \rho]\} \tag{5}$$

$$L[i, \lambda + h_i, \rho] = T[i - 1, \lambda, \rho] + \text{Left}(p_i, \lambda) \tag{6}$$

$$R[i, \lambda, \rho + h_i] = T[i - 1, \lambda, \rho] + \text{Right}(p_i, \rho) \tag{7}$$

Table T can be computed by dynamic programming. Having computed table T, the desired minimum total leader length is given by $\min_{0<a,b\leqslant H} T[n,a,b]$. We can recover the label placement which realizes the minimum total leader length by maintaining an additional table containing information regarding the routing of the ith leader (to the left or to the right side). The dynamic programming algorithm that computes table T takes $O(nH^2)$ time and space. □

3.3.4. One-side labeling with type-po leaders and uniform labels

In this subsection we first describe how to compute a legal labeling with leaders of type-*po* and uniform labels at fixed positions; see Fig. 2. Then, we show that the computed labeling also minimizes the total leader length. We restrict ourselves to attaching labels to one side s of R. For our description we assume that s is the right vertical side of R, and that the sites $p_1,\ldots,p_n$ are sorted according to increasing y-coordinate.

Our algorithm is very simple: we simply stack labels to the right of s in the same vertical order as the corresponding sites. Then we process the sites (and the corresponding labels) from bottom to top. Assume we have already placed non-intersecting leaders for the first $i-1$ sites. Then, we connect p_i to l_i by a leader c_i of type *po*, i.e., by a vertical segment (possibly of length zero) followed by a horizontal segment. If c_i intersects previously placed leaders,[1] we determine the rightmost site p_j whose leader c_j intersects c_i and reroute as in Fig. 12: we connect p_j to l_i and p_i to l_j. We observe that the new leader c'_j of p_j does not intersect any other leader. This is due to the fact that the vertical part of c'_j is contained in c_j, the horizontal part of c'_j is contained in c_i, and p_j was the rightmost site whose leader intersected c_i. By going through the sites $p_1,\ldots,p_{i-1}$ from right to left (i.e., in order of decreasing x-coordinate), we test their leaders for intersection with c_i and possibly reroute. This is detailed in Algorithm 3, where we refer to a leader as *disturbing* if its horizontal segment intersects other leaders in $\{c_1,\ldots,c_i\}$. To compute a legal *po*-routing, we call Algorithm 3 for $i=2,\ldots,n$. Clearly, each call takes $O(i)$ time, resulting in an $O(n^2)$ algorithm. The correctness rests on our observation above and on the invariant specified in the loop of Algorithm 3. Thus we have the following result:

Theorem 6. *Given a rectangle R, a side s of R, a set $P\subset R$ of n sites in general position and a rectangular uniform label for each site, there is an $O(n^2)$-time algorithm that attaches the labels to s and connects them to the corresponding sites with non-intersecting type-po leaders.*

Our claim that the computed labeling also minimizes the total leader length is based on the following lemma:

Lemma 4. *Rerouting two po-leaders as described in Fig.* 12 *leaves the sum of their lengths unchanged.*

Proof. The length of the horizontal segments does not change. Thus, to prove the lemma, we show that the sum of the lengths of the vertical segments of the two leaders remain unchanged. For the case of fixed ports this is obvious, however, it also holds for the case of sliding ports, assuming that we use as port the point of the left side of the label that is closest to the site. Fig. 12 shows that a crossing can only occur if the ports of both labels lie below both sites or both ports lie above both sites (equal y-coordinates are allowed but due to our assumption concerning general position at most one site and one port can have the same y-coordinates). In the case of sliding ports, the leaders use as ports the top points of the left side of the labels (or, in the symmetric case, the bottom points). After rerouting of the leaders, the same ports are used and the sum of their lengths remains unchanged.

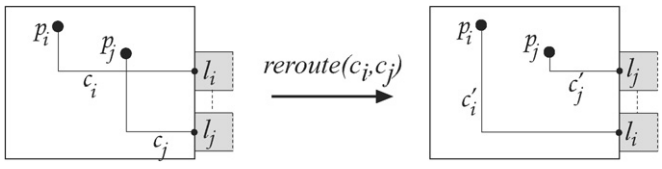


Fig. 12. Rerouting of crossing leaders.

[1] The case where leader c_i passes through a site is treated as an intersection. In the case where the labels are attached to the vertical (horizontal) sides of the rectangle, the site and the port have the same y-coordinate (x-coordinate).

Input: a *po*-placement for sites $p_1, \ldots, p_i$ and their labels such that:

1. Leaders $c_1, \ldots, c_{i-1}$ out of sites $p_1, \ldots, p_{i-1}$ are mutually disjoint, and
2. Leader c_i out of site p_i is the only (possibly) disturbing leader.

Output: a legal *po*-placement for sites $p_1, \ldots, p_i$ and their labels.

Let $p^1, \ldots, p^i$ be an ordering of $p_1, \ldots, p_i$ such that $x(p^1) > \cdots > x(p^i)$.
Let $c^1, \ldots, c^i$ be the corresponding leaders.
Let j be the index with $p^j = p_i$, i.e., c^j is the only (possibly) disturbing leader.
$k = 1$
while $k < i$ **and** $k < j$ **do**
  {there are more leaders to examine for possible intersection with c^j}
  if $c^j \cap c^k \neq \emptyset$ **then** reroute(j, k)
  invariant c^j is the only (possibly) disturbing leader, and c^j does not intersect any of $\{c^1, \ldots, c^k\}$
  $k = k + 1$
end
return $c_1, \ldots, c_i$

Algorithm 3. UNIFORMLABEL1SIDEROUTEPOCROSSINGELIMINATION.

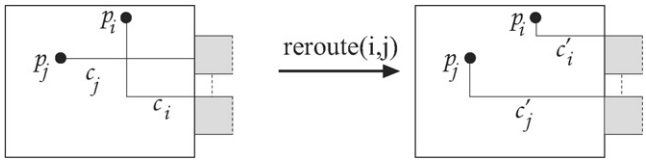


Fig. 13. Rerouting of crossing leaders in the case of sliding ports.

To complete the proof for the case of sliding ports, we have to examine the case where the port is somewhere along the left side of the label. This can happen only in the case that the site can be connected to the label by a horizontal leader (assuming leaders of minimum length), and as it can be easily verified (see Fig. 13) the rerouting still works fine. (However, notice that this case will never occur if the labels are placed in the same order with their corresponding sites.) □

Theorem 7. *Given a rectangle R, a side s of R, a set $P \subset R$ of n sites in general position and a rectangular uniform label for each site, there is an $O(n^2)$-time algorithm that produces a legal type-po labeling of minimum total leader length.*

Proof. The proof of this theorem is based on the algorithm used to construct a legal one-side type-*po* labeling for rectangular uniform labels (see Theorem 6). The algorithm repeatedly invokes the crossing-reducing step; see Algorithm 3. By Lemma 4, the total leader length is left unchanged (for fixed and sliding ports). Thus, the total leader length of the labeling is identical to the total leader length of the initial labeling, before any effort to remove the crossings was made.

Recall that in the initial labeling the ith site is connected with a *po*-leader to the ith label (if we go through both sites and labels from bottom to top). This labeling possibly has crossings. We want to show that it has minimum total leader length. To see this, consider the case where, instead of type-*po* leaders, we use type-*opo* leaders. Observe that the length of a type-*opo* leader connecting a site to its corresponding label does not change when the leader is converted to type-*po* (we ignore the width of the track-routing area). However, as we pointed out in Remark 1 a legal labeling with type-*opo* leaders is unique and, thus, of minimum total leader length. □

3.3.5. Two-side labeling with type-po leaders and uniform labels of maximum height

Our next result deals with two-side placement of uniform labels of maximum height. We consider type-*po* leaders and we again aim to minimize the total leader length. We obtain the following theorem:

Theorem 8. *Given a rectangle R with $n/2$ uniform labels of maximum height on each of its left and right sides, and a set $P \subset R$ of n sites in general position, there is an $O(n^2)$-time algorithm that attaches each site to a label with non-intersecting type-po leaders such that the total leader length is minimized.*

Proof. We use the dynamic-programming algorithm of Theorem 3 for the case of type-*opo* leaders to obtain the label placement of minimum total leader length. It runs in $O(n^2)$ time. As before (proof of Theorem 7), we observe that connecting a site to its label with a type-*opo* or a type-*po* leader requires the same leader length, namely, the Manhattan distance of site and port. So after obtaining the label placement (for type-*opo* leaders) we use type-*po* leaders routed in the way described in Section 3.3.4. Possible crossings of leaders to the same side are resolved as in Section 3.3.4 without changing the total length, while crossings of leaders that go to opposite sides cannot occur. This is due to the fact that swapping labels between a pair of sites with crossing leaders would result in a solution with smaller total leader length, a contradiction since we assume that the original solution minimizes the total leader length. □

4. Straight-line leaders

In this section we investigate straight-line or type-*s* leaders, i.e., we relax the rectilinearity constraint on the leaders. We first give a simple algorithm that computes a legal one-side labeling. Then we show how this algorithm can be improved either in terms of runtime or in terms of total leader length. Finally we describe how it can be applied to four-side labeling.

4.1. One-side labeling

We adopt the scenario of Section 3.1. Let R be the bounding rectangle and let P be the set of sites inside R. We want to attach labels to the right side of R. We assume that labels are uniform and that their heights add up to the height of R. We also assume that the port m_i where the leader is connected to its label l_i is fixed, say m_i is in the middle of the left label edge. Thus the only task is to assign ports to sites such that no two leaders intersect. Let $M = \{m_1, \ldots, m_n\}$ be the ports sorted by y-coordinate from bottom to top. Simple examples show that a bottom-to-top assignment of the sites to the ports might lead to crossing leaders (for example, see Fig. 12).

Lemma 5. *A legal one-side type-s leader-label placement for fixed labels with fixed ports can be constructed in $O(n^2)$ time.*

Proof. For $i = 1, \ldots, n$ we assign to m_i the first unlabeled site $p \in P$ that is hit by a ray r_i that emanates from m_i and is rotated around m_i in clockwise order. Initially r_i is pointing vertically downwards.

We prove correctness by contradiction: if a crossing would occur between the first and second line, the rotating line would have found the second site first and connected it to the first label. A straightforward implementation yields a time complexity of $O(n^2)$ if we perform a linear search for site p each time. □

The time complexity of Lemma 5 can be improved to $O(n \log n)$ by using a semi-dynamic convex-hull algorithm.

Theorem 9. *A legal one-side type-s leader-label placement for fixed labels with fixed ports can be computed in $O(n \log n)$ time.*

Proof. Let CH be the convex hull of $P \cup M$. Note that CH has an edge between the lowest port m_1 and the first site p reached by the rotating ray r_1. This edge is the first leader. Removing p and m_1 from CH yields the next leader and so on. Using a semi-dynamic convex-hull data structure which only supports *deletion* of points [13] yields a total running time of $O(n \log n)$. This algorithm is correct since it mimics the $O(n^2)$-time algorithm in the proof of Lemma 5. □

Now we tackle the optimization problem.

Theorem 10. *A one-side type-s leader-label placement of minimum total leader length for fixed labels can be computed in $O(n^{2+\varepsilon})$ time for any $\varepsilon > 0$ in the case of fixed ports and in $O(n^3)$ time in the case of sliding ports.*

Proof. For fixed ports, we proceed as described in the proof of Theorem 4, except now we use *Euclidean* minimum-cost bipartite matching between sites and ports in the case of fixed ports. This takes $O(n^{2+\varepsilon})$ time [1], where $\varepsilon > 0$ can be chosen arbitrarily small. For sliding ports we again use minimum-cost bipartite matching in an appropriately defined auxiliary graph, which takes cubic time.

In both cases, fixed and sliding ports, the triangle inequality ensures that the leaders corresponding to the matching do not intersect. □

4.2. Four-side labeling

In this subsection, we consider four-side type-s labeling. We describe how to obtain a legal labeling for fixed labels with fixed ports and a minimum total leader length labeling for fixed labels. Our results are extensions of the results on one-side type-s labeling.

Theorem 11. *A legal four-side type-s leader-label placement for fixed uniformly distributed labels of equal size and fixed ports can be computed in* $O(n \log n)$ *time.*

Proof. We partition the rectangle into convex polygons, such that the sites in each polygon can be connected to the labels on the boundary of the polygon using the $O(n \log n)$ one-side routing in the proof of Theorem 9. Note that the only assumption we used about the relative position of sets P and M of sites and ports, respectively, was that M is contained in an edge of the convex hull of $P \cup M$. To make the one-side routing algorithm work, the convex polygons must be chosen such that they contain exactly as many sites as there are labels on their boundary. We construct our partition as follows:

1. Rotate a straight line ℓ through the center of the rectangle R until on each side of ℓ there are exactly $n/2$ sites. Since ℓ is rotated through the center of the rectangle, and the labels are uniformly distributed around the rectangle's boundary, there are always $n/2$ labels on each side of line ℓ. For simplicity, we assume that ℓ intersects the top and bottom side of the rectangle R; see the solid line in Fig. 14.
2. For the left half, we sweep a horizontal line ℓ_{left} from bottom to top until both polygons contain as many sites as there are labels on their boundaries. We proceed similarly for the right half.
3. From each of the corners v_1 to v_4 of R we rotate a line ℓ_i $(1 \leqslant i \leqslant 4)$ which divides the corresponding partitioned area into two adjacent polygons until both contain as many sites as there are labels on their boundaries.

Since we always divide a convex polygon with a straight line, the resulting polygons are also convex. We did not succeed to partition the rectangle into just four convex polygons but we need two polygons for each side which makes eight in total. Moreover, by construction, the number of sites in each polygon exactly equals the number of adjacent labels. Thus, the one-side type-s $O(n \log n)$-time labeling algorithm of Theorem 9 can be applied, leading to an $O(n \log n)$-time algorithm for legal four-side type-s labeling. □

Since the partition procedure is independent of the scheme that assigns the sites to the label ports the quality of the resulting leaders is not always good. A labeling of minimum total leader length can be obtained by using the method based on minimum bipartite matching in a way identical to that for the one-side type-s leader-label placement. Thus, we can state the following theorem:

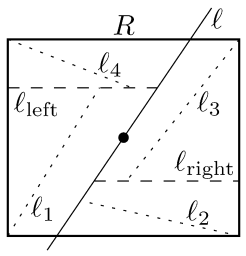


Fig. 14. Partition of R for straight-line leaders.

Theorem 12. *A four-side type-s leader-label placement of minimum total leader length for fixed labels can be computed in* $O(n^{2+\varepsilon})$ *time for any* $\varepsilon > 0$ *in the case of fixed ports and in* $O(n^3)$ *time in the case of sliding ports.*

5. Examples

In this section, we present some characteristic drawings obtained by implementations of the algorithms presented in this paper. Fig. 1 in the introduction shows a map of the city of Karlsruhe boundary-labeled with the names and addresses of kindergartens. Fig. 15 depicts a relatively small medical map of a skeleton. The original labels and leaders are on the right side of the drawing. We have mirrored the sites at the vertical line through the spine and have applied our algorithm (presented in Section 3.1, Theorem 1, Algorithm 1) for type-*opo* leaders such that labels were placed to the left of the drawing and the number of bends is minimum.

Fig. 16 shows two boundary labelings for the map of Italy. We use uniform labels of maximum size placed to the left and the right of the map and we minimize the total leader length. The top labeling uses type-*opo* leaders and was obtained by an implementation of Algorithm 2 (presented in Section 3.3.1, Theorem 3), while the bottom drawing uses type-*po* leaders and was obtained by an implementation of the algorithm presented in Section 3.3.5 (Theorem 8). Although in the type-*po* labeling (bottom figure) the number of bends is smaller and the bends are better distributed, the relative top-to-bottom order between the sites and the labels is not preserved, which might lead to confusion. This order is preserved in type-*opo* labelings.

6. Conclusion

We have defined *boundary labeling* and have presented a series of models and algorithms for efficient boundary labeling of site sets. Originally, we were motivated by a map of the infrastructure network of the Greek school system; see Fig. 17. This example indicates some possible generalizations of our model: graph labeling with objectives like the minimization of crossings between graph edges and leaders.

Fig. 15. A medical map with original labels and leaders (right) as well as labels and type-*opo* leaders computed by our algorithm that minimizes the number of leader bends (left). Drawing from the Internet service of the Vorarlberger Bildungsserver [23].

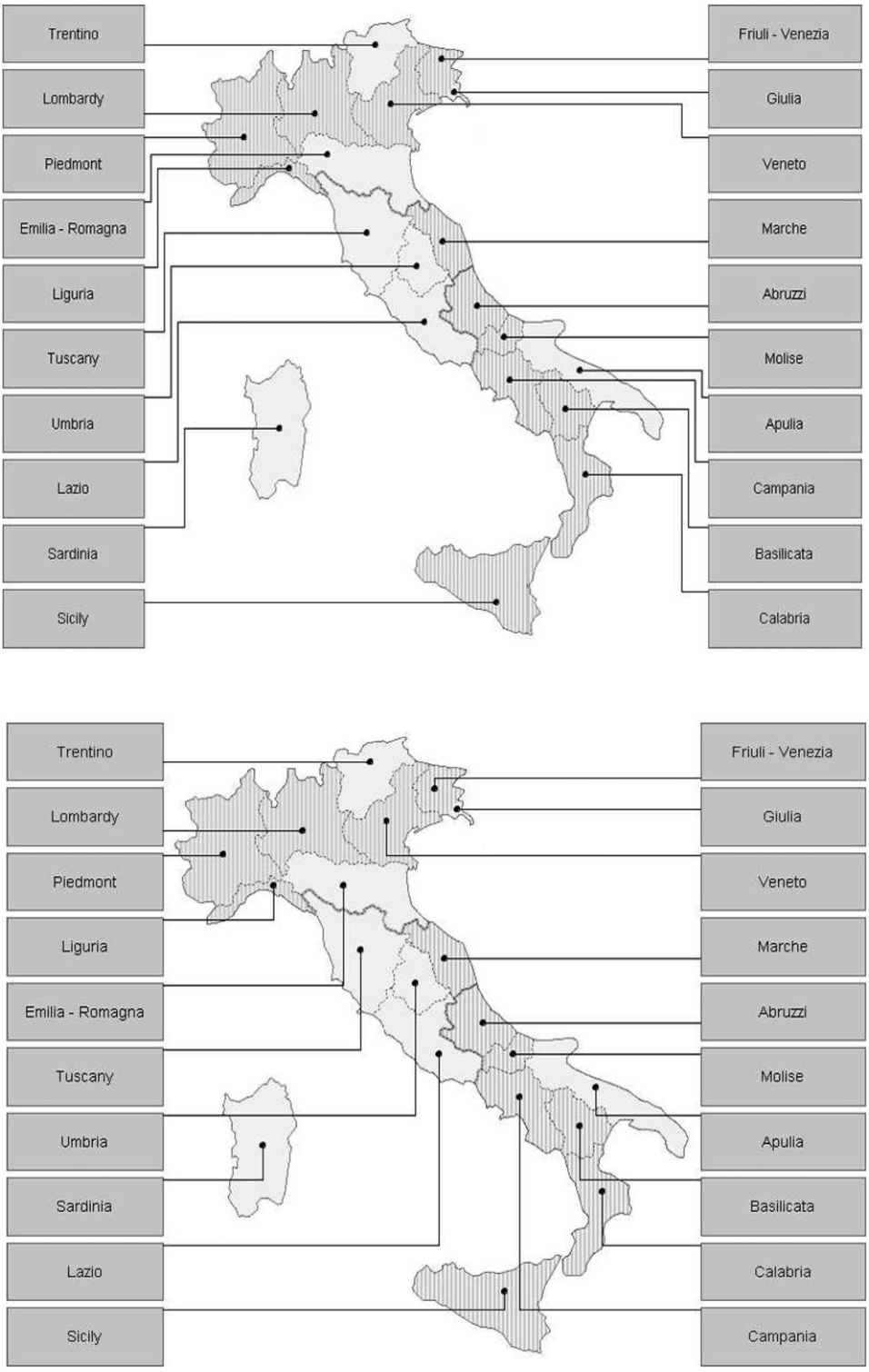


Fig. 16. Two boundary labelings of the map of Italy. We attach labels to opposite sides of the map. We assume uniform labels of maximum size and leaders of type *opo* (top figure) and *po* (bottom figure). The total leader length is minimized in both cases.

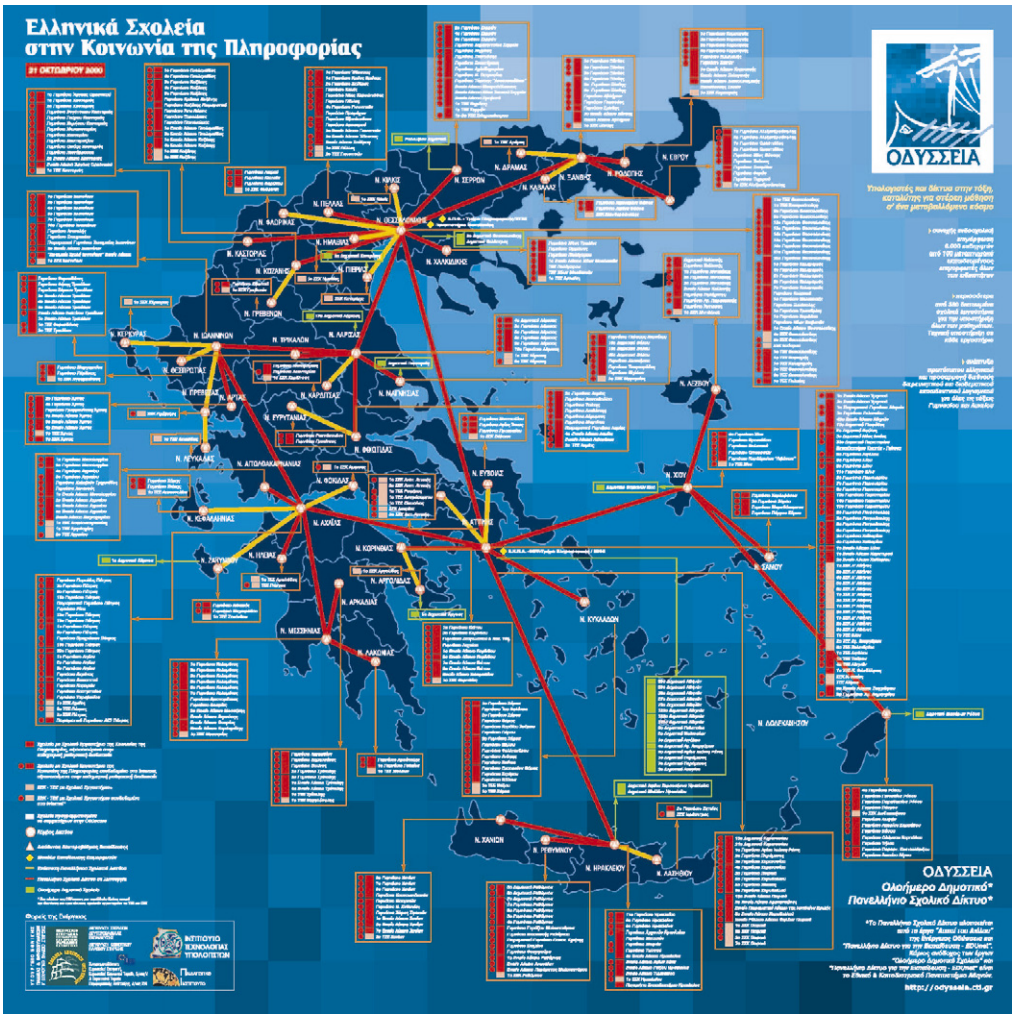


Fig. 17. A map of the infrastructure network of the Greek school system.

Here is a list of interesting open problems:

- The minimum-weight bipartite matching algorithm for four-side labeling is quite powerful, but it is not very efficient in practice.
- The dynamic programming algorithms for two-side labeling (minimizing the total leader length as well as the number of bends) should be generalized to three and four boundary sides.
- The examples for type-*opo* and type-*po* leaders show advantages and also some disadvantages of both types. A practical solution may be to mix both types in order to cope with disadvantages while keeping advantages.
- Type-*opo* minimum-bend routing can be computed efficiently for the case where labels are attached only to one side of the enclosing rectangle. What about the cases where the labels are placed on the two opposite sides, or on all four sides?
- In all of the type-*opo* drawings presented in the paper, the p-segment of the leaders is routed inside the track-routing area. Can this restriction be relaxed, i.e., can the routing of the leaders be done exclusively within the rectangle?

Acknowledgements

We thank Pankaj Agarwal for information about geometric matching and both anonymous referees for their very helpful comments.

References

[1] P.K. Agarwal, A. Efrat, M. Sharir, Vertical decomposition of shallow levels in 3-dimensional arrangements and its applications, SIAM Journal on Computing 29 (3) (1999) 912–953.

[2] J. Ahn, H. Freeman, AUTONAP—an expert system for automatic map name placement, in: Proc. International Symposium on Spatial Data Handling (SDH'84), 1984, pp. 544–569.

[3] Arbeiterwohlfahrt Kreisverband Karlsruhe-Stadt e.V., http://www.awo-karlsruhe.de/sides/dienste/map3.htm, Accessed 2005-11-08.

[4] C. Binucci, W. Didimo, G. Liotta, M. Nonato, Orthogonal drawings of graphs with vertex and edge labels, Computational Geometry: Theory and Applications 32 (2) (2005) 71–114.

[5] G. Booch, J. Rumbaugh, I. Jacobson, The Unified Modeling Language User Guide, Addison-Wesley, Reading, MA, 1999.

[6] B. Chazelle, 36 co-authors, The computational geometry impact task force report, in: B. Chazelle, J.E. Goodman, R. Pollack (Eds.), Advances in Discrete and Computational Geometry, vol. 223, American Mathematical Society, Providence, RI, 1999, pp. 407–463.

[7] H. Freeman, S. Marrinan, H. Chitalia, Automated labeling of soil survey maps, in: Proc. ASPRS–ACSM Annual Convention, Baltimore, vol. 1, 1996, pp. 51–59.

[8] J.-D. Fekete, C. Plaisant, Excentric labeling: Dynamic neighborhood labeling for data visualization, in: Proc. Conference on Human Factors in Computer Systems (CHI'99), ACM, New York, 1999, pp. 512–519.

[9] M. Formann, F. Wagner, A packing problem with applications to lettering of maps, in: Proc. 7th Annual ACM Symposium on Computational Geometry (SoCG'91), 1991, pp. 281–288.

[10] M.Á. Garrido, C. Iturriaga, A. Márquez, J. Ramon Portillo, P. Reyes, A. Wolff, Labeling subway lines, in: P. Eades, T. Takaoka (Eds.), Proc. 12th Annual International Symposium on Algorithms and Computation (ISAAC'01), in: Lecture Notes in Computer Science, vol. 2223, Springer-Verlag, Berlin, 2001, pp. 649–659.

[11] M.R. Garey, D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, W.H. Freeman, New York, 1979.

[12] S.A. Hirsch, An algorithm for automatic name placement around point data, The American Cartographer 9 (1) (1982) 5–17.

[13] J. Hershberger, S. Suri, Applications of a semi-dynamic convex hull algorithm, BIT 32 (1992) 249–267.

[14] C. Iturriaga, A. Lubiw, Elastic labels around the perimeter of a map, Journal of Algorithms 47 (1) (2003) 14–39.

[15] C. Iturriaga, Map labeling problems, PhD thesis, University of Waterloo, 1999.

[16] G.W. Klau, P. Mutzel, Automatic layout and labelling of state diagrams, in: W. Jäger, H.-J. Krebs (Eds.), Mathematics—Key Technology for the Future, Springer-Verlag, Berlin, 2003, pp. 584–608.

[17] S. Kwon Kim, C.-S. Shin, T.-C. Yang, Labeling a rectilinear map with sliding labels, International Journal of Computational Geometry and Applications 11 (2) (2001) 167–179.

[18] K.G. Kakoulis, I.G. Tollis, A unified approach to automatic label placement, International Journal of Computational Geometry and Applications 13 (1) (2003) 23–59.

[19] E.L. Lawler, Combinatorial Optimization: Networks and Matroids, Holt, Rinehart & Winston, New York, 1976.

[20] K. Mehlhorn, Data Structures and Algorithms 3: Multi-dimensional Searching and Computational Geometry, EATCS Monographs on Theoretical Computer Science, vol. 3, Springer-Verlag, Heidelberg, 1984.

[21] J.L. Morrison, Computer technology and cartographic change, in: D.R.F. Taylor (Ed.), The Computer in Contemporary Cartography, Johns Hopkins University Press, 1980.

[22] P.M. Vaidya, Geometry helps in matching, SIAM Journal on Computing 18 (1989) 1201–1225.

[23] Vorarlberger Bildungsserver, http://www.vobs.at/bio/a-phys/pdf/a-skelett-a.jpg, Accessed 2004-07-22.

[24] A. Wolff, T. Strijk, The map-labeling bibliography, http://i11www.ira.uka.de/map-labeling/bibliography/, 1996.

[25] S. Zoraster, The solution of large 0–1 integer programming problems encountered in automated cartography, Operations Research 38 (5) (1990) 752–759.

[26] S. Zoraster, Practical results using simulated annealing for point feature label placement, Cartography and GIS 24 (4) (1997) 228–238.