Boundary Labelling: Models and Efficient Algorithms for Rectangular Maps*

Michael A. Bekos† Michael Kaufmann‡ Antonios Symvonis†
Alexander Wolff§

Technical Report 2004-15
Fakultät für Informatik, Universität Karlsruhe

Abstract

In this paper, we present *boundary labelling*, a new approach for labelling point sets with large labels. We first place disjoint labels around an axis-parallel rectangle that contains the point sites. Then, we connect each label to its site such that no two connections, so-called *leaders*, intersect. Such an approach is common e.g. in technical drawings and medical atlases, but so far the problem has not been studied in the literature. The new problem is interesting in that it is a mixture of a label-placement and a graph-drawing problem.

We consider attaching labels to one, two or all four sides of the rectangle. We investigate rectilinear and straight-line leaders. We present simple and efficient algorithms that minimize the total length of the leaders or, in the case of rectilinear leaders, the total number of bends.

Keywords

Automated label placement, boundary labelling, straight-line and rectilinear leaders.

*This work has been partially supported by the DFG grants Ka 512/8-2 and WO 758/4-1, 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%).

†National Technical University of Athens, Dept. of Mathematics, Athens, Greece. Email: { mikebekos | symvonis }@math.ntua.gr

‡University of Tübingen, Institute for Informatics, Sand 13, 72076 Tübingen, Germany. Email: mk@informatik.uni-tuebingen.de

§Fakultät für Informatik, Universität Karlsruhe, P.O. Box 6980, 76128 Karlsruhe, Germany. WWW: http://i11www.ira.uka.de/people/awolff

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 labelled to convey information. The interest 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 [7], cartographers, graph drawers, and computational geometers have suggested numerous approaches, such as expert systems [2], zero-one integer programming [22], approximation algorithms [7], simulated annealing [23] and force-driven algorithms [12] to name only a few. An extensive bibliography about label placement can be found at [21]. The ACM Computational Geometry Impact Task Force report [5] denotes label placement as an important research area. Manually labelling a map is a tedious task that is estimated to take 50% of total map production time. [19].

In this paper, we deal with labelling dense point sets with comparatively large labels. This is common 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. 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 either allow to attach labels to one, two or all four sides of R, and we either use straight-line or rectilinear leaders. We propose efficient algorithms that find *some* non-intersecting leader-label placement, but we also consider two natural objectives: minimize the total length of the leaders and, if leaders are not straight lines, minimize the total number of bends over all leaders.

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 [4]. Similar layout problems arise when labels are placed *after* the layout of the graph structure [15]. The reason might be that the layout algorithm does not support immediate labelling 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 [17] have coined the term "graph labelling" 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. [3] who additionally allow edge labels.

Leaders have so far only been used by Zoraster [23] and Freeman et al. [8]. Zoraster [23] investigates the labelling of seismic survey maps. Such maps are special in that the sites typically lie on a few seismic lines, which also must be labelled, 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 favours labels that are close to the object they annotate.

Freeman et al. [8] present ALPS, a software system for automated labelling 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 unlabelled. Label positions are determined by an iterative raster-based method.

An example of interactive label placement is the widely used infotip mechanism, which consists of supplying the user with additional information about screen objects whenever the mouse pointer rests a certain amount of time in their vicinity. Fekete and Plaisant [6] extend the infotip paradigm to cope with labelling 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 labelling 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 [13] 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 labelling problem. In Section 3, we are concerned with rectilinear leaders. We present algorithms for legal leader-label placement, leader-bend minimization, 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 modelling 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 but touch the boundary of R.
3. Leader c_i connects site p_i with label l_i for $1 \leq i \leq n$.
4. Intersections of leaders with other leaders, sites or labels are not allowed.
5. The *ports* where leaders touch labels may be prescribed (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 *labellings*) 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 [23], 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 straight-line. Clearly, minimizing the number of bends does not make sense for straight-line leaders.

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 *opo* and *po*, see Figures 1 and 2, respectively. For each type-*opo* leader we further insist that the parallel p-segment is immediately outside the bounding rectangle R and is routing in the so-called *track routing area*. We consider type-o leaders to be of type *opo* and of type *po* as well. In accordance with this notation we refer to straight-line leaders as type-s leaders, see Figure 3.

In 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.

We start with a negative result. Assume that not all label heights are equal, that labels must be attached either to the right or left side of the rectangle R,

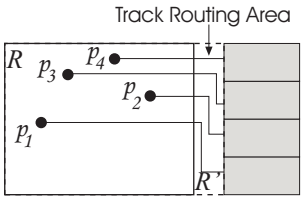


Figure 1: Type-*opo* leaders.

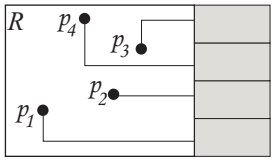


Figure 2: Type-*po* leaders.

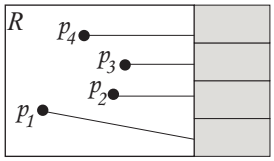


Figure 3: Type-*s* leaders.

and 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 [9]. Due to this observation we will often assume that labels are uniform.

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 *opo* and *po*, see Figures 1 and 2, respectively.

The algorithms which deal with the routing of type-*opo* leaders, are mainly considered with label placement and ignore the routing of the *opo* leaders. The routing of the leaders is done by assuming a rectangle R' broader than R and using the area of R' which is not part of R as a fixed width area, called *track routing area*, in which the routing of all leader segments which are *parallel to the side of the rectangle* is performed. If the track routing area is not of fixed width, then the algorithms become more complicated and have to take into account the width of the track area, resulting in an $O(n)$-time slowdown (which is the worst case track width) of most type-*opo* algorithms which are presented in this section.

3.1 Leader-bend minimization

3.1.1 One-side labelling with type-*opo* leaders.

We consider the problem of attaching labels of variable height to one, say the right, side of the rectangle R. We consider the case of sliding ports, i.e., the leader connecting the site to the label has to simply touch some point in the perimeter of the 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 labelling 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 them to the corresponding sites with non-intersecting type-opo leaders.*

Proof. By construction. 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 their corresponding sites) immediately to the right of the track routing area (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 Figure 1. The $O(n \log n)$ time complexity is due to the assumption that the sites are sorted according to increasing y-coordinate. □

Note 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 labelling. Thus, the algorithm described in the proof of Lemma 1 also yields a labelling 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. [16] have observed that it is trivial to decide whether a set of points on a line can be labelled 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 labelled 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*

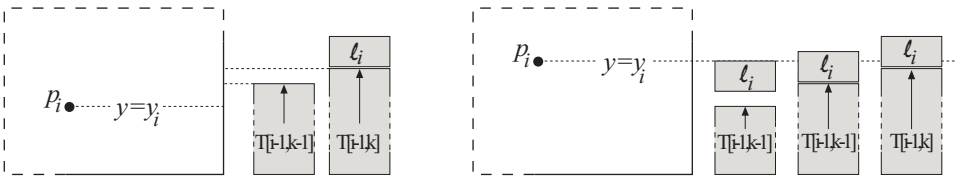


Figure 4: Label placements that the dynamic programming algorithm takes into account when computing $T[i, k]$.

$O(n^2)$-time algorithm that attaches the labels to s and connects them to the corresponding sites with non-intersecting type-opo leaders 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 i-th site, by h_i we denote the height of the i-th label, $1 \leq i \leq n$, 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 (see Algorithm 1) employs a table T of size $(n + 1) \times (n + 1)$. For $0 \leq k \leq i \leq n$ 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:

$$T[i, k] \leq T[i, k + 1] \tag{1}$$

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 $y_i \leq T[i - 1, k - 1]$.

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

Refer to Figure 4(a). In the case where $y_i \leq T[i - 1, k - 1]$ 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 k^{th} horizontal leader.

So, $T[i, k]$ 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 get 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 "$\infty + h_i = \infty$", both subcases can be described by the equation:

$$T[i, k] = T[i - 1, k] + h_i \tag{2}$$

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

Refer to Figure 4(b). Consider first the subcase where $y_i \leq T[i - 1, k - 1] + h_i$. 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 with a horizontal leader with its corresponding label, then $T[i,k] = \max\big(y_i, T[i-1,k-1] + h_i\big)$.

If $T[i-1,k]$ is finite, 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:

$$T[i,k] = \min\Big(T[i-1,k] + h_i, \max\big(y_i, T[i-1,k-1] + h_i\big)\Big) \qquad (3)$$

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

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

Algorithm 1 computes the maximum possible number of horizontal leaders. A placement of maximum number of horizontal leaders has minimum number of bends. The algorithm is directly based on the recurrence relation computed above (see block 1 of the algorithm). Block 2 of Algorithm 1 computes the maximum possible number of zero-bend leaders by identifying the largest j, $0 \le j \le n$, such that $T[n,j] \le t_R$, that is, all labels fit on the side of rectangle R.

It is obvious that Algorithm 1 runs in $O(n^2)$ time using $O(n^2)$ space. By using an extra table of the same size as T, the algorithm can be modified such that it computes the label and leader positions of an optimal solution.

□

3.2 Legal leader-label placement

3.2.1 Four-side labelling with type-*opo* leaders.

Our approach for attaching labels to all sides of the rectangle R is very simple. We partition R into four disjoint regions such that the algorithms 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. If we manage to find a partition of R into four regions such that each region A contains the side s_A of R and A is monotone in the direction of s_A then obviously both requirements are fulfilled.

We introduce some notation, see Figure 5. 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

Algorithm 1: 1SideRouteOPO

Input: n sites $p_i = (x_i, y_i),\ i = 1, 2, \ldots, n$, inside rectangle R and, for each site p_i, a label l_i of height h_i. The sites are sorted in order of increasing y-coordinate. The top and the bottom sides of rectangle R lie on lines $y = t_R$ and $y = t_B$, respectively.
In order to keep the description of the algorithm simple, we assume that access to table entries which are meaningless (for example, $T[i-1, i]$) is possible and yields the value of ∞. We also assume that adding any number to ∞ is possible and yields the value of ∞.
Output: The maximum possible number of horizontal leaders.

$T[0,0] = b_R$
for $i = 1$ **to** n **do**
  $T[i,0] = T[i-1,0] + h_i$
  for $k = 1$ **to** i **do**
1   **if** $T[i-1,k-1] \geq y_i$ **then**
      $T[i,k] = T[i-1,k] + h_i$
    else
      $T[i,k] = \min\Big(T[i-1,k] + h_i, \max\big(y_i, T[i-1,k-1] + h_i\big)\Big)$

2 $j = 0$
  while $T[n,j] \leq t_R$ **do**
    $j = j + 1$
  return $j - 1$

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 an NP-hard partition problem 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 respective sides 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 Figure 5.

2. Now we split A_{12} into two regions A_1 and A_2 such that A_1 contains side s_1 and n_1 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 Figure 5.

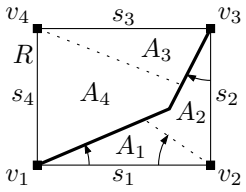


Figure 5: Partition into monotone regions.

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 the previous subsection 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.*

In general we will get less 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

We focus on obtaining 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 labelling 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, and hence their position at each side is determined. We assume, again, that 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 i-th site p which is assigned to s_{left} is connected to the i-th label of s_{left} with a type-*opo* leader. Since the location of each label is determined (and fixed) the length of the leader to the i-th label of s_{left} is defined. We denote it by $\text{Left}(p, i)$. Similarly, we denote by $\text{Right}(p, i)$ the length of the type-*opo* leader which connects site p with the i-th label of s_{right}. Note that the distance functions $\text{Left}(p, i)$ and $\text{Right}(p, i)$ are appropriately defined to incorporate fixed and sliding ports. In the case of fixed ports the distance of the label port closer to the site is computed while in the case of sliding port the functions compute the distance of the label point (sliding port) closer to the site. We obtain the following result:

Theorem 3 *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-opo leaders such that the total leader length is minimized.*

Proof. To compute a label placement of minimum total leader length, we use a dynamic programming algorithm (see Algorithm 2). Without loss of generality, assume that n is even. We will describe later on how to deal with the case where 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 where l of them have labels on s_{left} and the rest r on s_{right}.

For the correctness of the algorithm, it can easily be proven by induction that $T[l, r]$ satisfies the following recurrence relation for $l, r \leq n/2$:

$$
\begin{aligned}
T[0,0] &= 0 && (4)\\
T[0,r] &= T[0,r-1] + \text{Right}(p_r, r) && (5)\\
T[l,0] &= T[l-1,0] + \text{Left}(p_l, l) && (6)\\
T[l,r] &= \min\{T[l,r-1] + \text{Right}(p_{l+r}, r),\ T[l-1,r] + \text{Left}(p_{l+r}, l)\} && (7)
\end{aligned}
$$

Having computed table T, entry $T[n/2, n/2]$ corresponds to a label placement of minimum total leader length. In order to recover the actual placement

and not only the cost of it, we need to maintain an additional table T'. For $0 \leq l, r \leq n/2$ entry $T'[l, r]$ contains information regarding the side of the rectangle which contains the label connected to site $p_{\,l+r}$ and is determined by the term which was minimum in Equation 7.

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 $a + b \leq 1$. The recurrence relation which must be satisfied by T can be easily obtained as an extension of the one for the case of even number of sites. Finally, observe that the size of the table remains $O(n^2)$, leaving the time and the space complexity unchanged. □

Algorithm 2: UniformLabel2SideRouteOPO

Input: Even number of n sites $p_i = (x_i, y_i), i = 1, 2, \ldots, n$, sorted in order of increasing y-coordinate.
In order to keep the description of the algorithm simple, we assume that access to table entries which do not exist, for example $T[-1, 1]$ or $T[1, -1]$, is possible and yields the value of ∞.

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

3.3.2 Four-side labelling with type-*opo* leaders.

We present a polynomial-time algorithm which finds type-*opo* leaders of minimum total length when the labels can be placed on all four sides of the boundary of the rectangle. We assume labels of uniform size and sliding ports.

Before we proceed with the description of our algorithm we make some observations regarding *opo*-labelling (which might contain crossings) of minimum total leader length for the case of four-side labelling with labels of uniform size and sliding ports. Consider an *opo*-leader c which originates from point p and is connected with a label on side AB of the rectangle at port q (see Figure 6). 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 that leader c *is oriented towards* corner A of the rectangle if port q and corner

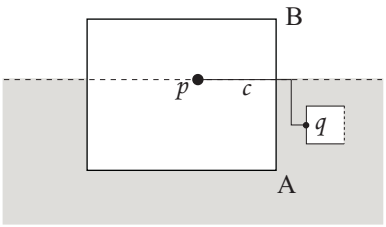


Figure 6: Orientation of a leader with respect to a corner. Leader c is oriented *towards* corner A and *away of* corner B.

A are on the same half-plane, otherwise, we say that leader c *is oriented away of* 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 labelling with labels of uniform size and sliding ports and let L be an opo-labellings (which might contain crossings) of minimum total leader length. Let c_i and c_j be two leaders originating from sites p_i and p_j, respectively, which cross each other. Then it holds:*
(i) The labels associated with leaders c_i and c_j are located at 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.

Proof. Showing that *"the labels associated with leaders c_i and c_j are located at adjacent sides of the rectangle"* is easy. We simply have to show that it is not possible to have the labels located at the same side or 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 (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), a contradiction since we assumed that the total leader length of the labelling is minimum (see Figure 7).

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, a 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, assuming that we can identify two crossing leaders, their associated labels lie on adjacent sides of the rectangle.

Let A be the corner which is incident to the two sides of the rectangle

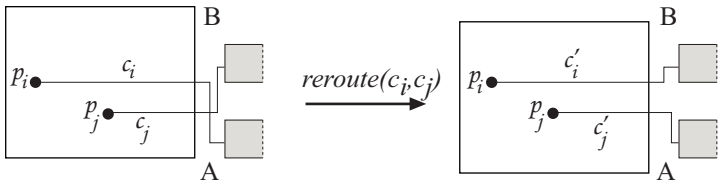


Figure 7: Rerouting used to prove that in an *opo*-labelling (where crossings are allowed) of minimum total leader length, the labels associated with two crossing leaders do not lie on the same side of the rectangle.

containing the labels associated with leaders c_i and c_j. In order to show that in a labelling of minimum total leader length both *"leaders c_i and c_j are oriented towards the same corner"*, it is enough to show that (in a labelling of minimum total leader length) it is impossible to have one or both leaders oriented away of corner A. We proceed to consider these two cases.

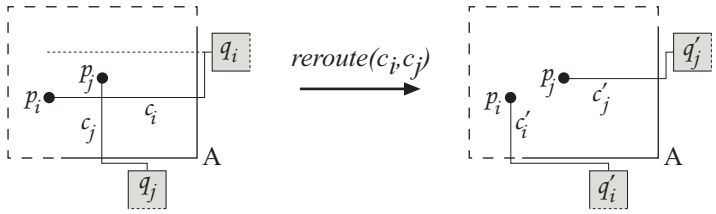
Case 1: Exactly one leaders, say c_i, is oriented away of corner A. This case is described in the left-hand side of Figure 8.a. Rerouting the leaders as described in Figure 8.a results in a reduction of the total leader length, a contradiction since we assumed that the total leader length of the labelling 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 labelling of minimum total leader length does not contain two crossing leaders where one of them is oriented away of the corner corner A incident to the sides containing their associated labels.

Case 2: Both leaders c_i and c_j are oriented away of corner A. When both leaders are oriented away of corner A, rerouting results in higher reduction of the total leader length, compared to Case 1 where only one leader was oriented away of corner A. The rerouting of the leaders is described in Figure 8.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 to reduction of the total leader length, we conclude that a labelling of minimum total leader length does not contain two crossing leaders where both of them are oriented away of the corner corner A incident to the sides containing their associated labels.

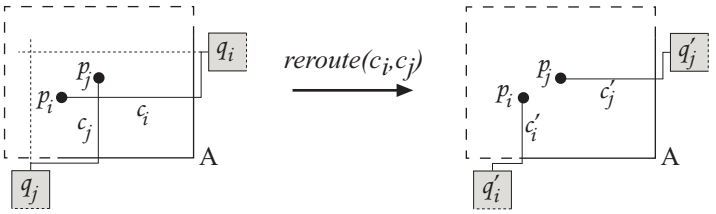
Having eliminated the cases where one or both crossing leaders are oriented away of corner A, implies that (assuming that we can identify two crossing leaders) they are both oriented towards corner A.

Showing 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"* is easy. In the rerouting described in Figure 8.c, 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, leaving their sum unchanged.

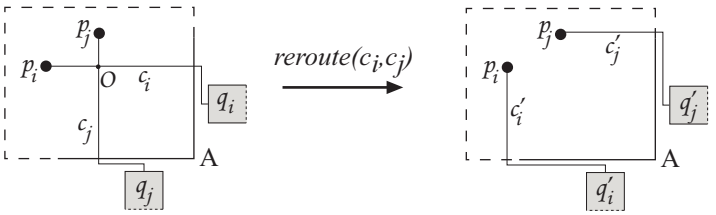
To complete the proof of the lemma, we note that in whenever we performed a rerouting, we never changed the position of a port. So, since the used port would also be available in the case where the sliding-port model is used, the lemma applies to sliding ports, as stated. □



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



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



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

Figure 8: Rerouting used to prove that in an *opo*-labelling (where crossings are allowed) of minimum total leader length, two crossing leaders are oriented towards the corner incident to the sides of the rectangle containing the associated labels and that their crossing can be eliminated without reducing the sum of their leader length.

Lemma 3 *Consider opo-labelling of n sites with uniform labels and sliding ports where crossings are allowed. Then, given a labelling L of minimum total leader length, we can always identify a crossing-free opo-labelling L' with total leader length equal to that of L. Moreover, labelling L' can be obtained in $O(n \log n)$ time.*

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 in the left-to-right and one in the right-to-left direction.

Consider first the left-to-right pass. In the left-to-right pass of labelling L, we consider all sites with labels on the right side of the rectangle. We examine the sites in order from left-to-right and focus only on those which are incident to crossing leaders. Let p be the leftmost such site and let c be the leader that connects it with its corresponding label on the right side of the

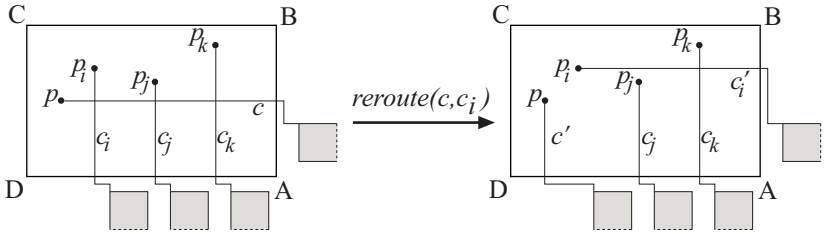


Figure 9: Rerouting used to eliminate crossings in an *opo*-labelling of minimum total leader length. The crossings to be eliminated are identified in a left-to-right pass of the sites, followed by a right-to-left pass. See proof of Lemma 3.

rectangle (see Figure 9). Given that L is an *opo*-labelling of minimum total leader length, Lemma 2 (i) implies that leader c intersects only with leaders that are connected with 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 (Figure 9). 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 total number of crossings is reduced and, more importantly, the leftmost site incident to an intersecting leader connected to a label on the right side of the rectangle is located to the right of site p. Continuing in the same 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.

Another important property is that *"it is impossible to introduce during the left-to-right pass any crossing that has to be examined during a right-to-left pass"*. 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 (Figure 10). Given that the rerouting does not increase the total leader length, the labelling 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 of corner D (and towards corner A).

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 labelling L' without any crossings and of total leader length equal to that of L, that is, minimum.

To complete the proof of the theorem, it remains to explain how to obtain in $O(n \log n)$ time the new labelling L', given labelling 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 $O(n \log n)$

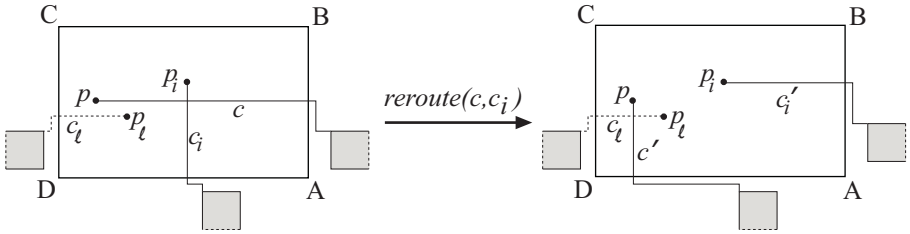


Figure 10: It is impossible to introduce any right-to-left crossing during the left-to-right pass described in the proof of Lemma 3.

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 storing "vertical line segments" and supporting *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 $y = y_0$"*, as well as *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 [18, pp. 211]. This results in a total of $O(n \log^2 n)$ time for the left-to-right pass and, consequently, for the elimination of all crossings. However, 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. So, 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 $x > x_0$, $y \leq y_0$, and $x < r_R$. The *MinXinRectangle* query just described can be answered in time $O(\log n)$ by employing a dynamic priority search tree based on half-balanced trees [18, pp. 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 to a total time complexity of $O(n \log n)$ for the production of the crossing free boundary labelling L'. □

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

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

Proof. Let M be the set of the n labels around the boundary of the rectangle. We construct a complete bipartite graph $G = (P \cup M, E)$ between all the sites $p \in P$ and all the labels $m \in M$, with edge weights to be the Manhattan length of the corresponding leaders. Note that the length of each leader depends on the type of the port. For the case of sliding ports, the leader typically connects the site to one of the corners of the label. We proceed by applying the Vaidya's algorithm [20] for minimum-cost bipartite matching for points in the plane under the Manhattan metric. It runs in $O(n^2 \log^3 n)$ time and finds a matching between sites and labels that minimizes the total Manhattan distance of the matched pairs. The leaders in the produced solution might overlap. However, based on 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. 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, managing to place the labels accounts to solving the partition problem. So, we cannot expect an algorithm that runs in polynomial time only to the number of sites. Instead we get an algorithm that runs in polynomial time to the height of rectangle R, which can be considered to be the equivalent of the pseudo-polynomial solution to the partition problem.

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

Proof. We say that label l is *placed at height h* if its bottom edge has y-coordinate h. Assume that the i-th 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 defined. Call it Left(p_i, y). Similarly, we define Right(p_i, y).

We denote by $T[i, y_L, y_R]$ the total length of the type-*opo* leaders of the i lowest sites, where the left side of the rectangle is occupied up to y_L and right side is occupied up to y_R. By $T_L[i, y_L, y_R]$ we denote the total leader length for the case where the i-th site has its label on the left side, the left side of the rectangle is occupied up to y_L (including label l_i) and right side is occupied up to y_R. Similarly we define $T_R[i, y_L, y_R]$. Then, by induction we can show that the following recurrence relations hold (we omit the boundary conditions):

$$T[i, y_L, y_R] = \min\{T_L[i, y_L, y_R],\ T_R[i, y_L, y_R]\} \quad (8)$$

$$T_L[i, y_L + h_i, y_R] = T[i-1, y_L, y_R] + \text{Left}(p_i, y_L) \quad (9)$$

$$T_R[i, y_L, y_R + h_i] = T[i-1, y_L, y_R] + \text{Right}(p_i, y_R) \quad (10)$$

Table T can be computed by a dynamic programming algorithm (see Algorithm 3). Having computed table T, the minimum table entry of the form $T[n, a, b]$, where $0 < a, b \leq H$ is the minimum total leader length. We can recover the label placement which realizes the computed total leader length by maintaining an additional table containing information regarding the routing of the i-th leader (to the left or to the right side). Algorithm 3 terminates in $O(nH^2)$-time and uses $O(nH^2)$ space. □

Algorithm 3: NonUniformLabel2SideRouteOPO

Input: n sites $p_i = (x_i, y_i),\ i = 1, 2, \ldots, n$, inside rectangle R of height H and, for each site p_i, a label l_i of height h_i. The sites are sorted in order of increasing y-coordinate.
In order to keep the description of the algorithm simple, we assume that all the necessary initializations have been done and that access to table entries which do not exist (for example, $T[-1, 0, 0]$) is possible and yields the value of ∞. For the same reason we do not make any effort to avoid meaningless assignments (for example, computing $T_L(i, H+1, 0)$ or $T_R(i, 0, H+1)$).

```
for i = 1 to n do
  for y_L = 0 to H do
    for  y_R = 0 to H do
      T_L[i, y_L + h_i, y_R] = T[i − 1, y_L, y_R] + Left(p_i, y_L)
      T_R[i, y_L, y_R + h_i] = T[i − 1, y_L, y_R] + Right(p_i, y_R)

  for y_L = 0 to H do
    for  y_R = 0 to H do
      T[i, y_L, y_R] = min{T_L[i, y_L, y_R], T_R[i, y_L, y_R]}
```

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

In this subsection we first describe how to compute a legal labelling with leaders of type-*po* and uniform labels, see Figure 2. Then, we show that the computed

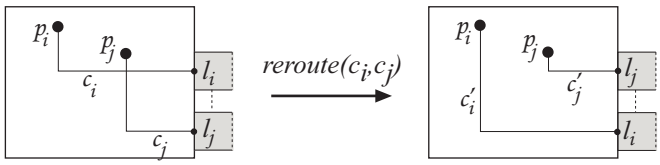


Figure 11: Rerouting of crossing leaders.

labelling 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 Figure 11: 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 4, 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 4 for $i = 2, \ldots, n$. Clearly each call takes $O(i)$ time, resulting to an $O(n^2)$ algorithm. The correctness rests on our observation above and on the invariant specified in the loop of Algorithm 4. 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 labelling also minimizes the total leader length is based on the following lemma:

Lemma 4 *Rerouting two po-leaders as described in Figure 11 leaves the sum of their lengths unchanged.*

Proof. The size of the horizontal segments remains identical. Thus, to prove the lemma, simply observe that the sum 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

[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).

Algorithm 4: UNIFORMLABEL1SIDEROUTEPOCROSSINGELIMINATION

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 $x(p^1) > \cdots > x(p^i)$ and **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 $k < ik < j$ **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 $\{c^1, \ldots, c^k\}$
  $k = k + 1$
return $c_1, \ldots, c_i$

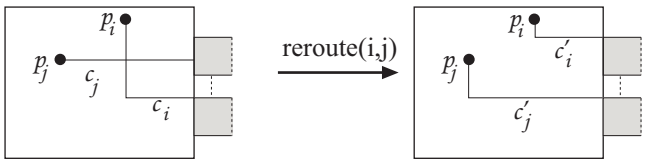


Figure 12: Rerouting of crossing leaders in the case of sliding ports.

point of the left side of the label that is closer to the site. As it can be seen in Figure 11, in order for a crossing to occur, the ports (fixed or sliding) used by the two leaders must have smaller or equal y-coordinate[2] (or, in the symmetric case, greater of equal y-coordinate) than both sites. 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 length remains unchanged.

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 Figure 12) 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.) □

[2]Given that the sites are considered to be in general position, only one site can have the same y-coordinate with an given port.

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, there is an $O(n^2)$-time algorithm that produces a legal type-po labelling of minimum total leader length.*

Proof. The algorithm that support the proof of the theorem is the algorithm used to construct a legal one-side type-*po* labelling for rectangular uniform labels (see Theorem 6). It is based on successive application of the step which resolves crossings (see Algorithm 4), which by Lemma 4, leaves the total leader length unchanged (for fixed and sliding ports). Thus, the total leader length of the labelling is identical to the total leader length of the initial labelling, before any effort to remove the crossings was made.

So, it only remains to show that the initial (with crossings) labelling where we simply stack the labels to the right of s in the same vertical order as the corresponding sites and connect them with type *po*-leaders is of 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 it was pointed out in Note 1 the labelling with type-*opo* leaders is unique and, thus, of minimum total leader length. ◻

3.3.5 Two-side labelling 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 get 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 opposites 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 allow skewed lines but forbid bends. We first give a simple algorithm that computes a legal one-side labelling. 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 labelling.

4.1 One-side labelling

We adopt the scenario of Section 3.1.1. 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.

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 unlabelled 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 nicely be improved to $O(n \log n)$ by using a 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 [11] yields a total running time of $O(n \log n)$. This algorithm is correct since it mimics the (n^2) algorithm in the proof of Lemma 5. □

It is also possible to compute a solution of minimum total leader length which applies to the case of sliding ports, as indicated in the next theorem.

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

Proof. To compute an assignment that is minimum in terms of total leader length and supports sliding ports, we proceed as described in the proof of Theorem 4, except now we use *Euclidean* minimum-cost bipartite matching for the sets of sites and ports. This takes $O(n^{2+\delta})$ time [1], where $\delta > 0$ can be chosen arbitrarily small. The weight of each edge of bipartite graph is the Euclidean distance from the corresponding site to the closest point of the corresponding label, supporting in this way sliding ports. Fortunately, the solution of the minimum-weight matching implies a crossing-free solution of the leader assignment since a crossing between two leaders could have been resolved such that the total length would have decreased. To see that, observe that the two sites and the two ports which correspond to two crossing leaders define a convex 4-gon which has the two crossing leaders as its diagonals. In any convex 4-gon, it holds that the sum of the length of its two opposite sides is smaller that the sum of the length of its two diagonals. Thus, a solution of minimum total leader length cannot have any crossing leaders. ◻

4.2 Four-side labelling

In this subsection, we consider four side type-s labelling. We describe how to obtain a legal labelling for fixed labels with fixed ports and a minimum total leader length labelling for fixed labels with sliding ports. These results can be considered to be extensions of the results on one-side type-s labelling.

Theorem 11 *A legal four-side type-s leader-label placement for fixed uniformly distributed labels with 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 Figure 13.

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 $(1 \leq i \leq 4)$ which divides the corresponding partitioned area into two adjacent polygons until both contain as many sites as there are labels on their boundaries.

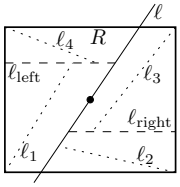


Figure 13: Partition of R for straight-line leaders.

Since we always divide a convex polygon with a straight line, the resulting polygons are also convex. We did not succeed to partition the rectangles into just four convex polygons but we need two polygons for each side which makes eight in total. Moreover, by the construction, the number of sites in each polygon is equal to the number of labels on its side that is part of the perimeter of the original rectangle. Thus, the one side type-*s* $O(n \log n)$ algorithm of Theorem 9 can be applied, leading to an $O(n \log n)$ algorithm for legal four side type*s* labelling. ◻

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 labelling 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 labelling case for fixed labels with sliding ports. Thus, we can state the following theorem:

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

Proof. Similar to the proof of Theorem 10. ◻

5 Examples

In this section, we present drawing obtained by our algorithms some characteristic examples. Figure 14 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 for type-*opo* leaders such that labels were placed to the left of the drawing and the number of bend is minimum.

Figure 15 shows two boundary labellings 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 labelling uses *opo* leaders, while the bottom one *po*. Although in the type *po* labelling (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 the type *opo* labellings.

6 Conclusion

We have defined *boundary labelling* and have presented a series of models and algorithms for efficient boundary labelling of site sets. Originally, we were motivated by a map of the infrastructure network of the Greek school system, see Figure 16. This examples indicates some possible generalizations of our model: graph labelling with objectives like the minimization of crossings between graph edges and leaders.

There are several issues that should be considered in future work:

- The minimum weight matching algorithm for four-side labelling seems to be quite powerful, but it is not very efficient in praxis.
- The dynamic programming algorithms for two-side labelling (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 might be to mix both types in order to cope with disadvantages while keeping advantages.

Acknowledgments

We thank Pankaj Agarwal for information about geometric matching.

References

[1] P. K. Agarwal, A. Efrat, and M. Sharir. Vertical decomposition of shallow levels in 3-dimensional arrangements and its applications. In *Proc. 11th Annu. ACM Sympos. Comput. Geom.*, pages 39–50, 1995.

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

[3] C. Binucci, W. Didimo, G. Liotta, and M. Nonato. Orthogonal drawings of graphs with vertex and edge labels. *Computational Geometry: Theory and Applications*, 2005. To appear.

[4] G. Booch, J. Rumbaugh, and I. Jacobson. *The Unified Modeling Language User Guide*. Addison-Wesley, 1999.

[5] B. Chazelle and 36 co-authors. The computational geometry impact task force report. In B. Chazelle, J. E. Goodman, and R. Pollack, editors, *Advances in Discrete and Computational Geometry*, volume 223, pages 407–463. American Mathematical Society, Providence, RI, 1999.

[6] J.-D. Fekete and C. Plaisant. Excentric labeling: Dynamic neighborhood labeling for data visualization. In *Proc. Conference on Human Factors in Computer Systems (CHI'99)*, pages 512–519. ACM New York, 15–20 May 1999.

[7] M. Formann and F. Wagner. A packing problem with applications to lettering of maps. In *Proc. 7th Annuual ACM Symposium on Computational Geometry (SoCG'91)*, pages 281–288, 1991.

[8] H. Freeman, S. Marrinan, and H. Chitalia. Automated labeling of soil survey maps. In *Proc. ASPRS-ACSM Annual Convention, Baltimore*, volume 1, pages 51–59, 1996.

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

[10] M. Á. Garrido, C. Iturriaga, A. Márquez, J. R. Portillo, P. Reyes, and A. Wolff. Labeling subway lines. In P. Eades and T. Takaoka, editors, *Proc. 12th Annual International Symposium on Algorithms and Computation (ISAAC'01)*, volume 2223 of *Lecture Notes in Computer Science*, pages 649–659. Springer-Verlag, 19–21 Dec. 2001.

[11] J. Hershberger and S. Suri. Applications of a semi-dynamic convex hull algorithm. *BIT*, 32:249–267, 1992.

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

[13] C. Iturriaga. *Map Labeling Problems*. PhD thesis, University of Waterloo, 1999.

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

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

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

[17] G. W. Klau and P. Mutzel. Automatic layout and labelling of state diagrams. In W. Jäger and H.-J. Krebs, editors, *Mathematics—Key Technology for the Future*, pages 584–608. Springer-Verlag, Berlin, 2003.

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

[19] J. L. Morrison. Computer technology and cartographic change. In D. Taylor, editor, *The Computer in Contemporary Cartography*. Johns Hopkins University Press, 1980.

[20] P. M. Vaidya. Geometry helps in matching. *SIAM J. Comput.*, 18:1201–1225, 1989.

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

Figure 14: 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 `http://www.vobs.at/bio/a-phys/pdf/a-skelett-a.jpg`.

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

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

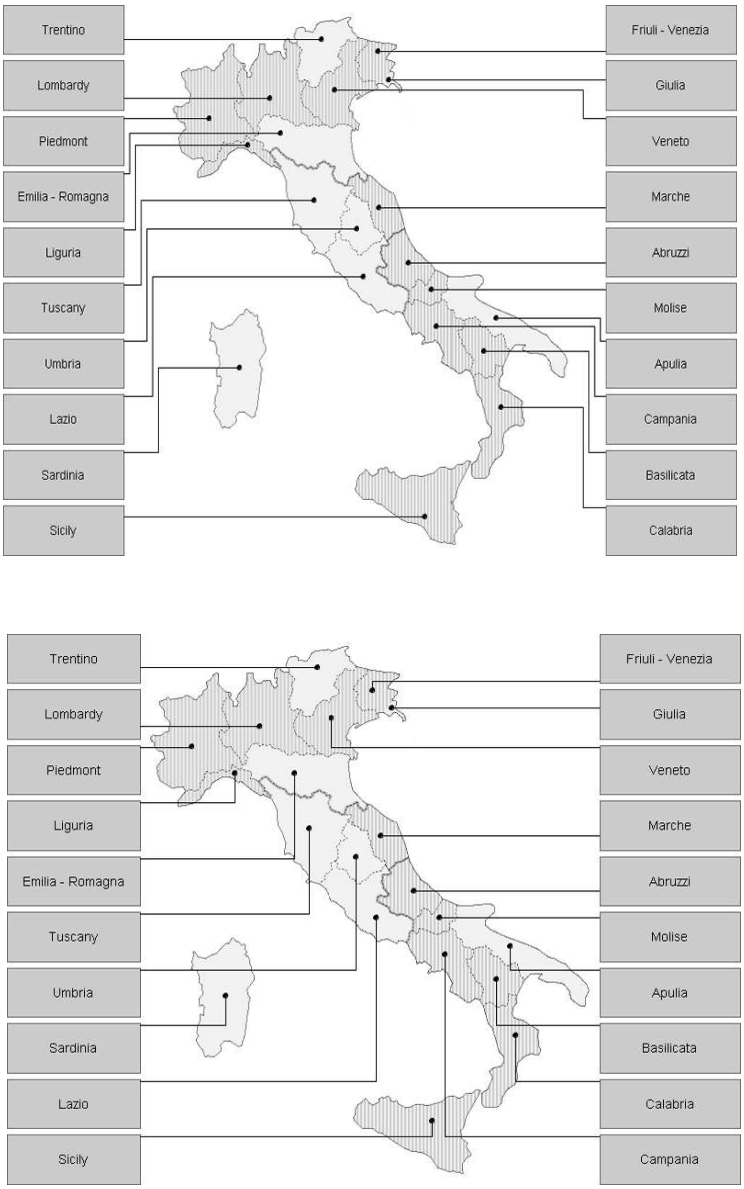


Figure 15: Two boundary labellings for the map of Italy. We attach labels to opposite sides of the map. We assume uniform labels of maximum size and leaders of type *opo* (top figure) and *po* (bottom figure). The total leader length is minimized in both cases.

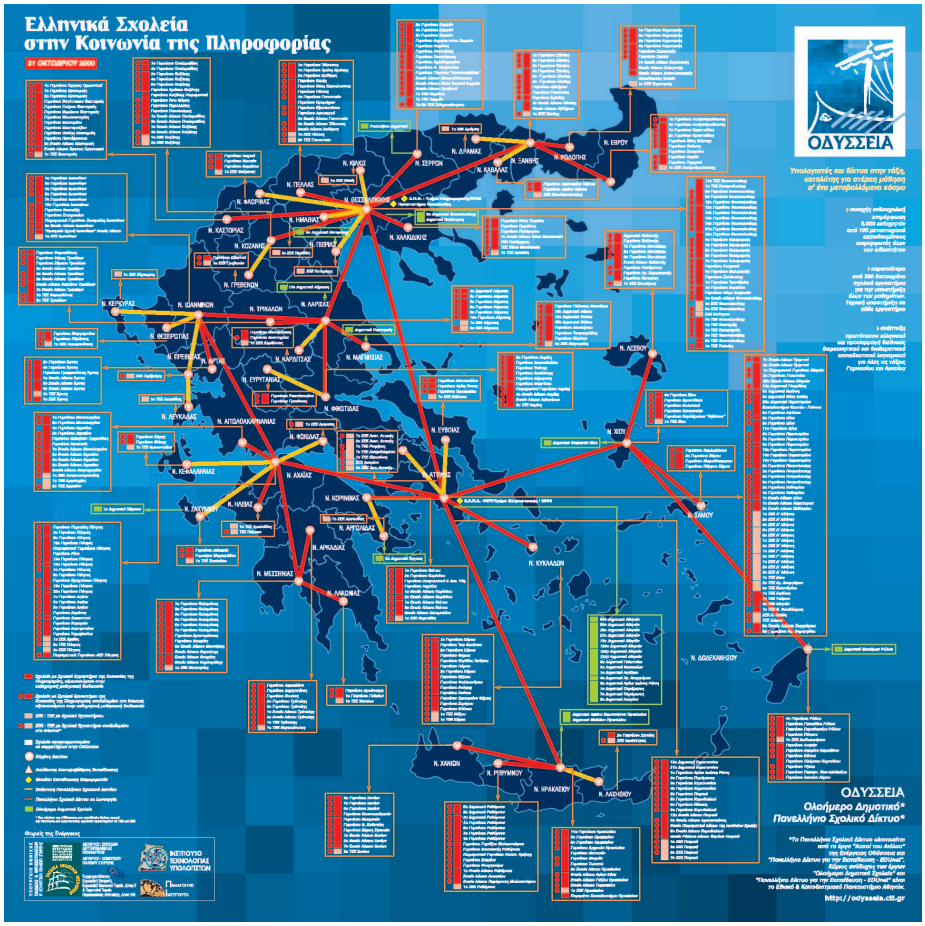


Figure 16: A map of the infrastructure network of the Greek school system.