Boundary Labeling: Models and Efficient Algorithms for Rectangular Maps*

Michael A. Bekos[1], Michael Kaufmann[2], Antonios Symvonis[1], Alexander Wolff[3]

[1] National Technical University of Athens, Dept. of Mathematics, Athens, Greece
{ mikebekos | symvonis } @math.ntua.gr
[2] University of Tübingen, Institute for Informatics, Sand 13, 72076 Tübingen, Germany
mk@informatik.uni-tuebingen.de
[3] Faculty of Informatics, Karlsruhe University, P.O. Box 6980, 76128 Karlsruhe, Germany
http://i11www.ira.uka.de/people/awolff

Abstract. In this paper, we present *boundary labeling*, a new approach for labeling point sets with large labels. We first place disjoint labels around an axis-parallel rectangle that contains the points. Then we connect each label to its point such that no two connections 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.

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 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 [5], cartographers, graph drawers, and computational geometers have suggested numerous approaches, such as expert systems, zero-one integer programming, approximation algorithms, simulated annealing, and force-driven algorithms to name only a few. An extensive bibliography about label placement can be found at [14]. The ACM Computational Geometry Impact Task Force report [3] denotes label placement as an important research area.

In this paper, we deal with labeling dense point sets with 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

* This work has partially been supported by the DFG grants Ka 512/8-2 and WO 758/4-1, by the German-Greek cooperation program GRC 01/048 and the EPEAEK program Pythagoras 89181(28).

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. Due to the complexity of either step there are still very few publications that combine graph drawing and label placement. Klau and Mutzel [11] have coined the term "graph labeling" for this discipline and have given a mixed-integer program for computing orthogonal graph layouts with node labels.

Leaders have so far only been used by Zoraster [15], Freeman et al. [6], and Fekete and Plaisant [4]. Zoraster [15] uses simulated annealing to label points and lines in seismic survey maps, while Freeman et al. [6] use an iterative raster-based method to determine positions for area labels in soil survey maps. Fekete and Plaisant [4] extend the *infotip paradigm* to cope with labeling dense point sets interactively. They draw a circle of fixed radius around the current cursor position, the so-called *focus circle*, and label only the sites that fall into the circle. Labels are left-aligned and placed in two stacks to the left and the right of the circle. To connect sites to their labels, Fekete and Plaisant use non-crossing leaders that consist of two or three line segments: one segment goes radially from the site to its projection on the focus circle and one or two axis-aligned segments go from there to the corresponding label.

Iturriaga and Lubiw [10] give an $O(n^4)$-time decision algorithm for attaching *elastic* labels to n points 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. Iturriaga [9] also briefly investigates the inverse problem, where elastic labels must be attached to the 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 around R, her algorithm takes $O(n)$ time.

This paper is structured as follows. In Section 2 we model and define our problem. In Section 3 we are concerned with rectilinear leaders. We investigate algorithms for non-intersecting leader-label placement, for leader-bend and leader-length minimization. In Section 4 we consider straight-line leaders. For the one-side and the four-side case, we compute legal leader-label placements and we minimize (with a slower algorithm) the total leader length. We have implemented some of our algorithms. In Section 5 we give an example layout. A full version of this paper with more examples and proofs is available at [2].

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 points $p_i = (x_i, y_i)$, 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 point p_i with label l_i for $1 \leq i \leq n$.
4. Intersections of leaders with other leaders, points 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.

In this paper we present algorithms that compute legal leader-label placements 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. 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. We consider type-o leaders to be of type *opo* and of type *po* as well. We refer to straight-line leaders as type-s leaders.

In this paper we assume that input points are in general position, i.e. no three points lie on a line and no two points 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, and that the 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 [7]. Due to this observation we first make some simplifying assumptions like uniform labels and then generalize our algorithms by adding more requirements.

3 Rectilinear leaders

In this section we investigate different ways of drawing rectilinear leaders. We present algorithms for legal leader-label placement, leader-bend 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.

3.1 Leader-bend minimization

One-side labeling with type-*opo* leaders. We first consider the problem of attaching labels to one, say the right, side of the rectangle R. We assume that the sum of the label heights is at most the height of R and that the sites are sorted according to non-decreasing y-coordinate. If we use a slightly wider rectangle R' and leaders of type *opo*, then we can attach labels to the right side of R' and place non-crossing leaders in R' as follows. We first stack the labels on top of each other such that the lower left corner of l_1 is incident to the lower right corner of R'. Then we connect each site p_i by a horizontal segment $y_i \times [x_i, r_R]$ to the right side of R. Finally we use the gap between the right sides of R and R' to lay out the remaining parts of the leaders from the right side of R to the, say, midpoints of the left label edges, see Figure 1. This can be done with at most two bends per leader and without any crossing, since the vertical orders of sites and labels are identical and since we assume that no two sites have the same y-coordinate. Thus a legal one-side type-*opo* leader-label placement can be computed in $O(n \log n)$ time.

Clearly this approach is not optimal in terms of the total number of leader bends. Given the restriction to leaders of type *opo* and the trick with the extra space at the right side of R, routing the leaders is easy, and the remaining problem 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. 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).

Theorem 1. *A legal one-side type-opo leader-label placement with the minimum number of bends can be computed in $O(n^2)$ time and space.*

Proof. We use dynamic programming with a table T of size $n \times (n+1)$. For $k \leq i$ the entry $T[i,k]$ will contain the minimum y-coordinate that is needed to accommodate the first i labels such that at least k of them use horizontal leaders. If it is impossible to connect k out of the first i labels with horizontal leaders, we set $T[i,k]$ to ∞. As usual, the table entries are computed bottom-up.

To compute a new entry $T[i,k]$, observe that there are only three interesting positions of the label l_i: (a) directly on top of l_{i-1} using a horizontal leader, (b) directly on top of l_{i-1} using a 2-bend leader, and (c) such that the top edge of l_i lies on the horizontal line through the i-th site. These cases and the case $T[i,k] = \infty$ can be distinguished in constant time. Thus T can be computed

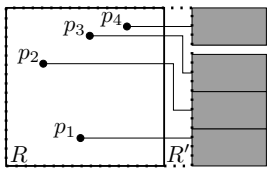

Fig. 1. Type-*opo* leaders.

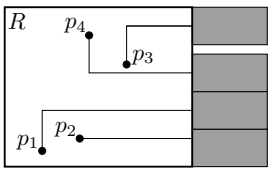

Fig. 2. Type-*po* leaders.

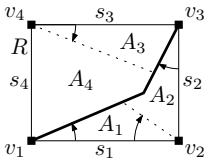

Fig. 3. Partition into monotone regions.

in $O(n^2)$ time. Given T, the number of horizontal leaders is the largest k that fulfills $T[n,k] \leq t_R$. By using an extra table of the same size as T, label and leader positions of an optimal solution can be computed as well. □

3.2 Legal leader-label placement

Four-side labeling 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. Points that lie on boundaries of our partition in the interior of R can be connected to a side of R via both incident regions. Thus we ignore the problem of how to distribute these boundaries.

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 point 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.

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. To simplify the presentation, we assume 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 want to partition R into four regions $A_1, \ldots, A_4$ as described above, such that $|A_i \cap P| = n_i$ for $i = 1, \ldots, 4$. We do this by rotating rays around the rectangle corners until these conditions are fulfilled, see Figure 3.

The rotations 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:

Lemma 1. *Given a rectangle R of sufficient size, a set $P \subset R$ of n points in general position, square uniform labels, one per point, 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 points with non-intersecting type-opo leaders.*

One-side labeling with type-*po* leaders. In this subsection we describe how to compute a legal labeling with leaders of type-*po*, see Figure 2. We restrict

ourselves to attaching labels to one side s of R. W.l.o.g., 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. We consider uniform labels. Since we do not attempt to minimize the number of bends, we simply stack labels to the right of s in the same vertical order as the corresponding sites.

Our algorithm is very simple: we go through the sites 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. Clearly c_i can be routed such that c_i does not contain any sites except p_i. Now we go through the sites $p_1, \ldots, p_{i-1}$ from right to left and test their leaders for intersection with c_i. Let p_j be the rightmost site p_j whose leader c_j intersects c_i. Then we reroute as in Figure 4: 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 and that the new leader c'_i of p_i can only intersect leaders of sites to the left of p_j. For placing the leader of p_i we have to reroute at most $i-1$ times, and after this process of rerouting no two leaders intersect any more. Thus we have:

Theorem 2. *A legal one-side type-po leader-label placement can be computed in $O(n^2)$ time given uniform labels.*

3.3 Leader-length minimization

In the remainder of this section we focus on obtaining label placements of minimum total leader length. We attach labels to the left and the right side of the rectangle R, and we treat uniform and non-uniform labels.

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. 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 are given points $p_1, \ldots, p_n$ that 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. Here we ignore the subproblem of routing. This can be done as for the one-side label placement in Section 3.1. Again we assume the existence of a slightly wider rectangle R'. The i-th point 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. Call it $Left[p, i]$. We define $Right[p, i]$ analogously.

Theorem 3. *Given a rectangle R with $n/2$ uniform labels of maximum height on its left and on its right side, and a set $P \subset R$ of n points in general position, there is an $O(n^2)$-time algorithm that connects all points to their labels with non-intersecting type-opo leaders such that the total leader length is minimum.*

Proof. To compute a label placement of minimum total leader length, we use a dynamic programming algorithm. We assume that n is even. The case that

n is odd is slightly more involved, see [2]. The algorithm constructs a table $T[0 : n/2, 0 : n/2]$. Entry $T[l, r]$ contains the minimum total leader length for the $l + r$ lowest points where l of them have labels on s_{left} and r on s_{right}. It is easy to prove by induction that $T[l, r]$ satisfies the following recurrence relation for $l, r \leq n/2$:

$$T[0, 0] = 0 \tag{1}$$
$$T[0, r] = T[0, r-1] + \mathit{Right}[p_r, r] \tag{2}$$
$$T[l, 0] = T[l-1, 0] + \mathit{Left}[p_l, l] \tag{3}$$
$$T[l, r] = \min\{\, T[l, r-1] + \mathit{Right}[p_{l+r}, r],\ T[l-1, r] + \mathit{Left}[p_{l+r}, l] \,\} \tag{4}$$

Having computed table T, entry $T[n/2, n/2]$ corresponds to a label placement of minimum total leader length. The actual placement can be easily recovered by maintaining an additional table. The running time is obvious. □

Type-*po* leaders and uniform labels. Our next result also deals with two-side placement of uniform labels.

Theorem 4. *Given a rectangle R with $n/2$ uniform labels of maximum height on each of its left and right side, and a set $P \subset R$ of n points in general position, there is an $O(n^2)$-time algorithm that attaches each point to a label with non-intersecting type-po leaders such that the total leader length is minimum.*

Proof. We use the dynamic-programming algorithm of Theorem 3 for the case of type-*opo* leaders to get the label placement. It runs in $O(n^2)$ time. Observe that connecting a site to its label (at a fixed port) 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 3.2. Possible crossings of leaders to the same side are resolved as in Section 3.2 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 points with crossing leaders would result in a solution with smaller total leader length.

Four-side labeling with type-*opo* leaders. We give 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 only assume that the labels have uniform size, the positions of the labels are disjoint, and the label ports are predefined. We have the following planarity result:

Lemma 2. *For any one-side solution of type-opo leaders with crossings there exists a crossing-free one-side solution of type-opo leaders which does not have a larger total leader length.*

Now we can use Vaidya's algorithm [13] 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 ports that minimizes the total Manhattan distance of the matched pairs.

Theorem 5. *A crossing-free four-side solution of type-opo leaders with minimum total length can be computed in polynomial time.*

Proof. Assume now that the solution of the minimum-weight matching implies a crossing between two leaders. Clearly this crossing is between two segments inside of the rectangle. Replacing the crossing by an appropriate "knock-knee" [12] gives two leaders which might not be of type-*opo*. Rerouting the leaders in type-*opo* shape does not increase the leader lengths, and applying Lemma 2 to each of the two affected sides of the rectangle will provide a new solution of type-*opo* with at most the same total leader length. An argument similar to that used for the crossing resolution for type-*po* leaders shows that the process of crossing resolution terminates in polynomial time. □

Type-*opo* leaders and non-uniform labels. We focus on two-side label placement of type-*opo* leaders. We are given n points $p_i = (x_i, y_i), i = 1, 2, \ldots, n$, 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 points. 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'.

Theorem 6. *Given a rectangle R of height H, a set $P \subset R$ of n points in general position where point 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 points 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 point p_i is connected to s_{left} and its label l_i is placed at height y then the length of the edge 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 points, where the left side of the rectangle is occupied up to y_L and the 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 point has its label on the left side, the left side of the rectangle is occupied up to y_L (including label l_i) and the 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):

$$T[i, y_L, y_R] = \min\{T_L[i, y_L, y_R],\ T_R[i, y_L, y_R]\} \tag{5}$$

$$T_L[i, y_L + h_i, y_R] = T[i-1, y_L, y_R] + Left[p_i, y_L] \tag{6}$$

$$T_R[i, y_L, y_R + h_i] = T[i-1, y_L, y_R] + Right[p_i, y_R] \tag{7}$$

Based on them, we can compute table T by dynamic programming. After this computation, the minimum table entry of the form $T[n, a, b]$, where $0 < a, b \leq H$, gives the minimum total leader length. We can recover the label placement which realizes the computed total leader length by maintaining an additional table with dimensions equal to those of T. The dynamic programming algorithm will use $O(nH^2)$ time and space. □

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 labeling. Then we show how this algorithm can be improved either in terms of runtime or in terms of total leader length. Finally we sketch how it can be applied to four-side labeling.

One-side labeling. We adopt the scenario of Section 3.1. Let R be the bounding rectangle. 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 points such that no two leaders intersect.

Let $M = \{m_1, \ldots, m_n\}$ be the ports sorted by y-coordinate from bottom to top. For $i = 1, \ldots, n$ we assign to m_i the first unlabeled point $p \in P$ 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. The proof of correctness is trivial.

Clearly the algorithm can be implemented in $O(n^2)$ time, but we can do better. 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 point 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 [8] yields a total running time of $O(n \log n)$. This algorithm is correct since it mimics the slow one.

To compute an assignment that is minimum in terms of total leader length we proceed as described just before Theorem 5, 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. For type-s leaders length minimization automatically ensures planarity. Thus we have:

Theorem 7. *A legal one-side type-s leader-label placement can be computed in $O(n \log n)$ time. Minimizing total leader length takes $O(n^{2+\delta})$ time for any $\delta > 0$.*

Four-side labeling. In this subsection, 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 one-side routing algorithm of the previous subsection. We assume uniform labels. 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

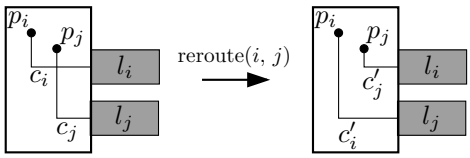

Fig. 4. Rerouting of crossing leaders.

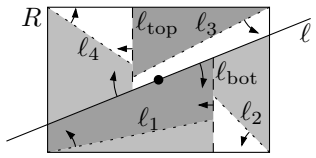

Fig. 5. Partition for straight-line leaders.

contain exactly as many sites as there are labels on their boundary. We construct in $O(n \log n)$ time eight polygons with this property by rotating ℓ, moving ℓ_{top} and ℓ_{bot}, and rotating ℓ_1 to ℓ_4 as indicated in Figure 5.

As in the one-side case Euclidean minimum-cost bipartite matching yields a placement of minimum total leader-length. Thus we conclude:

Theorem 8. *A legal four-side type-s leader-label placement can be computed in $O(n \log n)$ time. Minimizing total leader length takes $O(n^{2+\delta})$ time for any $\delta > 0$.*

5 Examples

We have implemented some of the presented algorithms, but due to space constraints we can give only one example here. Figure 6 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. For more examples, see [2].

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 ACM Symp. Comp. Geom. (SoCG'95)*, pages 39–50, 1995.
2. M. A. Bekos, M. Kaufmann, A. Symvonis, and A. Wolff. Boundary labeling: Models and efficient algorithms for rectangular maps. Technical Report 2004-15, Fakultät für Informatik, Universität Karlsruhe, 2004. Available at `http://www.ubka.uni-karlsruhe.de/cgi-bin/psview?document=/ira/2004/15`.
3. 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*, vol. 223, pp. 407–463. AMS, 1999.
4. 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, 1999. ACM New York.
5. M. Formann and F. Wagner. A packing problem with applications to lettering of maps. In *Proc. 7th ACM Symp. Comp. Geom. (SoCG'91)*, pages 281–288, 1991.
6. H. Freeman, S. Marrinan, and H. Chitalia. Automated labeling of soil survey maps. In *Proc. ASPRS-ACSM Annual Convention, Baltimore*, vol. 1, pp. 51–59, 1996.
7. 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.
8. J. Hershberger and S. Suri. Applications of a semi-dynamic convex hull algorithm. *BIT*, 32:249–267, 1992.

Fig. 6. A medical map with original labels and leaders (left) as well as labels and type-*opo* leaders computed by our algorithm (right). Drawing from `http://www.vobs.at/bio/a-phys/pdf/a-skelett-a.jpg`.

9. C. Iturriaga. *Map Labeling Problems*. PhD thesis, University of Waterloo, 1999.
10. C. Iturriaga and A. Lubiw. Elastic labels around the perimeter of a map. *Journal of Algorithms*, 47(1):14–39, 2003.
11. 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.
12. T. Lengauer. *Combinatorial Algorithms for Integrated Circuit Layout*. B. G. Teubner, 1990.
13. P. M. Vaidya. Geometry helps in matching. *SIAM J. Comput.*, 18:1201–1225, 1989.
14. A. Wolff and T. Strijk. The Map-Labeling Bibliography. `http://i11www.ira.uka.de/map-labeling/bibliography/`, 1996.
15. S. Zoraster. Practical results using simulated annealing for point feature label placement. *Cartography and GIS*, 24(4):228–238, 1997.