Labeling a Rectilinear Map More Efficiently*

Tycho Strijk
Dept. of Computer Science
Utrecht University
tycho@cs.uu.nl

Marc van Kreveld
Dept. of Computer Science
Utrecht University
marc@cs.uu.nl

Abstract

Given a rectilinear map consisting of n disjoint line segments, the corresponding labeling problem is to place a rectangle at each segment, allowing three possible positions, such that the rectangles do not intersect. This problem has a decision and a height maximization version. This paper improves results from Poon, Zhu and Chin: *A polynomial time solution for labeling a rectilinear map* (*IPL* **65** (1998), pp. 201–207). An algorithm is proposed that improves the running time of the decision problem from $O(n^2)$ to $O(n \log n\, \alpha(n))$, with $\alpha(n)$ the inverse of the Ackermann function. An algorithm for the height maximization problem is presented that lowers the running time from $O(n^2 \log n)$ to $O(n \log^2 n\ \alpha(n))$. These bounds are for the pointer machine model; in the RAM model we can drop the $\alpha(n)$ factor. We also describe an algorithm to solve the decision labeling problem where segments of arbitrary directions are allowed in $O(n^{\frac{4}{3}} \text{polylog}\, n)$ time and space.

1 Introduction

Map labeling is an important aspect of Geographic Information Systems. It has been studied both by the cartographic community as well as the computer science community [5, 6, 10, 14]. For a bibliography of map labeling papers see [15]. Besides point feature map labeling, there are a few papers which deal with line labeling. Barrault [3, 4] proposed an algorithm for river labeling and for road network labeling. Kakoulis and Tollis [11] have looked at edge labelings of graphs. Poon et al. [13] studied rectilinear line segment labeling.

*This research is supported by the Dutch Organization for Scientific Research (N.W.O.) and by the ESPRIT IV LTR Project No. 21957 (CGAL).

Figure 1: The three positions allowed for a label.

A rectilinear map consists of n disjoint horizontal and vertical line segments. We want to give each line segment a label that consists of a rectangle of height B and length the same as the segment. We allow three positions for the label, see Figure 1. The corresponding decision problem is called 3-position Rectilinear Segment Labeling (3RSL). The maximization problem is finding the largest height B such that there still exist a labeling of all segments.

Poon et al.[13] solve the decision problem by constructing a 2-SAT formula which exactly corresponds to the decision problem. For each segment they introduce two boolean variables and a clause to model the fact that the label can only be placed at one of the three allowed positions. They also construct clauses that prevent the intersection of labels of different segments. Each intersecting pair of two labels gives at most four clauses. Since the number of pairwise intersections of n line segments is $O(n^2)$ the maximum number of clauses is $O(n^2)$.

In our approach we do not construct intersection clauses explicitly. The intersections are only computed when needed by the algorithm.

Poon et al. solve the maximization problem by doing a binary search on a set of possible maximal values of B. The size of this set is $O(n^2)$ and it is constructed explicitly, leading to an $O(n^2 \log n)$ time algorithm. We are able to circumvent quadratic running time by constructing the set implicitly as a constant number of sorted matrices and by doing a binary search on these matrices. This yields $O(n \log^2 n)$ running time.

An extension is the problem in which we drop the restriction that segments must be axis parallel. This variant can be solved in $O(n^{\frac{4}{3}} \text{polylog}\, n)$ time using partition/cutting trees.

2 Fixed height labeling

Given are n disjoint horizontal and vertical segments and a label height B. Each segment i has two binary variables x_{i1} and x_{i2}. Around each segment there are four regions r_{i1}, r_{i2}, r'_{i1} and r'_{i2}, see Figure 2. The regions belonging to a horizontal

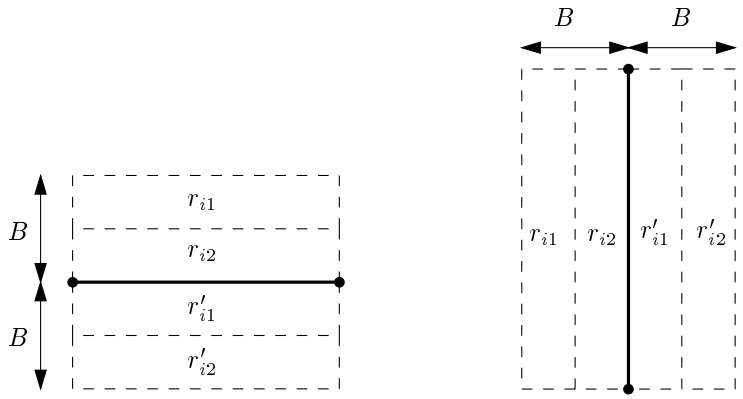

Figure 2: A horizontal and a vertical segment with their regions.

segment are called the horizontal regions. The vertical regions are the regions belonging to a vertical segment. Choosing r_{i2} and r'_{i1} is now the same as placing the label at the middle position. If region r_{i1} is part of the solution then $x_{i1} = 1$. If region r'_{i1} is part of the solution then $x_{i1} = 0$. This is defined analogously for x_{i2}. Thus regions r_{i1} and r'_{i1} cannot both be chosen at the same time. The clause $x_{i1} \Rightarrow x_{i2}$ takes care that only two adjacent regions can be chosen. Thus n clauses are needed to get the correct behavior at each individual segment. It remains to look at the interactions between different segments.

We can remove a region if it intersects a line segment s or if there is a line segment s between the region and the line segment to which the region belongs. We build a horizontal decomposition on the vertical segments using a sweep in $O(n \log n)$ time. During the sweep we also insert each horizontal segment into the face in which it lies. Since the segments are disjoint, a horizontal segment lies in exactly one face of the horizontal decomposition. Given this horizontal decomposition, which has $O(n)$ faces, we can determine for any height B in $O(n)$ time the vertical regions that intersect a vertical segment and the vertical regions that intersect a horizontal segment. Similarly, we build a vertical decomposition on the horizontal segments to determine the horizontal regions that intersect a horizontal segment and the horizontal regions that intersect a vertical segment.

To prevent the intersection of labels, we must determine all intersections between between regions $\{r_{i1}, r_{i2}, r'_{i1}, r'_{i2} \mid 1 \leq i \leq n\}$. All regions together form a set R. We can distinguish two types of intersection. The first type is the containment intersection in which one region is contained in another. The second type is the edge intersection in which the edges of two regions intersect. We next show that the containment intersections can be ignored, because these intersections are prevented if there are no edge intersections and the individual segment clauses are fulfilled.

Lemma 1 *There are no containment intersections if there are no edge intersections and the individual segment clauses are fulfilled.*

Proof: Horizontal regions cannot be contained in each other because they have the same height. Similarly vertical regions cannot be contained in each other. So we only have to look at vertical regions contained in horizontal regions and vice versa. Suppose a horizontal region belonging to a horizontal segment h is contained in a vertical region belonging to a vertical segment v. Since we removed regions that intersected a segment, we have that the horizontal region is either a r_{h1} or a r'_{h2} region. Without loss of generality we suppose that it is a r_{h1} region, see Figure 3. For the vertical region there is no restriction so we denote it by r_v. Suppose in a solution both regions r_{h1} and r_v were chosen. Then we must also choose r_{h2} because the segment clauses were fulfilled. But this is a contradiction since now r_v and r_{h2} are chosen and their edges intersect. ■

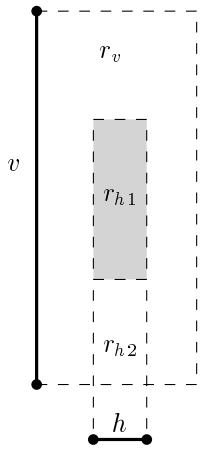

Figure 3: The intersection between regions r_{h1} and r_v is prevented implicitly.

So to solve the labeling problem we only have to fulfill the segment clauses and we have to prevent edge intersections of rectangles.

Imai and Asano [9] showed how to solve a 2-SAT problem where the only clauses were the intersections of rectilinear segments in $O(n \log n\ \alpha(n))$ time and $O(n \log n)$ space in the pointer machine model. In the RAM model the running time is reduced to $O(n \log n)$. Their algorithm is an adaptation for rectilinear segments of an algorithm from Masuda et al. [12] for graphs that has running time $O(m)$ with m the number of edges. Imai and Asano also mentioned how the algorithm could be used for intersections of rectangles in $O(n \log^2 n)$ time and $O(n \log n)$ space. Aonuma et al. [2] and later Formann and Wagner [7] used this latter algorithm for solving a point labeling problem with two positions allowed per point.

Many graph algorithms, such as depth first search and breadth first search, need two essential operations. The first operation is LIST, which, given a visited node u, returns an unvisited node v such that (u, v) is an arc of the graph. The second operation is DELETE which marks a node u visited and updates data structures such that the LIST operation does not report the visited node u any more. If the graph is given explicitly these two operations are standard. If the graph is the intersection graph of rectilinear line segments, the operations LIST and DELETE use a data structure that is described in Imai and Asano [9].

The algorithm of Masuda can be used to solve the 2-SAT problem on graphs. This algorithm determines the existence of an independent set of maximally possible size given a matching. The nodes of the graph are the normal and negated variables of the 2-SAT formula. The matching consists of all unnegated/negated pairs of variables. Each clause is represented by an arc in the graph. In our problem we have nodes for the variables $\{x_{i1}, x_{i2}, \neg x_{i1}, \neg x_{i2} \mid 1 \leq i \leq n\}$, corresponding to the regions in the problem. Each individual segment clause is an arc. If two regions intersect the corresponding nodes must be connected by an arc. We proved that we only have to look at edge intersections of regions. Thus a region intersects another region if one of its four edges intersects an edge of the other region.

We use a combination of the graph algorithm of Masuda et al. and the rectilinear segment algorithm of Imai and Asano. The graph G represents the individual segment clauses and thus has n edges. The edges of the regions give the rectilinear segments for the data structure S that represents the intersection graph. The basic operation LIST is composed of a LIST on G to detect an individual segment clause and a LIST for its four edges on S. The DELETE operation is decomposed similarly as one LIST on G and four LIST's on S. It is easily checked that this can be done with running time the sum of the running times of both original algorithms, and the same holds for the space requirements.

We must remark that the algorithm at this point is not entirely correct, because now regions belonging to the same segment are also considered to be intersecting. We can prevent this by adapting the algorithm of Imai and Asano a little. The operation LIST for an edge e on S must skip all edges that belong to the same segment as e. A careful look at the algorithm shows that this takes per LIST operation an additional time proportional to the number of skipped edges. Since an edge of a region can intersect at most five other edges of the same segment, this adaptation only takes a constant factor more time.

Theorem 1 *The 3RSL problem for n line segments can be solved in $O(n \log n\ \alpha(n))$ time in the pointer machine model and in $O(n \log n)$ time in the RAM model. The space needed in both models is $O(n \log n)$.*

Proof: The removal of regions which intersect segments takes $O(n \log n)$ time. The combined algorithm takes $O(n \log n\ \alpha(n))$ time in the pointer machine model and $O(n \log n)$ time in the RAM model. ∎

3 Optimal height labeling

Given the rectilinear line segments, the maximum height rectilinear segment labeling problem as defined in Poon et al. [13] is to find the maximum B such that there exists a placement for n non-overlapping rectangular valid labels, each attached to a distinct line segment, of height B and of length same as its line segment.

We solve this problem by performing a binary search on an implicit list W of possible values of maximum label height. Each search decision is based on the the result of the 3RSL problem for fixed label height.

When the height is optimal the labels of some segments nearly touch other segments. So we only have to look at the distances between all pairs of two horizontal, two vertical and the two distances between a horizontal and a vertical segment, see Poon et al. [13]. For two horizontal segments with distance d the heights we have to look at are $\frac{1}{2}d, \frac{2}{3}d, d$, and $2d$. This is analogous for two vertical segments. For a horizontal segment and a vertical segment we have x-axis distance d_1 and the y-axis distance d_2. The heights we must look at are $d_1, d_2, 2d_1$ and, $2d_2$.

In Poon et al. the list W is constructed explicitly, which has size $O(n^2)$. We construct W implicitly by using sorted matrices. We construct a sorted list H_y that consists of the y-coordinates of the horizontal segments. Further a sorted list H_x that consists of the x-coordinates of the endpoints of the horizontal segments. The lists V_x and V_y are constructed in the same way for the vertical segments.

Now let for two sets X and Y the set $X - Y$ be defined by $\{x - y \mid x \in X, y \in Y\}$. Furthermore let cX be $\{cx \mid x \in X\}$. Using this notation we can easily describe W as a subset of $\frac{1}{2}(H_y - H_y) \cup \frac{2}{3}(H_y - H_y) \cup (H_y - H_y) \cup 2(H_y - H_y) \cup \frac{1}{2}(V_x - V_x) \cup \frac{2}{3}(V_x - V_x) \cup (V_x - V_x) \cup 2(V_x - V_x) \cup (H_y - V_y) \cup (V_y - H_y) \cup 2(H_y - V_y) \cup 2(V_y - H_y) \cup (H_x - V_x) \cup (V_x - H_x) \cup 2(H_x - V_x) \cup 2(V_x - H_x)$.

Theorem 2 *The three position maximum height rectilinear segment labeling problem for n line segments can be solved in $O(n \log^2 n\ \alpha(n))$ in the pointer machine model and $O(n \log^2 n)$ time in the RAM model.*

Proof: Using the results of Frederickson and Johnson [8] we can select an ele-

ment with rank k in a constant number of $n \times n$ sorted matrices in $O(n)$ time. We have to do that $O(\log n)$ times. So the total time that the selecting part of the binary search takes is $O(n \log n)$. We have to solve the 3RSL problem at most $O(\log n)$ times. This takes in total $O(n \log^2 n\ \alpha(n))$ in the pointer machine model and $O(n \log^2 n)$ time in the RAM model. ∎

4 Labeling arbitrary segments

In this section we drop the axis-parallel constraint on the segments, so segments can have arbitrary orientations. Two segments may only intersect at an endpoint, otherwise a labeling is not possible. We solve this decision problem for labeling with fixed height B by constructing a 2-SAT formula which describes all intersections between regions. We do not generate a clause for each individual intersection, but use clauses to describe the intersections of groups of regions.

We use the following lemma, resulting from research on partition trees and cutting trees, see Agarwal and Erickson [1].

Lemma 2 *There exists a data structure that can answer segment-segment intersection queries in $O(n^{\frac{1}{3}}\text{polylog}\, n)$ time and $O(n^{\frac{4}{3}}\text{polylog}\, n)$ space. The construction time is $O(n^{\frac{4}{3}}\text{polylog}\ n)$. A data structure for rectangle-point intersection queries exists with the same bounds.*

First we remove all regions which contain a segment endpoint. This can be done using the data structure for rectangle–point intersection queries from Lemma 2. This takes in total $O(n^{\frac{4}{3}}\text{polylog}n)$ time. Similar to the axis-parallel case we can ignore the containment intersections, because they are automatically fulfilled if the individual segment clauses are fulfilled and there are no edge intersections of regions.

The next step is to build a data structure on the edges of the regions. This data structure is the standard partition/cutting tree data structure for segment-segment intersection searching, see Lemma 2.

When we search in this data structure with an edge e of a region, the solution consists of the intersecting edges which are associated with a subset U_e of $O(n^{\frac{1}{3}}\text{polylog}\, n)$ nodes of the data structure. At each node μ of the data structure we have two lists. The first is E_μ which stores all edges associated with node μ. The second is a list F_μ which consists of all edges e with $\mu \in U_e$. The

idea is to generate for each node μ a number of clauses for the 2-SAT formula. Suppose we have an edge $f \in F_\mu$. Then if we choose an edge e of E_μ than we cannot choose edge f any more and vice versa. This is modeled by the following clauses: $x_e \Rightarrow y_\mu$ for all $e \in E_\mu$, $x_f \Rightarrow z_\mu$ for all $f \in F_\mu$ and $\neg(y_\mu \wedge z_\mu)$ with y_μ and z_μ new variables. This gives in total $O(n^{\frac{4}{3}}\text{polylog}\, n)$ clauses because $\sum_\mu(|E_\mu| + |F_\mu|) = O(n^{\frac{4}{3}}\text{polylog}\, n)$ since each edge is stored $O(n^{\frac{1}{3}}\text{polylog}\, n)$ times in lists E_μ and F_μ.

Furthermore we have the same individual segment clauses as in the axis-parallel case. Also we have clauses which take care that if one edge of a region is chosen then the other edges of the same region are chosen as well. The number of them is $O(n)$.

The algorithm described until now also generates clauses for edge-edge intersections of edges belonging to the same region or the same segment. We can prevent this by making a binary decomposition of the edges in E_μ which are stored at each node μ. Since there are only five edges which belong to the same segment that an edge f can intersect, we can always choose $O(\log n)$ groups at μ that together form the edges in E_μ without those belonging to the same segment as f. The number of clauses is multiplied by a factor $\log n$ since for all $O(\log n)$ groups chosen we need a clause.

Theorem 3 *The decision segment labeling problem for arbitrary segments can be solved in $O(n^{\frac{4}{3}}\text{polylog}\, n)$ time and space.*

5 Conclusions and remarks

We have improved the time needed to label a rectilinear map to $O(n \log n\ \alpha(n))$ for the decision version and $O(n \log^2 n\ \alpha(n))$ for the maximization problem. For the k-position RSL problem where k positions are allowed for a label can be solved in time $O(kn \log n\ \alpha(n))$ in a similar way. The k-position maximum height labeling can be solved in time $O(kn \log^2 n\ \alpha(n))$ time. In the RAM-model we can drop the $\alpha(n)$ from the time bounds. For arbitrary segments we can solve the decision problem in $O(n^{\frac{4}{3}}\text{polylog}\, n)$ time and space. It is not clear how to solve the optimization problem in less than quadratic time.

References

[1] P. K. Agarwal and J. Erickson. Geometric range searching and its relatives. Tech. Report CS-1997-11, Department of Computer Science, Duke Univer-

sity, 1997.

[2] H. Aonuma, H. Imai, and Y. Kambayashi. A visual system of placing characters appropriately in multimedia map databases. In *Proceedings of the IFIP TC 2/WG 2.6 Working Conference on Visual Database Systems*, pages 525–546. North-Holland, 1989.

[3] M. Barrault. An automated system for name placement which complies with cartographic quality criteria: The hydrographic network. In *Proceedings of the Conference on Spatial Information Theory*, volume 1329 of *Lecture Notes Comput. Sci.*, pages 499–500. Springer-Verlag, 1997.

[4] M. Barrault and F. Lecordix. An automated system for linear feature name placement which complies with cartographic quality criteria. In *Proc. Auto-Carto 12*, pages 321–330, 1995.

[5] J. Christensen, J. Marks, and S. Shieber. An empirical study of algorithms for point-feature label placement. *ACM Transactions on Graphics*, 14(3):203–232, 1995.

[6] J. S. Doerschler and H. Freeman. An expert system for dense-map name placement. In *Proc. Auto-Carto 9*, pages 215–224, 1989.

[7] M. Formann and F. Wagner. A packing problem with applications to lettering of maps. In *Proc. 7th Annu. ACM Sympos. Comput. Geom.*, pages 281–288, 1991.

[8] G. N. Frederickson and D. B. Johnson. Generalized selection and ranking: sorted matrices. *SIAM J. Comput.*, 13:14–30, 1984.

[9] H. Imai and T. Asano. Efficient algorithms for geometric graph search problems. *SIAM Journal on Computing*, 15(2):478–494, 1986.

[10] C. Jones. Cartographic name placement with Prolog. *IEEE Computer Graphics & Applications*, 9(5):36–47, 1989.

[11] K. G. Kakoulis and I. G. Tollis. An algorithm for labeling edges of hierarchical drawings. In *Proceedings of the Symposium on Graph Drawing '97*, volume 1353 of *Lecture Notes in Computer Science*, pages 169–180, 1997.

[12] S. Masuda, S. Kimura, T. Kashiwabara, and T. Fujisawa. On the Manhattan wiring problem. Technical Report CAS 83-20, Institute of Electronics and Communication Engineers of Japan, 1983. in Japanese.

[13] C. K. Poon, B. Zhu, and F. Chin. A polynomial time solution for labeling a rectilinear map. *Information Processing Letters*, 65(4):201–207, 1998.

[14] M. van Kreveld, T. Strijk, and A. Wolff. Point set labeling with sliding labels. In *Proc. 14th Annu. ACM Sympos. Comput. Geom.*, pages 337–346, June 1998.

[15] A. Wolff and T. Strijk. A map labeling bibliography, 1998. http://www.inf.fu-berlin.de/map-labeling/bibliography.html.