# Labeling a rectilinear map with sliding labels

Sung Kwon Kim
Dept. of Computer Science and Engineering, Chung-Ang U., Korea
skkim@cau.ac.kr

Chan-Su Shin
Dept. of Computer Science
HKUST, Hong Kong
cssin@cs.ust.hk

Tae-Cheon Yang
School of Information Science
Kyungsung U., Korea
tcyang@csd.kyungsung.ac.kr

July 2, 1999

## 1 Introduction

Map labeling in geographic information systems has been considered to be important and well studied by the cartographic and computer science researchers [2, 5, 7]. There is a bibliography of papers on map labeling [11], maintained by Wolff and Strijk. Labeling points in the plane with rectangular labels has been an important problem and well investigated by many people [4, 6, 7].

In recent papers [3, 8, 10], the problem of labeling a rectilinear map was studied. A rectilinear map consists of a set of $n$ mutually non-intersecting rectilinear (horizontal or vertical) line segments, and each segment is allowed to use a rectangular label of height $B$ and length the same as the segment. A label can be placed at one of three positions, thus the problem is called the 3-position rectilinear segment labeling problem.

Poon et al. [8] solved the decision version of the problem (i.e., decide whether all segments can be assigned labels of height $B$) by constructing a 2-SAT formula that consists of $O(n^2)$ clauses, taking $O(n^2)$ time. They solved the optimization problem (i.e., find the maximum possible value of $B$) by doing binary searches over a sorted list of candidate values of $B$, which consists of $O(n^2)$ values, resulting an $O(n^2 \log n)$ time algorithm.

Recently, Strijk and van Kreveld [10] reduced the time of the decision algorithm to $O(n \log n)$ by *not* explicitly computing all possible clauses, but by considering clauses when needed only by the algorithm. They also invented a way of implicitly storing the list of candidate values of $B$ as a constant number of sorted matrices, allowing binary searches over sorted matrices, and thus obtain an $O(n \log^2 n)$ running time algorithm.

As a generalization of the 3-position labeling problem, $k$-position problem, $k \geq 1$, was also studied by Poon et al. [8] and Strijk and van Kreveld [10]. Strijk and van Kreveld [10] gave a decision algorithm with running time $O(kn \log n)$ and an optimization algorithm with running time $O(kn \log^2 n)$.

Van Kreveld, Strijk and Wolff [7] introduced the notion of *sliding* labels, where labels are not restricted to three (or any finite number of) predefined positions but can slide and be placed at any position as long as it intersects the object. In [7], they labeled points with sliding labels and introduced three different models of point labeling with sliding labels depending on the number of label sides allowed to touch points. Maximizing the number of points labeled was their objective, and they showed that the problem is NP-complete under the most general model of sliding labeling. A simple and efficient $\frac{1}{2}$-factor approximation algorithm was given.

In this paper, we will consider labeling rectilinear maps with sliding rectangular labels, i.e, the $\infty$-position rectilinear segment labeling problem will be considered. Unlike van Kreveld, Strijk and Wolff [7], our objective is to maximize the height of labels. Depending on the type of input segments, three different versions are possible.

**Problem 1D** A set of $n$ non-intersecting horizontal segments is given and the segments are intersected by a common vertical line.

**Problem 2D** A set of $n$ non-intersecting horizontal segments is given and there is no common vertical line that intersects all of the segments.

**Problem HV** A set of $n$ non-intersecting horizontal and vertical segments is given.

In this paper efficient algorithms for the optimizations of these three problems will be given. In Section 2, Problem **1D** will be considered, and a linear time algorithm is sufficient to solve it, after sorting the segments according to their $y$-coordinates. In Section 3, Problem **2D** will be considered, and a quadratic time algorithm will be given. Problem **HV** will be considered in Section 4. For the problem, a polynomial time "exact" algorithm and faster constant factor approximation algorithms will be given.

# 2 Problem 1D

In this section, we are given a set of $n$ horizontal segments, and there is a vertical line that intersects all segments. This problem is *one-dimensional* in the sense that the optimal height of the rectangles can be computed from the intersections of the vertical line with the segments. Compute the $y$-coordinates of the intersections of the vertical line with the segments and sort them in increasing order. Let $a_1, \ldots, a_n$ be the sorted list. We first decide whether $n$ closed intervals $I_i$ of length $B$ can be assigned to $a_i$ so that

- $a_i \in I_i$ for $1 \leq i \leq n$, and
- the intervals do not overlap, but may touch at their end points.

This decision problem can be decided by the following greedy algorithm: Place $I_1$ so that its upper end point coincides $a_1$; and for $i = 2, \ldots, n$, place $I_i$ one by one at a lowest possible position as far as the above two conditions are satisfied. It is easy to see that this greedy algorithm correctly decides.

Consider the optimization problem. Suppose that $B^\star$ is the maximum value of $B$ such that the above two conditions are satisfied.

**Lemma 1** *In an optimal assignment of intervals, there exist two indices $i, j$ with $i < j$ such that $a_j - a_i = (j - i - 1)B^\star$.*

**Proof:** Wlog assume that the intervals $I_i$ of length $B^\star$ are placed by the above greedy algorithm. Let $u_i$ be the coordinate of the upper end point of $I_i$. Then, the coordinate of the lower end point of $I_i$ is $u_i - B^\star$.

Consider a connected component which consists of intervals, $I_k, I_{k+1}, \ldots, I_{k'}$ with $k \leq k'$. Then, $u_k = a_k$. Otherwise, i.e., if $u_k > a_k$, we can push $I_k$ downward since $I_k$ is the first interval in the connected component and there is a room below $I_k$. The lower end point of the second interval, $I_{k+1}$, if exists, coincides $a_k$. So, $I_{k+1}$ cannot move downward and is *lower-fixed* by $a_k$. In fact, the second interval of each connected component consisting of two or more intervals is always lower-fixed.

We check if there is an *upper-fixed* interval by testing if $I_l$ for $k < l \leq k'$ such that $u_l = a_{l+1}$. If yes, then we are done because $a_{l+1} - a_k = (l-k)B^\star$. Note that the upper end point of $I_l$, if exists, is $a_{l+1}$. In other words, $I_l$ cannot move upward and is upper-fixed by $a_{l+1}$. Two intervals $I_{k+1}$ and $I_l$ are fixed by $a_k$ and $a_{l+1}$, and there are $l-k$ intervals of length $B^\star$ between them.

If there is no upper-fixed interval, all intervals in the connected component can be made longer. $I_{k+1}, I_{k+2}, \ldots, I_{k'}$ can be made longer by increasing upward by a small amount $\epsilon$, and $I_k$ can also made be longer by increasing downward by $\epsilon$.

If all connected components have no upper-fixed interval, then we can conclude that all intervals $I_1, \ldots, I_n$ can be made longer. This is a contradiction to the optimality of $B^\star$. Q.E.D.

By Lemma 1, the candidates for $B^\star$ are in the set

$$A = \{\frac{a_j - a_i}{j-i-1} \mid 1 \leq i < j \leq n\}.$$

In fact, $B^\star$ is the minimum one in $A$.

**Lemma 2** $B^\star = \min A$.

**Proof:** Let $B' = \min A = \frac{a_j - a_i}{j-i-1}$ for some $i, j$. If $B^\star > B'$, then $a_j - a_i = (j-i-1)B' < (j-i-1)B^\star$. So, $j-i-1$ intervals of length $B^\star$ cannot be placed between $a_i$ and $a_j$. Thus, $B^\star = B'$. Q.E.D.

To find $B^\star = \min A$, we transform this into a geometric problem as follows: Given a list $a_1, \ldots, a_n$, sorted in increasing order, construct $n$ blue points $b_i = (i, a_i)$ and $n$ red points $r_i = (i-1, a_i)$ for $1 \leq i \leq n$ in the plane. Consider lines connecting a blue point $b_i$ and a red point $r_j$ with $i < j$, whose slopes are $\frac{a_j - a_i}{j-i-1}$. The slopes of these lines are exactly in the set $A$. So, computing $B^\star$ is equivalent to finding the minimum slope line among these lines.

Let $B_i = \{b_1, \ldots, b_i\}$ and $R_i = \{r_1, \ldots, r_i\}$. Let $H_i$ be the upper hull of $B_i$. Let $\sigma_i$ be the minimum slope of the lines which connect blue points in $B_i$ and red points in $R_i$. Consider a line $\ell_i$ of slope $\sigma_i$ that is tangent to $H_i$. Define $b_{\lambda_i}$ to be the tangent point at which $\ell_i$ touches $H_i$. In the following, given $\sigma_i$, $\lambda_i$, $\ell_i$ and $H_i$, we will show how $\sigma_{i+1}, \lambda_{i+1}$, $\ell_{i+1}$ and $H_{i+1}$ can be computed. Note that $\sigma_i \geq \sigma_{i+1}$, and $\lambda_i \leq \lambda_{i+1}$. Consider a tangent line which passes through $b_{i+1}$ and touches $H_i$. Let $b_{\beta_{i+1}}$ be the tangent point. $\beta_{i+1}$ is needed to compute $H_{i+1}$. Also consider a tangent line which passes through $r_{i+1}$ and touches $H_i$. Let $b_{\rho_{i+1}}$ be the tangent point. $\rho_{i+1}$ is needed to compute $\sigma_{i+1}$.

Computing $H_{i+1}$ from $H_i$ is easily done by computing $\beta_{i+1}$. This procedure is usually called Graham's scan in the literature [1, 9]. Graham's scan computes $H_i$'s for all $i$ in $O(n)$ time. So, we will not describe how to compute $H_{i+1}$ and $\beta_{i+1}$.

Case (i): $r_{i+1}$ is below $\ell_i$. ($b_{i+1}$ is also below $\ell_i$).

In this case, $\sigma_{i+1} < \sigma_i$, so we have to compute $\rho_{i+1}$. Note that $\lambda_i \leq \rho_{i+1} \leq \beta_{i+1}$. Refer to Figure 1.

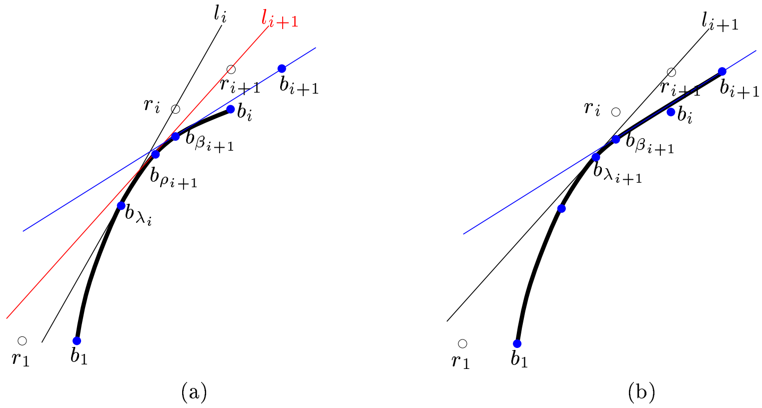



Figure 1: An illustration of Case (i): (a) Computing a tangent line $l_{i+1}$ by finding $b_{\rho_{i+1}}$. (b) Changing $\lambda_{i+1} = \rho_{i+1}$ and computing $H_{i+1}$ from $b_{\beta_{i+1}}$.

To compute $\rho_{i+1}$, we proceed, starting from $b_{\lambda_i}$, to the right along $H_i$, checking for each blue point $b_j$ on $H_i$ if the line connecting it and $r_{i+1}$ is tangent to $H_i$. Let $n_i$ be the number of points on $H_i$ between $b_{\lambda_i}$ and $b_{\rho_{i+1}}$. Since we never compute $\rho_{i+1}$ in cases (ii) and (iii), $\sum_{i=1}^{n} n_i$ corresponds to time spent computing all $\rho_i$'s throughout the execution of the algorithm. Note that in this case, $\lambda_{i+1} = \rho_{i+1}$. Thus, $b_{\lambda_{i+1}}$ is the only point on $H_{i+1}$ that will be checked while computing $\rho_{i+2}$ and that is checked while computing $\rho_{i+1}$. So, $\sum_{i=1}^{n} n_i \leq 2n$. As a result, $\ell_{i+1}$ is the line connecting $b_{\rho_{i+1}}$ and $r_{i+1}$; $\sigma_{i+1}$ is the slope of $\ell_{i+1}$.

Case (ii): $r_{i+1}$ is above $\ell_i$ and $b_{i+1}$ is below $\ell_i$.

In this case, $\sigma_{i+1} = \sigma_i$, $\lambda_{i+1} = \lambda_i$, and $\ell_{i+1} = \ell_i$. We do not need to compute $\rho_{i+1}$.

Case (iii): Both $r_{i+1}$ and $b_{i+1}$ are above $\ell_i$.

In this case, we have $\sigma_{i+1} = \sigma_i$, $\lambda_{i+1} = i + 1$, and $\ell_{i+1}$ is the line of slope $\sigma_i$ that passes through $b_{i+1}$. We also do not need to compute $\rho_{i+1}$.

This completes a proof of the following lemma.

**Lemma 3** *$B^\star = \min A$ can be found in $O(n)$ time.*

After computing $B^\star$, our linear time greedy algorithm can be used to assign rectangular labels of height $B^\star$ to the horizontal segments; so we have the following theorem.

**Theorem 1** *Problem* **1D** *can be solved in $O(n)$ time if the line segments are given sorted according to their $y$-coordinates.*

## 3 Problem 2D

We are given a set of $n$ horizontal line segments $s_1, \ldots, s_n$, and there is no vertical line that intersects all of the segments. The decision problem is to decide whether each segment $s_i$ can be assigned a (closed) rectangular label $L_i$ of height $B$ and length the same as $s_i$ so that

- $s_i \in L_i$ for $1 \leq i \leq n$; and
- the labels do not overlap, but may touch at their sides.

A greedy decision algorithm similar to the one in Section 2 works here. Sort the segments in increasing order of their $y$-coordinates and let $s_1, \ldots, s_n$ be the sorted list. Place $L_1$ so that its upper side coincides $s_1$; and for $i = 2, \ldots, n$, place $L_i$ one by one at a lowest possible position as far as the above two conditions are satisfied.

We now solve the optimization problem, in which we are to compute the maximum possible label height $B^\star$. To figure out what values could be candidates for $B^\star$, construct a directed acyclic graph $G$ as follows: the segments are the nodes of $G$, and there is a directed edge from $s_i$ to $s_j$ if $s_i$ and $s_j$ are visible and $s_i$ is below $s_j$. Two segments are visible if there is a vertical line which intersects both but none between them. In $O(n \log n)$ time, $G$ can be obtained by using methods known as trapezoidal decomposition or vertical decomposition [1, 9].

Define $m_{i,j}$ to be the number of nodes on the longest path from $s_i$ to $s_j$, excluding $s_i$ and $s_j$, if there is a directed path, consisting of two or more edges, from $s_i$ to $s_j$. Otherwise, $m_{i,j}$ is undefined. Since $G$ is planar, directed and acyclic, the values of $m_{i_0,j}$ for a fixed segment $s_{i_0}$ can be obtained in $O(n)$ time by noting

$$m_{i_0,j} = 1 + \max\{m_{i_0,j'} \mid \text{there is an edge from } s_{j'} \text{ to } s_j\}.$$

Notice that this is a variation of single-source all-destination *longest* path problem on a directed, acyclic graph. Thus, $m_{i,j}$ for all $i, j$ can be found in $O(n^2)$ time.

Let $D_{i,j}$ be the vertical distance between $s_i$ and $s_j$.

**Lemma 4** *In an optimal assignment of rectangles, there exist two indices $i, j$ with $i < j$ such that $D_{i,j} = m_{i,j} B^\star$.*

**Proof:** Assume that the labels are assigned by the above greedy method. Find the connected components of the labels. Then, by a similar argument as in the proof of Lemma 1, there must be a connected component which has a lower-fixed label and an upper-fixed label, which proves the lemma. Q.E.D.

By Lemma 4, the candidate values for $B^\star$ are in the set

$$A = \{\frac{D_{i,j}}{m_{i,j}} \mid m_{i,j} \text{ is defined in } G\}.$$

Since there are $m_{i,j}$ segments to be labeled between $s_i$ and $s_j$, the optimal height cannot be larger than $D_{i,j}/m_{i,j}$. Thus, we have the following.

**Lemma 5** $B^\star = \min A$.

After finding $B^\star$ in $O(n^2)$ time, our greedy algorithm assigns rectangular labels of height $B^\star$ to the horizontal segments; so we have the following theorem.

**Theorem 2** *Problem* **2D** *can be solved in $O(n^2)$ time.*

# 4 Problem HV

Given is a set of $n$ non-intersecting horizontal and vertical line segments $s_1, \ldots, s_n$, to which we want to assign rectangular labels of optimal height $B^\star$ so that the rectangles do not intersect. For Problem **HV**, an efficient algorithm will be given in Section 4.2, and then faster approximation algorithms will be given in Section 4.3. We will start with a problem whose solution will used as a subroutine in our algorithm.

## 4.1 Variable-position labeling

A set of $n$ non-intersecting horizontal and vertical line segments $s_1, \ldots, s_n$ is given. Each segment has a set of predefined positions, regardless of the label sizes, at which a rectangular label may be assigned. To be specific, each horizontal (resp., vertical) segment $s_i$ has a set $E_i$ of $y$- (resp., $x$-) coordinates: a label assigned to $s_i$ must have *either* its top *or* bottom (resp., left or right) side at one of the values in $E_i$.

Given a value of $B$, we want to know whether it is possible to assign a label of height $B$ to each segment. This is called the *variable-position* rectilinear labeling problem, and will be used as a subroutine in Section 4.2.

For each segment $s_i$ remove from $E_i$ the values that are more than $B$ away from $s_i$. These are no longer valid positions as the size of labels is $B$. That is, if $s_i$ is horizontal, remove from $E_i$ the values $> y(s_i) + B$ and the values $< y(s_i) - B$, where $y(s_i)$ is the $y$-coordinate of $s_i$. Do similar removals when $s_i$ is vertical.

For each segment $s_i$ add to $E_i$ the values that are exactly $B$ away from those in $E_i$. In other words, if $s_i$ is horizontal, for each value $y \in E_i$ add $y - B$ to $E_i$ if $y > y(s_i)$, and add $y + B$ to $E_i$ if $y < y(s_i)$. Do similar additions when $s_i$ is vertical. These added values are matching top and bottom sides, and left and right sides.

At this point, each segment has a set of fixed positions at which its label may locate. In Poon et al. [8] they generalize the 3-position problem to the $k$-position problem in which a label may locate at one of $k$ equidistant positions. The number of positions is fixed to $k$ but the positions are not fixed because they are dependent on $B$. The distance between two consecutive positions is $B/k$. Poon et al. [8] solve the decision version of the problem in $O(kn^2)$ time by using the 2-SAT formulation, and Strijk and van Kreveld [10] reduce the time to $O(kn \log n)$.

In our case each segment has different number of positions which are fixed in advance (given as a set of $x$- or $y$-coordinates, $E_i$) and thus independent of $B$. Moreover, they are not equidistant. However, it is easy to check that it is still possible to use the 2-SAT formulation of Poon et al. [8] and employ the algorithm by Strijk and van Kreveld [10].

**Lemma 6** *The variable-position rectilinear labeling problem can be decided in $O(k_{\max} n \log n)$ time, where $k_{\max} = \max_{i=1}^{n}\{|E_i|\}$.*

## 4.2 "Exact" algorithm

Given is a set of $n$ non-intersecting horizontal and vertical line segments $s_1, \ldots, s_n$, to which we want to assign rectangular labels of optimal height $B^\star$ so that the rectangles do not intersect. To compute $B^\star$ we first find out what values could be $B^\star$.

Consider an optimal assignment of labels in which each segment is labeled with a rectangular label of height $B^\star$. A label is called *horizontal* (resp., *vertical*) if its segment is horizontal (resp., vertical). Two horizontal (resp., vertical) labels are *stacked* if the bottom (resp., left) side of one label touches the top (resp., right) side of the other. A *vertically* (resp., *horizontally*) *stacked component* (*VSC* or *HSC*, for short) is a set of horizontal (resp., vertical) labels that are stacked. As in the two previous sections, VSCs and HSCs play an important role in determining the optimal height. We say that $B^\star$ occurs in horizontal (resp., vertical) labels if there is a "critical" horizontal (resp., vertical) label such that increasing its height by a small amount no more guarantees an optimal assignment of labels to the segments. A *critical* VSC is one containing a "critical" horizontal label. A *critical* HSC is similarly defined.

Consider first what values could be candidates for $B^\star$ when $B^\star$ occurs in horizontal labels. For this, we need to know all potential critical VSCs the horizontal segments might induce, and their label heights. To do this, as in Section 3, construct a directed acyclic graph $G_1$ as follows: the nodes of $G_1$ are the *horizontal* segments, and there is a directed edge from a horizontal segment $s_i$ to a horizontal segment $s_j$ if

- $s_i$ and $s_j$ are visible,
- $s_i$ is below $s_j$, and
- there is no vertical segment that is visible from $s_i$ and $s_j$.[1]

Define $P_{i,j}$ to be the longest path from $s_i$ to $s_j$ in $G_1$, excluding $s_i$ and $s_j$, if there is a directed path, consisting of two or more edges, from $s_i$ to $s_j$. In case of more than one longest path, take one arbitrarily. Otherwise, $P_{i,j}$ is undefined. Define $m_{i,j}$ and $D_{i,j}$ on $G_1$ as in Section 3, i.e., $m_{i,j} = |P_{i,j}|$ and $D_{i,j}$ is the vertical distance between $s_i$ and $s_j$. Let $A_1 = \{\frac{D_{i,j}}{m_{i,j}} \mid P_{i,j} \text{ is defined in } G_1\}$. The segments on $P_{i,j}$ might constitute a critical VSC, and $\frac{D_{i,j}}{m_{i,j}}$ is the label height of the critical VSC.

We augment $G_1$ to get $G_2$ by adding some horizontal lines (half-lines, line segments). For each end point of each vertical line segment, draw a *maximal* horizontal segment passing through it until it hits any other segments. A maximal segment could be a horizontal line (if it hits no segment), a horizontal half-line (if it hits a segment at one of its end points), or a horizontal segment (if it hits segments at both of its end points). The nodes of $G_2$ are the horizontal segments, colored *blue*, plus the maximal horizontal segments, colored *red*, $s_{n+1}, s_{n+2}, \ldots, s_{n'}$, and there is a directed edge from an (any-colored) horizontal segment $s_i$ to an (any-colored) horizontal segment $s_j$ if

- $s_i$ and $s_j$ are visible, and
- $s_i$ is below $s_j$.

In $G_2$, assign a 1 to each blue segment and a 0 to each red segment. Compute the longest paths from any-colored $s_i$ to any-colored $s_j$ in $G_2$, if there is a directed path from $s_i$ to $s_j$. In case of more than one longest paths, take one arbitrarily. Define $Q_{i,j}$ to be the set of blue segments on the longest path from $s_i$ to $s_j$, excluding $s_i$ and $s_j$. If there is no directed path from $s_i$ to $s_j$ or $Q_{i,j} = \emptyset$, then $Q_{i,j}$ is said to be undefined. Define $l_{i,j} = |Q_{i,j}|$ and $D_{i,j}$ is the vertical distance between $s_i$ and $s_j$. Let $A_2 = \{\frac{D_{i,j}}{l_{i,j}} \mid Q_{i,j} \text{ is defined in } G_2\}$. Again, the segments on $Q_{i,j}$ might constitute a critical VSC, and $\frac{D_{i,j}}{l_{i,j}}$ is the label height of the critical VSC.

[1]Here, two objects are visible if there is a vertical line that intersects both but none between them.

**Lemma 7** *If $B^\star$ occurs in horizontal labels, then $B^\star \in A_1 \cup A_2$.*

**Proof:** Consider an optimal label assignment $\mathcal{L}$ with optimal height $B^\star$. In $\mathcal{L}$, without moving any vertical labels, move the horizontal labels at their lowest possible positions as did in Sections 2 and 3 by the greedy methods. Compute the VSCs. Then, there must be a critical VSC in $\mathcal{L}$ which contains two horizontal labels such that either

- one is lower-fixed by a horizontal segment and the other is upper-fixed by a horizontal segment;
- one is lower-fixed by a horizontal segment and the other is upper-fixed by a vertical segment or a vertical label;
- one is lower-fixed by a vertical segment or a vertical label and the other is upper-fixed by a horizontal segment; or
- one is lower-fixed and the other is upper-fixed by vertical segments or vertical labels.

If there is no such VSC, then the height of the horizontal labels can be increased, which is a contradiction to that $B^\star$ occurs in horizontal labels.

$A_1$ corresponds to the first case, and $A_2$ corresponds to the last three cases. Q.E.D.

Next, consider the case when $B^\star$ occurs in vertical labels. $A_3$ and $A_4$ can be defined on $G_3$ and $G_4$ and computed analogously as $A_1$ and $A_2$ on $G_1$ and $G_2$, respectively.

**Lemma 8** *If $B^\star$ occurs in vertical labels, then $B^\star \in A_3 \cup A_4$.*

Combining these two lemmas, the following lemma has been proved. Let $A = A_1 \cup A_2 \cup A_3 \cup A_4$.

**Lemma 9** *$B^\star \in A$.*

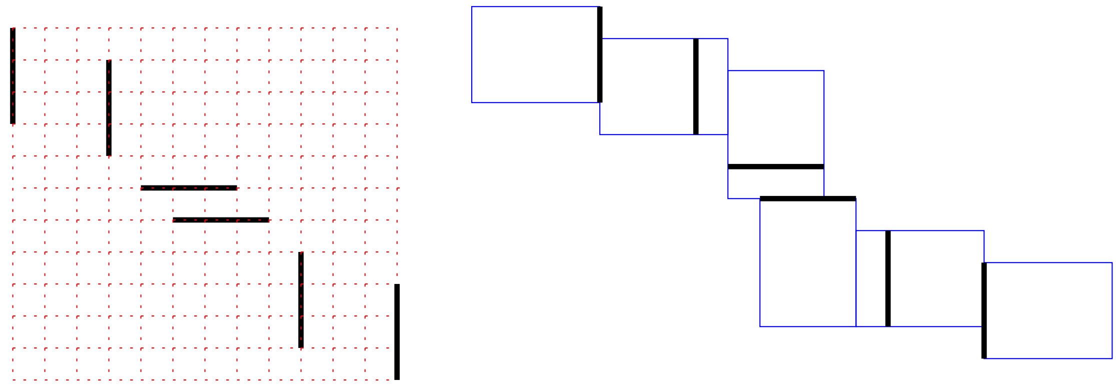
Unlike Lemmas 2 and 5, $B^\star$ is not necessarily $\min A$; Figure 2 shows such an example. To find $B^\star$ in $A$, we need to develop a decision algorithm which determines whether $B \leq B^\star$ for a given value $B > 0$. The decision algorithm will be used to do binary searches over $A$ to locate $B^\star$.

To develop our decision algorithm, Lemma 6 will be employed. Firstly, we need to determine for each segment the positions at which its label may locate, i.e., $E_k$ for each $s_k$ as in Section 4.1.

Consider a horizontal segment $s_k$ in $G_1$. Remember that $P_{i,j}$ is the longest path between two horizontal segments $s_i$ and $s_j$ excluding $s_i$ and $s_j$ in $G_1$. If $s_k \in P_{i,j}$ for some $1 \leq i, j \leq n$, then $s_k$ might be assigned a *virtual* label of height $D_{i,j}/m_{i,j} \in A_1$. ¿From $P_{i,j}$ and $D_{i,j}/m_{i,j}$, the position of the virtual label can be determined. By *position* we mean the $y$-coordinates of the top and bottom sides of the virtual label.

For each horizontal segment $s_k$, compute the positions of its virtual labels in $G_1$. This can be done in $O(n^3)$ time since there are at most $O(n^2)$ directed paths in $G_1$, and each path consists of at most $n$ segments.

Now consider $G_2$. For each horizontal segment $s_k$ in $G_2$, compute the positions of all virtual labels that might be assigned to it. If $s_k \in Q_{i,j}$ for some $1 \leq i, j \leq n'$, then $s_k$ might be assigned a *virtual* label of height $D_{i,j}/l_{i,j} \in A_2$. ¿From $Q_{i,j}$ and $D_{i,j}/l_{i,j}$, the position of the virtual label can be determined. This also can be done in $O(n^3)$ time.



(a) (b)

Figure 2: An example meaning that $B^\star$ may not be $\min A$. (a) $A_1 = \emptyset, A_2 = \{1.5, 2\}, A_3 = \emptyset, A_4 = \{4\}$, so $A = \{1.5, 2, 4\}$. (b) An optimal labeling with $B^\star = 4$.

If the bottom (resp., top) side of a virtual label is higher (resp., lower) than $s_k$, then it can be discarded as it is not valid.

Each $s_k$ has the positions computed in $G_1$ and the positions computed in $G_2$. Union them and this is $E_k$. Finally, add $y(s_k)$ to $E_k$, if it is not already in. Then, each horizontal segment $s_k$ has $E_k$.

A similar computation finds $E_k$, the positions of each vertical segment $s_k$. Now each segment $s_k$ has its predefined positions, $E_k$, at which its label may locate. Apply Lemma 6 to the segments.

**Theorem 3** *Problem* **HV** *can be decided in* $O(n^3 \log n)$ *time.*

**Proof:** *Time complexity:* Trapezoidal decomposition constructs $G_1$ and $G_2$ in $O(n \log n)$ time [1, 9]. Computing $m_{i,j}$ and $l_{i,j}$ on $G_1$ and $G_2$, respectively, are again variations of the longest path problem on directed acyclic graphs, taking $O(n^2)$ time. As explained above, computing the positions of the virtual labels takes $O(n^3)$ time. For horizontal segments, $k_{\max} = |A_1 \cup A_2|$ and for vertical segments, $k_{\max} = |A_3 \cup A_4|$. Since each $|A_i| = O(n^2)$, time complexity follows from Lemma 6.

*Correctness proof:* Problem **HV** is about *sliding* labels, while Lemma 6 applies to the case of labels with predefined positions. So, it is needed to prove that given $B > 0$, Lemma 6 decides *yes* if and only if Problem **HV** is *yes*.

If the lemma decides *yes*, then Problem **HV** is trivially *yes*.

Suppose that Problem **HV** is *yes*, and consider a "witness" labeling $\mathcal{L}$ of label height $B$. It will be shown that each label in $\mathcal{L}$ can be shifted to one of its predefined positions so that the resulting labels do not overlap.

We first show that this is true for $B = B^\star$. In $\mathcal{L}$, there exist critical VSCs or HSCs. The labels in the critical VSCs or HSCs are already at predefined positions, and thus their segments are said to be *resolved*.

To resolve the unresolved segments, consider a new optimal labeling $\mathcal{L}'$ of the *unresolved* segments only. In this new labeling $\mathcal{L}'$, the critical VSCs and HSCs of the previous paragraph are "prohibited area" where new labels are not allowed to overlap but may touch their boundary. Let $B'$ be the optimal height of $\mathcal{L}'$. Clearly, $B^\star < B'$. Again, there exist critical VSCs and HSCs in $\mathcal{L}'$, in which each label is of height $B'$. Note that $B' \in A$ as $A$ contains the label heights of all possible critical stacked components. Consider a horizontal segment $s_k$ and its label $L'_k$ in the critical VSCs and HSCs of $\mathcal{L}'$. Then, $L'_k$ is at one of the predefined positions in $E_k$. Since $B^\star < B'$, it is possible to assign a label $L_k$ of height $B^\star$ inside $L'_k$ as follows: If the vertical distance between $s_k$ and the top side of $L'_k$ is at most $B^\star$, place $L_k$ so that its top side coincides the top side of $L'_k$. Otherwise, place $L_k$ so that its bottom side coincides $s_k$. These two positions are in $E_k$. Similar placement works for vertical segments. So, all $s_k$ in the VSCs and HSCs of $\mathcal{L}'$ are now resolved.

Continue the above procedure with the still unresolved segments until the segments are all resolved.

Next consider $B < B^\star$. Let $\mathcal{L}$ be the optimal labeling with label height $B^\star$ in which all labels are at predefined positions. Consider a horizontal segment $s_k$ and its label $L_k$. Since $B < B^\star$, assign a label $L''_k$ of height $B$ inside $L_k$ as follows: If the vertical distance between $s_k$ and the top side of $L_k$ is at most $B$, place $L''_k$ so that its top side coincides the top side of $L_k$. Otherwise, place $L''_k$ so that its bottom side coincides $s_k$. These two positions are in $E_k$. Similar placement works for vertical segments. So, the labels $L''_k$ are at predefined positions.

It is shown that any (valid) labeling with label height $B \leq B^\star$ can be transformed into another labeling in which each label is at one of its predefined positions. This completes our correctness proof. Q.E.D.

A binary search over $A$, after sorting it, is sufficient to locate $B^\star$ in $A$.

**Theorem 4** *Problem* **HV** *can be solved (i.e., optimized) in* $O(n^3 \log^2 n)$ *time.*

### 4.3 Approximation algorithms

An approximation algorithm for Problem **HV** works by employing optimization algorithms for the 3-position labeling problem. Let $B^\star_3$ be the maximum height of rectangle labels assigned by optimization algorithms for the 3-position labeling problem in [8, 10]. The algorithm of Strijk and van Kreveld [10] runs in $O(n \log^2 n)$ time.

**Lemma 10** $B^\star_3 / B^\star \geq 2/3$.

**Proof:** Consider an optimal labeling of the segments with sliding labels. Consider a rectangular label $L_i$ assigned to $s_i$. $s_i$ divides $L_i$ into two rectangles. Let $p_i$ and $q_i$ be the heights of these two rectangles. Wlog assume $p_i \geq q_i$. Let $B' = \min_i\{\max\{p_i, 2q_i\}\}$.

For each $i$, create another label $L'_i$ of height $B'$. If $p_i \geq 2q_i$, then let $L'_i$ be the rectangle of height $B'$ that is labeled either above or below $s_i$ depending on in which side of $s_i$, $p_i$ occurs. Otherwise, let $L'_i$ be the rectangle of height $B'$ with $s_i$ in the middle of $L'_i$.

Since each $L'_i \subseteq L_i$ and is at one of the three positions, indicated in the 3-position labeling problem, the new labeling is a 3-position labeling for the segments. Thus $B^\star_3 \geq B'$. Since $\max\{p_i, 2q_i\} \geq (2/3)B^\star$ for all $i$, $B' \geq (2/3)B^\star$. Therefore, $B^\star_3 \geq (2/3)B^\star$. Q.E.D.

If $k$-position labeling for $k \geq 3$ are used instead of 3-position labelings, then tighter bounds can be obtained. Let $B^\star_k$ be the maximum height of rectangle labels assigned by optimization algorithms for the $k$-position labeling problem in [8, 10]. The algorithm of Strijk and van Kreveld [10] takes $O(kn \log^2 n)$ time.

**Theorem 5** *$B^\star_k/B^\star \geq 1 - 1/k$ for $k \geq 3$.*

**Proof:** $L_i$, $p_i$, and $q_i$ are defined as in the proof of Lemma 10. Define, for $1 \leq i \leq n$, $c_i = \max\{p_i, t_{i,j} \mid 1 \leq j \leq \lfloor \frac{k-1}{2} \rfloor, t_{i,j} \leq B^\star\}$, where $t_{i,j} = (1 + \frac{k-j-1}{j})q_i$. Let $B' = \min_i\{c_i\}$.

For each $i$, create another label $L'_i$ of height $B'$ as follows: If $c_i = p_i$, then let $L'_i$ be the rectangle that is labeled either above or below $s_i$ depending on in which side of $s_i$, $p_i$ occurs. If $c_i = t_{i,j}$ for some $j$, let $L'_i$ be the rectangle with $s_i$ in position $j$ of $L'_i$.[2]

Since each $L'_i \subseteq L_i$ and is at one of the $k$ positions, indicated in the $k$-position labeling problem, the new labeling is a $k$-position labeling for the segments. Thus $B^\star_k \geq B'$. Since $c_i \geq (1 - 1/k)B^\star$ for all $i$, $B' \geq (1 - 1/k)B^\star$. Therefore, $B^\star_k \geq (1 - 1/k)B^\star$. Q.E.D.

# 5 Conclusions

We have combined the rectilinear labeling problem with the sliding label, and presented efficient solutions for several versions of the problem. An interesting problem remaining is to reduce the time complexity of Problem **2D**, currently $O(n^2)$ in this paper. To do this we should devise a way of reducing the number of candidate values in $A$, or a method of storing them implicitly and evaluating values only when they are really needed. Developing faster algorithms for Problem **HV** is also interesting.

# References


[1] M. de Berg, M. van Kreveld, M. Overmars and O. Schwarzkopf, *Computational Geometry: Algorithms and Applications*, Springer-Verlag, 1997.

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

[3] S. Doddi, M. Marathe, A. Mirzaian, B. Moret, and B. Zhu, Map labeling and its generalizations, *Proc. 8th ACM-SIAM Symp. on Disc. Alg.,* pages 148–157, 1997.

[4] M. Formann and F. Wagner, A packing problem with applications to lettering of maps, *Proc. Annu. 7th Symp. on Comp. Geom.*, 281–288, 1991.

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

[6] D.E. Knuth and A. Raghunathan, The problem of compatible representatives, *SIAM J. Discr. Math.*, 5(3) pp. 422–427, 1992.


[2]Suppose that $L'_i$ is partitioned into $k-1$ equal-size rectangles of height $B'/(k-1)$ by $k-2$ equidistant horizontal line segments. $s_i$ is said to be in position $j$ if $s_i$ coincides the $j$-th horizontal line segment from the bottom.

[7] M. van Kreveld, T. Strijk, and A. Wolff, Point set labeling with sliding labels, *Proc. Annu. 14th Symp. on Comp. Geom.*, pages 337–346, 1998. For a full version, see Technical Report UU-CS-1998-40, Department of Computer Science, Utrecht University, 1998.

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

[9] F.P. Preparata and M.I. Shamos, *Computational Geometry: An Introduction*, Springer-Verlag, 1985.

[10] T. Strijk, and M. van Kreveld, Labeling a rectilinear map more efficiently, *Information Processing Letters*, 69(1) pp. 25–30, 1999.

[11] A. Wolff and T. Strijk, The map labeling bibliography, 1999. `http://www.inf.fu-berlin.de/map-labeling/bibliography.html`.