Labeling Subway Lines*

Mari Ángeles Garrido[1], Claudia Iturriaga[2], Alberto Márquez[1], José Ramón Portillo[1], Pedro Reyes[1], and Alexander Wolff[3]

[1] Dept. de Matemática Aplicada I, Universidad de Sevilla, Spain
[2] Faculty of Computer Science, University of New Brunswick, Canada
[3] Institut für Mathematik und Informatik, Universität Greifswald, Germany
Email: vizuete@us.es, citurria@unb.ca, almar@us.es, josera@us.es, preyes@us.es, awolff@uni-greifswald.de

Abstract. Graphical features on map, charts, diagrams and graph drawings usually must be annotated with text labels in order to convey their meaning. In this paper we focus on a problem that arises when labeling schematized maps, e.g. for subway networks. We present algorithms for labeling points on a line with axis-parallel rectangular labels of equal height. Our aim is to maximize label size under the constraint that all points must be labeled.
Even a seemingly strong simplification of the general point-labeling problem, namely to decide whether a set of points on a horizontal line can be labeled with sliding rectangular labels, turns out to be weakly NP-complete. This is the first labeling problem that is known to belong to this class. We give a pseudo-polynomial time algorithm for it.
In case of a sloping line points can be labeled with maximum-size square labels in $O(n \log n)$ time if four label positions per point are allowed and in $O(n^3 \log n)$ time if labels can slide. We also investigate rectangular labels.

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, cartographers, graph drawers, and computational geometers have suggested numerous approaches, such as expert systems [1], zero-one integer programming [16], approximation algorithms [7, 12–14], simulated annealing [5] and force-driven algorithms [9] to name only a few. An extensive bibliography about label placement can be found at [15]. The ACM Computational Geometry Impact Task Force report [4] denotes label placement as an important research area. Manually labeling a map is a tedious task that is estimated to take 50% of total map production time.

* Partially supported by *PAI* project FQM—0164

When producing schematized maps [3], e.g. for road or subway networks, an interesting new label-placement problem has to be solved: that of labeling points on a line, e.g. stations on a specific subway line. We assume that all labels are parallel to each other and contain text of the same font size, so we can model labels by axis-parallel rectangles of equal height. We investigate two different labeling models, 4P and 4S, that were introduced in [7] and [14], respectively. In the *fixed-position model* 4P a label must be placed such that one of its four corners coincides with the point site to be labeled. The *slider model* 4S is less restrictive in that a label can be placed such that *any* point of its boundary coincides with the site. See Figure 1 for a variety of point-labeling models that have been studied previously [14, 12]. In that figure, each rectangle stands for a feasible label position. An arrow between two rectangle indicates that additionally all label position are feasible that arise when moving one rectangle on a straight line onto the other.

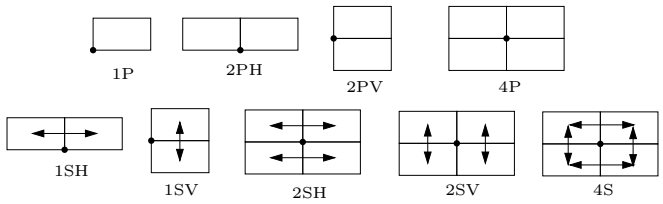

Fig. 1. Each model has an abbreviation of the form xMD where $M \in \{\mathrm{P}, \mathrm{S}\}$ stands for fixed-position model (P) or slider model (S), $x \in \{1, 2, 4\}$ refers to the number of fixed positions or sliding directions, and $D \in \{\emptyset, \mathrm{H}, \mathrm{V}\}$ indicates the horizontal or vertical direction in which fixed-position labels are arranged or labels can slide.

While most point-labeling problems are computationally hard [7], one would expect to be in a better situation if the input points are not scattered all over the plane but lie on a line. We show, however, that this is not necessarily true: labeling points on a horizontal line with sliding rectangles remains NP-hard. We do give a pseudo-polynomial time algorithm for that problem and show that several simplifications—square labels or no sliding—all have efficient algorithms. Other point-labeling problems that are not NP-hard include labeling points with maximum-size rectangles in one of two positions [7] or with maximum-size rectangles of aspect ratio 1:2 in one of four special positions [13]. There is also a polynomial-time algorithm that decides whether points on the boundary of a rectangle can be labeled with so-called elastic labels, i.e. rectangular labels of fixed area but flexible length and height [10]. Last but not least the problems 1d-kPH and 1d-1SH of labeling points on a horizontal line with labels in a constant number of positions and with sliding labels, respectively, have been studied under the restriction that all labels must be placed on top of the line [11, 12].

Our paper is structured as follows. In Section 2 we investigate the problem of labeling points on a horizontal line with axis-parallel rectangular labels that touch the line. In Section 3 we consider labeling points on sloping lines with

squares and sketch how some of these can be extended to rectangular labels. Throughout the paper we consider labels topologically open, i.e. they may touch other labels or input points. An *M-labeling* maps each input point to a label position that is allowed in labeling model M such that no two labels intersect. For a variety of labeling models, refer to Figure 1. In our paper the names of the models in Figure 1 are prefixed with "1d-" or "Slope-", in order to refer to the corresponding problems where all input points lie on a horizontal or sloping line, respectively. An *optimal* labeling will refer to a labeling where all labels are scaled by the same factor and this factor is maximum (prefix "Max-").

2 Points on a horizontal line

So far only Poon et al. have explicitely given algorithms for labeling points on a horizontal line [12]. They assume points with weights and investigate algorithms for maximizing the weighted sum of points that can be labeled *above* the line. Their aim actually is to label points with unit-height labels in the plane, but they reduce the difficult two-dimensional rectangle-placement problem into simpler one-dimensional interval-placement problems by means of line stabbing. Solving the 1d-problems (near-) optimally then gives approximation algorithms for the 2d-problem. See Figure 1 for the labeling models Poon et al. consider. The discrete case 1d-kPH, where each point has only a constant number of feasible label positions, is a special case of maximum-weight independent set (MWIS) on interval graphs. They use a MWIS algorithm to solve 1d-kPH in $O(kn \log n)$ time. In the weighted case the problem 1d-1SH where labels can slide horizontally above the given line (see Figure 1) is equivalent to a job scheduling problem, namely single-machine throughput maximization. It is not known whether a polynomial-time algorithm for 1d-1SH exists in the weighted case. Poon et al. modify a fully polynomial-time approximation scheme (FPTAS) for single-machine throughput maximization to approximate 1d-1SH: for each $\varepsilon > 0$ they obtain a factor-$(1+\varepsilon)$ approximation algorithm that runs in $O(n^2/\varepsilon)$ time and uses $O(n/\varepsilon)$ space. They also give an exact pseudo-polynomial time algorithm based on dynamic programming for 1d-1SH with a bounded number of different weights and an exact $O(n^2 \log n)$-time algorithm for the special case of square labels (i.e. intervals of fixed length). A similar approach can be used to approximate 1d-2SH (and, equivalently 1d-4S) in the weighted case: there is a factor-$(1.8+\varepsilon)$ approximation algorithm for 2-machine throughput maximization that runs in $O(n^2/\varepsilon)$ time [2].

Kim et al. have investigated algorithms for labeling axis-parallel line segments with sliding maximum-width rectangles [11]. They also consider the 1d-case first and show that 1d-1SH can be decided in linear time for unit squares (unit-length intervals) in the unweighted case if points are given in left-to-right order. In the same paper they also give a linear-time algorithm for the problem Max-1d-1SH, where the label length is maximized under the restriction that all labels have the same length.

In this section we will investigate a problem that is more difficult than 1d-1SH: we allow to place labels both above and below the horizontal line, say the x-axis, on which the input points are given. Let us start by introducing some notions that we will use throughout the paper, both for horizontal and sloping lines. We will direct the line from left (bottom) to right (top) and process the points in this linear order.

Definition 1. *Given a set $P = \{p_1, \ldots, p_n\}$ of n points on a line ℓ in lexicographical order, we refer to ℓ as the* input line *and direct it according to the order on P. Given an axis-parallel label L_i for each $p_i \in P$ and a labeling model M, a k-tuple $R = (r_1, \ldots, r_k)$ is a* k-realization *of P if each entry r_i encodes a position of L_i that is valid in M and no two labels intersect.*

For the 4P-model an entry of a k-realization is simply an integer $r_i \in \{1, 2, 3, 4\}$ that specifies in which of the four quadrants (in canonical order) L_i lies relative to a coordinate system with origin p_i. For the 4S-model we take $r_i \in [1, 5[$ with the obvious meaning that e.g. 2.5 is half way in between position 2 and 3. In order to express minimality among realizations we need at least a partial order on the set of possible k-realizations and thus on the label positions r_i. Intuitively, a minimum k-realization should be a k-realization that leaves the maximum amount of freedom for the placement of label L_{k+1}. This leads to the concept of the *shadow* of a k-realization—space that cannot be used for placing L_{k+1}. Our definition depends on the fact that our labels are always axis-parallel rectangles.

Definition 2. *The* foremost vertex *of a label L is the point on the boundary of L that is furthest in the direction of the input line ℓ. In case of a tie a point on ℓ wins. The* shadow $s(L)$ of a label L *is the quadrant of the plane that contains L and is defined by the foremost vertex of L and the two adjacent edges of L. The* shadow of a k-realization R *is $s(R) = \cup_{i=1}^{k} s(L_i)$. Two k-realizations are* equivalent *if they have the same shadows.*

For shadows of labels, see Figure 2. A shadow of a k-realization given a sloping line is depicted in Figure 5. Let us now focus on horizontal lines.

Definition 3. *If ℓ is a horizontal line, the* dual *of a k-realization R is the k-realization R^* with $r_i^* = 5 - r_i$ for $i = 1, \ldots, k$. We write $R \leq R'$ if $s(R) \subseteq s(R')$ or $s(R^*) \subseteq s(R')$. $R = (r_1, \ldots, r_k)$ is a* minimum k-realization *if $(r_1, \ldots, r_i) \leq R'$ for all i-realizations R' and for each $i = 1, \ldots, k$.*

If ℓ is horizontal, the shadow of a realization R can be denoted by (t, b) where t (b) is the x-coordinate of the right edge of the rightmost label in R above (below) ℓ. The dual R^* of R is obtained by mirroring R at ℓ. For a minimum and a non-minimum 4-realization, see Figure 3.

Lemma 1. *If there is a k-realization R of P then there is also a minimum k-realization R' of P.*

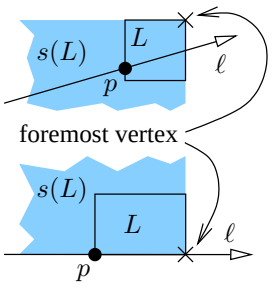

Fig. 2. Shadows of labels.

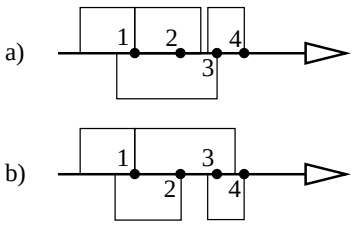

Fig. 3. A minimum (a) and a non-minimum realization (b).

Proof. Let $R_{|i}$ be the i-realization (for $i = 1, \ldots, k-1$) obtained from R by removing its $k - i$ last entries (i.e. labels). Our proof is by induction over k. The claim is certainly valid for $k = 1$: in this case $R' = (2)$ or $R' = (3)$, i.e. place the label of p_1 leftmost. For $k > 1$ if $R_{|k}$ is not a minimum k-realization then by our induction hypothesis we have a minimum $(k-1)$-realization R''. Clearly $R'' \leq R_{|k-1}$, thus adding the label L_k of R to R'' gives a k-realization R'. To make sure that R' is in fact minimum, we push L_k as far left as possible, checking positions both below and above ℓ. If this new R' was not a minimum k-realization we would have a contradiction to the minimality of R''. □

Thus it is enough to keep track of minimum k-realizations to solve the decision problem. Among these only non-equivalent k-realizations are of interest. Their number can be bounded as follows.

Lemma 2. *Given* 1d-4P *there are at most two non-equivalent minimum k-realizations for $k = 1, \ldots, n$.*

This is proved by induction over k and by going through all different possibilities according to the position of p_k. The duality of two realizations is important here.

Theorem 1. *If points are given in left-to-right order,* 1d-4P *with rectangles can be decided in linear time and space.*

Proof. We label points in the given order starting with the two minimum 1-realizations that we get from labeling p_1 leftmost, i.e. in positions 2 and 3. Then in each step we only have to combine each minimum $(k-1)$-realization R with the two leftmost placements of label k that R allows (if any) and compare the resulting at most four realizations with each other. According to Lemma 2 at most two of these are minimum and have to be kept. If at some point label k cannot be combined with any of the minimum $(k-1)$-realizations, then Lemma 1 guarantees that no k-realization exists. In this case the algorithm outputs "no", otherwise it returns a minimum n-realization. □

The maximization version Max-1d-4P of this problem is the following: given a set P of n points $p_1 = (x_1, 0), \ldots, p_n = (x_n, 0)$ each with a label of length l_i and unit height, find the largest stretch factor $\lambda_{\max}$ such that there is a 1d-4P-labeling of P with labels of length $\lambda_{\max} l_1, \ldots, \lambda_{\max} l_n$ and determine the corresponding labeling.

Theorem 2. Max-1d-4P *can be solved in* $O(n^2 \log n)$ *time using* $O(n^2)$ *space or in* $O(n^3)$ *time using linear space.*

Proof. First we sort the input points lexicographically. Let $\Delta x_{i,j} = |x_j - x_i|$. Since at least two labels must touch each other in an optimal labeling, $\lambda_{\max}$ must be in the list $L = \{\Delta x_{i,j}/l_i, \Delta x_{i,j}/l_j, \Delta x_{i,j}/(l_i + l_j) : 1 \leq i < j \leq n\}$ of all potentially optimal stretch factors. We compute L, sort L, and do a binary search on L calling our decision algorithm in each step. This takes $O(n^2 \log n)$ time and uses quadratic space for L. Instead we could also compute the elements of L on the fly and test them without sorting. □

The problem becomes much harder when we allow labels to slide horizontally. We will show this by reducing a special variant of Partition to 1d-4S. Actually sliding vertically does not help when the input line is horizontal, so 1d-4S is equivalent to 1d-2SH, where only horizontal sliding is allowed. For the same reason the problem 1d-2SV, where only vertical sliding is allowed, is equivalent to 1d-4P, see Figure 1.

Theorem 3. 1d-4S *is NP-complete.*

Proof. The problem is in $\mathcal{NP}$ since we have the following non-deterministic polynomial-time decision algorithm. First guess an n-tuple c with entries $c_i \in \{a, b\}$ that encode whether label L_i lies $a$bove or $b$elow the input line in the solution. Then go through the points from left to right and place L_i at position 1 if $c_i = a$ and at position 4 if $c_i = b$. Push L_i left until it either hits a previously placed label or is in its leftmost position. If L_i cannot be placed without intersection, there cannot be a solution that conforms to c. However, if there is a 1d-4S-labeling of the given points, it is found with non-zero probability.

To show the NP-hardness we reduce the following NP-hard variant of Partition [8] to 1d-4S. Given positive integers $a_1, \ldots, a_{2m}$ is there a subset I of $J = \{1, \ldots, 2m\}$ such that I contains exactly one of $\{2i-1, 2i\}$ for $i = 1, \ldots, m$ and $\sum_{i \in I} a_i = \sum_{i \in J \setminus I} a_i$? We will reduce an instance A of this problem to an instance (P, L) of 1d-4S such that A can be partitioned if and only if there is a 1d-4S-labeling of P with the corresponding labels from L.

First let C be a very large and c a very small number, e.g. $C = 1000 \sum_{i \in J} a_i$ and $c = \min_{i \in J} a_i/1000$. Our point set P consists of 4 stoppers and $2m$ usual points a, b, p_1, p_2, …, p_{2m}, y, z from left to right with distances $\overline{ab} = \overline{yz} = c$, $\overline{bp_1} = \overline{p_{2m}y} = C/2$, $\overline{p_{2i-1}p_{2i}} = (a_{2i-1}+a_{2i})/2$, and $\overline{p_{2i}p_{2i+1}} = C$, thus $\overline{by} = mC + \sum_{i \in J} a_i/2$, see Figure 4. The corresponding labels have length $l_a = l_b = l_y = l_z = \overline{by}$ and $l_i = C + a_i$, thus $\sum_{i \in J} l_i = 2mC + \sum_{i \in J} a_i = 2\overline{by}$. Due to the long labels of the stoppers, the other points must be labeled between these stoppers.

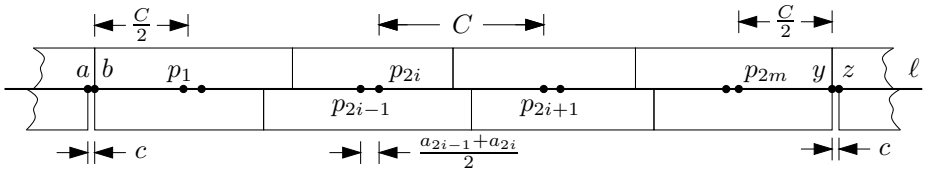

Fig. 4. Instance of 1d-4S to which Partition is reduced.

The total space available above and below the input line is $\overline{by} + \overline{az} = 2\overline{by} + 2c$ and thus just slightly more than the total length of the labels.

If there is a labeling for this instance then it must be tight (neglecting the $2c$ extra space) and the number of labels above and below the input line ℓ must be equal. Due to its length a label is attached to its point roughly in its center. Thus the labels of p_{2i-1} and p_{2i} lie on opposite sides of ℓ. Therefore the indices of the points whose labels lie above ℓ give the desired partition I of J.

On the other hand if there is a partition I of J then we can label P as follows. For each p_i with $i \in I$ we place its label with the lower left corner at $x_b + \sum_{j \in I, j<i} l_j$, where x_b is the x-coordinate of b. The labels of the other m points are placed analogously below ℓ. □

We needed extremely long labels and point distances to construct the reduction from Partition to 1d-4S above. In practice such labels are not common, which makes it worthwhile to design a pseudo-polynomial time algorithm whose running time depends not only on n, the size of the input, but also on $l_{\max}$, the length of the longest label. Pseudo-polynomial time algorithms have been suggested for point labeling before. There is a scheduling algorithm that can be used for weight maximization given 1d-1SH and runs in $O(dn \log\log d)$ time, where $d = x_n + l_n$ [2]. Another example is an approximation algorithm that labels points with circles of radius at least $R^*/3.6$ in $O(n \log n + n \log R^*)$, where R^* is the maximum label radius [6].

Theorem 4. *If the input consists exclusively of integers and points are sorted from left to right,* 1d-4S *can be solved in* $O(nl_{\max}^2)$ *time and space, where* $l_{\max}$ *is the length of the longest label.*

Proof. We will use dynamic programming with a table T of size $(n+1) \times (2l_{\max}+1) \times (2l_{\max}+1)$. Let a k-realization be *leftmost* if all its labels are pushed as far left as possible. Note that a leftmost realization is not necessarily minimum. Now an entry $T[k,t,b]$ is a boolean that answers the question "Is (t,b) the shadow of a leftmost k-realization?" where $t, b \in \{-l_{\max}, \ldots, l_{\max}\}$ are measured relative to x_{k+1} assuming $x_{n+1} = x_n + l_n$. Initially all table entries are false except $T[0, -l_{\max}, -l_{\max}]$. The entries of level k are computed from those in level $k-1$ as follows. Let $f_k(x) = \max\{-l_{\max}, x - \Delta x_{k,k+1}\}$ be the function that maps a point x measured relative to x_k to a point $f_k(x)$ measured relative to x_{k+1} with a lower bound of $-l_{\max}$. For each level-$(k-1)$ entry $T[k-1,t,b]$ that is true we

switch at most two entries in level k to true: if $t \leq 0$ we generate a new leftmost k-realization by placing L_k leftmost above ℓ. If additionally $t \geq -l_k$ then the new label touches the last label above ℓ and we set $T[k, f_k(t+l_k), f_k(b)]$ to true, otherwise $T[k, f_k(0), f_k(b)]$. The case $b \leq 0$ is symmetric. The algorithm returns true if and only if there is an entry of value true at level n.

It is not difficult to modify the algorithm within the time and space bound of $O(nl_{\max}^2)$ such that it actually computes a labeling if one exists. The proof of correctness is by induction over k. □

3 Points on a sloping line with square labels

In this section we will investigate labeling problems where the input line has positive slope and labels are equal-size squares. We will give decision and label-size maximization algorithms for both the discrete and the continuous versions Slope-4P and Slope-4S, respectively. We allow labels to intersect the input line, otherwise only two label positions per point would be valid, and the decision version could simply be reduced to 2-SAT and solved in linear time [7]. We start with the more difficult problem Slope-4S and later show how it can be simplified to Slope-4P. Since we do not have the notion of duality for sloping lines as in Definition 2, we redefine minimality as follows.

Definition 4. *Let R, R' two k-realizations. We write $R \leq R'$ if $s(R) \subseteq s(R')$. $R = (r_1, \ldots, r_k)$ is a* minimum k-realization *if $(r_1, \ldots, r_i) \leq R'$ for all i-realizations R' and for each $i = 1, \ldots, k$. A $(k+1)$-realization is the* child *of a k-realization if their first k entries agree.*

Lemma 3. *A k-realization has at most two children that are minimum.*

Proof. Given a k-realization R and a point p_{k+1} with a square label L, R has a child iff $p_{k+1} \notin s(R)$. In this case place L in position 1. There are two paths in which L can be slid towards its optimal position 3, either left-down on a path via position 2 or down-left on a path via position 4. Here optimality refers to the resulting $(k+1)$-realization, and sliding means that the label is moved continuously from one position to the other while touching p_{k+1}. All shadows of L on one path are comparable to each other, while no shadow of L on one path is comparable with a shadow of L on the other path except for the two endpoints. Thus sliding L as far as possible without intersecting $s(R)$ on each of the two paths gives minimum shadows for L, see Figure 5. Appending these positions of L to R yields at most two children that can be minimum $(k+1)$-realizations. □

We will continue to use the terms left-down and down-left from the previous proof. We will say that a k-realization $R = (r_1, \ldots, r_k)$ is of type LD if $r_k \leq 3$ and of type DL if $r_k \geq 3$. Extending this notation, we will say that R is of type X-X if the last two entries of R are both at most 3 or both at least 3, X-Y otherwise. The following two lemmas are easy to prove.

Lemma 4. *The shadow of a realization is determined by its last two labels.*

Lemma 5. *Two k-realizations of type* DL-DL *are comparable to each other, and their minimum can be determined in constant time.*

Due to symmetry the same holds for k-realizations of type LD-LD.

Theorem 5. *Given unit-square labels,* Slope-4S *can be solved in quadratic time and space.*

Proof. We sort the input points lexicographically and process them in this order. In each step we maintain a superset $\mathcal{R}$ of the set of minimum k-realizations. The idea to bound the size of $\mathcal{R}$ is the following. In step $k+1$ each of the k-realizations in $\mathcal{R}$ yields at most two $(k+1)$-realizations according to Lemma 3, one of type LD and one of type DL. Of these $(k+1)$-realizations at most $|\mathcal{R}|$ will be of type X-Y. All $(k+1)$-realizations of type DL-DL are comparable to each other according to Lemma 5. Finding the minimum among them takes linear time, analogously for those of type LD-LD. Thus we keep at most $|\mathcal{R}|$ $(k+1)$-realizations of type X-Y and 2 of type X-X. According to Lemma 3, the minimum $(k+1)$-realizations must be among them. Since the number of realizations increases by two in each step, the total running time and space consumption are quadratic. □

There are point sets with a linear number of minimum n-realizations of type X-Y, for an example see Figure 6, so one cannot hope to do better using the concept of minimum realizations.

The following lemma is shown in a similar way as Lemma 1 in [11]. The idea is that in an optimal solution there are paths of labels that touch each other, and among these paths there is a path whose first and last label touch input points—either with their top and bottom edge or with their left and right edge. Otherwise all labels could be slid and enlarged by a small factor.

Lemma 6. *Given* Max-Slope-4S *the maximum label size lies in the set* $L = \{\Delta x_{i,j}/m, \Delta y_{i,j}/m : 1 \le i < j \le n, 1 \le m \le j - i\}$.

This immediately yields an algorithm for Max-Slope-4S using binary search on the list L of potential label sizes and the decision algorithm of Theorem 5.

Corollary 1. *Given square labels,* Max-Slope-4S *can be solved in $O(n^3 \log n)$ time using $O(n^2)$ space.*

As expected the discrete version Slope-4P can be solved faster.

Theorem 6. *Given unit-square labels and points sorted lexicographically,* Slope-4P *can be solved in linear time and space.*

Proof. Lemma 4 holds in the discrete case Slope-4P as well. Here, however this immediately implies that there are at most $4^2 = 16$ (a case analysis reduces this to 2) different shadows in each step. When placing label L_{k+1} we get at most 32 $(k+1)$-realizations by Lemma 3, but of these at most 16 can have different shadows and must be kept. Filtering these out can be done in constant time, which yields a linear-time algorithm. □

Corollary 2. *Given square labels,* Max-Slope-4P *can be solved in* $O(n \log n)$ *time using linear space.*

Proof. It is not hard to see that in an optimal solution of Max-Slope-4P only labels of neighboring points or points that have a common neighbor can touch, see Figure 7. Thus there is a list $L = \{\Delta x_{i,j}/m, \Delta y_{i,j}/m : \ 1 \leq i \leq n, \ j \in \{i+1, i+2\}, \ m \in \{1, 2\}\}$ of linear length that contains the maximum label size. The algorithm is as follows: first sort P, then compute L, sort L, and finally do a binary search on L. □

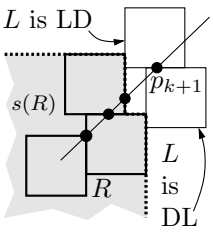

Fig. 5. The two paths of label L given 4S.

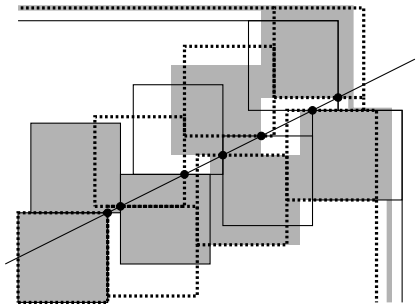

Fig. 6. Three incomparable shadows of type LD-DL.

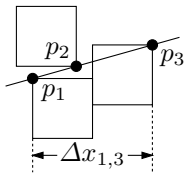

Fig. 7. Touching 4P-labels.

The problem Slope-4P can be solved similarly for unit-height rectangles in linear time. The maximization version, however, takes longer, namely $O(n^2 \log n)$ time, since there is a quadratic number of possibly optimal stretch factors.

Conclusions

In this paper we have studied problems that arise when labeling schematic maps such as subway networks where the points to be labeled lie on a line. Even among these seemingly strong simplifications of the general point-labeling problem there are cases that remain weakly NP-hard.

There are plenty of open problems left in 1d-labeling. First of all we would like to see the time complexity of our algorithm for Max-Slope-4S reduced. Then it would be interesting to see how much the reduction to one dimension helps to solve the label-number or label-weight maximization problem. Given a set of points in the plane, each with its own label length but fixed height, there is a PTAS for finding the largest subset of points that can be labeled [14], while for weight maximization only a factor-$(\frac{1}{2} - \varepsilon)$ approximation algorithm is known [12].

References

1. J. Ahn and H. Freeman. AUTONAP—an expert system for automatic map name placement. In *Proceedings International Symposium on Spatial Data Handling*, pages 544–569, 1984.
2. P. Berman and B. DasGupta. Multi-phase algorithms for throughput maximization for real-time scheduling. *Journal of Combinatorial Optimization*, 4(3):307–323, Sept. 2000.
3. S. Cabello, M. de Berg, S. van Dijk, M. van Kreveld, and T. Strijk. Schematization of road networks. In *Proceedings of the 17th Annual ACM Symposium on Computational Geometry (SoCG'01)*, 2001. To appear.
4. 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, 1999.
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. S. Doddi, M. V. Marathe, and B. M. Moret. Point set labeling with specified positions. In *Proc. 16th Annu. ACM Sympos. Comput. Geom. (SoCG'00)*, pages 182–190, Hongkong, 12–14 June 2000.
7. M. Formann and F. Wagner. A packing problem with applications to lettering of maps. In *Proc. 7th Annu. ACM Sympos. Comput. Geom. (SoCG'91)*, pages 281–288, 1991.
8. 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.
9. S. A. Hirsch. An algorithm for automatic name placement around point data. *The American Cartographer*, 9(1):5–17, 1982.
10. C. Iturriaga and A. Lubiw. Elastic labels around the perimeter of a map. In *Proceedings of the 8th International Workshop on Algorithms and Data Structures (WADS'99)*, volume 1663 of *Lecture Notes in Computer Science*, pages 306–317, Vancouver, B.C., Canada, 12–14 Aug. 1999. Springer-Verlag.
11. 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.
12. S.-H. Poon, C.-S. Shin, T. Strijk, and A. Wolff. Labeling points with weights. In *Proc. 12th Annual International Symposium on Algorithms and Computation (ISAAC'01)*, Lecture Notes in Computer Science, Christchurch, 19–21 Dec. 2001. Springer-Verlag. To appear.
13. Z. Qin, A. Wolff, Y. Xu, and B. Zhu. New algorithms for two-label point labeling. In *Proc. 8th Annu. Europ. Symp. on Algorithms (ESA'00)*, volume 1879 of *Lecture Notes in Computer Science*, pages 368–379, Saarbrücken, 5–8 Sept. 2000. Springer-Verlag.
14. M. van Kreveld, T. Strijk, and A. Wolff. Point labeling with sliding labels. *Computational Geometry: Theory and Applications*, 13:21–47, 1999.
15. A. Wolff and T. Strijk. The Map-Labeling Bibliography. `http://www.math-inf.uni-greifswald.de/map-labeling/bibliography/`, 1996.
16. S. Zoraster. The solution of large 0-1 integer programming problems encountered in automated cartography. *Operations Research*, 38(5):752–759, 1990.