Labeling collinear sites*

Michael A. Bekos†
National Technical University of Athens,
School of Applied Mathematical & Physical Sciences,
15780 Zografou, Athens, Greece

Michael Kaufmann‡
University of Tübingen,
Institute for Informatics,
Sand 13, 72076 Tübingen, Germany

Antonios Symvonis§
National Technical University of Athens,
School of Applied Mathematical & Physical Sciences,
15780 Zografou, Athens, Greece

ABSTRACT

We consider a map labeling problem, where the sites to be labeled are restricted on a line L. This is quite common e.g. in schematized maps for road or subway networks. Each site s_i is associated with an axis-parallel $w_i \times h_i$ label l_i, which can be placed anywhere on the "*boundary*" of the input line L. The main task is to place the labels in distinct positions, so that they do not overlap and do not obscure the site set, and to connect each label with its associated site through a *leader*, such that no two leaders intersect. We propose several variations of this problem and we investigate their computational complexity under certain optimization criteria.

Keywords: Labeling, Sites, Labels, Leaders, Line.

1 INTRODUCTION

Automated map labeling is a well-known problem, which has received considerable attention due to the large number of applications in both Cartography and Graphical Information Systems. Manual label placement is a time-consuming task, which is estimated to take 50% of total map production. Apart from that, the ACM Computational Geometry Impact Task Force report [8] denotes label placement as an important research area.

Recent research on map labeling has been primarily focussed on labeling point-features. In order to ensure readability, unambiguity and legibility, cartographers suggest that the labels should be pairwise disjoint and close to the point (also referred to as *site* or *anchor*) to which they belong [18, 27]. Unfortunately, the majority of map labeling problems are shown to be *NP*-complete [1, 11, 19, 20, 24]. Due to this fact, the map labeling community has suggested various approaches, among them expert systems [2], gradient descent [16], approximation algorithms [11, 25], zero-one integer programming [28] and simulated annealing [29]. An extensive bibliography about label placement can be found at [26].

There are many variations of the point labeling problem, regarding the shape of the labels, the location of the sites or some optimization criterion, e.g. maximizing the size of labels. In this paper, we consider the case where all sites lie on the same line and are to be labeled with axis-parallel rectangular labels. This is a quite common approach e.g. in schematized maps for road or subway networks. Most of known labeling models for this problem produce quite legible labelings, when the input sites are sparsely distributed on the input line. However, when the site set contains a dense 5-tuple of sites they fail to produce labelings. To address this problem, we propose a more flexible labeling model, according to which the labels are placed on the "boundary" of the input line and are connected to their associated sites in a simple and elegant way by using non-intersecting polygonal lines, called *leaders*.

*The work is co - funded by the European Social Fund (75%) and National Resources (25%) - Operational Program for Educational and Vocational Training II (EPEAEK II) and particularly the Program PYTHAGORAS.

†e-mail: mikebekos@math.ntua.gr
‡e-mail: mk@informatik.uni-tuebingen.de
§e-mail: symvonis@math.ntua.gr

1.1 Problem Definition

Our labeling model in its primitive form can be described as follows: We are given a straight line L and a set S of n sites $s_i = (x_i, y_i)$ on L. Each site s_i is associated with an axis-parallel rectangular label l_i of dimensions $w_i \times h_i$. The "*boundary of L*" is defined by two lines L^T and L^B (one on top and one bellow L), that are translations of L by c_0 towards to $(0, \infty)$ and $(0, -\infty)$, respectively, where c_0 is a fixed, predefined, positive constant (see Figures 1 and 2). Labels have to be placed on the boundary of L, so that they do not overlap and do not obscure the site set, and, to be connected to their associated sites by non-intersecting polygonal lines, called *leaders*. Such labelings are referred to as *legal* or *crossing-free labelings* (for brevity, they are simply referred to as *labelings*).

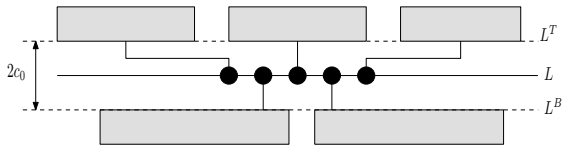


Figure 1: Horizontal input line.

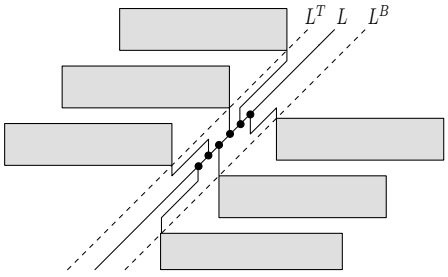


Figure 2: Sloping input line.

Our labeling model consists of several parameters (sites, labels, leaders, input line). So, it is reasonable to exist several variations of the primitive form discussed above, each giving rise to different labeling models.

Input line: The input line L may be horizontal (see Figure 1) or may have a positive slope (see Figure 2).

Labels: In general, the labels are of arbitrary sizes, i.e. label l_i associated with site s_i has width w_i and height h_i (*non-uniform labels*). However, in real applications labels usually contain text of the same font size. So, it is reasonable to separately consider the case, where the labels are of the same width and/or height (*uniform labels*). In our model, we further assume that each label can be placed anywhere on the boundary of L, so that either its bottom right or top left corner coincides with L^T or L^B, respectively. This implies that the labels do not overlap the input line and therefore do not obscure the site set.

Leaders: The leaders which connect the sites to their corresponding labels can also be of several types. In our approach, we focus on leaders of type-opo, which result in simple and easy to visualize labelings. Leaders of type-opo consist of three line segments. The first and third ones are orthogonal (o) to x-axis, whereas the second one is parallel (p) to the input line (see Figures 1 and 2). Degenerated case of a type-opo leader is a leader of type-o, which consists of only one line segment orthogonal to x-axis (i.e. the length if the p-segment is zero). Additionally, for each type-opo leader, we insist that its p-segment is located inbetween L^T and L^B (in the so-called *track routing area*) and does not intersect L. We further assume that the *thickness* $2c_0$ of the track routing area is large enough to accommodate all leaders.

Ports: The point where each leader touches its corresponding label is referred to as *port*. We assume either *fixed ports*, where each leader is only allowed to use a fixed set of ports on some label side (e.g. the middle point of a label side or some corner of the label; see Figure 2) or *sliding ports*, where the leader can touch any point of the label's side (see Figure 1).

Under a labeling model, one can define several optimization problems, adopting one of the following optimization criteria:

Minimize the total number of bends: Find a labeling, such that the total number of bends is minimum.

Minimize the total leader length: Find a labeling, such that the total leader length is minimum. Note that only the p-segments of the leaders contribute to the total leader length, since we assume that the thickness $2c_0$ of the track routing area is fixed.

1.2 Related Literature

The problem of labeling points on a single line has so far been studied by Garrido et al. [14] and Chen et al. [9], along two different labeling models, 4P and 4S. In the *fixed-position model* 4P a label must be placed, so that the site to be labeled coincides with one of its four corners (see Figure 3), whereas in the *sliding model* 4S a label can be placed so that the site lies on one of the boundary edges of the label (see Figure 4). One can also use prefixes $1d-$ and SLOPE- combined with each model to denote the type of the input line; $1d$ denotes a horizontal or vertical line, whereas SLOPE denotes a sloping line.

Garrido et al. showed that the $1d$-4S problem is NP-complete and they presented a pseudo-polynomial time algorithm to solve it. They also showed that several simplifications, e.g. square labels or no sliding, all have efficient algorithms. Chen et al. showed that the SLOPE-4P problem with rectangular labels of fixed-height can be solved in linear time, when the order of the input sites is given. They also showed that the problem of maximizing the size of the rectangular equal-width labels of the points on a horizontal line whose top or bottom edge coincide with the input line under the 4S model can be solved in $O(n^2 \log n)$ time.

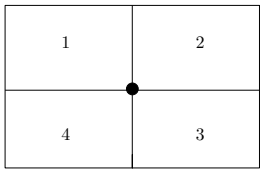


Figure 3: Illustration of 4P model.

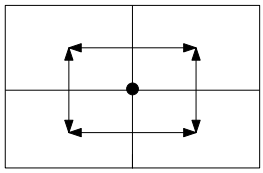


Figure 4: Illustration of 4S model.

Labeling where the labels are connected to their associated features by leaders has so far been studied in the map labeling literature by Bekos et al. [4, 5, 7], Fekete and Plaisant [10], Freeman et al. [12], Müller and Schödl [23] and Zoraster [29]. Our labeling model is quite similar to the *boundary labeling model* proposed by Bekos et al. [7] (see also [6]). In boundary labeling, the labels are placed on the boundary of a rectangle R (referred to as *enclosing rectangle*), which encloses the set of sites, and are connected to their associated sites by non-intersecting leaders. In most of the algorithms presented for boundary labeling, the labels are considered to be placed in predefined positions and their sizes are bounded by the dimensions of R. In this paper, we tackle both restrictions, since we do not assume the existence of the enclosing rectangle.

The paper is structured as follows: Section 2 reviews preliminary results required for the development of our algorithms. In Section 3, we consider the problem of labeling points on a horizontal line with axis-parallel rectangular labels. We propose efficient algorithms to determine labelings of either minimum total leader length or of minimum number of bends for the case, where the labels are placed above the input line. For the general case, where the labels can be placed on both sides of the input line we show that both problems are NP-complete. In Section 4, we extend the results of Section 3 to the case, where the input line has a positive slope. We conclude in Section 5 with open problems and future work.

2 PRELIMINARIES

A key component, that is heavily used in the description of our algorithms, is a formulation of our problem as a *Single Machine Scheduling problem*[1] *with due windows and symmetric earliness and tardiness penalties*, according to which: We are given a set of n jobs $J_1, J_2, \ldots, J_n$, which are to be executed on one machine. Each job J_i is associated with a processing time p_i and a time window (b_i, d_i). If a job J_i is processed entirely within its time window, it occurs no penalty. On the other hand, if the starting time σ_i of J_i commences prior to b_i (or the completion time $c_i = \sigma_i + p_i$ of J_i exceeds d_i), an earliness (tardiness) penalty E_i (T_i) incurs equal to the corresponding deviation. Thus, $E_i = \max\{b_i - \sigma_i, 0\}$ and $T_i = \max\{c_i - d_i, 0\}$. There are no restrictions on time windows, preemption is not allowed and the machine is continuously available. The objective is to determine a schedule, so that either the total earliness-tardiness penalty $\sum_{j=1}^{n}(E_j + T_j)$ or the number of penalized jobs is minimized.

Scheduling to minimize the total-earliness tardiness penalty: The general case of the problem of determining a schedule, so that the total earliness-tardiness penalty is minimized, is shown to be NP-hard, since it can be viewed as a generalization of the single-machine earliness-tardiness problem with distinct due dates, which is a well-known NP-complete

[1] Extensive surveys on the most important aspects of scheduling research are given at [3, 15, 17].

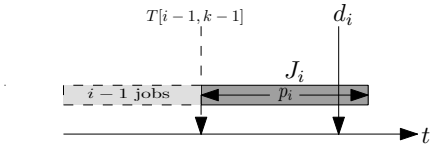
problem [13]. However, for the special case, in which the jobs are to be scheduled in a fixed predefined order, Koulamas [21] has proposed an efficient algorithm, which determines an optimal schedule in $O(n \log n)$ time by inserting idle time between jobs.

Scheduling to minimize the number of penalized jobs: In general, the problem of determining a schedule, so that the total number of penalized jobs is minimized can be solved in $O(n^2)$ time by employing a greedy algorithm of Lann and Mosheiov [22]. If the jobs are required to be scheduled in a predefined order, Lann and Mosheiov [22] have also proposed a dynamic programming based algorithm, which determines an optimal schedule in $O(n^2)$ time, only for the case of distinct due dates (i.e. time windows of zero length). In the following Theorem, we generalize their algorithm to support time windows.

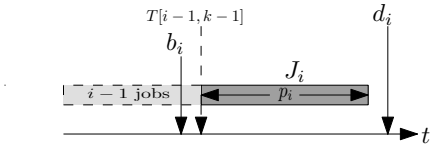
Theorem 1 *Given a set of n jobs $J_1, J_2, \ldots, J_n$, which are to be executed on one machine in this order, a processing time p_i and a time window (b_i, d_i) for each job J_i, we can compute in $O(n^2)$ time a schedule, so that the number of penalized jobs is minimized.*

Proof: Our dynamic programming algorithm employs a table T of size $(n+1) \times (n+1)$. For $0 \leq k \leq i \leq n$, entry $T[i,k]$ contains the minimum completion time for the subproblem consisting only of the first i jobs, such that at least k out of them are scheduled *on time* (i.e. they do not incur a penalty). If it is impossible to obtain a schedule for this setting, we set $T[i,k]$ to ∞. Therefore, all table entries $T[i,k]$, with $i < k$ are ∞.

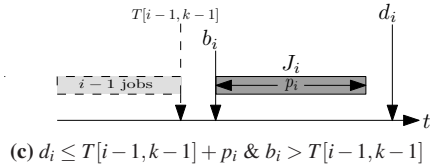
As usual, the table entries are computed in a bottom-up fashion. Assuming that we have scheduled the first $i-1$ jobs, we try to schedule the i-th job J_i. We distinguish two cases based on whether J_i is scheduled on time or not. From the two alternatives, we select the one, which minimizes the total completion time. Thus, for computing entry $T[i,k]$ we only need to know entries $T[i-1,k-1]$ and $T[i-1,k]$.



(a) $d_i \leq T[i-1,k-1] + p_i$



(b) $d_i \leq T[i-1,k-1] + p_i$ & $b_i \leq T[i-1,k-1]$



(c) $d_i \leq T[i-1,k-1] + p_i$ & $b_i > T[i-1,k-1]$

Figure 5: Different schedules obtained for the i-th job J_i.

Case 1: $d_i \leq T[i-1,k-1] + p_i$.
Refer to Figure 5a. In this case, it is obvious that J_i cannot be scheduled on time. Therefore, $T[i,k]$ can have a finite value only if $T[i-1,k]$ is finite. In this subcase, we simply schedule job J_i exactly after the $i-1$ already scheduled jobs, and obtain a schedule of total completion time $T[i-1,k] + p_i$. If on the other hand, $T[i-1,k] = \infty$, no solution with k on time jobs exists and thus $T[i-1,k] = \infty$. Both subcases can be described by the equation:

$$T[i,k] = T[i-1,k] + p_i$$

Case 2: $d_i > T[i-1,k-1] + p_i$.
Consider first the subcase where $b_i \leq T[i-1,k-1]$ (refer to Figure 5b). In this subcase, the total completion time is $T[i-1,k-1] + p_i$. In the subcase where $b_i > T[i-1,k-1]$ (refer to Figure 5c), we can schedule J_i, so that $\sigma_i = b_i$. Both subcases can be described by the equation: $T[i,k] = \max\{T[i-1,k-1], b_i\} + p_i$. However, if $T[i-1,k]$ is finite, then a different solution is also possible. The total completion time of this solution is $T[i-1,k] + p_i$. The above subcases can be expressed by the equation:

$$T[i,k] = \min\{T[i-1,k],\ \max\{T[i-1,k-1], b_i\}\} + p_i$$

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

$$T[i,k] = \begin{cases} T[i-1,k] + p_i, & \text{if } d_i \leq T[i-1,k-1] + p_i \\ \min\{T[i-1,k], \max\{T[i-1,k-1], b_i\}\} + p_i, & \text{if } d_i > T[i-1,k-1] + p_i \end{cases}$$

Algorithm 1 outputs the maximum possible number of non-penalized jobs and it is directly based on the above recurrence relation (see block 1 of the algorithm). Block 2 of Algorithm 1 computes the maximum possible number of non-penalized jobs by identifying the largest j with $0 \leq j \leq n$ such that $T[n,j] < \infty$.

Algorithm 1 needs $O(n^2)$ time and space, since it maintains a $(n+1) \times (n+1)$ table and each entry of this table needs a constant effort to be computed. By using an extra table of the same size as T, the algorithm can easily be modified, such that it also computes the starting times $\sigma_1, \sigma_2, \ldots \sigma_n$ of jobs $J_1, J_2, \ldots J_n$, respectively in the optimal solution.

1: MINPENALIZEDJOBS

input : A set of n jobs $J_1, J_2, \ldots, J_n$, which are to be executed on one machine, a deterministic processing time p_i and a time window (b_i, d_i) for each job s_i.
output : The maximum number of non-penalized jobs.
require: Job J_i should be executed before J_j, if $i < j$.

1 {Fill dynamic programming table T}
$T[0,0] = 0$
for $i = 1$ **to** n **do**
  $T[i,0] = T[i-1,0] + p_i$
  $T[i-1,i] = \infty$
  for $k = 1$ **to** i **do**
    if $d_i > T[i-1,k-1] + p_i$ **then**
      $T[i,k] = T[i-1,k] + p_i$
    else
      $T[i,k] = \min\{T[i-1,k],\ \max\{T[i-1,k-1], b_i\}\} + p_i$

2 {Compute maximum possible number of non-penalized jobs}
for $j = n$ **down to** 0 **do**
  if $T[n,j] < \infty$ **then**
    return j

□

3 SITES ON A HORIZONTAL LINE

In this section, we consider the case, where the sites to be labeled are restricted on a horizontal line[2]. W.l.o.g. we assume that L is the x-axis, i.e. $L : y = 0$. We want to obtain legal type-*opo* labelings either of minimum total leader length or of minimum number of bends. Recall that in the case of a horizontal line, the boundary of L is defined by lines $L^T : y = c_0$ and $L^B : y = -c_0$. This implies that either the bottom or the top boundary edge of each label should coincide with either L^T or L^B, respectively. Moreover, each type-*opo* leader should have its p-segment either between L^T and L or between L and L^B (see Figure 1).

Before we proceed with the description of our algorithms, we make some observations regarding *opo*-labelings. It is easy to see that the problem of determining a labeling of minimum total leader length is equivalent to the problem of determining a labeling, so that the sum of the lengths of the p-segments of all leaders is minimum. This is because we assumed that the thickness $2c_0$ of the track routing area is fixed and large enough to accommodate all leaders. We can also observe that in any legal *opo*-labeling, the horizontal order of the sites with labels positioned above (or below) the input line is identical to the horizontal order of their corresponding labels.

3.1 Labels above the input line

We first consider the case where the labels are restricted on the same side of the input line L. W.l.o.g. we assume that all labels will be placed above L. This implies that the bottom boundary edge of each label should coincide with L^T (see figure 6). We consider the more general case of labels with sliding ports, i.e. the leader connecting the site to the label has simply to touch some point in the perimeter of the label.

3.1.1 Total leader length minimization:

We describe how to compute in $O(n \log n)$ time a labeling with leaders of type-*opo*, so that the total leader length is minimum. To solve this problem, we will reduce it to the single-machine scheduling problem with due windows and symmetric earliness and tardiness penalties. The reduction we propose can be achieved in linear time. For each site $s_i = (x_i, 0)$, we introduce a job J_i. The processing time p_i of job J_i is equal to the width w_i of label l_i. The corresponding due window (b_i, d_i) of job J_i is $(x_i - w_i, x_i + w_i)$ and its length is equal to $2w_i$ (see Figure 6).

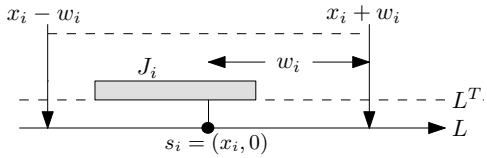


Figure 6: For each site s_i, a job J_i of processing times w_i is introduced.

We proceed by applying Koulamas [21] algorithm to obtain a schedule σ_{opt}, which minimizes the total earliness-tardiness penalty. The exact positions of labels are then determined based on the starting times $\sigma_1, \sigma_2, \ldots, \sigma_n$ of jobs $J_1, J_2, \ldots, J_n$, respectively, under schedule σ_{opt}. More precisely, the x-coordinate of the lower left corner of label l_i is σ_i, and since, the y-coordinate of each of the lower left corners is equal to c_0, the exact positions of all labels are well-specified.

If a job J_i is placed entirely within its time window, the corresponding leader c_i, which connects label l_i with site s_i, is of type-o, which implies that leader c_i does not contribute to the total leader length. On the other hand, if job J_i deviates its time window, then leader c_i contributes to the total leader length a penalty equal to the corresponding deviation. So, the total leader length is equal to the total earliness-tardiness penalty of the implied scheduling problem. The above result is summarized in Theorem 2.

Theorem 2 *Given a set S of n sites on a horizontal line L, each associated with a rectangular $w_i \times h_i$ label l_i that can be placed above L, we can compute in $O(n \log n)$ time a legal opo-labeling of minimum total leader length.*

3.1.2 Leader bend minimization:

We use the same reduction to obtain a labeling of minimum number of bends. In this case, we proceed by applying the algorithm of Theorem 1 to obtain a schedule of minimum number of penalized jobs. Observe that if a job J_i is *on time* (i.e. it does not incur a penalty), the corresponding leader c_i, which connects label l_i with site s_i, is of type-o, which implies that leader c_i does not contribute to the total number of bends. On the other hand, if job J_i is either early or tardy, then leader c_i contributes two bends to the total number of bends. So, the total number of leader bends is equal to twice the total number of penalized jobs of the implied scheduling problem. The above result is summarized in Theorem 3.

Theorem 3 *Given a set S of n sites on a horizontal line L, each associated with a rectangular $w_i \times h_i$ label l_i that can be placed above L, we can compute in $O(n^2)$ time a legal opo-labeling of minimum number of bends.*

3.2 Labels on both sides of the line

In this section, we show that when non-uniform labels can be placed on both sides of L the problem of determining a legal labeling of either minimum total leader length or of minimum number of bends is *NP*-hard.

3.2.1 Total leader length minimization:

We will show that the decision problem "Is there a labeling with total leader length no more than k?" is *NP*-complete and hence the corresponding optimization problem is at least as hard. We also do give a pseudo-polynomial time algorithm for that problem, establishing that the problem is NP-hard in the ordinary sense.

Theorem 4 *Given $k \in \mathbb{Z}^+$ and a set S of n sites on a horizontal line L, each associated with a rectangular label l_i of dimensions $w_i \times h_i$, it is NP-complete to determine if there exists a legal opo-labeling with labels on both sides of L, such that the total leader length is at most k.*

Proof: The reduction we propose is by restriction, since our problem can be viewed as a generalization of the $1d$-$4S$ sliding model, which is shown to be *NP*-complete [14]. Recall that in $1d$-$4S$, all sites lie on a horizontal line and each label must be placed, so that the site lies on one of the boundary edges of the label. Simply observe that in the case where $k = nc_0$ (i.e. the total length of all p-segments is equal to zero) the problem is restricted to the $1d$-$4S$ problem. □

Theorem 4 implies that we can not expect to find an algorithm, which runs in polynomial time with respect to the number of sites, unless $P = NP$. So, we assume that the input consists exclusively of integers and propose a pseudo-polynomial time algorithm, which runs in polynomial time to both the number of sites n and $W = 2\sum_{i=1}^{n} w_i + x_n - x_1$, where x_i is the x-coordinate of site s_i.

Theorem 5 *Given a set P of n sites on a horizontal line L, each associated with a rectangular $w_i \times h_i$ label l_i, there is an $O(nW^2)$ time algorithm that places all labels on both sides of L and attaches each point to its label with non-intersecting type-opo leaders, such that the total leader length is minimum, where $W = 2\sum_{i=1}^{n} w_i + x_n - x_1$.*

[2] Sites positioned on a vertical line are treated similarly.

Proof: We can observe that in an optimal solution, the lower left (right) corner of the leftmost (rightmost) label can not commence prior to $W_1 = x_1 - \sum_{i=1}^{n} w_i$ (exceed $W_2 = x_n + \sum_{i=1}^{n} w_i$). So, our problem can be easily formulated as a boundary labeling problem of minimum total leader length with non uniform sliding labels [7]. The total time needed to compute a legal labeling of minimum total leader length is equal to $O(nW^2)$. □

3.2.2 Leader bend minimization:

Following similar arguments as in proof of Theorem 4, we will show that the decision problem "Is there a labeling with total number of bends no more than k?" is *NP*-complete and hence the corresponding optimization problem is at least as hard.

Theorem 6 *Given $k \in \mathbb{Z}^+$ and a set S of n sites on a horizontal line L, each associated with a rectangular label l_i of dimensions $w_i \times h_i$, it is NP-complete to determine whether there exists a legal opo-labeling with labels on both sides of L, such that the total number of bends is at most k.*

Proof: By restriction. Simply observe that in the case where $k = 0$ (i.e. all leaders are of type-o) the problem is restricted to the $1d$-$4S$ problem. □

4 Sites on a Sloping Line

In this section, we extend the results of Section 3.1 to the case, where the input line has a positive slope ϕ (i.e. $0 < \phi < 90$). We assume that each label can be placed anywhere on the boundary of L, so that its bottom right corner coincides with L^T (recall that L^T is a translation of L by c_0 towards to $(0,\infty)$; see Figure 2). We further assume that each leader can touch its label only at the bottom right corner of the label (i.e. the point which slides along L_T). We want to obtain legal *opo*-labelings of either minimum number of bends or of minimum total leader length. We first consider the case of unit height labels and later on we will describe how to extend our approach to support non-uniform labels in general.

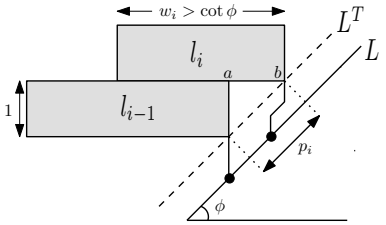
4.1 Labels of unit height

4.1.1 Total leader length minimization:

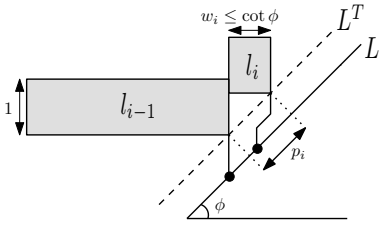
We describe how to compute in $O(n \log n)$ time a labeling with leaders of type-*opo*, so that the total leader length is minimum. Our approach is quite similar to the one presented for the case of a horizontal line in Section 3.1. In this case, we will reduce our problem to the single-machine scheduling problem with distinct due dates and symmetric earliness and tardiness penalties. Note that since we assume fixed ports the time windows have distinct due dates. The reduction we propose can be achieved in linear time. For each site $s_i = (x_i, y_i)$, we introduce a job J_i. The due date d_i of job J_i is x_i. The processing time p_i of job J_i is equal to the minimum Euclidean distance between the bottom right corner v_{i-1} of label l_{i-1} and the bottom right corner v_i of label l_i, when the y coordinate of v_{i-1} is less than the y coordinate of v_i and labels l_{i-1} and l_i do not overlap (see Figures 7a and 7b). We will refer to corners v_{i-1} and v_i as *sliding corners* of labels l_{i-1} and l_i, respectively. Since we assumed that all labels are of unit height, the computation of the minimum Euclidean distance between the sliding corners of labels l_{i-1} and l_i demands only a geometric analysis of the possible positions of labels l_{i-1} and l_i. It is easy to see that $\cot\phi$ is equal to the corresponding horizontal distance between the sliding corners of labels l_{i-1} and l_i (in Figure 7a, $\cot\phi$ is the length of the line segment ab). We distinguish two cases based on whether the width w_i of label l_i is greater than $\cot\phi$ or not.

Case 1 ($w_i > \cot\phi$)**:** Refer to Figure 7a. In this case, the minimum Euclidean distance between the sliding corners of labels l_{i-1} and l_i can be computed by placing label l_i on top of label l_{i-1} and is equal to $\frac{h_{i-1}}{\sin\phi}$ or equivalently equal to $\frac{1}{\sin\phi}$, since we assumed that all labels are of unit height.

Case 2 ($w_i \leq \cot\phi$)**:** Refer to Figure 7b. In this case, the minimum Euclidean distance between the sliding corners of labels l_{i-1} and l_i can be computed by placing label l_i next to label l_{i-1} and is equal to $\frac{w_i}{\cos\phi}$.



(a) $w_i > \cot\phi$



(b) $w_i \leq \cot\phi$

Figure 7: Processing time p_i of job J_i.

Based on the above cases, the processing time p_i of Job J_i is computed by using the following relation:

$$p_i = \begin{cases} \frac{1}{\sin\phi}, & \text{if } w_i \leq \cot\phi \\ \frac{w_i}{\cos\phi}, & \text{if } w_i > \cot\phi \end{cases}$$

We proceed by applying Garey et. al [13] algorithm to obtain a schedule σ_{opt}, which minimizes the total earliness-tardiness penalty. The exact positions of labels $l_1, l_2, \ldots, l_n$ are then determined based on the starting times $\sigma_1, \sigma_2, \ldots, \sigma_n$ of jobs $J_1, J_2, \ldots, J_n$, respectively, under schedule σ_{opt}. More precisely, the x-coordinate of the sliding corner v_i of label l_i is σ_i, and since, the y-coordinate of v_i is implied by the slope of L^T which is given, the exact positions of all labels are well-specified.

Next we show that the total earliness-tardiness penalty of the scheduling problem is equal to the total leader length of our labeling problem. Observe that if a job J_i is on time (i.e. it does not incur a penalty), the corresponding leader c_i, which connects label l_i with site s_i, is of type-o, which implies that leader c_i does not contribute to the total leader length. On the other hand, if job J_i is either early or tardy, then leader c_i contributes to the total leader length a penalty equal to the corresponding deviation. The processing times p_i of jobs J_i, $i = 1, 2, \ldots, n$ ensure that in an optimal solution no two labels overlap and hence the implied labeling is legal. The above result is summarized in Theorem 7.

Theorem 7 *Given a set S of n sites on a sloping line L, each associated with a rectangular $w_i \times 1$ label l_i that can be placed above L, we can compute in $O(n \log n)$ time a legal opo-labeling of minimum total leader length.*

4.1.2 Leader bend minimization:

We use the same reduction to obtain a labeling of minimum number of bends. In this case, we proceed by applying the algorithm of Lann and Mosheiov [22] to obtain a schedule of minimum number of penalized jobs. Observe that if a job J_i is on time, the corresponding leader c_i, which connects label l_i with site s_i, is of type-o, which implies that leader c_i does not contribute to the total number of bends. On the other hand, if job J_i is either early or tardy, then leader c_i contributes two bends to the total number of bends. So, the total number of leader bends is equal to twice the total number of penalized jobs of the implied scheduling problem. Moreover, the processing times p_i of jobs J_i, $i = 1, 2, \ldots, n$ ensure that in an optimal solution no two labels overlap and hence the implied labeling is legal. The above result is summarized in Theorem 8.

Theorem 8 *Given a set S of n sites on a sloping line L, each associated with a rectangular $w_i \times 1$ label l_i that can be placed above L, we can compute in $O(n^2)$ time a legal opo-labeling of minimum number of bends.*

4.2 Non-uniform labels

As already mentioned both algorithms can be extended to support non-uniform labels. In this case, a label of large height can effect the placement of a label later on the order (see Figure 8). Thus, the processing time p_i of job J_i can not be computed based only on the previous label l_{i-1}.

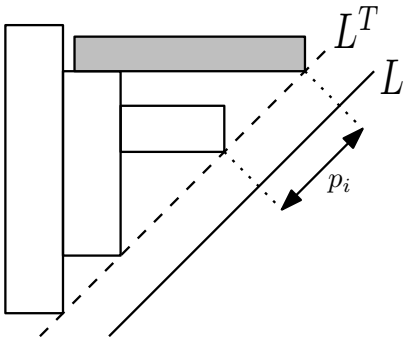


Figure 8: A label of large height effects the placement of a label later on the order

A straightforward computation of the processing time p_i corresponding to label l_i can be done in $O(i)$ time by considering all previous labels $l_1, l_2, \ldots, l_{i-1}$. Thus, the computation of all processing times requires $O(n^2)$ time and therefore the complexity of algorithm of Theorem 7 becomes $O(n^2)$, instead of $O(n \log n)$. The following Theorems summarize our results.

Theorem 9 *Given a set S of n sites on a sloping line L, each associated with a rectangular $w_i \times h_i$ label l_i that can be placed above L, we can compute in $O(n^2)$ time a legal opo-labeling of minimum total leader length.*

Theorem 10 *Given a set S of n sites on a sloping line L, each associated with a rectangular $w_i \times h_i$ label l_i that can be placed above L, we can compute in $O(n^2)$ time a legal opo-labeling of minimum number of bends.*

5 Open Problems and Future Work

1. In this paper, we presented results only for leaders of type-*opo*. No results are known regarding straight line leaders.

2. Another line of research is to design good approximation algorithms that solve the problems, that are proved to be *NP*-hard.

3. It is also intuitive that the quality of the labelings can be improved by allowing the labels to be placed anywhere on the boundary of L, without restricting them to slide along the lines L^T and L^B. No algorithms exist for this model.

References

[1] P. K. Agarwal, M. van Kreveld, and S. Suri. Label placement by maximum independent set in rectangles. In *Proc. 9th Canad. Conf. Comput. Geom.*, pages 233–238, 1997.

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

[3] K. R. Baker and G. D. Scudder. Sequencing with earliness and tardiness penalties: a review. *Operations Research*, 38:22–36, 1989.

[4] M. A. Bekos, M. Kaufmann, K. Potika, and A. Symvonis. Boundary labelling of optimal total leader length. In P. Bozanis and E. Houstis, editors, *10th Panhellenic Conference on Informatics (PCI'05)*, pages 80–89, 2005.

[5] M. A. Bekos, M. Kaufmann, K. Potika, and A. Symvonis. Polygons labelling of minimum leader length. In M. Kazuo, S. Kozo, and T. Jiro, editors, *Asia Pacific Symposium on Information Visualisation (APVIS2006), CRPIT 60*, pages 15–21, 2006.

[6] M. A. Bekos, M. Kaufmann, A. Symvonis, and A. Wolff. Boundary labeling: Models and efficient algorithms for rectangular maps. *Computational Geometry: Theory and Applications*. To appear.

[7] M. A. Bekos, M. Kaufmann, A. Symvonis, and A. Wolff. Boundary labeling: Models and efficient algorithms for rectangular maps. In J. Pach, editor, *Proc. 12th Int. Symposium on Graph Drawing (GD'04), LNCS 3383*, pages 49–59, New York, 2005.

[8] B. Chazelle et al. Application challenges to computational geometry: CG impact task force report. Technical Report TR-521-96, Princeton Univ., Apr. 1996.

[9] Y.-S. Chen, D. T. Lee, and C.-S. Liao. Labeling points on a single line. *International Journal of Computational Geometry and Applications*, 15(3):261–277, June 2005.

[10] J.-D. Fekete and C. Plaisant. Excentric labeling: Dynamic neighborhood labeling for data visualization. Technical Report CS-TR-3946, UMIACS-TR-98-59, Department of Computer Science, University of Maryland, 1998.

[11] 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.

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

[13] M. Garey, R. Tarjan, and G. Wilfong. One-processor scheduling with symmetric earliness and tardiness penalties. *Mathematics of Operations Research*, 13:330–348, 1988.

[14] M. Á. Garrido, C. Iturriaga, A. Márquez, J. R. Portillo, P. Reyes, and A. Wolff. Labeling subway lines. In P. Eades and T. Takaoka, editors, *Proc. 12th Annual International Symposium on Algorithms and Computation (ISAAC'01)*, volume 2223, pages 649–659, 2001.

[15] V. Gordon, J.-M. Proth, and C. Chu. A survey of the state-of-the-art of common due date assignment and scheduling research. *European Journal of Operational Research*, 139(1):1–25, 2002.

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

[17] H. Hoogeveen. Multicriteria scheduling. *European Journal of Operational Research*, 167(3):592–623, 2005.

[18] E. Imhof. Positioning names on maps. *The American Cartographer*, 2(2):128–144, 1975.

[19] C. Iturriaga and A. Lubiw. NP-hardness of some map labeling problems. Technical Report CS-97-18, University of Waterloo, Canada, 1997.

[20] T. Kato and H. Imai. The NP-completeness of the character placement problem of 2 or 3 degrees of freedom. In *Record of Joint Conference of Electrical and Electronic Engineers in Kyushu*, page 1138, 1988. In Japanese.

[21] C. Koulamas. Single-machine scheduling with time windows and earliness/tardiness penalties. *European Journal of Operational Research*, 91(1):190–202, 1996.

[22] A. Lann and G. Mosheiov. Single machine scheduling to minimize the number of early/tardy jobs. *Computers and Operations Research*, 23:769–781, 1996.

[23] S. Müller and A. Schödl. A smart algorithm for column chart labeling. In *Smart Graphics: 5th International Symposium (SG 2005), LNCS 3638*, pages 127–137, 2005.

[24] S. H. Poon, C.-S. Shin, T. Strijk, T. Uno, and A. Wolff. Labeling points with weights. *Algorithmica*, 38(2):341–362, 2003.

[25] M. van Kreveld, T. Strijk, and A. Wolff. Point labeling with sliding labels. *Conputational Geomentry: Theory and applications*, 13:21–47, 1999.

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

[27] P. Yoeli. The logic of automated map lettering. *The Cartographic Journal*, 9:99–108, 1972.

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

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