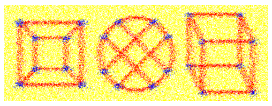

Journal of Graph Algorithms and Applications
http://jgaa.info/ vol. 12, no. 3, pp. 357–380 (2008)

Efficient Labeling of Collinear Sites

Michael A. Bekos[1] *Michael Kaufmann*[2] *Antonios Symvonis*[1]

[1]National Technical University of Athens
School of Applied Mathematics & Physical Sciences, Greece
[2]University of Tübingen, Institute for Informatics, Germany

Abstract

In this paper we study the map labeling problem where the sites to be labeled are restricted to a line L. Previous models studied in the map labeling literature fail to produce label placements (i.e. place each label *next to* the site it describes) without label overlaps for certain instances of the problem with dense point sets. To address this problem, we propose a new approach according to which, given n sites each associated with an axis-parallel rectangular label, we aim to place the labels in distinct positions on the "*boundary*" of L 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 evaluate our labeling model under two minimization criteria: (i) total leader length and (ii) total number of leader bends. We show that both problems are NP-complete if the labels can be placed on both sides of L, while we present polynomial time algorithms for the case where the labels can be placed on only one side of L.

Submitted:	Reviewed:	Revised:	Reviewed:	Revised:
May 2007	August 2007	January 2008	March 2008	July 2008
	Accepted:	Final:	Published:	
	August 2008	September 2008	October 2008	
	Article type:	Communicated by:		
	Regular Paper	S.-H. Hong		

An extended abstract of this paper has appeared in [6]. This work is partially funded by project PENED-2003. PENED-2003 is co-funded by the European Social Fund (75%) and Greek National Resources (25%). The work of M. Kaufmann is also supported by the German Research Foundation DFG Ka 812/8-3.

E-mail addresses: mikebekos@math.ntua.gr (Michael A. Bekos) mk@informatik.uni-tuebingen.de (Michael Kaufmann) symvonis@math.ntua.gr (Antonios Symvonis)

1 Introduction

Due to its extensive use in information visualization, automated map labeling has been identified as an important research area by the ACM Computational Geometry Impact Task Force report [8]. Map labeling has received considerable attention due to the large number of applications that stem from diverse areas such as Cartography and Geographical Information Systems.

Current research on this topic has been primarily focussed on labeling point-features. According to the cartographic literature [18, 27], there exist several parameters that determine the readability and unambiguity of a map. Among them they suggest that the labels should be pairwise disjoint and close to the point (also referred to as *site* or *anchor*) to which they belong. 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]. Strijk and Wolff [26] present an extensive online bibliography on label placement.

There exist 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 a case which arises when drawing schematized maps for road or subway networks. We assume that all sites lie on the same line and are to be labeled with axis-parallel rectangular pairwise-disjoint labels. Most of the known labeling algorithms for this problem produce quite legible labelings, when the input sites are sparsely distributed on the input line. However, if the site set contains a dense 5-tuple of sites they fail to produce non-overlapping labelings. To see this, w.l.o.g. assume that the points lie on a horizontal line and define a point set to be *dense* if the distance between the leftmost and the rightmost points is smaller than the width of the labels. Then, is is easy to label a dense set of 4 points (two labels are placed above the line and two labels are place below it), however, it is not possible to label a dense set of 5 points with non-overlapping labels (the label corresponding to the 5^{th} point will overlap one of the previously placed labels). To address the inability to label dense point sets, we propose a more flexible labeling model, according to which pairwise-disjoint 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*. Such labelings are referred to as *legal* or *crossing-free labelings* (for brevity, they are simply referred to as *labelings*).

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 $w_i \times h_i$. Without loss of generality, we assume that L is not parallel to the y-axis. The

"*boundary of L*" is defined by two lines L^T and L^B (one on top and one below L), that are translations of L by c_0 towards $(0, \infty)$ and $(0, -\infty)$, respectively, where c_0 is a predefined, positive constant (see Figure 2a). Labels have to be placed on the boundary of L, so that they do not overlap and do not obscure the site set. They also have to be connected to their associated sites by using leaders.

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

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

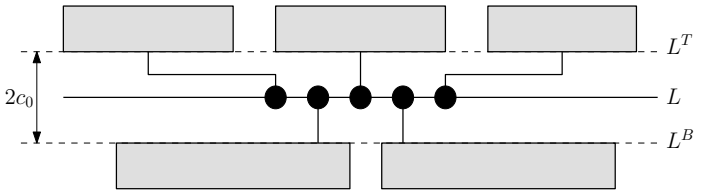


Figure 1: Horizontal input line.

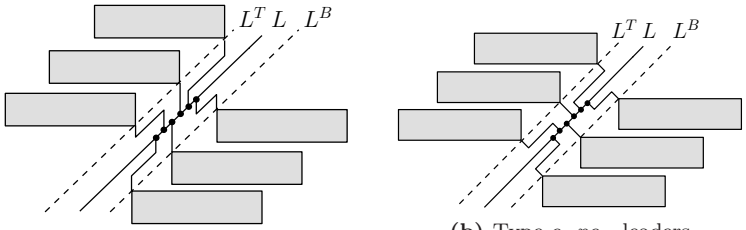
In general, the labels are of arbitrary sizes, i.e. label l_i associated with site s_i has width w_i and height h_i (*non-uniform labels*). However, in real applications labels usually contain text of the same font size. So, it is reasonable to separately consider the case, where the labels are of the 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 lies on 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.

The leaders which connect the sites to their corresponding labels can also be of several types. In our approach, we focus on two different types of leaders, which result in simple and easy-to-visualize labelings:

Type-$o_{\mathrm{x}}po_{\mathrm{x}}$ leaders: Leaders of type $o_{\mathrm{x}}po_{\mathrm{x}}$ (for simplycity, in the rest of the paper they are referred to as *opo-leaders*) consist of three line segments. The first and third line segments are orthogonal (o) to the X-axis, whereas the second one is parallel (p) to the input line (see Figure 2a). A degenerate 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 of the p-segment is zero).

Type-$o_{L}po_{L}$ leaders: Following the same notation scheme, leaders of type $o_{L}po_{L}$ consist of three line segments, where the first and third line segments are orthogonal to the input line L, whereas the second one is parallel to L (see Figure 2b). Again, a degenerate case of a type-$o_{L}po_{L}$ leader is a leader of

[1] The cases of a vertical input line and of a negative slope line are handled symmetrically.



(a) Type-*opo* leaders **(b)** Type-$o_L po_L$ leaders

Figure 2: Sloping input line

type o_L, which consists of only one line segment orthogonal to the input line.

Note that in the case of a horizontal input line, terms "$o_{\mathrm{x}}po_{\mathrm{x}}$ leader" and "$o_L po_L$ leader" both refer to the same type of leader (see Figure 1). Additionally, for each 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. This means that we ignore resolution issues and we allow the distance between adjacent parallel (p)-segments (belonging to different leaders) to be as small as required, that is, in the worst case smaller than c_0/n .

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 Figures 2a and 2b) 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:

1. **Minimize the total number of bends:** Find a labeling, such that the total number of bends is minimum. This is equivalent to maximizing the number of type-o leaders.

2. **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.

The paper is structured as follows: Section 2 reviews basic tools on scheduling 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

Input line	Labeling	Tllm		Tlbm	
		FixedP.	SlidP.	FixedP.	SlidP.
horizontal	1-side	open	$O(n \log n)$	open	$O(n^2)$
sloping	1-side	$O(n \log n)$	open	$O(n^2)$	open
horizontal	2-sides	open	$O(nW^2)^*$	open	NP-complete
sloping	2-sides	open	open	open	open

Table 1: Running times of our algorithms (in big-Oh-Notation) for various models of the problem. The polynomial time bounds refer to the case of non-uniform labels. Thus, they can be also applied for the case of uniform labels, too. The NP-completeness results refer to the case of non-uniform labels only. The problem marked by * is NP-complete. The pseudo-polynomial solution assumes that the input consists exclusively of integers. $W = 2\sum_{i=1}^{n} w_i + x_n - x_1$, where w_i is the width of label l_i, x_1 is the x-coordinate of leftmost site and x_n is the x-coordinate of rightmost site. Tllm, Tlbm, FixedP and SlidP stand for "Total Leader Length Minimization", "Total Leader Bend Minimization", "Fixed Ports" and "Sliding Ports", respectively.

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 also propose a new algorithm for the case of *opo*-labelings of minimum total leader length with non-uniform labels that have to be placed on one side of the input line which reduces the time complexity of [6] from $O(n^2)$ to $O(n \log n)$. We conclude in Section 5 with open problems and future work. Table 1 gives an overview over our results.

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[2] (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[3] (Figure 4 illustrates the $4S$ model, where the label can be shifted continuously as indicated by the arrows). 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 problem of determining a legal label placement under the $1d$-$4S$ model 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 fixed position, all have efficient algorithms. Chen et

[2]The model is called $4P$ due to the fact that each site has 4 candidate label positions.

[3]The model is called $4S$ due to the fact that each site must be labeled such that any edge of the label contains the site, i.e. the site is "sliding" along a label's edge.

al. showed that the problem of determining a legal labeling under the SLOPE-$4P$ model with rectangular labels of fixed height can be solved in linear time, if 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 sites on a horizontal line with top or bottom edges coinciding with the input line under the $4S$ model can be solved in $O(n^2 \log n)$ time.

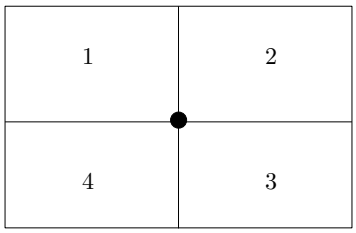


Figure 3: Illustration of $4P$ model.

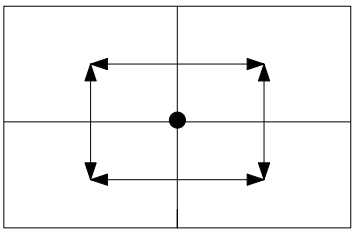


Figure 4: Illustration of $4S$ model.

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]. 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 they 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 along a side of the enclosing rectangle R and they do not extend beyond the rectangle corners defining that side. In this paper, we tackle both restrictions, since we do not assume the existence of the enclosing rectangle.

2 Preliminaries on Scheduling

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 with due windows and symmetric earliness-tardiness penalties*[4], 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 incurs 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

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

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 and tardiness penalty is minimized, is shown to be NP-complete, since it can be viewed as a generalization of the single-machine earliness-tardiness problem with distinct due dates $d_1, d_2 \ldots d_n$, which is a well-known NP-complete problem [13]. In the case of distinct due dates, a job J_i is considered to be on time if its completion time c_i is equal to d_i. In the case, where c_i is greater than d_i, J_i is considered to be tardy. Finally, a job J_i is early if its completion time c_i is smaller that d_i. In other words, this problem is equivalent to the one where each job J_i is associated with a time window of the form $[d_i - p_i, d_i]$, i.e. the length of each time window is equal to the processing time of its associated job. If the jobs are to be scheduled in a fixed predefined order, Garey et al. [13] have proposed an efficient algorithm, which determines an optimal schedule –by inserting idle time between jobs– in $O(n \log n)$ time for the cases of distinct due dates. Koulamas [21] has extended this result for the case of time windows.

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]. For the special case, in which 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 for the special case of distinct due dates $d_1, d_2 \ldots d_n$ (instead of time windows). In the following theorem, we generalize the algorithm of Lann and Mosheiov [22] to support time windows of any length.

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 $T[i - 1, k]$.

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

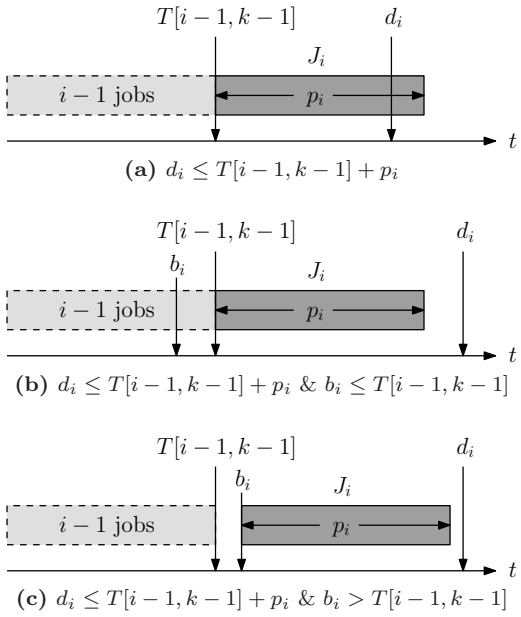


Figure 5: Different schedules obtained for the i-th job J_i.

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 $T[i,k]=\infty$. Both subcases can be described by the equation:

$$T[i,k] = T[i-1,k] + p_i \tag{1}$$

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 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 MinPenalizedJobs 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 MinPenalizedJobs 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 MinPenalizedJobs needs $O(n^2)$ time and space, since it maintains a $(n+1)\times(n+1)$ table and each entry of this table is computed in constant time. 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.

Algorithm 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] = ∞
    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] < ∞ then
      return j
      break
```

□

3 Sites on a Horizontal Line

In this section, we consider the case, where the sites to be labeled are restricted to a horizontal line[5] L. W.l.o.g. we assume that L is the X-axis, i.e. $L : y = 0$.

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

We want to obtain legal type-*opo*[6] 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 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 to 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 simply has 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).

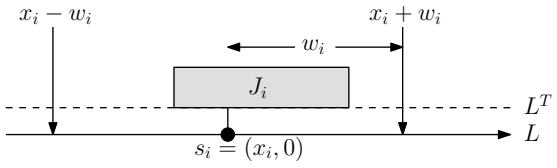


Figure 6: For each site s_i, a job J_i of processing times w_i is introduced.

[6]Note that in the case of a horizontal line, a leader of type-*opo* is identical to a leader of type-$o_L po_L$.

We proceed by applying Koulamas' algorithm [21] 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 from 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 has to 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 algorithm MinPenalizedJobs 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 has to 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 if 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-complete.

3.2.1 Total leader length minimization

We 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 this problem, establishing that the problem is NP-complete 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: Membership in NP follows from the fact that a nondeterministic algorithm needs only guess for each label l_i, $i = 1, 2, \ldots, n$ whether it lies above or below the horizontal line, defining in this way two independent one-sided subproblems, which can be solved in polynomial time by applying the algorithm of Section 3.1.1 twice, i.e., once for the labels that lie above the input line and once for the corresponding labels that lie below it. Then, we can trivially check that the sum of the lengths of the p-segment of all leaders[7] is no more than k, by summing the total leader lengths of the solutions of the two subproblems.

We will reduce the problem of determining a legal labeling under the $1d$-$4S$ sliding model, which is known to be NP-complete [14], to our problem. 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. Let $I_{1d\text{-}4S}$ be an instance of the problem of determining a legal labeling under the $1d$-$4S$ sliding model. $I_{1d\text{-}4S}$ consists of n sites $s_1, s_2 \ldots s_n$ on a horizontal line L. Each site s_i is associated with a rectangular, axis-parallel, $(w_i \times h_i)$-label l_i. We use the same setting to construct an instance I_L of our problem.

The reduction we propose is "somewhat" by restriction. It is based on the fact that a legal labeling for $I_{1d\text{-}4S}$ implies a solution of I_L, where each site is attached to its label through a type-o leader and vice versa. More precisely, we can easily observe that there exists a legal labeling for $I_{1d\text{-}4S}$ if and only if there exists a solution of I_L with total leader length at most $k = 0$, i.e., the total length of all p-segments is equal to zero. □

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 in terms of 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 *Assume a set P of n sites on a horizontal line L, each associated with a rectangular $(w_i \times h_i)$-label l_i and let $W = 2\sum_{i=1}^{n} w_i + x_n - x_1$, where x_i is the x-coordinate of site s_i. Then, 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.*

[7] Recall that we consider only the lengths of the p-segments of all leaders when measuring the total leader length.

Proof: We observe that in an optimal solution, the lower left (right) corner of the leftmost (rightmost) label cannot have x-coordinate smaller than $W_1 = x_1 - \sum_{i=1}^{n} w_i$ (greater than $W_2 = x_n + \sum_{i=1}^{n} w_i$). So, our problem can be easily formulated as a boundary labeling problem of minimum total leader length with non uniform sliding labels. Bekos et al. [7] showed how to compute a legal boundary labeling of minimum total leader length in $O(nW^2)$ time. □

3.2.2 Leader bend minimization

In the following theorem, we show that the decision problem "Is there a labeling with the 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: Membership in NP follows from the fact that a nondeterministic algorithm needs only guess for each label l_i, $i = 1, 2, \ldots, n$ whether it lies above or below the horizontal line, defining in this way two independent one-sided subproblems, which can be solved in polynomial time by applying the algorithm of Section 3.1.2 twice, i.e., once for the labels that lie above the input line and once for the corresponding labels that lie below it. Then, we can trivially check that the total number of leader bends is no more than k, by summing the total number of leader bends of the solutions of the two subproblems.

As in the proof of Theorem 4, we will reduce the problem of determining a legal labeling under the $1d$-$4S$ sliding model to our problem. The reduction is again by restriction. Let $I_{1d\text{-}4S}$ be an instance of the problem of determining a legal labeling under the $1d$-$4S$ sliding model, consisting of n sites $s_1, s_2 \ldots s_n$ on a horizontal line L, each associated with a rectangular, axis-parallel label. We use the same setting to construct an instance I_L of our problem. Observe now that there exists a legal labeling for $I_{1d\text{-}4S}$ if and only if there exists a solution of I_L with total number of leader bends at most $k = 0$, i.e., all sites are connected to their associated labels through leaders of type-o. □

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 lies on L^T (recall that L^T is a translation of L by c_0 towards to $(0, \infty)$; see Figure 2a). 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 labelings of either minimum number of bends or of minimum total leader length. We first consider the case of unit height labels with type-*opo* leaders

(Section 4.1) and later on we describe how to extend our approach to support non-uniform labels (Section 4.2) and leaders of type $o_{\scriptscriptstyle L}po_{\scriptscriptstyle L}$ (Section 4.3).

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 label ports, we use distinct due dates instead of time windows.

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 horizontal 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 assume that all labels are of unit height, the computation of the minimum horizontal 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 the case where label l_i is of unit width (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.

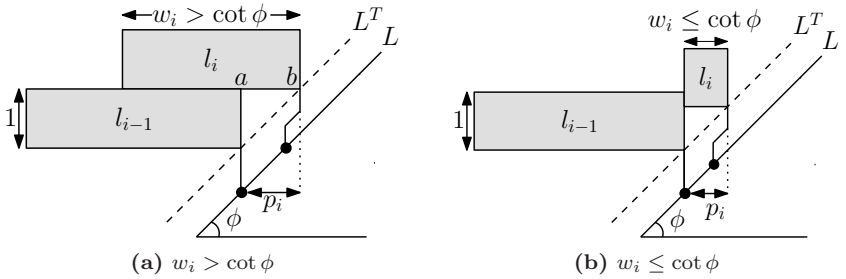
Case 1 ($w_i > \cot\phi$)**:** Refer to Figure 7a. In this case, the minimum horizontal distance between the sliding corners of labels l_{i-1} and l_i can be computed by placing label l_i on top of label l_{i-1} and is equal to ab or equivalently equal to $\cot\phi$, since we assume that all labels are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

Case 2 ($w_i \leq \cot\phi$)**:** Refer to Figure 7b. In this case, the minimum horizontal 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 w_i.

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

$$p_i = \min\{w_i, \cot\phi\}$$

We proceed by applying the algorithm of Garey et al. [13] to obtain a schedule σ_{opt} that minimizes the total earliness-tardiness penalty. The exact positions of labels $l_1, l_2, \ldots, l_n$ are then determined based on the completion times $c_1, c_2, \ldots, c_n$ 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 c_i, and since the



(a) $w_i > \cot\phi$ **(b)** $w_i \leq \cot\phi$

Figure 7: Processing time p_i of job J_i.

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 leader length of our labeling problem is equal to $1/\cos\phi$ times the total earliness-tardiness penalty of the scheduling 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 $1/\cos\phi$ times 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 has to 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 to 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

The algorithms of unit height labels of Section 4.1 can be extended to support non-uniform labels. In the case of non-uniform labels, a label of large height can effect the placement of a label later in the order (see Figure 8). Thus, the processing time p_i of job J_i cannot be computed based only on the previous label l_{i-1}.

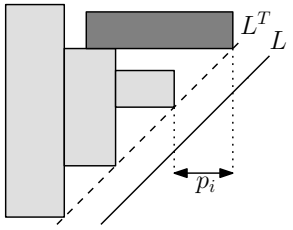


Figure 8: A label of large height affects the placement of a label later in the order

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 would become $O(n^2)$, instead of $O(n \log n)$.

However, it is possible to compute the processing times of all jobs in $O(n \log n)$ time by employing a binary search like technique. To achieve this, we have to introduce the notions of the *shadow* of a label l and the *frontier* of k already placed labels (see Figures 9 and 10, respectively).

Definition 1 *The* shadow $s(l)$ *of a label* l *is the quadrant of the plane that contains* l *and is defined by the top right corner of* l *and the two adjacent sides of* l*.*

Definition 2 *The* frontier $F(k)$ *of* k *already placed labels* $l_1, l_2, \ldots, l_k$ *is the union of their shadows, i.e.* $F(k) = \cup_{i=1}^{k} s(l_i)$*.*

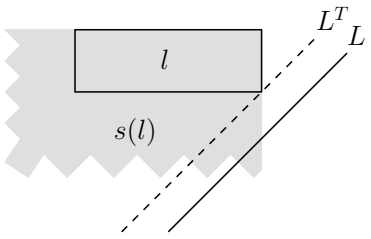


Figure 9: Shadow of label l.

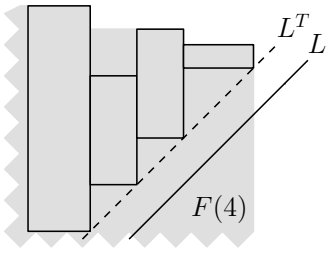


Figure 10: The frontier $F(4)$ of 4 labels.

Assume that we have placed the labels $l_1, l_2, \ldots, l_k$, which correspond to sites $s_1, s_2, \ldots, s_k$, respectively, so that the horizontal distance between the sliding

corners of labels l_{j-1} and l_j is minimum, for each pair of labels l_{j-1} and l_j, $j \in \{2, 3, \ldots k\}$.

The placement of label l_{k+1} has to satisfy the following: (i) the bottom right corner of l_{k+1} is on L^T and to the right of all previously placed labels, (ii) l_{k+1} does not overlap with previously placed labels and (iii) the x-coordinate of the bottom right corner of l_{k+1} is as small as possible, i.e. l_{k+1} touches some previously placed label but does not overlap with it.

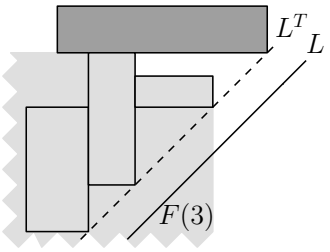


Figure 11: Step B.i of LABEL PLACEMENT procedure: the case where v_f does not exist.

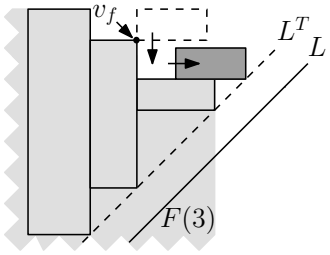


Figure 12: Step B.ii of LABEL PLACEMENT procedure: the case where v_f does exist.

Our approach consists of two steps and is described in Procedure LABEL PLACEMENT. Figures 11 and 12 illustrate Steps B.i and B.ii, respectively, of the LABEL PLACEMENT procedure.

4.2.1 The frontier data structure

We use a data structure which maintains the *frontier of a set of labels* and supports the following operations:

a. ***add(label)*:** The new label is placed so that:

   i. Its bottom right corner is on L^T.

   ii. Its bottom right corner is to the right of all previously placed labels.

   iii. It does not overlap with previously placed labels.

   iv. The x-coordinate of its bottom right corner is as small as possible, i.e. the label touches some previously placed label but does not overlap with it.

b. ***position(label)*:** Returns the x-coordinate of the lower left corner of the label (the y-coordinate is implied by the slope of L^T which is given).

Based on the discussion in the preceding section, we can implement the *add(label)* operation in logarithmic time with respect to the number of labels that *"contribute"* to the frontier, i.e., their top-right corners are on the boundary of the frontier. We say that these labels are "*on the frontier*".

Procedure `Label Placement`(l_{k+1})

Input : A positive sloped input line L, the Frontier $F(k)$ of k already placed labels $l_1, l_2 \dots l_k$ and a label l_{k+1} to be properly placed.

{**Step A.**}
Determine a corner v_f of $F(k)$ (if any), which: (i) is convex, (ii) has the greatest possible x-coordinate and (iii) if we place the bottom-left corner of l_{k+1} on it, then label l_{k+1} does not intersect L_T. Note that since we insist only on convex corners, v_f will be the top-right corner of some label.

{**Step B.** *Based on the existence of corner v_f, we proceed as follows*}
if corner v_f does not exist then
{Case i: Refer to Figure 11} We place l_{k+1}, so that (i) its sliding corner touches L_T and (ii) its bottom boundary edge coincides with the topmost horizontal edge of $F(k)$.
else
{Case ii: Refer to Figure 12} We initially place the bottom-left corner of l_{k+1} on that vertex and we start sliding it vertically until either its sliding corner "hits" L_T or its bottom boundary edge touches the horizontal boundary edge of $F(k)$ immediately below and to the right of v_f. In the latter case, we may also have to slide the label horizontally until the sliding corner of l_{k+1} "hits" L_T, resulting into the desired placement.

More precisely, the frontier data structure is implemented by employing a sorted array, where we store the labels on the frontier in increasing order of the x-coordinates of their top-right corners. Note that since we only store the labels on the frontier, this also implies that the labels are ordered in decreasing order of the y-coordinates of their top-right corners. By applying a binary search on this array, Step A of Procedure Label Placement can be implemented in $O(\log n)$ time. Having determined corner v_f, the position of the label can be determined in constant time following a similar geometric analysis as the one in Section 4.1.

However, the *add(label)* operation requires some additional effort, so that the data structure represents the new frontier (i.e. the one obtained after the addition of the new label). This is done as follows: Initially, the new label –since it is on the frontier– has to be placed in the proper position in the array representing the frontier.

If corner v_f does not exist (case B.i of Procedure Label Placement), the array should only contain the new label. This case is illustrated in Figure 11.

If corner v_f does exist (case B.ii of Procedure Label Placement), there exists two alternatives:

- If the new label is placed so that its top-right corner is below v_f, the

new label has to be placed after the label containing v_f. Moreover, all labels to the right of the new label have to be removed from the array representing the frontier, since they are not on the frontier any more. This case is illustrated in Figure 13.

- In the case where the top-right corner of the new label is placed above v_f, the new label has to replace the label containing v_f. This is because the label containing v_f is not on the frontier any more. As in the previous case, all labels to the right of the new label have to be removed from the array representing the frontier. However, this case requires some additional effort. More precisely, all labels to the left of the new label whose top-right corners lie below the top-right corner of the new label have to be removed from the array. This is not performed by searching all elements of the array. We only search the (immediate) left neighbors of the new label until we identify the rightmost one whose top-right corner lie above the top-right corner of the new label. This case is illustrated in Figure 14.

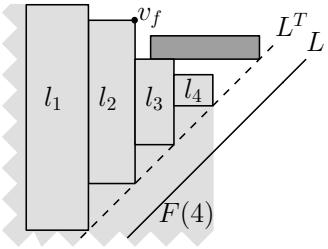


Figure 13: Labels l_3 and l_4 are not on the new frontier.

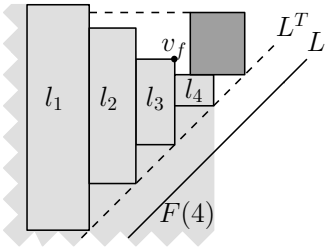
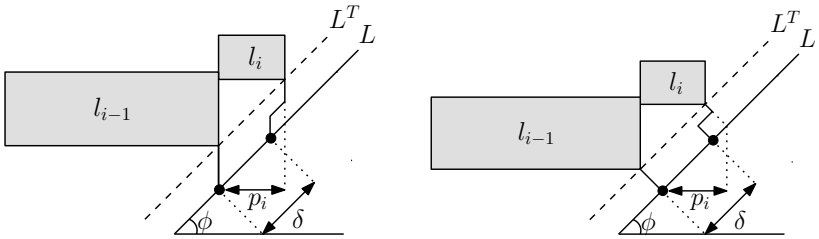


Figure 14: Labels l_2, l_3 and l_4 are not on the new frontier.

Since each label is removed at most once from the array representing the frontier, the total time needed to perform all updates concerning the data structure is $O(n)$. Therefore the total time needed to perform all *add(label)* operations is $O(n \log n)$. Note that having implemented the *add(label)* operation the operation *position(label)* implies an $O(1)$ additional time. 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 has to be placed above L, we can compute in $O(n \log n)$ 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 has to be placed above L, we can compute in $O(n^2)$ time a legal opo-labeling of minimum number of bends.*



(a) The sites are connected with their labels using leaders of type *opo*

(b) The sites are connected with their labels using leaders of type $o_{\scriptscriptstyle L}po_{\scriptscriptstyle L}$

Figure 15: The total earliness-tardiness penalty incurred is equal in both cases. Note that the sites are equally spaced in both cases, denoted by δ in the Figure.

4.3 The $o_{\scriptscriptstyle L}po_{\scriptscriptstyle L}$ Model

As mentioned earlier, our approach can be extended to support leaders of type $o_{\scriptscriptstyle L}po_{\scriptscriptstyle L}$. In the case of $o_{\scriptscriptstyle L}po_{\scriptscriptstyle L}$-leaders, the processing time p_i of job J_i can be calculated as in Section 4.1 assuming uniform labels, or, as in Section 4.2 assuming non-uniform labels. This ensure that in an optimal solution no two labels overlap and hence the implied labeling is legal. However, the due date d_i of job J_i is now the x-coordinate of the projection of the site s_i to the line L^T. This is interpreted as follows: A job J_i is considered to be on time, if its associated label l_i can be connected with site s_i through a leader of type $o_{\scriptscriptstyle L}$. On the other hand, if job J_i is either early or tardy, then leader c_i contributes i) two bends to the total number of leader bends and ii) a penalty equal to $1/\cos\phi$ times the corresponding deviation to the total leader length.

Note also that under this setting, the total earliness-tardiness penalty incurred in the case of $o_{\scriptscriptstyle L}po_{\scriptscriptstyle L}$-leaders is equal to the penalty incurred in the case where *opo*-leaders are used to connect the sites with their labels. This is illustrated in Figure 15. The following theorems summarize our results.

Theorem 11 *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 has to be placed above L, we can compute in $O(n \log n)$ time a legal $o_{\scriptscriptstyle L}po_{\scriptscriptstyle L}$-labeling of minimum total leader length.*

Theorem 12 *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 has to be placed above L, we can compute in $O(n^2)$ time a legal $o_{\scriptscriptstyle L}po_{\scriptscriptstyle L}$-labeling of minimum number of bends.*

5 Open Problems and Future Work

In this paper, we considered the problem of labeling collinear sites with "floating" labels on the boundary on the input line. We presented efficient algorithms to determine *opo*- and $o_{\scriptscriptstyle L}po_{\scriptscriptstyle L}$-labelings of either minimum total leader length or of minimum number of leader 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 showed that both problems are NP-complete.

Several problems arise from our research that need to be addressed. Among them, we distinguish the following:

- Straight line leaders (instead of *opo-* or $o_{_L}po_{_L}$) can be used to connect each site with its corresponding label. Moreover, a combination of different types of leaders could also be of particular interest.
- It would be interesting to derive labeling that combine the traditional labeling models $4P$ and $4S$ with our model, i.e., labelings that use leaders to connect labels with their corresponding sites only in the case where it is not possible to place the labels using the $4P$ and/or $4S$ models.
- It is intuitive that the quality of the labelings can be improved by allowing the labels to be placed "beyond" the boundary of L, without restricting them to slide along the lines L^T and L^B. No algorithms exist for this model.
- Several labeling problems were shown to be NP-complete if labels can be placed at both sides of the line. It is worth trying to derive approximation algorithms for these problems.
- To the best of our knowledge no algorithms exist –in the map labeling literature in general– regarding the case of labeling sites on a (rectilinear) polygonal line. So, another line of research is to try to evaluate our labeling model for this setting.

Acknowledgment. The authors would like to thank the anonymous referees for their comments, and particularly for correcting an error in the original algorithm for the total leader length minimization in the case of a sloping line and for the suggestion to include in our study the case of $o_{_L}po_{_L}$ leaders.

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. AUTONAP an 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 (APVIS'06), CRPIT 60*, pages 15–21, 2006.

[6] M. A. Bekos, M. Kaufmann, and A. Symvonis. Labeling collinear sites. In *6th Asia Pacific Symposium on Visualization 2007 (APVIS'07)*, pages 45–51, 2007.

[7] M. A. Bekos, M. Kaufmann, A. Symvonis, and A. Wolff. Boundary labeling: Models and efficient algorithms for rectangular maps. *Computational Geometry: Theory and Applications*, 36(3):215–236, 2007.

[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 of *LNCS*, pages 649–659. Springer-Verlag, 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 *Proc. 5th International Symposium on Smart Graphics (SG'05)*, volume 3638 of *LNCS*, pages 127–137. Springer-Verlag, 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.