Maximizing the Number of Independent Labels in the Plane*

Kuen-Lin Yu[1], Chung-Shou Liao[2,**], and D. T. Lee[2,**]

[1] Department of Computer Science and Information Engineering, National Chiao Tung University, HsinChu, Taiwan
[2] Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan
dtlee@ieee.org

Abstract. In this paper, we consider a map labeling problem to maximize the number of independent labels in the plane. We first investigate the point labeling model that each label can be placed on a given set of anchors on a horizontal line. It is known that most of the map labeling decision models on a single line (horizontal or slope line) can be easily solved. However, the label number maximization models are more difficult (like 2SAT vs. MAX-2SAT). We present an $O(n \log \Delta)$ time algorithm for the four position label model on a horizontal line based on dynamic programming and a particular analysis, where n is the number of the anchors and Δ is the maximum number of labels whose intersection is nonempty. As a contrast to Agarwal et al.'s result [Comput. Geom. Theory Appl. 11 (1998) 209-218] and Chan's result [Inform. Process. Letters 89(2004) 19-23] in which they provide $(1 + 1/k)$-factor PTAS algorithms that run in $O(n \log n + n^{2k-1})$ time and $O(n \log n + n\Delta^{k-1})$ time respectively for the fixed-height rectangle label placement model in the plane, we extend our method to improve their algorithms and present a $(1 + 1/k)$-factor PTAS algorithm that runs in $O(n \log n + kn \log^4 \Delta + \Delta^{k-1})$ time using $O(k\Delta^3 \log^4 \Delta + k\Delta^{k-1})$ storage.

1 Introduction

In cartographic literature, the main approach to conveying information concerning what is on the map is to attach texts or labels to geographic features on the map. Automated label placement subject to the constraint that the labels are pairwise disjoint is a well-known important problem in geographic information systems (GIS). In the ACM Computational Geometry Impact Task Force

* This work was supported in part by the National Science Council under the Grants NSC95-2221-E-001-016-MY3, NSC-94-2422-H-001-0001, and NSC-95-2752-E-002-005-PAE, and by the Taiwan Information Security Center (TWISC) under the Grants NSC NSC95-2218-E-001-001, NSC95-3114-P-001-002-Y, NSC94-3114-P-001-003-Y and NSC 94-3114-P-011-001.

** Also with the Institute of Information Science, Academia Sinica, Nankang, Taipei, Taiwan.

F.P. Preparata and Q. Fang (Eds.): FAW 2007, LNCS 4613, pp. 136–147, 2007.
© Springer-Verlag Berlin Heidelberg 2007

report[4] the map label placement is listed as an important research area. Since this problem in general is known to be NP-complete, many heuristics or special cases for which polynomial time algorithms are given have been presented. For instance, there are many algorithms that have been developed for labeling points that are on lines[5,9,11,12,19,22] or in a region[7,8,13,14,15,16,17,18,20,21].

Let $\mathcal{A}$ denote a set of points $\{A_1, A_2, \ldots, A_n\}$ in the plane, called *anchors*. Associated with each anchor there is an axis-parallel rectangle, called *label*. The *point-feature label placement problem* or simply *point labeling problem*, is to determine a placement of these labels such that the anchors coincide with one of the corners of their associated labels and no two labels overlap. The point labeling problem for labeling an arbitrary set of points has been shown to be NP-complete[8,14,15,18] and there were some heuristic algorithms[6,8,21].

There are many variations of the point labeling problem, including shapes of the labels, locations of the anchors to be labeled and where the labels are placed. Consider the case that the placement of the labels are restricted. For instance, one is *fixed-position model*, denoted 4P model, in which a label must be placed so that the anchor coincides with one of its four corners; and another is *slider model*, denoted 4S model, in which a label can be placed so that the anchor lies on one of the four boundary edges of the label. The coordinate positions {1, 2, 3, 4} in 4P model denote the corner positions of labels coincident with the anchor, and the arrows in 4S model indicate the directions along which the label can slide, maintaining contact with the anchor.

In this paper we consider the case when the anchors lie on a line and are to be labeled with rectangular labels. This problem has been studied previously [9,16,19,5]. The prefix 1d or Slope refers to the problem in which the anchors lie on a horizontal or a sloping line, respectively. Garrido et al.[9] gave linear time algorithms for 1d4P rectangle label, 1d4S square label, and Slope4P square label models, and a quadratic time algorithm for Slope4S square label model as well. They also showed 1d4S rectangle label is NP-complete and consider the maximization version to maximize the size of labels. Chen et al.[5] further provided linear time algorithms for the decision version of Slope4P fixed-height(or width) rectangle label and elastic rectangular label (of a given area) models. They also presented a lower bound $\Omega(n \log n)$ time and a different method to maximize the label size for 1d4S square label model. Maximizing the number of labels that can be placed or the so-called maximum independent set problem, is yet another common problem. Although the label size maximization model is as easy as the decision model, the label number maximization model has been considered to be harder. Most of cases where we can tackle the decision models in polynomial time are more or less comparable to 2SAT, and yet the label number maximization model is relatively more difficult (like MAX-2SAT). In 1998, Agarwal et al.[1] provided a $(1 + 1/k)$-factor algorithm that runs in $O(n \log n + n^{2k-1})$ time, for any integer $k \geq 1$, for fixed-height rectangle label placement model in the plane and an $O(\log n)$-factor approximation algorithm that runs in $O(n \log n)$ time for arbitrary rectangle labels. Poon et al.[19] further considered the weighted case in which each label is associated with a given weight and provided the

same approximation result for 4P fixed-height weighted rectangle model. They also gave a $(2+\epsilon)$-factor approximation algorithm that runs in $O(n^2/\epsilon)$ time for 1d4S weighted rectangle label. As for arbitrary rectangle label, Berman et al.[2] presented a $\lceil O(\log_k n)\rceil$-factor approximation algorithm that runs in $O(n^{k+1})$ time, for any integer $k \geq 2$. In 2004, Chan[3] improved the previous results an gave a $(1 + 1/k)$-factor algorithm that runs in $O(n \log n + n\Delta^{k-1})$ time, where $\Delta \leq n$ denotes the maximum number of rectangles whose intersection is nonempty, for fixed-height rectangle label model and a $\lceil O(\log_k n)\rceil$-factor approximation algorithm that runs in $n^{O(k/\log k)}$ time for arbitrary rectangle label.

We first investigate the maximization version of the feasible number of labels when the anchors lie on a horizontal line. That is, we want to maximize the number of labels whose associated anchors lie on a horizontal line for which a *feasible* placement exists that no two labels overlap. In other words, these labels form an independent set. We refer to this model as *Max-1d4P rectangle label model*, or *Max-1d4P* for short. Since most of the decision model of map labeling problems on a single line are easily solved in a greedy manner, we are looking for an almost linear time algorithm for the Max-1d4P model. As a contrast to previous related results[1,3] in which the maximum independent set of label placement problem in the plane was considered and polynomial time approximation schemes (PTAS) were provided using the *line stabbing* technique and the shifting idea, we present a faster approach based on a different form of dynamic programming strategy and a particular analysis to solving this Max-1d4P model in $O(n \log \Delta)$ time which improves previously known results that run in $O(n^2)$ and $O(n\Delta)$ time in the worse case. We also point out an implicit difference between point labeling problem and label placement problem, mentioned in the intuitive proof of the reduction[19]. In addition, we further extend our method to solve the fixed-height rectangle label placement model in the plane and present a $(1 + 1/k)$-factor polynomial time approximation scheme (PTAS) algorithm that runs in $O(n \log n + kn \log^4 \Delta + \Delta^{k-1})$ time, using $O(k\Delta^3 \log^4 \Delta + k\Delta^{k-1})$ storage.

This paper is organized as follows. In Section 2, we introduce some definitions. Then we present in Section 3 an $O(n \log \Delta)$ time algorithm for the Max-1d4P model. In Section 4 we specify an implicit difference of point labeling problem and label placement problem and give a $(1+1/k)$-factor PTAS algorithm for the fixed-height rectangle label placement model in the plane. Finally we conclude in Section 5 with some discussions of future work.

2 Preliminaries

Consider a set of anchors $\mathcal{A} = \{A_1, A_2, \ldots, A_n\}$ on a horizontal line, and each anchor A_k is associated with its position (in x-coordinate) x_k and label size l_k. The aim is to maximize the number of *feasible* labels so that they do not overlap with each other. A feasible solution to the point labeling problem is called a *realization*.

Since we consider the problem on a horizontal line and put the label either *above* or *below* the line, we can simply associate a 2-tuple, namely (a, b), to represent the current labeling state of a realization R, with $R.a = a$ and $R.b = b$, representing respectively the coordinates of the right edge of the rightmost label *above* the line and *below* the line. A realization R will also contain a specification of the label placement at feasible positions associated with a subset of anchors. To be more precise, we can use $A_i.\ell \in R$, where $A_i.\ell \in \{0, 1, 2, 3, 4\}$ indicates the label position for anchor A_i included in R, with $A_i.\ell = 0$ representing anchor A_i is not labeled. If R contains k-feasible labels, i.e., it contains k non-zero $A_i.\ell$'s, then R is called a k-realization, and we use $R.c$ to denote the cardinality of the subset of feasible labels. We shall use the notation R to not only represent a realization R of $\mathcal{A}$, which corresponds to a subset, $\Re$, of feasible anchors, i.e., $\Re \subseteq \mathcal{A}$, but also use $R.a$, $R.b$ and $R.c$ to represent the state of its configuration and its size, respectively. Let us assume that the set of anchors has been ordered so that their x-coordinates are in strictly increasing order. That is, $x_1 < x_2 < \ldots < x_n$. Let $\mathcal{A}_i$ denote the subset of anchors $\{A_1, A_2, \ldots, A_i\}$, for $i = 1, 2, \ldots, n$, and R^i denote a realization of $\mathcal{A}_i$ for some i. An optimal solution is a realization R^n such that $R^n.c$ is maximum among all possible realizations of $\mathcal{A}_n$.

We shall process the anchors, and their associated labels, in ascending order of their x-coordinates, i.e., in the order of $A_1, A_2, \ldots, A_n$. Given a realization R^{i-1} of $\mathcal{A}_{i-1}$, and the next anchor, A_i, $i > 1$, the placements of the label of A_i that do not overlap the last label both above and below the line in R^{i-1} are called *feasible label placements*. Before proceeding we define the notion of *equivalence* of two realizations:

Definition 1. *Given two realizations R_1^i and R_2^i of $\mathcal{A}_i$ such that $R_1^i.c$ and $R_2^i.c$ are equal, if $\{R_1^i.a, R_1^i.b\} = \{R_2^i.a, R_2^i.b\}$, we say that the two realizations are* **equivalent in size***, or simply* **equivalent** *to each other.*

Based on the above definition, for a realization R^i with $R^i.a < R^i.b$, we always swap the upper and lower sides of the realization. That is, a realization will be represented in a *normal form* in which the coordinate *above* the line is no less than the coordinate *below* the line, i.e., $R^i.a \geq R^i.b$ without loss of generality. Here we define the *comparability* of two realizations.

Definition 2. *For any two realizations R_i^k and R_j^k, $1 \leq k \leq n$, if the following statements hold,*

(1). $R_i^k.c = R_j^k.c$, and

(2). $R_i^k.a \leq R_j^k.a$, and $R_i^k.b \leq R_j^k.b$ then we say that the two realizations are **comparable** *and R_i^k is* **better** *than R_j^k. Otherwise, they are* **incomparable***.*

Lemma 1. *Let $\mathcal{R}$ be an optimal realization of $\mathcal{A}$. Suppose R_i^k and R_j^k are two comparable realizations for $\mathcal{A}_k$ and R_i^k is better than R_j^k for some $n \geq k \geq 1$. If $\mathcal{R}$ contains R_j^k as a subset, then there exists another optimal solution that contains R_i^k.*

By using a 2-tuple to represent the labeling state of a realization, we can transform it into a point in the two-dimensional plane. To be more precise, given a

realization R represented by a 2-tuple $(R.a, R.b)$, we transform it into a point $P(x, y)$, where $x = R.a$ and $y = R.b$, in the plane. The two equivalent realizations will then be transformed into two points that are symmetric with respect to the line $x = y$. From now on, we use $P.x$ and $P.y$ to represent the x and y coordinates of a point P in the plane, and $P.c$ to represent its associated cardinality. We assume the point labeling on a line starts at the origin without loss of generality, which means the transformed points in the plane are all in the first quadrant. Using the normal form representation of a realization, all realizations will be mapped to points that are all located in the first quadrant below the line $x = y$.

Based on the *comparability* definition between two realizations, if one is better than the other, then the transformed points will carry the relationship of *domination*. That is, if realization R_s is *better* than realization R_t, then point P_t *dominates*[1] point P_s in the plane. On the other hand, if they are *incomparable* realizations, the transformed points in the plane do not dominate each other.

We shall also transform each anchor A_k with its position x_k and its given label size (or length) l_k, $1 \leq k \leq n$, into a point of another kind $P^{A_k}(x_k, y_k)$ located on line $x = y$ in the plane, that is, $x_k = y_k$. We define the operations in the plane as follows.

Definition 3. *Given a point $P(x, y)$ representing a realization and a point $P^{A_k}(x_k, x_k)$ representing a new label $A_k.\ell$ of length l_k in the plane, $x_k - l_k \geq x$, we have the following operations depending on how we select the placement of label $A_k.\ell$ of the anchor A_k for the realization P.*

1. *$A_k.\ell = 1$, then $P(x, y)$ generates $P'(x_k + l_k, y)$.*
2. *$A_k.\ell = 2$, then $P(x, y)$ generates $P'(x_k, y)$.*
3. *$A_k.\ell = 3$, then $P(x, y)$ generates $P'(x, x_k)$.*
4. *$A_k.\ell = 4$, then $P(x, y)$ generates $P'(x, x_k + l_k)$.*

The cardinality associated with point P' will be one more than that with point P. If the y-coordinate of P' is greater than the x-coordinate, we do the swapping operation to exchange the x- and y- coordinates. We call point P the **parent point** *of P' and the generated point P' the* **child point** *of P.*

Property 1. Given a parent point $P(x, y)$ and a point $P^{A_k}(x_k, x_k)$ with label length l_k,

1. If $y > x_k$, the point cannot apply label at any position.
2. If $x_k \geq y > x_k - l_k$, the point can apply label at position 4.
3. If $x_k - l_k \geq y$, the point can apply label at positions 3 and 4.
4. If $x > x_k$, the point can apply label at neither position 1 nor 2.
5. If $x_k \geq x > x_k - l_k$, the point can apply label at positions 1 and 4.
6. If $x_k - l_k \geq x$, the point can apply label at positions 1, 2, 3 and 4.

[1] If $P_t.x \geq P_s.x$, and $P_t.y \geq P_s.y$, then P_t is said to *dominate* P_s.

3 Point Labeling on a Single Line

We adopt a greedy method to solve the model Max-1d4P, namely, we will process the anchors in sequential manner, and maintain a set of realizations that reflect the best possible labeling, ignoring those that are known to be no better than the present set of realizations, after each anchor is processed. The following lemma is obvious.

Lemma 2. *Given two realizations R_s^i and R_t^i, for $i = 1, 2, \ldots, n$, we will select R_s^i over R_t^i, either if $R_s^i.c > R_t^i.c$, or if $R_s^i.c = R_t^i.c$, and R_s^i is better than R_t^i.*

Due to space constraint, we shall skip the proofs. The details of the proofs can be found in Yu et al.[22].

Lemma 3. *Given a realization R^{i-1}, if both label placements of the next anchor A_i at positions 2 and 3 (respectively, positions 1 and 4) are feasible, the selection of label at position 2 (respectively, position 1), above the line will yield a better realization R^i.*

Agarwal et al.[1] provided a standard dynamic programming method to solve the fixed-height rectangle label placement model in $O(n \log n + n^{2k-1})$ time if all the rectangles in the plane are exactly stabbed by k horizontal lines. Poon et al.[19] used a similar approach to solving the fixed-height rectangle 4P model (we will specify an implicit difference between these two models later). They both associated a polygonal line consisting of $2k-1$ orthogonal segments to specify all the possible $(2k-1)$-dimensional realizations. For the Max-1d4P model, since all the rectangles are stabbed by two horizontal lines exactly, it only needs a two-dimensional table $R[x, y]$ which stores the cardinality of the realization R with $R.a = x$ and $R.b = y$ (as we introduced in Section 2), and their solution leads to an $O(n^2)$ time algorithm. Chan[3] presented a different form of dynamic programming and improved the time complexity to $O(n \log n + n\Delta^{k-1})$ for the same model. The form $R[i, S]$ stores the cardinality of the realization R associated with the vertical line $x = x_i$ and a set S of disjoint rectangles intersecting the line $x = x_i$, where x_i, $1 \leq i \leq n$, denotes the abscissas of the left boundaries of all the rectangles, and R is a maximum independent set of rectangles to the right of $x = x_i$ and intersecting none of S with $|S| \leq k - 1$. For the Max-1d4P model, since all the rectangles are exactly stabbed by two horizontal lines, there is at most Δ choices for $|S| = k - 1$ (as $k = 2$), and thus it takes $O(n\Delta)$ time. Although their dynamic programming methods are apparently different, the operations (inserting or discarding the next rectangle) are executed iteratively for every possible realization. We provide another form of dynamic programming strategy and tackle the operations for partial representative realizations instead. It solves the Max-1d4P model in $O(n \log \Delta)$ time based on a particular analysis and improves their quadratic time results ($O(n^2)$ and $O(n\Delta)$, respectively) in the worse case. We first observe some properties of the model Max-1d4P.

Proposition 1. *Given a realization R^{i-1}, when both placements at label positions 1 and 2 (respectively, positions 3 and 4) of next anchor A_i are feasible,*

the selection of label at position 2 (respectively, position 3) will yield a better realization R^i.

Lemma 4. *Given a realization R^{i-1}, if the label placement of the next anchor A_i at position 3 is feasible, then label of A_i must be included in an optimal realization R^n that contains R^{i-1}, or $A_i.\ell \neq 0$.*

Corollary 1. *Given a realization R^{i-1}, when the label placement of next anchor A_i at position 2 is feasible, we have $A_i.\ell \neq 0$ in an optimal realization R^n.*

We introduce our main idea as follows. Let $S[i,j]$ denote a set of incomparable realizations R^j of cardinality i, for $1 \leq i \leq j \leq n$ ($i > 0$ since $A_1.\ell = 2$ without loss of generality). We shall apply a dynamic programming method to process the anchors and record the 'better' realizations of each possible cardinality. To find an optimal realization R^n, we may need to maintain intermediate realizations $S[i,j]$ for $1 \leq i \leq j \leq n$, that have the potential leading to an optimal realization. As we shall show later, for each $j \leq n$ we only need to maintain at most *five* subsets $S[k,j]$, $S[k+1,j]$, $S[k+2,j]$, $S[k+3,j]$, and $S[k+4,j]$ for some k, which is a *key* result of this paper. We shall process the table from $j = 1$ till n and fill each entry $S[i,j]$ with a set of incomparable realizations at each step.

The realizations in an incomparable set form a "*point chain*" in the plane without having any point in the chain dominate another. When we encounter a new anchor A_j, some of the points in this chain of cardinality k, for some k, will generate new child points, thus getting *upgraded* to a realization of cardinality $k+1$, some will remain as *non-upgraded* with cardinality k, and are kept as potential candidates without including A_j, leading possibly to an optimal solution, and some get eliminated due to some new child points upgraded from points of cardinality $k-1$. At the end after anchor A_n is processed, the realizations in the non-empty entry $S[i,n]$ with the largest i are optimal solutions.

To sum up, some points in the set $S[i,j]$ may simply move to $S[i,j+1]$ without increasing cardinality, following what we call a *non-upgrading process*. Other points in the set may generate points which are included in $S[i+1,j+1]$, whose cardinality is incremented, following what we call an *upgrading process*. When a point moves from one entry to another, it should be compared with other points in the target entry, and only *better* ones are kept. We repeat such operations until we have processed all anchors. The following is the algorithm for the model Max-1d4P.

Algorithm 1M4P. Find the maximum cardinality of map labeling for the model Max-1d4P.

Input. A set of anchors $\mathcal{A} = \{A_1, \ldots, A_n\}$ sorted by x-coordinates and associated set of labels.
Output. The maximum cardinality of an optimal realization R^n for Max-1d4P.
Method.
0. /*Use dynamic programming method on two parameters $S[i,j]$ with the anchor ordering in column and the cardinality of possible solutions in row.

Initialize the first entry $S[1,1]$ with the label placement of the first anchor at position 2, that is, $A_1.\ell = 2.*/$

1. For $j = 2$ **to** n

Let the largest cardinality of non-empty entries in column $j-1$ be k;

For $i = k$ **down to** $\max\{k-4, 0\}$

1-1. Classify the points in $S[i, j-1]$ into upgrading and non-upgrading classes according to A_j;

1-2. Move the non-upgraded points into $S[i, j]$;

1-3. Move the upgraded points into $S[i+1, j]$;

1-4. Compare the newly upgraded points with existing points in $S[i+1, j]$ and keep the better ones;

2. Output the largest i of the nonempty entry $S[i, n]$;

In what follows we will prove a few results that help establish the correctness of our algorithm. Let P_s and P_t be two incomparable points, and $P^{A_k}(x_k, x_k)$ be a point associated with the next anchor A_k.

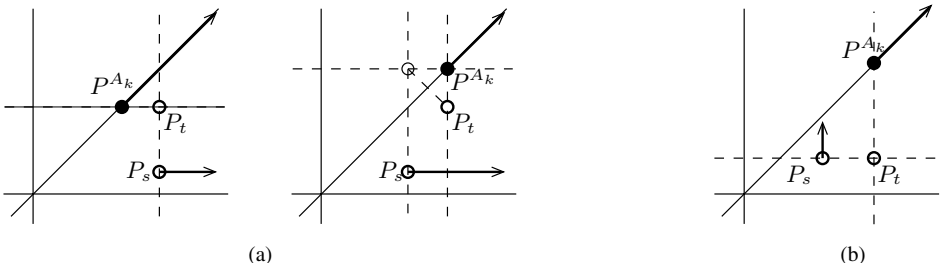

Fig. 1. Illustration of Lemma 5

Lemma 5. *For the following two cases, where P_s and P_t correspond to two realizations, and the next anchor is A_k associated with $P^{A_k}(x_k, x_k)$,*

(a) $P_s.x \geq P_t.x$, $P_s.y < P_t.y$, $P_s.c < P_t.c$ and $x_k \geq P_t.y$
(b) $P_s.y \geq P_t.y$, $P_s.x < P_t.x$, $P_s.c < P_t.c$ and $x_k \geq P_t.x$

P_t is better than P_s.

Lemma 6. *The points in an incomparable set going through an upgrading process collectively generate at most two incomparable child points.*

Theorem 1. *After processing an anchor A_u, if there is a point with cardinality k which is the ancestor of a point with cardinality $k+5$, then no point with cardinality k will lead to an optimal solution. That is, the difference in cardinality of incomparable points is at most four.*

By Theorem 1, in computing $S[i, j]$ for $1 \leq i \leq j \leq n$, it is sufficient to maintain at most five consecutive sets of incomparable realizations $S[k, *]$, $S[k+1, *]$, $S[k+2, *]$, $S[k+3, *]$ and $S[k+4, *]$. We have the following based on the above two results.

Theorem 2. *The number of points in an incomparable set is bounded by $O(\Delta)$, where Δ is the maximum number of labels whose intersection is nonempty.*

Lemma 7. *The following operations each take $O(\log \Delta)$ time.*

(1) Classifying points into the upgrading and non-upgrading classes.
(2) Finding two incomparable points among all upgraded points.
(3) The comparison between upgraded points and $O(\Delta)$ incomparable points.

Theorem 3. *The time complexity of Algorithm* 1M4P *is $O(n \log \Delta)$.*

When the algorithm terminates, any point in nonempty $S[i, n]$ with the largest i is an optimal solution (of maximum cardinality i). The actual placement of labels can be obtained if we record the processing history when a point is upgraded.

4 Fixed-Height Rectangle Label Placement in the Plane

First, we point out an implicit difference between point labeling problem and label placement problem. As point labeling problem was considered, where a constant number of label positions is allowed for each anchor, all label positions of each anchor were regarded as pairwise intersecting and the reduction from point labeling problem to label placement problem seemed intuitive[1,19]. However, for common 4P model, if there are more than one anchor lying on some horizontal line (or vertical line) with nonempty label intersection, we have the following implicit difference. Figure 2 shows that for anchor A_i, the selection of label at position 1 will affect the selection of the label at position 3 for anchor A_j. They could be regarded as intersecting, but in fact they are not. This problem can be resolved by set manipulation instead. Let a selection set of each anchor consist of all its label positions. We then allow at most one label position of each selection set be included in the solution. This doesn't affect the asymptotic running time but increase the implementation complexity.

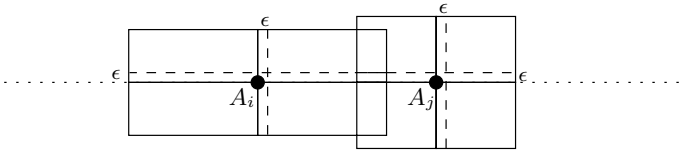

Fig. 2. An example shows an implicit difference between point labeling problem and label placement problem

For the fixed-height label placement model in the plane, Agarwal et al.[1] provided a $(1+1/k)$-factor PTAS algorithm running in $O(n \log n + n^{2k-1})$ time, and later Chan[3] improved it and presented a $(1+1/k)$-factor PTAS algorithm running in $O(n \log n + n\Delta^{k-1})$ time, for an integer $k \geq 1$. They both use the *line stabbing* technique and the shifting idea of Hochbaum and Maass[10]. Assume all

the fixed-height rectangles are stabbed by m horizontal lines and each rectangle is stabbed by one line exactly, and let C^i, $0 \leq i \leq k$, be the sub-collection of all rectangles which are not stabbed by any horizontal line $y = y_l$ with $l \equiv i \mod (k+1)$. Then C^i is a union of groups of rectangles, where each group can be stabbed by k horizontal lines and no two rectangles from different groups intersect. If we can solve the fixed-height label placement model for each group of n' rectangles for a given integer $k \geq 1$ in $O(t(k, n'))$ time, then we take the union of all groups to obtain a solution of C^i, $\forall i$, and select one of the solutions, O', of maximum cardinality. The total time complexity is bounded by $O(t(k, n))$ time and it is trivial to show that $k|O^*| \leq (k+1)|O'|$, where O^* is an optimal solution[3]. Hence we consider the fixed-height label placement model for n rectangles stabbed by k horizontal lines, for a given integer $k \geq 1$, from now on. Given a set of n fixed-height rectangles $r_1, r_2, \ldots, r_n$ sorted by x-coordinates of the left boundaries of all rectangles, stabbed by k horizontal lines $L_1, \ldots, L_k$, the following property is immediate by the line stabbing technique.

Property 2. From left to right, as we process r_i stabbed by L_l, r_i intersects at most two rectangles among $r_1, \ldots, r_{i-1}$: r_f, r_g, or r_g, r_h, where r_f, r_g, and r_h, are last rectangles of $r_1, \ldots, r_{i-1}$ on lines L_{l-1}, L_l, and L_{l+1}, respectively.

We associate a polygonal line consisting of $2k-1$ orthogonal segments to specify all the possible $(2k-1)$-dimensional realizations as used in Agarwal et al.[1]. However, based on the above property, we only need to consider the last rectangles on lines L_{l-1}, L_l, and L_{l+1} (i.e., three labels, and all the possible five-dimensional realizations), as we process each rectangle r_i, $1 \leq i \leq n$, stabbed by line L_l, for some l. In addition, we use the same dynamic programming method in Section 3 and extend our two-dimensional transformation for point labeling on a single line to $(2k-1)$-dimensional transformation for this model as follows. We transform a $(2k-1)$-tuple representation $(R.x_1, R.x_2, \ldots, R.x_{2k-1})$ of the labeling state of a realization R into a point $P(x_1, x_2, \ldots, x_{2k-1})$ in the $(2k-1)$-dimensional space. Therefore, as we process each rectangle r_i, $1 \leq i \leq n$, stabbed by line L_l, for some l, we only need to consider the five-dimensions $(x_{2l-2}, x_{2l-1}, x_{2l}, x_{2l+1}, x_{2l+2})$ of present incomparable points. The next lemma is an extension of Theorem 1.

Lemma 8. *After processing a rectangle r_i, if there is a point in $(2k-1)$-dimensional space with cardinality c which is the ancestor of a point with cardinality $c + (4k+1)$, then no point with cardinality c will lead to an optimal solution. That is, the difference in cardinality of incomparable points is at most $4k$. More precisely, after processing a rectangle r_i, if there is an ancestor point P with cardinality c upgraded by at least five rectangles stabbed by the same line, then no point with cardinality c will lead to an optimal solution.*

By Lemma 8, as we compute $S[i, j]$ for $1 \leq i \leq j \leq n$, it is sufficient to maintain at most $\min\{4k+1, n\}$ consecutive sets of incomparable points $S[c, *], S[c+1, *], \ldots, S[c+\min\{4k, n-1\}, *]$. We further extend Theorem 2 to the following lemma.

Lemma 9. *The number of points in an incomparable set is bounded by* $O(\Delta^{k-1})$*, where* Δ *is the maximum number of rectangles whose intersection is nonempty.*

We construct two three-dimensional range trees T_1 and T_k, and $k-2$ five-dimensional range trees $T_2, \ldots, T_{k-1}$ to determine whether the placement of next rectangle r_i stabbed by line L_j is feasible or not, by a range query of T_j in five-dimension $(x_{2j-2}, x_{2j-1}, x_{2j}, x_{2j+1}, x_{2j+2})$. According to the above lemmas, we extend Lemma 7 and Theorem 3 to obtain the result.

Theorem 4. *Given an integer* $k \geq 1$*, the label placement problem of* n *fixed-height rectangles stabbed by* k *horizontal lines can be solved in* $O(n \log n + kn \log^4 \Delta + \Delta^{k-1})$ *time and* $O(k\Delta^3 \log^4 \Delta + k\Delta^{k-1})$ *space.*

5 Concluding Remarks

We have extended the decision version of the map labeling problem on a horizontal line to an optimization version where the number of feasible labels is to be maximized. It is a variation of maximum independent set problem on interval graphs. Improving the previous related results, we have presented a faster $O(n \log \Delta)$ time algorithm for the Max-1d4P model by dynamic programming on two parameters: the anchor ordering and the cardinality of possible solutions. In addition, we have further extended our method to improve the previous results for the fixed-height rectangle label placement model in the plane and presented a $(1 + 1/k)$-factor PTAS algorithm that runs in $O(n \log n + kn \log^4 \Delta + \Delta^{k-1})$ time, using $O(k\Delta^3 \log^4 \Delta + k\Delta^{k-1})$ storage.

We conclude with two open questions concerning map label number maximization model. First, whether there exist solutions for Max-Slope4P fixed-height(or width) rectangle label model remains to be seen. Secondly, how to improve Algorithm 1M4P to obtain a linear time algorithm solving the Max-1d4P model given sorted anchors, and further extend it to reduce the time complexity of PTAS for label placement problem in the plane is worthwhile.

References

1. Agarwal, P.K., van Kreveld, M., Suri, S.: Label placement by maximum independent set in rectangles. Computational Geometry: Theory and Applications 11, 209–218 (1998)
2. Berman, P., DasGupta, B., Muthukrishnan, S., Ramaswami, S.: Efficient approximation algorithms for tiling and packing problems with rectangles. Journal of Algorithms 41, 443–470 (2001)
3. Chan, T.M.: A note on maximum independent sets in rectangle intersection graphs. Inform. Process. Letters 89, 19–23 (2004)
4. Chazelle, B., 36 co-authors: The computational geometry impact task force report. In: Chazelle, B., Goodman, J.E., Pollack, R. (eds.) Advances in Discrete and Computational Geometry, vol. 223, pp. 407–463. American Mathematical Society, Providence (1999)

5. Chen, Y.-S., Lee, D.T., Liao, C.-S.: Labeling points on a single line. International Journal of Computational Geometry and Applications (IJCGA) 15(3), 261–277 (2005)
6. Christensen, J., Marks, J., Shieber, S.: An empirical study of algorithms for point feature label placement. ACM Transactions on Graphics 14(3), 203–232 (1995)
7. Duncan, R., Qian, J., Vigneron, A., Zhu, B.: Polynomial time algorithms for three-label point labeling. Theoretical Computer Science 296(1), 75–87 (2003)
8. Formann, M., Wagner, F.: A packing problem with applications in lettering of maps. In: Proceedings of the 7th ACM Symposium on Computational Geometry, pp. 281–288. ACM Press, New York (1991)
9. Garrido, M.Á., Iturriaga, C., Márquez, A., Portillo, J.R., Reyes, P., Wolff, A.: Labeling subway lines. In: Eades, P., Takaoka, T. (eds.) ISAAC 2001. LNCS, vol. 2223, pp. 649–659. Springer, Heidelberg (2001)
10. Hochbaum, D.S., Maass, W.: Approximation schemes for covering and packing problems in image processing and VLSI. J. ACM 32(1), 130–136 (1985)
11. Iturriaga, C., Lubiw, A.: Elastic labels: The two-axis case. In: DiBattista, G. (ed.) GD 1997. LNCS, vol. 1353, pp. 181–192. Springer, Heidelberg (1997)
12. Iturriaga, C., Lubiw, A.: Elastic labels around the perimeter of a map. Journal of Algorithms 47(1), 14–39 (2003)
13. Jiang, M., Qian, J., Qin, Z., Zhu, B., Cimikowski, R.: A simple factor-3 approximation for labeling points with circles. Inform. Process. Letters 87(2), 101–105 (2003)
14. Kato, T., Imai, H.: 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, Japanese, p. 1138 (1988)
15. Knuth, D., Raghunathan, A.: The problem of compatible representatives. SIAM Disc. Math. 5(3), 422–427 (1992)
16. van Kreveld, M., Strijk, T., Wolff, A.: Point labeling with sliding labels. Computational Geometry: Theory and Applications 13, 21–47 (1999)
17. Kim, S.K., Shin, C.-S., Yang, T.-C.: Labeling a rectilinear map with sliding labels. International Journal of Computational Geometry and Applications 11(2), 167–179 (2001)
18. Marks, J., Shieber, S.: The computational complexity of cartographic label placement, Technical Report TR-05-91. Harvard University CS (1991)
19. Poon, S.-H., Shin, C.-S., Strijk, T., Uno, T., Wolff, A.: Labeling points with weights. Algorithmica 38(2), 341–362 (2003)
20. Strijk, T., Wolff, A.: Labeling points with circles. International Journal of Computational Geometry and Applications 11(2), 181–195 (2001)
21. Wagner, F., Wolff, A.: A practical map labeling algorithm. Computational Geometry: Theory and Applications. 7, 387–404 (1997)
22. Yu, K.-L., Liao, C.-S., Lee, D.T.: Maximizing the number of independent labels in the plane, manuscript, A preliminary version appeared in proceedings of the Frontiers of Algorithmics Workshop (April 2007), `http://www.iis.sinica.edu.tw/~shou794/research/1DMIS_p15_0416.pdf`