Optimal algorithm for a special point-labeling problem ☆

Sasanka Roy [a], Partha P. Goswami [b], Sandip Das [a], Subhas C. Nandy [a,*]

[a] *ACM Unit, Indian Statistical Institute, Calcutta 700 108, India*
[b] *Computer Center, Calcutta University, Calcutta 700 020, India*

Abstract

A special class of map labeling problem is studied. Let $P = \{p_1, p_2, \ldots, p_n\}$ be a set of point sites distributed on a 2D map. A label associated with each point p_i is an axis-parallel rectangle r_i of specified width. The height of all r_i, $i = 1, 2, \ldots, n$ are same. The placement of r_i must contain p_i at its top-left or bottom-left corner, and it does not obscure any other point in P. The objective is to label the maximum number of points on the map so that the placed labels are mutually non-overlapping. We first consider a simple model for this problem. Here, for each point p_i, the corner specification (i.e., whether the point p_i would appear at the top-left or bottom-left corner of the label) is known a priori. We show that the time complexity of this problem is $\Omega(n \log n)$, and then propose an algorithm for this problem which runs in $O(n \log n)$ time. If the corner specifications of the points in P are not known, our algorithm is a 2-approximation algorithm. Here we propose an efficient heuristic algorithm that is easy to implement. Experimental evidences show that it produces optimal solutions for most of the randomly generated instances and for all the standard benchmarks available in http://www.math-inf.uni-greifswald.de/map-labeling/.

Keywords: Map labeling; Chordal graphs; Algorithms; Complexity; Analysis of algorithms

1. Introduction

Labeling a point set is a well-studied problem in the geographic information systems, where the points represent cities on a map which need to be labeled with city names. The point set labeling problem finds many important statistical applications, e.g., scatter plot of principal component analysis [5], in spatial statistics where the aim is to post the field measures against the points, etc.

In general, the label placement problem includes positioning labels for area, line and point features on a 2D map. The basic requirements of a labeling algorithm are:

(i) the label of a site should touch the site at its boundary,
(ii) the labels of two sites must not overlap, and
(iii) the label of one site should not obscure the other sites on the map.

Many other aesthetic requirements for map labeling are listed in [6]. Here, two major types of problems are considered:

☆ The primary version of this paper has been accepted in SWAT 2002.

* Corresponding author.
E-mail address: nandysc@isical.ac.in (S.C. Nandy).

(i) label as many sites as possible where label size of each point is specified, and
(ii) find the largest possible size of the label such that all the sites can be labeled in a non-overlapping manner.

In general, both of these problems are NP-hard [2]. We shall consider a special case of the first variation of the point-site labeling problem.

Let $P = \{p_1, p_2, \ldots, p_n\}$ be a set of n points in the plane. A label associated with each point p_i is an axis-parallel rectangle r_i of a constant height but of variable width. Here height of a label indicates the font size and width indicates the number of characters in that label. For each point $p_i \in P$, we have a set π_i of marked positions on the boundary of r_i. In the placement of label r_i, p_i must appear on one of the marked positions in π_i. A feasible configuration is a family of axis-parallel rectangles (labels) $R = \{r_{i_1}, r_{i_2}, \ldots, r_{i_k}\}$ $(i_1 \neq i_2 \neq \cdots \neq i_k)$, such that the members in R are mutually non-overlapping. Here r_{i_j} is represented by a tuple $\{(p_{i_j}, \alpha) \mid p_{i_j} \in P,\ \alpha \in \pi_{i_j}$ and r_{i_j} is placed with p_{i_j} at the position α on its boundary$\}$. The label placement problem is to maximize k, i.e., to find the largest feasible configuration. Typical choices of π_i include

(i) the end points of the left edge of r_i,
(ii) the four corners of r_i, or (iii) the four corners and the center points of four edges of r_i, etc.

In [1], an $O(\log n)$-approximation algorithm is proposed for this problem which runs in $O(n \log n)$ time. In particular, if the labels are of the same height, a dynamic programming approach produces a $(1 + \frac{1}{k})$-approximation result in $O(n \log n + n^{2k-1})$ time. This case is of particular importance since it models the label placement problem when all labels have the same font size. In [13], a simple heuristic algorithm for the point labeling problem is proposed which produces near-optimal solution, but its worst case running time is $O(n^2)$. The point labeling with sliding labels is studied in [7,11]. The decision theoretic version of the map labeling problem for horizontal and vertical line segments are solved in polynomial time in [8,10]. But, to our knowledge, no polynomial time algorithm exists for the optimization version of any restricted class of the point set labeling problem. Here, we identify a special type of point set labeling problem, whose worst case time complexity is $O(n \log n)$.

In our model, the labels of the points on the map are axis-parallel rectangles of a constant height (h) but of variable width. The width (w_i) of the label of a point p_i is pre-specified. The point p_i appears either on the top-left or the bottom-left corner of its label. A label is said to be *valid* if it does not contain any other point(s) of P. We consider the following two problems:

P1: For each point $p_i \in P$, the corner specification (i.e., whether p_i would appear at the top-left or bottom-left corner of the label) is known.
P2: The corner specifications of the points in P are not known.

Problem P1 is modeled as finding a maximum independent set of a chordal graph, and an algorithm is proposed which produces an optimal solution in $O(n \log n)$ time. The worst case time complexity of solving problem P1 is shown to be $\Omega(n \log n)$. Next, a minor modification of our algorithm is proposed which can produce a 2-approximation result for problem P2. Finally, an efficient heuristic algorithm for problem P2 is given. The experimental results justify that for most of the randomly generated examples, and for all the standard benchmarks [12], it produces optimum solution. Surely, we have encountered a few random instances where it fails to produce optimum solution. However, for all instances we have considered, our algorithm outputs better result than the algorithm presented in [1].

2. Problem P1

Let $P = \{p_1, p_2, \ldots, p_n\}$ be a set of points in the plane. Each point p_i is associated with a label r_i which is a closed region bounded by an axis-parallel rectangle. The heights of all r_i are the same, but their width may vary. The placement of label r_i must coincide with the point p_i at either its top-left or bottom-left corner which is specified. A label r_i is said to be *valid* if it does not contain any point $p_j \in P$ $(j \neq i)$ in its interior. Thus, each point in P is attached with at most one valid label. We construct a graph, called *label graph* $LG = (V, E)$, whose vertices correspond to the set of valid labels of the points in P. A pair

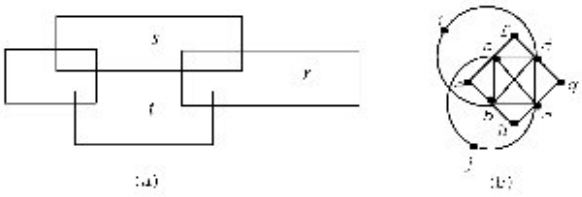

Fig. 1. Demonstration of Lemma 1.

of nodes v_i, v_j are adjacent if their corresponding labels have a non-empty intersection. Our problem is to find the largest subset $P' \subseteq P$ such that valid labels corresponding to the members of P' are mutually non-overlapping. Obviously, the above problem reduces to finding a maximum independent set of the graph *LG*.

2.1. Graph-theoretic characterization

Definition 1 [3]. An undirected graph is a *chordal graph* if and only if every cycle of length greater than or equal to 4 possesses a chord.

Definition 2 [3]. An *interval graph* is the intersection graph of a family of intervals on a real line.

Lemma 1. *The label graph (intersection graph of a set of valid labels) is a chordal graph.*

Proof. Assume to the contrary that there exists a label graph which contains a chordless cycle C of length greater than or equal to 4. Let R be the set of valid labels (of same height) corresponding to the vertices in C. Let $r \in R$ be the label whose left boundary is rightmost among the left boundaries of the members in R, and let s and t be two labels that intersect r. Here, s and t do not intersect since C does not have any chord. Since the left edge of r is rightmost, either s or t must contain the point that r labels (see Fig. 1(a)). This contradicts the validity of r. □

Note. It is easy to prove that the converse of Lemma 1 is not true. In other words, there exists some chordal graphs (for example see Fig. 1(b)) which are not label graph.

Lemma 2. *Any interval graph (IG) is a label graph (LG), but the converse is not true.*

Proof. Let $L = \{\lambda_1, \lambda_2, \ldots, \lambda_n\}$ be a set of intervals. In order to obtain a realization of a label graph from an interval graph, we place an interval λ_i above another interval λ_j if the left end point of λ_i is to the right of that of λ_j. Ties are broken arbitrarily. The vertical gap between two consecutive intervals are same. Next, we draw rectangles of equal height over all the intervals. See Fig. 2(a) for the demonstration of the steps. The proof of the first part of the lemma follows from the fact that, the placement of each label is valid, i.e., the top-left corner of each label is not inside any of the other labels.

To disprove the converse, consider a label graph *LG*, its label realization, and its complement $\overline{LG}$ (see Fig. 2(b)). By Lemma 1, *LG* is a chordal graph, but it is easy to observe that the graph $\overline{LG}$ does not have any transitive orientation (see [3, p. 105]). So, *LG* is not an interval graph (see [3, p. 149]). □

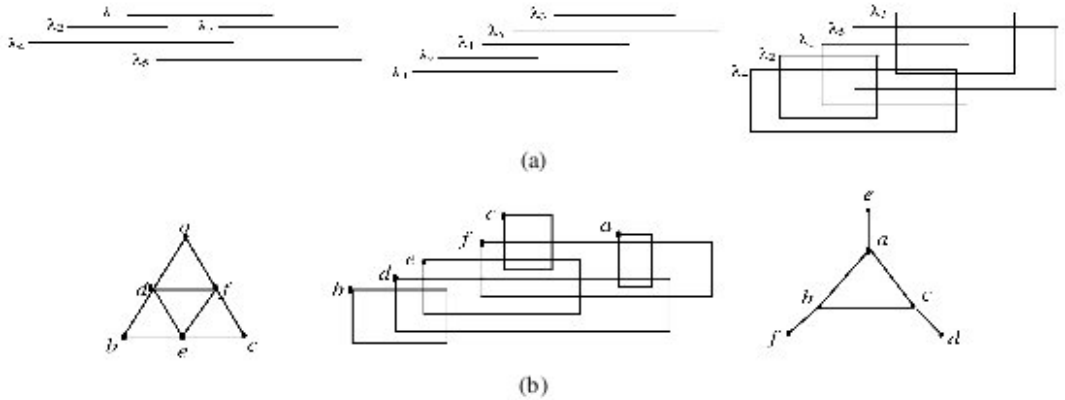

Fig. 2. Proof of Lemma 2.

Let $\mathcal{IG}$, $\mathcal{LG}$ and $\mathcal{CG}$ denote the classes of interval graphs, label graphs and chordal graphs, respectively. Now, we have the following theorem:

Theorem 1. $\mathcal{IG} \subset \mathcal{LG} \subset \mathcal{CG}$.

2.2. Algorithm

We propose an efficient algorithm for finding the placement of maximum number of labels for the points in P. For each point $p_i \in P$, the size of its label r_i, and the corner specification (top-left/bottom-left) of point p_i on r_i is already known. For each label, we check its *validity* (whether it obscures any other points or not) by searching in a 2D tree [9] with the set of points in P. This step requires $O(n \log n)$ time in total for all the labels.

Let $R = \{r_1, r_2, \ldots, r_N\}$ $(N \leqslant n)$ be a set of valid labels placed on the plane. The traditional line sweep technique may be used to construct the label graph LG in $O(n \log n + |E|)$ time. Our objective is to find a maximum independent set of LG, denoted by *MIS*. As LG is a perfect graph (see Lemma 1 and Theorem 4.1 of [3]), we define the perfect elimination order (*PEO*) among the vertices of the graph LG as follows:

Definition 3 [3]. A vertex v of a graph LG is a *simplicial vertex* if v and its adjacency set $Adj(v)$ induces a complete subgraph of LG.

Definition 4 [3]. Let $\sigma = \{v_1, v_2, \ldots v_n\}$ be an ordering of vertices of LG. We say that σ is a *perfect elimination order* (*PEO*) if each v_i is a simplicial vertex of the induced subgraph with the set of vertices $\{v_i, \ldots, v_n\}$. In other words, each set $X_i = \{v_j \in Adj(v_i) \mid j > i\}$ is a clique.

Lemma 3. *If R is a set of valid labels then the sorted order of the left boundaries of the members of R from the right to the left, gives a PEO of the graph LG.*

Proof. Let $L = \{\lambda_1, \lambda_2, \ldots, \lambda_N\}$ be the left boundaries of the labels in R, and let L^* denote the sorted sequence of the members in L in a right-to-left order. Consider the left boundary λ_i of a label r_i. Let R' be a subset of R such that the left boundaries of all the members in R' appear after λ_i in L^*, and all of them intersect r_i. We need to prove that $R' \cup \{r_i\}$ forms a clique. As the placement of each $r \in R'$ is valid, either all the members of R' contain the top-left corner of r_i or all of them enclose the bottom-left corner of r_i (in Fig. 3, r_j and r_k enclose the top-left corner of r_i). In other words, the corresponding point $p_i \in P$ is present either at the bottom-left corner or at the top-left corner of r_i. Hence all the members in $R' \cup \{r_i\}$ have a common region of intersection. □

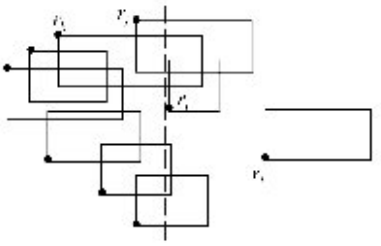

Fig. 3. Proof of Lemma 3.

The *PEO* of the graph $LG = (V, E)$ can be obtained in $O(n \log n)$ time (by Lemma 3). Let σ be a *PEO* of LG. We define inductively a sequence of vertices $y_1, \ldots, y_t$ in the following manner: $y_1 = \sigma(1)$; y_i is the first vertex in σ which follows y_{i-1}, and which is not in $X_{y_1} \cup X_{y_2} \cup \cdots \cup X_{y_{i-1}}$, where $X_v = \{x \in Adj(v) \mid \sigma^{-1}(v) < \sigma^{-1}(x)\}$. Hence, all the vertices following y_t are in $X_{y_1} \cup X_{y_2} \cup \cdots \cup X_{y_t}$, and $V = \{y_1, \ldots, y_t\} \cup X_{y_1} \cup X_{y_2} \cup \cdots \cup X_{y_t}$.

Lemma 4. *The vertices $\{y_1, \ldots, y_t\}$ form a maximum independent set in LG.*

Proof. Follows from Theorem 1 and Theorem 4.18 of [3]. □

The above method of obtaining a maximum independent set of LG needs $O(|V| + |E|)$ time [3]. We now show that if the placement of the labels corresponding to the vertices of LG is available in the plane, then a maximum independent set of LG can be determined in a faster way by simply sweeping the plane with a vertical line from right to left.

Algorithm Max_Indep_Set (* for finding a maximum independent set of LG *).

Data structure: An array L containing the line segments corresponding to the left and the right

boundaries of all the valid labels $r_i \in R$ in decreasing order of their x-coordinates. The right boundary of a label has a pointer to its left boundary. With each element of L, a *flag* bit is maintained and is initialized to 0. During execution, the *flag* of an element is set to 1 if it is selected as a member of *MIS* or if its corresponding label overlaps on the label of an existing element in *MIS*. The output of this algorithm is a maximum independent set of the intersection graph of R.

We maintain an interval tree $\mathcal{T}$ [9] with the y-coordinates of the top and bottom boundaries of the labels in R. It stores the vertical intervals of those labels that the sweep-line currently intersects, and does not overlap with any of the existing members in *MIS*.

Algorithm. We process the elements of the array L in order. If the *flag* bit of the current element $I \in L$ is 1, then ignore I; otherwise perform the following steps.

- If I corresponds to the right boundary of a label, then insert I in $\mathcal{T}$.
- If I corresponds to the left boundary of a label, then insert the label r (corresponding to I) in *MIS*. Next, Search $\mathcal{T}$ to find the set $X_r = \{J \mid J \in \mathcal{T}$ and the label corresponding to J overlaps with that of $I\}$. The *flag* bit of the left boundary element of each member of X_r is set to 1 (i.e., these elements will not be processed during the sweep).
- Next, remove all the intervals in $I \cup X_r$ from the interval tree $\mathcal{T}$.

Finally, report the elements of the array *MIS*.

Theorem 2. *Algorithm* **Max_Indep_Set** *computes a maximum independent set of the label graph LG in* $O(n \log n)$ *time.*

Proof. The *PEO* of the graph *LG* is obtained from the right to left sweep on the plane (see Lemma 3). When a label is selected as a member in the *MIS*, its adjacent labels are discarded by setting 1 in their *flag* bit. Now by Lemma 4, the correctness of the algorithm follows. Next, we discuss the time complexity of the algorithm.

The initial sorting requires $O(n \log n)$ time. A vertical interval corresponding to each label is inserted once in $\mathcal{T}$. After placing a label, say r_i, in the *MIS* array, it is deleted from $\mathcal{T}$. The set of labels X_{r_i}, whose corresponding vertical intervals are in $\mathcal{T}$ and which overlap on r_i, are recognized in $O(|X_{r_i}| + \log n)$ time. These intervals are deleted from $\mathcal{T}$, so that none of them will be recognized further. So, for every pair of elements $r_i, r_j \in MIS$, $X_{r_i} \cap X_{r_j} = \emptyset$, and if *MIS* contains t elements then $\sum_{i=1}^{t} |X_{r_i}| < n$. Each insertion/deletion in $\mathcal{T}$ requires $O(\log n)$ time. □

Theorem 3. *The worst case time complexity of the problem* P1 *is* $\Omega(n \log n)$.

Proof. Recall that an interval graph is a label graph (Lemma 2), and the time complexity of the problem of finding a maximum independent set of an interval graph is $\Omega(n \log n)$ [4]. If an algorithm exists which can compute a maximum independent set of a label graph in less than $O(n \log n)$ time, then it can be used for finding a maximum independent set of an interval graph, which is impossible. □

3. Problem P2

As in problem P1, here also the point p_i may appear either at the top-left or bottom-left corner of r_i, and for each point $p_i \in P$, the width of its label r_i is given, but unlike problem P1, the corner specification of the label r_i is not known in advance. Thus, a point p_i may not have any valid label, or it may have one or two valid labels. If a point p_i has two valid labels, say r_i and r_i', they must have an edge in the label graph *LG*. Here *LG* may contain cycle(s) of length $\geqslant 5$ (see Fig. 4), and hence *LG* is not a perfect graph in this case. So, the earlier algorithm cannot produce optimum result. But, a minor modification of our line sweep algorithm **Max_Indep_Set** produces a 2-approximation solution for problem P2. An efficient heuristic algorithm for the problem P2 is also proposed.

3.1. 2-approximation result

Let R be the set of all valid labels. During the right to left scan, let r_i and r_i' be a pair of valid labels (corresponding to point p_i) which are currently encountered by the sweep line. Prior to this instant of time some labels are already selected for solution, and for these selections, some labels are removed from

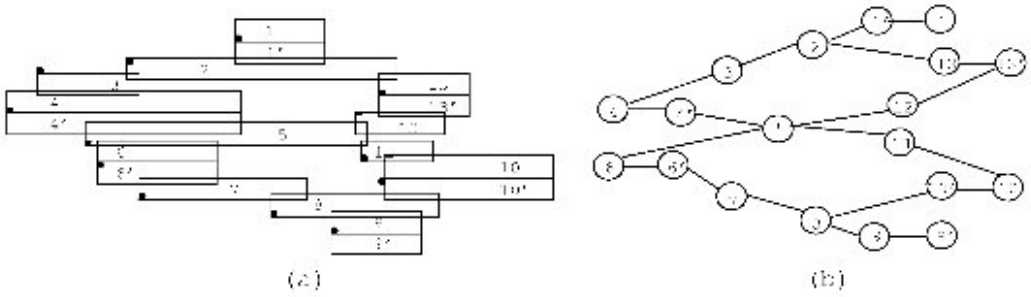

Fig. 4. The label graph corresponding to problem P2 is not a perfect graph.

the set of valid labels by setting their *flag* bit to 1. It is easy to show that there exists an optimal solution containing either r_i or r_i'. We select any one (say r_i) of them arbitrarily in our modified algorithm. Let R^* denote the remaining set of valid labels whose *flag* bit contain 0. R^* includes r_i and r_i'. LG^* denotes the label graph of R^*. R_i and R_i' denote the set of labels adjacent to r_i and r_i', respectively. It needs to mention that r_i, r_i' belong to both R_i and R_i'. We use $MIS(R)$ to denote the maximum independent set of the label graph corresponding to the set of valid labels R returned by our algorithm.

Lemma 5. $1 + \#(MIS(R^* \setminus R_i)) \leqslant \#(MIS(R^*)) \leqslant 2 + \#(MIS(R^* \setminus R_i))$, *where* $\#(A)$ *indicates the size of set* A.

Proof. The first part of the lemma is trivial. For the second part, consider the following argument.

If $r_i \in MIS(R^*)$, then $MIS(R^*) = \{r_i\} \cup MIS(R^* \setminus R_i)$. So the lemma follows in this case.

If $r_i' \in MIS(R^*)$ then $MIS(R^*) = \{r_i'\} \cup MIS(R^* \setminus R_i')$.

Again, $(R^* \setminus R_i') = (R^* \setminus \{R_i \cup R_i'\}) \cup (R_i \setminus \{r_i, r_i'\})$.

Thus, $\#(MIS((R^* \setminus \{R_i \cup R_i'\}) \cup (R_i \setminus \{r_i, r_i'\}))) \leqslant \#(MIS(R^* \setminus \{R_i \cup R_i'\})) + \#(MIS(R_i \setminus \{r_i, r_i'\}))$.

Again, $\#(MIS(R_i \setminus \{r_i, r_i'\})) = 1$ since $R_i \setminus \{r_i, r_i'\}$ forms a clique. Hence the lemma follows. □

If $r_i \in MIS(R)$, and we choose r_i' we lead to a non-optimal solution. But Lemma 5 says that the maximum penalty for doing a wrong choice is at most one. If k choices are made during the entire execution of the algorithm, and all the lead to a non-optimal result, the size of $MIS(R)$ may be at most $2k$. Thus we have the following theorem:

Theorem 4. *If the ties are resolved arbitrarily, then the size of the solution obtained by the algorithm* **Max_Indep_Set** *is no worse than* $\frac{1}{2} \times \#(MIS(R))$.

3.2. Heuristic algorithm

1. At each step, locate a simplicial vertex of the label graph.
2. If such a vertex is found then
   Select it as a member of maximum independent set.
3. else (* the rightmost point p_i has a pair of valid labels, say r_i and r_i' *)
   Select the vertex corresponding to any one label (say r_i) of p_i arbitrarily.
4. Remove the selected vertex and all its adjacent vertices from LG by setting their *flag* bit.
5. Repeat steps 1 to 4 until the *flag* bit of all the vertices are set to 1.

We implement the algorithm by sweeping two horizontal lines l_1 and l_2 and two vertical lines J_1 and J_2 simultaneously. l_1 (respectively l_2) is swept from the top (respectively bottom) boundary to the bottom (respectively top) boundary, and J_1 (respectively J_2) is swept from the left (respectively right) boundary to the right (respectively left) boundary. Below, we explain the sweeping of the vertical line J_2. The sweeping of the other lines are done in a similar manner.

We maintain an interval tree $\mathcal{T}$ with the y-coordinates of the end points of the vertical boundaries of n valid labels. During the sweep, when J_2 encounters a right boundary of a valid label, the corresponding interval is stored in the associated structure of the appropriate node of $\mathcal{T}$ and sweep continues. When a left boundary is faced by J_2, the sweep halts for

Table 1
Experimental results on the benchmarks cited in [12]

Examples	No. of sites	Height (pixels)	Optimum solution	Our algorithm		Algorithm [1]	
				No. of valid labels	No. of points labeled	No. of valid labels	No. of points labeled
Tourist shops in Berlin	357	4	216	401	216	799	165
		5	206	389	206	769	166
German railway stations	366	4	304	569	304	1133	258
		5	274	513	274	1030	243
American cities	1041	4	1036	2048	1036	4095	859
		5	1031	2027	1031	4053	878
Drill holes in Munich	19461	1000	***	27156	13895	54325	13730
		5000	***	10107	4737	20197	4678

*** Indicates optimum solution cannot be found for that example.

searching with the corresponding interval, say ℓ, in the interval tree to find the set of other intervals (present in the interval tree) which overlap on ℓ. If all these intervals are mutually overlapping, then the vertex of *LG* corresponding to the label having the above left boundary, is simplicial.

At each step I_1 (respectively I_2) proceeds until a bottom (respectively top) boundary of a valid label is faced, and J_1 (respectively J_2) proceeds until a left (respectively right) boundary of a valid label is faced. If any of this scan returns a simplicial vertex v_i, the corresponding label r_i is inserted in *MIS*. Otherwise, the label which is obtained by J_2 (in right to left scan) is selected for insertion in *MIS*. The vertex v_i, and all the vertices whose labels overlap on r_i are marked by setting their *flag* bit to 1. The corresponding intervals are removed from all the four interval trees if they are present there. The process is repeated until the *flag* bit of all the vertices are set to 1.

The time and space complexities of our algorithm are $O(n \log^2 n)$, and $O(n)$, respectively.

3.3. Experimental results

We executed this algorithm on many randomly generated examples and on all the benchmark examples available in [12]. In most of the examples, at each step we could locate a simplicial vertex, and finally arrived at an optimum solution. It needs to mention that we have also encountered few random instances where our algorithm could not produce optimum solution. For example, in Fig. 4, the optimum solution is $\{1, 3, 4', 6, 7, 9, 10', 11, 12, 13\}$, whereas our algorithm returns $\{1, 3, 4', 6', 8, 9', 10, 12, 13\}$. We have also compared our result with the labeling algorithm suggested in [1] (see Table 1). The algorithm of [1] assumes that the label of a point p_i may contain p_i at one of its four corner. Thus, each point may have at most four valid labels. In spite of this flexibility, it is observed that our algorithm can label more points than the algorithm proposed in [1].

Acknowledgement

We thank Dr. Alexander Wolff for providing us the benchmark examples. We also like to mention that the valuable comments of the anonymous referees helped us to improve the quality of presentation of the paper.

References

[1] P.K. Agarwal, M. van Kreveld, S. Suri, Label placement by maximum independent set in rectangles, Comput. Geometry: Theory Appl. 11 (1998) 209–218.
[2] M. Formann, F. Wagner, A packing problem with applications to lettering of maps, in: Proc. 7th Annual ACM Symp. on Computational Geometry, 1991, pp. 281–288.
[3] M.C. Golumbic, Algorithmic Graph Theory and Perfect Graphs, Academic Press, New York, 1980.
[4] U.I. Gupta, D.T. Lee, J.Y.-T. Leung, Efficient algorithms for interval graphs and circular arc graphs, Networks 12 (1982) 459–467.
[5] E.H. Isaaks, R.M. Srivastava, An Introduction to Applied Geostatistics, Oxford University Press, New York, 1989.
[6] E. Imhof, Positioning names on maps, The American Cartographer 2 (1975) 128–144.
[7] M. van Kreveld, T. Strijk, A. Wolff, Point labeling with sliding labels, Comput. Geometry: Theory Appl. 13 (1999) 21–47.

[8] C.K. Poon, B. Zhu, F. Chin, A polynomial time solution for labeling a rectilinear map, in: Proc. 13th ACM Symp. on Computational Geometry, 1997, pp. 451–453.
[9] F.P. Preparata, M.I. Shamos, Computational Geometry: An Introduction, Springer, Berlin, 1985.
[10] T. Strijk, M. van Kreveld, Labeling a rectilinear map more efficiently, Inform. Process. Lett. 69 (1999) 25–30.
[11] T. Strijk, M. van Kreveld, Practical extension of point labeling in the slider model, in: 7th Internat. Symp. on Advances in Geographical Information Systems (ACM-GIS'99), 1999, pp. 47–52.
[12] A. Wolff, Map labeling webpage, http://www.math-inf.uni-greifswald.de/map-labeling/.
[13] F. Wagner, A. Wolff, V. Kapoor, T. Strijk, Three rules suffice for good label placement, Algorithmica 30 (2001) 334–349.