

# Labeling Points with Given Rectangles \*

Joo-Won Jung      Kyung-Yong Chwa

Computer Science Division, Department of EECS  
Korea Advanced Institute of Science and Technology (KAIST)  
Daejeon, 305-701, Republic of Korea  
{jwjung,kychwa}@jupiter.kaist.ac.kr

## Abstract

In this paper, we consider the following problem: Given  $n$  pairs of a point and an axis-parallel rectangle in the plane, place each rectangle at each point in order that the point lies on the corner of the rectangle and the rectangles do not intersect. If the size of the rectangles may be enlarged or reduced at the same factor, maximize the factor. This paper generalizes the results of Formann and Wagner[3]. They considered the uniform squares case and showed that there is no polynomial time algorithm less than 2-approximation. We present a 2-approximation algorithm of the non-uniform rectangle case which runs in  $O(n^2 \log n)$  time and takes  $O(n^2)$  space. We also show that the decision problem can be solved in  $O(n \log n)$  time and space in the RAM model by transforming the problem to a simpler geometric problem.

## 1 Introduction

Label placement plays an important role in information visualization. On maps, diagrams, and graphs, properly placed text labels clarify the meaning of a point, a line, or an area. Many researchers in cartography and computational geometry have suggested various models and solutions. There is an extensive bibliography in Reference [9]. ACM Computational Geometry Impact Task Force Report[1] pointed out the label placement as an important research area.

Since most of the label placement problems are  $\mathcal{NP}$ -hard, one should consider its approximate solution. An algorithm is called an  $\varepsilon$ -approximation algorithm if it guarantees the value at least  $\sigma/\varepsilon$ , where  $\sigma$  is the maximum value.

Formann and Wagner[3] considered point-set labeling with axis-parallel uniform squares and the size maximization of the squares. They showed that there is no polynomial time algorithm better than 2-approximation and presented a 2-approximation algorithm. They also considered the decision problem of labeling with rectangles, but two positions per point are allowed. They showed that the problem can be solved in polynomial time by formulating an input as a 2-SAT problem. They also pointed out that there may exist  $\Omega(n^2)$  intersections

---

\*This work was supported by grant No. R01-2003-000-11676-0 from Korea Science and Engineering Foundation.

in rectangular labels, but the time complexity can be reduced using the result of Imai and Asano[4].

Wagner and Wolff[7] focused on the combinatorial characteristic of the labeling problem and presented several candidate deletion rules that does not harm the optimal solution. Wagner et al. [8] reduced the number of sufficient deletion rules to three. Qin and Zhu[6] considered the slider model which allows the point to lie on the boundary of the square. They presented a 2-approximation algorithm.

In this paper, we consider maximizing the size of non-uniform axis-parallel rectangular labels. We show 2-approximation algorithms, which are, as far as the authors know, the first solutions of the Problem R4 in [3] which has been stated as difficult. We also suggest an efficient decision algorithm by using the result in [4]. Our result implies that there exists a 4-approximation algorithm for the slider model since the optimal size of the slider model is no larger than two times the optimal size of this model.

## 2 Preliminaries

### 2.1 Definitions and Notations

We redefine Formann and Wagner’s Problem R4 for notational convenience and add the optimization problem.

**Definition 1 (4-position rectangular labeling: 4PRL).** A set  $P$  of points in the plane and a set  $\mathcal{R}$  of axis-parallel closed rectangles such that each point  $p \in P$  is associated with a rectangle  $R_p \in \mathcal{R}$  are given. For positive real  $\sigma$ , let  $\sigma R_p$  be the rectangle whose width and height are  $\sigma$  times  $R_p$ ’s, and let  $\sigma\mathcal{R} = \{\sigma R_p : p \in P\}$  preserving the association.

**Decision problem:** Determine if the rectangles can be placed such that all the rectangles are pairwise disjoint, and each point touches a corner of the associated rectangle.

**Optimization problem:** Find the largest value of  $\sigma$  such that  $(P, \sigma\mathcal{R})$  have a solution.

We denote an input of the problem as  $(P, \mathcal{R})$ . Note that there is an association between  $P$  and  $\mathcal{R}$ , but we hide it for notational convenience. We call the placement of rectangles that satisfy the condition of the problem *valid labeling*. By definition,  $R_p$  has no location information, only the size information. We call the rectangle at the feasible location a (label) *candidate*. We denote and classify candidates as follows. (See Figure 1a.)

**Definition 2 (candidate).**  $\sigma R_p^i$  ( $i \in \{0, \dots, 3\}$ ) denotes the rectangle  $\sigma R_p$  in which the point  $p$  is placed at its northwest, southwest, southeast, and northeast corners. We may omit  $\sigma$  when  $\sigma = 1$ . We call  $R_p^i$  a *candidate* of the point  $p$ . The index  $i$  is chosen like the enumeration of the quadrant. We use  $i$  as  $(i \bmod 4)$ .

We call a candidate  $R_p^i$   $\sigma$ -*dead* if there is a point  $q \in P - \{p\}$  in  $2\sigma R_p^i$ ,  $\sigma$ -*pending* if  $R_p^i$  is not  $\sigma$ -dead and  $\sigma R_p^i$  intersects  $\sigma R_q^j$  that is not  $\sigma$ -dead, and  $\sigma$ -*alive* if  $R_p^i$  is neither  $\sigma$ -dead nor  $\sigma$ -pending.

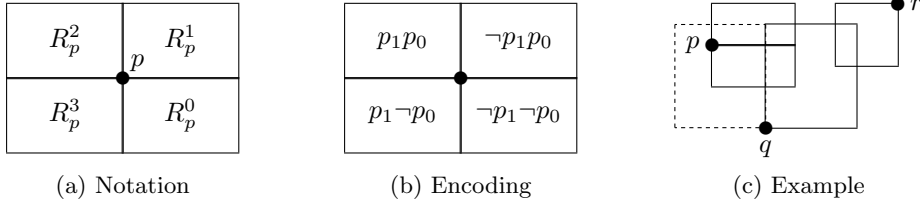


Figure 1: How to notate and encode candidates

## 2.2 Candidate Encoding Rule

Formann and Wagner[3] suggested an encoding rule to formulate the relations between intersecting candidates as a 2-SAT instance, when only two labels are allowed for each point. They assigned one Boolean variable for each point, and added a CNF clause of two variables for each candidate intersection. If one uses this encoding rule, he or she must reduce the number of candidates to two.

We suggest a new encoding rule. First, encode each candidate of a point  $p$  with two Boolean variables  $p_0$  and  $p_1$  as in Figure 1b. Note that two adjacent candidates can be represented together as one variable. For each pair of intersecting candidates, add a CNF clause such that all the corresponding four variables, two variables for each candidate, cannot be true. For each infeasible candidate, which contains another point, add a CNF clause such that the corresponding two variables cannot be true. We call the former clause an *intersection clause* and the latter an *infeasibility clause*.

For example, see Figure 1c. The intersection between  $R_q^1$  and  $R_r^3$  is encoded as  $\neg(\neg q_1 \wedge q_0 \wedge r_1 \wedge \neg r_0) = (q_1 \vee \neg q_0 \vee \neg r_1 \vee r_0)$ . Candidates of  $p$  and  $q$  are encoded as  $(p_1 \vee p_0 \vee q_1 \vee \neg q_0) \wedge (p_1 \vee \neg p_0 \vee q_1 \vee \neg q_0) \wedge (\neg q_1 \vee \neg q_0)$ . The first and second clauses are from the intersecting relation, and the third clause is from the infeasible candidate condition. Note that this expression can be reduced to  $(p_1 \vee \neg q_0) \wedge (\neg q_1 \vee \neg q_0)$ . On the other side, we cannot reduce the number of variables that represents the relation between  $q$  and  $r$ .

It is clear that the encoded expression is satisfiable if and only if there exists a valid labeling. We can solve the decision version of 4PRL by solving the encoded expression. However, we cannot solve it in polynomial time if  $\mathcal{P} \neq \mathcal{NP}$  since the resulting CNF expression may have a clause with three or four variables.

## 3 Simple Optimization Algorithm

In this section, we show the relation between the candidates that can be formulated as a 2-CNF by enlarging the candidates. Using this, we suggest a simple 2-approximation algorithm.

### 3.1 Property of Pending Candidate

**Lemma 3.** *If two pending candidates,  $R_p^i$  and  $R_q^j$ , intersect each other,  $2R_q^j$  intersects  $R_p^{i'}$ , where  $i'$  is  $(i+1)$  or  $(i-1)$ .*

*Proof.* Without loss of generality, assume that  $p$  is located at the origin. Note that  $q$  should not be in the  $(i+2)$ -th quadrant since  $R_q^j$  is not dead but pending.

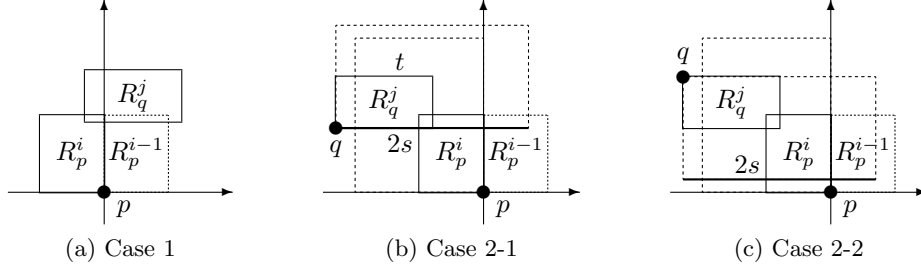


Figure 2: Proof of Lemma 3. The coarse-dotted rectangle is the one which is twice the size.

Otherwise,  $R_q^j$  contains  $p$ . The proof can be divided into two cases.

**Case 1:**  $q$  is in the  $i'$ -th quadrant (see Figure 2a). Note that the sides on the axis are shared by the adjacent candidate. If  $q$  is in the  $(i-1)$ -th quadrant, we can see that  $R_q^j$  intersects  $R_p^{i-1}$  as well as  $R_p^i$ . If  $q$  is in the  $(i+1)$ -th quadrant,  $R_q^j$  intersects  $R_p^{i+1}$ . Since  $R_q^j$  intersects  $R_p^{i'}$ ,  $2R_q^j$  also intersects  $R_p^{i'}$ .

**Case 2:**  $q$  is in the  $i$ -th quadrant. Since  $R_p^i$  is not dead,  $q$  is not in  $2R_p^i$ . Since  $R_q^j$  intersects  $R_p^i$ , either the width or height of  $R_q^j$  is longer than that of  $R_p^i$ . Otherwise,  $q$  is in  $2R_p^i$  or  $R_q^j$  does not intersect  $R_p^i$ . Call such side  $s$  and its parallel side  $t$ . Note that  $s$  is closer to  $p$  than  $t$ . Without loss of generality, assume that  $s$  (and  $t$ ) is parallel to the  $x$ -axis.

**Case 2-1:**  $q$  is on  $s$  (See Figure 2b). One of  $2R_q^j$ 's sides is the two times extended line segment of  $s$ . Call this line segment  $2s$ . Since the length of  $s$  is longer than the width of  $R_p^i$ ,  $2s$  crosses the  $y$ -axis. Note that  $2s$  crosses the line segment, say  $u$ , corresponding to  $R_p^{i'}$ 's side that lies on the  $y$ -axis since the  $y$ -coordinate of  $2s$  is the same as  $s$ . Since  $u$  is also one of the sides of  $R_p^{i'}$  and  $2s$  is a side of  $2R_q^j$ ,  $2R_q^j$  intersects  $R_p^{i'}$ .

**Case 2-2:**  $q$  is on  $t$  (See Figure 2c). One of  $2R_q^j$ 's sides,  $2s$ , that corresponds to  $s$  in  $R_q^j$  crosses the  $y$ -axis, and the  $y$ -coordinate of  $2s$  is positive. That is,  $2s$  crosses  $u$ , which is one of the sides of  $R_p^{i'}$ , as in the proof above. Otherwise,  $p$  is in  $2R_q^j$ , which contradicts the assumption that  $R_q^j$  is not dead, but pending. Therefore,  $2R_q^j$  intersects  $R_p^{i'}$ .  $\square$

Note that the relation of Lemma 3 is symmetric. That is,  $2R_p^i$  intersects either  $R_q^{j-1}$  or  $R_q^{j+1}$  as well as  $2R_q^j$  intersects either  $R_p^{i-1}$  or  $R_p^{i+1}$ . By showing that  $2R_p^{i'}$  and  $2R_q^{j'}$  also intersect, we can see that both  $2R_p^i$  and  $2R_p^{i'}$  intersect both  $2R_q^j$  and  $2R_q^{j'}$ .

**Lemma 4.** *Suppose that  $R_p^i, R_q^j$  are pending and intersect each other. Let  $R_p^{i'}$  be the candidate that  $2R_q^j$  intersects, and  $R_q^{j'}$  be the candidate that  $2R_p^i$  intersects ( $i' \neq i, j' \neq j$ ). Then,  $2R_p^{i'}$  and  $2R_q^{j'}$  intersect each other.*

*Proof.* Due to Lemma 3,  $R_p^{i'} (R_q^{j'})$  exists and is adjacent to  $R_p^i (R_q^j)$ . Let  $s$  be the line segment that  $R_p^i$  and  $R_p^{i'}$  share, and let  $t$  be the line segment that  $R_q^j$  and  $R_q^{j'}$  share. Then,  $2s$  crosses  $2t$ , in both cases of the proof of Lemma 3. Therefore,  $2R_p^{i'}$  and  $2R_q^{j'}$  intersect.  $\square$

**Corollary 5.** *If two pending candidates  $R_p^i$  and  $R_q^j$  intersect each other, both  $2R_p^i$  and  $2R_p^{i'}$  intersect  $2R_q^j$  and  $2R_q^{j'}$ .*

### 3.2 Simple Decision Algorithm

Using Corollary 5 and the fact that two adjacent candidates are encoded as one variable, we can get a simple decision algorithm which guarantees 2-approximation as follows.

**Algorithm A1** ( $P$ : point set,  $\mathcal{R}$ : associated rectangles)

- $e$ : Boolean expression (initially null).
- 1. Generate all the candidates.
- 2. For each dead candidate, delete it and add the infeasibility clause to  $e$ .
- 3. For each intersecting candidate pair,  $R_p^i$  and  $R_q^j$  ( $p \neq q$ ):
  - (a) Find  $i'$  and  $j'$  such that  $2R_q^j$  intersects  $R_p^{i'}$  and  $2R_p^{i'}$  intersects  $R_q^{j'}$ .
  - (b) Add  $(\neg v_p \vee \neg v_q)$  to  $e$  where  $v_p$  is the variable that  $R_p^i$  and  $R_p^{i'}$  are encoded to, and  $v_q$  is the variable that  $R_q^j$  and  $R_q^{j'}$  are encoded to.
- 4. Solve the 2-SAT instance  $e$ . If  $e$  is satisfiable, output the candidates that are true in the truth assignment. Otherwise, output ‘no solution’.

**Lemma 6.** *Algorithm A1 outputs a valid labeling for  $(P, \mathcal{R})$  if there exists a solution for  $(P, 2\mathcal{R})$ .*

*Proof.* It is clear that each clause consists of two variables. Let  $T$  be the truth assignment of a solution of  $(P, 2\mathcal{R})$ . Obviously,  $T$  is also a valid labeling of  $(P, \mathcal{R})$ . We will show that  $T$  satisfies the expression  $e$  in Algorithm A1. Note that the clauses in  $e$  are added at step 2 or step 3. Suppose that  $T$  does not satisfy a clause added at step 2. Then,  $T$  is not a valid labeling of  $(P, 2\mathcal{R})$  since  $2R_p^i$  contains  $q \in P - \{p\}$ . Suppose that  $T$  does not satisfy a clause added at step 3. Then,  $T$  is not the solution of  $(P, 2\mathcal{R})$  since  $T$  selects either  $2R_p^i$  or  $2R_p^{i'}$ , and either  $2R_q^j$  or  $2R_q^{j'}$  as a solution, but all four intersect each other due to Corollary 5. Therefore,  $T$  satisfies  $e$ .  $\square$

One may add some steps for practical reasons without harming the performance guarantee. For example, one may allocate just one variable for points that have two candidates remain to reduce the number of variables in  $e$ . To reduce the number of candidates left, one may add more deletion rules in step 2 such as deleting other candidates if one candidate is alive. For useful deletion rules, see [7] and [8].

### 3.3 Optimization Algorithm

We can construct an optimization algorithm from a decision algorithm. At the optimal value of  $\sigma$ , a candidate touches another candidate or point. Otherwise, we can increase  $\sigma$  by touching another one. Therefore, by calculating possible values for each point pair and running the decision algorithm for each value, we can get an optimal value of  $\sigma$ . Since the number of possible values are in

$O(n^2)$  and we can search the optimal value by binary search, we can get it in  $O(n^2 \log n)$  time and  $O(n^2)$  space if the decision algorithm runs in  $O(n^2)$  time and space.

**Theorem 7.** *A valid labeling of 4PRL with at least half of the optimal size can be computed in  $O(n^2 \log n)$  time with  $O(n^2)$  space.*

*Proof.* By using Algorithm A1, an optimization algorithm guarantees half of the optimal size due to Lemma 6. It is clear that Algorithm A1 runs in  $O(n^2)$  time and space.  $\square$

## 4 Efficient Decision Algorithm

In this section, we present an efficient decision algorithm that is an improvement of Algorithm A1. The idea of the efficient decision algorithm is to represent labeling constraints as a simpler geometric problem, whose exact solution can be gotten efficiently. For ease of explanation, we define a problem.

**Definition 8 (disjoint orthogonal line segments selection: DOLS).** Given  $n$  pairs of orthogonal line segments, select  $n$  line segments, one for each pair, which do not intersect each other.

Our decision algorithm transforms 4PRL to DOLS, solves DOLS, and transforms back to 4PRL. We show that a solution of the transformed DOLS is 2-approximation solution of 4PRL, and show the time and space complexity of DOLS and of the transformation are in  $O(n \log n)$ .

Since the number of possible values of  $\sigma$  is in  $O(n^2)$  and we should sort the values to do binary search, we still get  $O(n^2 \log n)$  time optimization algorithm. But this implies that we can improve the optimization algorithm by finding an efficient way to search the possible values of  $\sigma$ .

### 4.1 Approximate Transformation to DOLS

Let  $s_p^i$  ( $i \in \{0, \dots, 3\}$ ) be the half open line segment ( $i$  that is the intersection of  $R_p^i$  and  $R_p^{i+1}$  except for the point  $p$ ). The following rules transform a problem instance of 4PRL to DOLS.

**Rule 1:** For each point  $p \in P$ , add  $(2s_p^0, 2s_p^2)$  and  $(2s_p^1, 2s_p^3)$  to the instance.

**Rule 2:** For each *dead* candidate  $R_p^i$ , add a pair of line segments such that one intersects only  $2s_p^{i-1}$  and the other intersects only  $2s_p^i$ .

The output of DOLS is transformed back to 4PRL as follows. If the output of DOLS is ‘no solution’, 4PRL outputs ‘no solution’. Otherwise, the output consists of disjoint line segments, one for vertical segments and one for horizontal segments for each point. Then, 4PRL outputs the candidate surrounded by the two line segments. That is, if  $2s_p^{i-1}$  and  $2s_p^i$  are selected by DOLS, 4PRL outputs  $R_p^i$ . Note that only the line segments created by Rule 1 are transformed to 4PRL.

**Lemma 9.** *If  $R_p^i$  and  $R_q^j$  intersect each other but neither  $2s_p^{i-1}$  nor  $2s_p^i$  intersects either  $2s_q^{j-1}$  or  $2s_q^j$ , then either  $R_p^i$  or  $R_q^j$  is dead.*

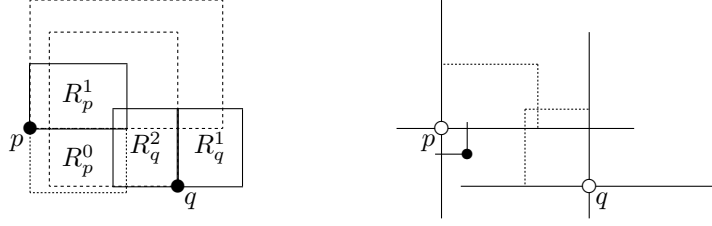


Figure 3: Example of transformation to DOLS. A pair of line segments are added since  $R_p^0$  is dead.

*Proof.* Suppose  $p$  is at the origin. Then  $q$  must be in the  $i$ -th quadrant. If  $q$  is in the  $(i-1)$ -th or the  $(i+1)$ -th quadrant, either  $2s_q^{j-1}$  or  $2s_q^j$  intersects either  $2s_p^{i-1}$  or  $2s_p^i$ . If  $q$  is in the  $(i+2)$ -th quadrant,  $R_q^j$  is dead since  $R_q^j$  contains  $p$  to intersect  $R_p^i$ .

Assume that  $q$  is in the second quadrant and  $R_p^i$  is not dead. Then either  $2s_q^{j-1}$  or  $2s_q^j$  intersects either  $2s_p^{i-1}$  or  $2s_p^i$  due to the similar argument of case 2 of Lemma 3. This contradicts the assumption. Therefore, either  $R_p^i$  or  $R_q^j$  is dead.  $\square$

**Lemma 10.** *A solution of the transformed DOLS problem is a 2-approximate solution of 4PRL.*

*Proof.* First, we will show that the output of DOLS is a valid labeling. It is clear that the labeling contains one rectangle for each point. Assume that there are intersecting rectangles, say  $R_p^i$  and  $R_q^j$ . Since the line segments of the output of DOLS do not intersect each other,  $2s_p^{i-1}$ ,  $2s_p^i$ ,  $2s_q^{j-1}$ , and  $2s_q^j$  do not intersect. Then, due to Lemma 9, either  $R_p^i$  or  $R_q^j$  is dead. Suppose, without loss of generality, that  $R_p^i$  is dead. Due to the line segments added by Rule 2 of the transformation, however, both  $2s_p^{i-1}$  and  $2s_p^i$  cannot be included simultaneously in the solution of DOLS. Therefore,  $R_p^i$  is not dead and neither is  $R_q^j$ . This contradicts the assumption.

Now we will show that DOLS outputs line segments if there exists a solution in  $(P, 2\mathcal{R})$ . Note that the contrapositive is that if DOLS outputs 'no solution', there is no solution in  $(P, 2\mathcal{R})$ . Consider the rectangles in the solution of  $(P, 2\mathcal{R})$  and its sides, especially the sides attached to the points. The sides do not intersect each other. Since the rectangles do not contain a point in  $P$ , the candidate corresponding to the rectangle is not dead. It implies that the line segments added by Rule 2 do not intersect both sides of the rectangles. Therefore, there exist disjoint line segments which are selected one by one from each pair, and they form a solution of DOLS.  $\square$

## 4.2 Solving DOLS

Imai and Asano[4] suggested an efficient data structure that handles an intersection graph of orthogonal line segments, whose vertices are the line segments and whose edges are the intersections between the line segments. They suggested several graph algorithms, including the procedure named INDEPENDENT\_SET that decides whether an independent set of size  $(n - |X|)$  exists where  $n$  is the

number of vertices and  $X$  is a matching.  $X$  is given as the function  $mate(u)$  such that, if  $(u, v)$  is in  $X$ ,  $mate(u) = v$  and  $mate(v) = u$ .

We can solve DOLS, with  $n$  pairs of line segments, using the independent set algorithm. We input all the line segments of DOLS as vertices and input all the pairs as a matching. Obviously, the procedure outputs ‘no independent set’ if and only if there is no solution to the DOLS instance. The output is  $n$  line segments that do not intersect each other. Therefore, the output is a valid solution of DOLS.

**Lemma 11.** *DOLS can be solved in  $O(n \log n)$  time in the RAM model, taking  $O(\min\{m, n \log n\})$  space where  $m$  is the number of intersections.*

*Proof.* By Proposition 6.1 of Imai and Asano[4]. Note that the procedure INDEPENDENT\_SET is not affected whether two line segments in an element of matching intersect each other or not.  $\square$

### 4.3 Classifying Candidates

It is clear that the transformation between DOLS and 4PRL takes linear time and space if dead candidates are identified already. The following lemma shows that, not only the dead candidates, but also pending and alive candidates can be identified in  $O(n \log n)$  time and  $O(n)$  space. This concludes that there is an efficient decision algorithm that takes  $O(n \log n)$  time and space.

**Lemma 12.** *Candidates are classified as dead, pending, and alive in  $O(n \log n)$  time and  $O(n)$  space.*

*Proof.* We use the sweeping with the interval tree [2, 5] with slight modification. It takes  $O(n)$  space,  $O(n \log n)$  time for construction,  $O(\log n)$  time for an insertion or a deletion operation, and  $O(\log n + k)$  time for a query where  $k$  is the number of reported intervals. The most problematic part is the number of reported intersections which may be  $O(n^2)$ . The ideas to solve this are to delete an interval as soon as an intersection is detected and to manage another tree for detecting the intersection.

Dead candidates can be classified by sweeping the points  $P$  and the candidates with twice the size. We can keep  $O(n \log n)$  time by deleting an interval and marking the corresponding candidate as dead when it contains a point, since each rectangle is reported at most once.

After classifying dead candidates, we can classify the remaining candidates as pending or alive by checking intersections among non-dead candidates. We can keep  $O(n \log n)$  time also in the sweeping by maintaining two interval trees,  $T_m$  for marking candidates and  $T_d$  for detecting intersections. The intervals in  $T_m$  are deleted as soon as the interval intersects. So, if an interval remains until the end of the scope of the corresponding candidate, the candidate is alive. The intervals in  $T_d$  are not deleted until the end of its scope. However, when detecting an intersection in  $T_d$ , we compare only the leftmost or rightmost element of the secondary structure. Therefore, we can keep  $O(n \log n)$  time for all the queries.  $\square$

**Theorem 13.** *There is a decision algorithm that outputs a valid labeling for  $(P, \mathcal{R})$  if there exists a solution for  $(P, 2\mathcal{R})$  and runs in  $O(n \log n)$  time and space in the RAM model.*

## Acknowledgements

The authors thank Alexander Wolff for introducing the map labeling area and invaluable comments. Thanks are also in order to Chong-Dae Park, Jae-Hoon Kim, and Chin-Wan Chung for their invaluable discussions.

## References

- [1] Bernard Chazelle and 36 co-authors. The computational geometry impact task force report. In B. Chazelle, J. E. Goodman, and R. Pollack, editors, *Advances in Discrete and Computational Geometry*, volume 223, pages 407–463. American Mathematical Society, Providence, 1999.
- [2] H. Edelsbrunner. Dynamic data structures for orthogonal intersection queries. Technical Report F59, Inst. Informationsverab., Tech. Univ. Graz, Graz, Austria, 1980.
- [3] Michael Formann and Frank Wagner. A packing problem with applications to lettering of maps. In *Proc. 7th Annu. ACM Sympos. Comput. Geom. (SoCG'91)*, pages 281–288, 1991.
- [4] Hiroshi Imai and Takao Asano. Efficient algorithms for geometric graph search problems. *SIAM Journal on Computing*, 15(2):478–494, 1986.
- [5] E. M. McCreight. Efficient algorithms for enumerating intersecting intervals and rectangles. Technical Report CSL-80-9, Xerox Palo Alto Research Center, Palo Alto, CA, 1980.
- [6] Zhongping Qin and Binhai Zhu. A factor-2 approximation for labeling points with maximum sliding labels. In Martti Penttonen and Erik Meineche Schmidt, editors, *Proc. 8th Scandinavian Workshop on Algorithm Theory (SWAT'02)*, volume 2368 of *Lecture Notes in Computer Science*, pages 100–109, Turku, 3–5 July 2002. Springer-Verlag.
- [7] Frank Wagner and Alexander Wolff. A combinatorial framework for map labeling. In Sue H. Whitesides, editor, *Proceedings of the Symposium on Graph Drawing (GD'98)*, volume 1547 of *Lecture Notes in Computer Science*, pages 316–331, Montréal, 13–15 August 1998. Springer-Verlag.
- [8] Frank Wagner, Alexander Wolff, Vikas Kapoor, and Tycho Strijk. Three rules suffice for good label placement. *Algorithmica*, 30(2):334–349, 2001.
- [9] Alexander Wolff and Tycho Strijk. The Map-Labeling Bibliography. <http://i11www.ira.uka.de/map-labeling/bibliography/>, 1996.