Scientia Iranica D (2016) 23(3), 1228–1238

Sharif University of Technology
Scientia Iranica
Transactions D: Computer Science & Engineering and Electrical Engineering
www.scientiairanica.com

Separating bichromatic point sets by two disjoint isothetic rectangles

Z. Moslehi and A. Bagheri*

Department of Computer Engineering and IT, Amirkabir University of Technology, Tehran, Iran.

Received 7 April 2015; received in revised form 12 October 2015; accepted 21 November 2015

KEYWORDS
Algorithm;
Computational geometry;
Separability;
Bichromatic point sets;
Isothetic rectangles.

Abstract. Given a set P of red points and a set Q of blue points in a plane with total size n, we investigate the problem of finding two disjoint isothetic rectangles containing all the points of Q, avoiding any points of P. Such rectangles are called *two separating disjoint isothetic rectangles*. We first compute *two separating disjoint axis-aligned rectangles* in $O(n \log n)$ time. Then, we relax the axis-aligned constraint and report all combinatorially different two separating disjoint isothetic rectangles. To compute these rectangles, we introduce some events by rotating the coordinate system and processing these events. Computing and processing all of the events are done in $O(n^2 \log n)$ time. Thus, our algorithm reports all combinatorially different separating rectangles in $O(n^2 \log n)$ time.

1. Introduction

In the basic separability problem, we are given two colored point sets P and Q in the plane, classified as red and blue points of total size n, and a geometric shape C, as a separator. The goal is to locate C such that all the blue points lie inside C and all the red points lie outside it.

Geometric separability has many applications, such as image analysis, statistics, and other fields in which classification is required [1]. Fundamental classification problems have been extensively studied in machine learning. Also, in many data analysis problems, two (usually disjoint) sets of points P and Q are given and one would like to find patterns which exactly intersect one of them. A systematic study of criteria for selecting the most useful patterns of data has been presented and it has been shown that one of the useful patterns is a rectangle.

There has been a fair amount of work on using different kinds of separators, such as rectangles, squares, circles, etc. The complete separability by a rectangle in any orientation (not only an axis-aligned rectangle) was investigated by van Kreveld et al. [2]. In some cases, complete separability is not possible. Eckstein et al. [3] worked on the problem of finding an axis-aligned rectangle B such that $B \cap P = \phi$ and the cardinality of $B \cap Q$ is maximized. Finding an axis-aligned rectangle B that maximizes the difference between the numbers of points of Q and P inside B, i.e. $||B \cap Q| - |B \cap P||$, has been studied by Liu and Nediak [4].

Finding the largest subset $S \subseteq P \cup Q$ enclosed by the union of two, not necessarily disjoint, isothetic rectangles B_P and B_Q such that $B_Q \cup P = \phi$ and $B_P \cap Q = \phi$ has been considered by Corts et al. [5,6]. Finding two axis-aligned unit squares, S_P and S_Q, with disjoint interiors, such that the number of red points covered by S_P plus the number of blue points covered by S_Q is maximized, has been considered in [7,8].

*. *Corresponding author. Tel.: +98 21 64542742; Fax: +98 21 66495521*
E-mail addresses: zahra_moslehi83@aut.ac.ir (Z. Moslehi); ar_bagheri@aut.ac.ir (A. Bagheri)

Sheikhi et al. [9,10] studied the separability problem by an L-shape separator.

Other variants of the problem, when we have monochromatic point set, have been studied in [11-15].

In this paper, we are given two sets P and Q of red and blue points, respectively, and we find two disjoint isothetic rectangles containing all the points of Q, avoiding any points of P, and report them in $O(n^2 \log n)$ time, if they exist.

The remainder of the paper is organized as follows: Some definitions are given in Section 2. The separating algorithm for finding two disjoint axis-aligned rectangles is introduced in Section 3. Then, in Section 4, the algorithm is extended to relax the axis-aligned constraint. Finally, the conclusion is given in the last section.

2. Definitions

The rectilinear axis-aligned convex hull of Q, $RCH(Q)$, is defined as follows: A quadrant is the intersection of two half-planes whose supporting lines are axis-aligned. We call a quadrant free of Q if its interior does not contain any point of Q. $RCH(Q)$ is defined as follows [16]:

$$RCH(Q) = \Re^2 - \bigcup_{q_u \text{ is a quadrant free of } Q} q_u \qquad (1)$$

Figure 1 shows a rectilinear convex hull. For a fixed orientation, the rectilinear convex hull can be constructed in $O(n\ log\ n)$ time [17].

Definition 1. *We call a point $p \in Q$ an extremal point, if there is no point $q \in Q$ such that $q_x > p_x$ and $q_y > p_y$, where p_x and p_y are the x- and y-coordinates of p in a specific coordinate system, respectively.*

Let $X(Q)$ denote the set of all extremal points of Q. For example, in Figure 2, all the cross points are extremal points. We define a total order $<$, on the extremal points, i.e. $q < p$ if $(q_x < p_x)$ or $(q_x = p_x$ and $q_y > p_y)$. For each pair q_i, $q_{i+1} \in X(Q)$, $1 \leq i < k$,

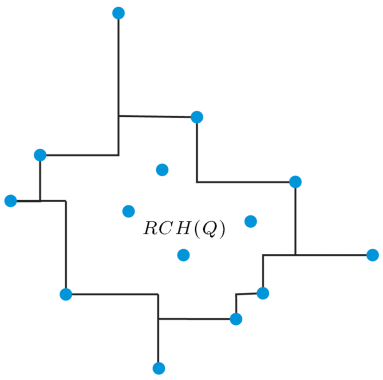

Figure 1. The rectilinear convex hull of Q, $RCH(Q)$.

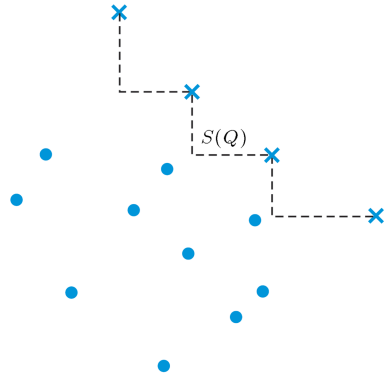

Figure 2. The extremal points of a point set and a staircase $S(Q)$.

where the indices are given in the order $<$, we draw two axis parallel rays, one downward from q_i and the other leftward from q_{i+1}. Then, we get a step between q_i and q_{i+1}, $1 < i < k$, and we denote such a sequence of steps by the staircase $S(Q)$ of Q. Considering Figure 2, such a staircase can be defined by a sequence of free quadrants supporting two consecutive extremal points. The rectilinear convex hull $RCH(Q)$ can be described by four staircases, $S_0(Q)$, $S_{\pi/2}(Q)$, $S_\pi(Q)$, and $S_{3\pi/2}(Q)$ [16], see Figure 3.

Let $R(Q)$ denote the minimum axis-aligned bounding rectangle of the blue point set Q. Consider the red points lying inside $R(Q)$ and outside $RCH(Q)$. Let R_0 denote the minimum bounding rectangle of red points which are within the restricted boundary of $S_0(Q)$ and $R(Q)$. Now, $R_0(P)$ denotes the smallest rectangle that contains R_0 and shares its top-right corner with $R(Q)$; see Figure 3. Similarly, we define $R_{\pi/2}(P)$, $R_\pi(P)$, and $R_{3\pi/2}(P)$ by using $S_{\pi/2}(Q)$, $S_\pi(Q)$, and $S_{3\pi/2}(Q)$, respectively.

If we rotate the coordinate system by θ, then $R(Q)$, $S_0(Q)$, $S_{\pi/2}(Q)$, $S_\pi(Q)$, $S_{3\pi/2}(Q)$ and

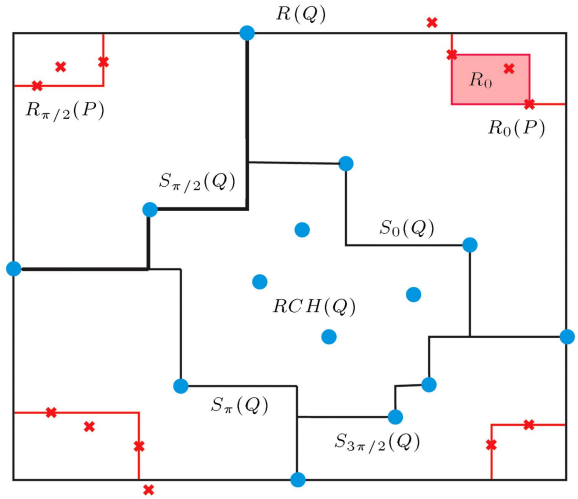

Figure 3. Representation of $R(Q)$, $RCH(Q)$, $R_0(P)$, ..., $R_{3\pi/2}(P)$.

$R_0(P), ..., R_{3\pi/2}(P)$ are computed as before. We compute them according to the previous definition, but their sides would be parallel to the new coordinate system.

We call two staircases $S_\theta(Q)$ and $S_{\theta'}(Q)$ *opposite staircases if* θ and θ' differ by π and *adjacent staircases* if θ and θ' differ by $\pi/2$ [16]. $R_\theta(P)$ and $R_{\theta'}(P)$ are called *adjacent rectangles* if θ and θ' differ by $\pi/2$ and they are called *opposite rectangles* if θ and θ' differ by π. For example, $R_0(P)$ and $R_{\pi/2}(P)$ are adjacent rectangles and $R_0(P)$ and $R_\pi(P)$ are opposite rectangles.

Let $CH(Q)$ denote the convex hull of Q. The convex hull of red points inside $R_0(P)$ is denoted by $CH_0(P)$.

3. Separating bichromatic point sets by two disjoint axis-aligned rectangles

In this section, we introduce an algorithm for separating bichromatic point sets by two disjoint axis-aligned rectangles. Assume that the coordinate system is at a fixed orientation, θ. To avoid many special cases, we assume that no two points (red or blue) have the same x-coordinate or y-coordinate. For example, we ignore the cases in which a red point is on the border of $RCH(Q)$ or on the border of $R(Q)$. These special cases can be handled easily, but make our text complicated; thus, we ignore them. First, we show the necessary and sufficient conditions for separability and then we introduce our algorithm.

3.1. Necessary and sufficient conditions

The separability of bichromatic point set is based on the position of red and blue points. For example, in Figure 4, the two point sets are separable by two disjoint axis-aligned rectangles when the rectilinear convex hull of Q, $RCH(Q)$, is bichromatic. Nevertheless, the red points inside $RCH(Q)$ could not exist freely anywhere inside it. We define S_v (respectively S_h) as the narrowest vertical (respectively horizontal) strip containing all red points in the interior $RCH(Q)$, which is inside $R(Q)$. Some of the necessary conditions for separability are introduced in Lemma 1.

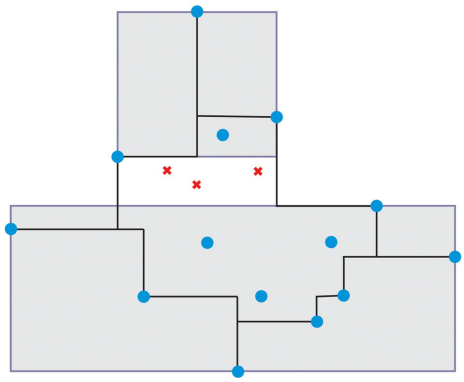

Figure 4. Two disjoint axis-aligned rectangles that separate P (red points) and Q (blue points) even if $RCH(Q)$ is bichromatic.

Lemma 1. *The following conditions are necessary for separability by two disjoint axis-aligned rectangles:*

1. *At least one of* S_v *and* S_h *should be monochromatic;*
2. *None of the corners of red rectangles* $R_0(P), ..., R_{3\pi/2}(P)$ *should be inside* $RCH(Q)$*;*
3. *At least two of the four red rectangles* $R_0(P), ..., R_{3\pi/2}(P)$ *should be empty.*

Proof. First, assume that condition 1 does not hold. If there exist some blue points in the interior of both S_h and S_v, then the two point sets are not separable because we should cover all of the blue points on one side of S_h (respectively S_v) by one rectangle and all of the points on the other side of S_h (respectively S_v) by another rectangle. Thus, the blue points lying in the interior of S_h (respectively S_v) are not covered by any rectangle. This means that we cannot extend the rectangle lying on one side of S_h (respectively S_v) to cover the blue points lying in the interior of S_h (respectively S_v), because S_h (respectively S_v) is defined by red points inside $RCH(Q)$.

Now, assume that both S_h and S_v are empty. This means that $RCH(Q)$ is monochromatic. Also, assume that condition 1 holds, but condition 2 does not hold. So, at least one of the extremal points is inside at least one of the red rectangles $R_0(P), ..., R_{3\pi/2}(P)$. If we cover all of the blue points in that red rectangle with one rectangle, the other blue points outside the red rectangle cannot be covered with another rectangle. So, we do not have two disjoint axis-aligned separating rectangles. As a result, condition 2 is a necessary condition, no matter if S_h and S_v are not empty, which is more general, or even both are empty.

Now, assume that condition 3 does not hold and we have three or four red rectangles. Assume that we have $R_0(P)$, $R_{\pi/2}(P)$, and $R_\pi(P)$. Then, at least three disjoint rectangles are necessary to cover all of the blue points. The first rectangle covers the topmost blue point, the second rectangle covers the leftmost blue point, and the third rectangle covers the rightmost and the bottommost blue points. These rectangles cannot be merged with each other because of the existence of the red rectangles. Thus, condition 3 is also a necessary condition.□

The three conditions of Lemma 1 are necessary but not sufficient for separability. We describe other necessary conditions for separating point sets by two disjoint axis-aligned rectangles in two different cases, as follows:

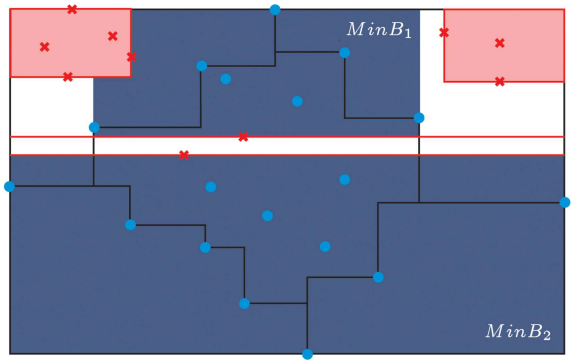

Figure 5. $MinB_1$ and $MinB_2$ when two adjacent red rectangles are empty.

Case A. When two adjacent red rectangles are empty.

First, assume that the first three conditions hold. Thus, we consider one of the monochromatic vertical or horizontal strips S_h or S_v. In Figure 5, S_h is monochromatic and we choose S_h. Considering this figure, one of the covering rectangles covers all of the blue points at one side of S_h and the other covering rectangle covers all of the blue points at the other side of it. Note that three sides of each covering rectangle are defined by $RCH(Q)$ and the fourth side is defined by S_h. We call these rectangles $MinB_1$ and $MinB_2$, which cover all of the blue points. $MinB_1$ and $MinB_2$ are defined similarly if we choose S_v. Lemma 2 shows a necessary condition in this case.

Lemma 2. *A necessary condition for separability of two disjoint axis-aligned rectangles in Case A is:*

(4-A) $((MinB_1 \cap (R_0(P) \cup R_{\pi/2}(P) \cup R_\pi(P) \cup R_{3\pi/2}(P)) = \phi)$ and $((MinB_2 \cap (R_0(P) \cup R_{\pi/2}(P) \cup R_\pi(P) \cup R_{3\pi/2}(P)) = \phi)$

i.e., intersection of $MinB_1$ and $MinB_2$ with red rectangles should be empty.

Proof. Three sides of $MinB_1$ and $MinB_2$ are defined by blue points of $RCH(Q)$. If they intersect red rectangles, we cannot find two rectangles covering the blue points avoiding the red points, because every other two covering rectangles must overlap the boundaries $MinB_1$ (and respectively $MinB_2$), which are defined by $RCH(Q)$.□

In Case A, the conditions of Lemma 1 and Lemma 2 are sufficient conditions for separability, because we can compute two covering rectangles $MinB_1$ and $MinB_2$ and report them as two disjoint axis-aligned separating rectangles. But, since $MinB_1$ and $MinB_2$ intersect the red points on S_h or S_v, we compute two open rectangles by removing the sides which intersect S_h or S_v and consider them as two separating rectangles. As more description, let all the conditions of Lemma 1 and Lemma 2 hold. Now, we consider one of the monochromatic strips S_h or S_v, then we compute $MinB_1$ and $MinB_2$ according to the selected strip. If condition (4-A) holds, these rectangles can define separating rectangles. If this condition does not hold, we consider another monochromatic strip, then we compute $MinB_1$ and $MinB_2$ and report two open rectangles as two disjoint axis-aligned separating rectangles. In Figure 5, $MinB_1$ and $MinB_2$ are not separating rectangles because $MinB_1$ intersects $R_{\pi/2}(P)$ (condition 4-A does not hold) and there is not any other monochromatic strip.

For computing two disjoint axis-aligned separating rectangles, when S_h and S_v are both empty, if both $R_\pi(P)$ and $R_{3\pi/2}(P)$ are empty, we can extend the bottom side of the red rectangle $R_0(P)$ or $R_{\pi/2}(P)$ that is lower than the other rectangle until it intersects $R(Q)$. We use the extended bottom edge as S_h. In the other cases, when the other two adjacent red rectangles are empty, we proceed in a similar manner.

Case B. When two opposite red rectangles are empty.

Considering Figure 6, in this case, each separating rectangle shares at least one corner of $R(Q)$. We define two rectangles $MaxB_1$ and $MaxB_2$ as follows:

Consider two opposite red rectangles. We extend a vertical side of one red rectangle and a horizontal side of the opposite red rectangle until they meet. $MaxB_1$ is defined by the extension of these sides and a corner of $R(Q)$ which does not overlap the red rectangles. Similarly, $MaxB_2$ is defined by the extension of a horizontal side of the first red rectangle and a vertical side of the other red rectangle and another corner of $R(Q)$ which does not overlap the red rectangles. We define rectangle B_1B_2 as the intersection of $MaxB_1$ and $MaxB_2$.

Now, consider $MinB_1$ (or $MinB_2$) to be the minimum bounding rectangle that covers all of the blue points outside B_1B_2 and be a sub-region of $MaxB_1$ (respectively $MaxB_2$); see Figure 6. Let B_1 and B_2

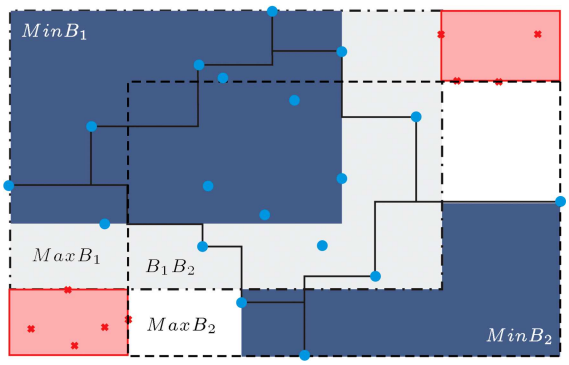

Figure 6. $MinB_1$, $MinB_2$, $MaxB_1$, $MaxB_2$, and B_1B_2 when two opposite red rectangles are empty.

be the two disjoint axis-aligned separating rectangles (if they exist), where B_1 is included in $MaxB_1$ and B_2 is included in $MaxB_2$. Note that the blue points within $MinB_1$ (respectively $MinB_2$) and outside B_1B_2 cannot be covered by B_2 (respectively B_1) and they should be covered only by B_1 (respectively B_2).

According to these definitions, $MinB_1 \subseteq B_1 \subseteq MaxB_1$. Lemma 3 shows a necessary condition in this case.

Lemma 3. *Necessary conditions for separability of two disjoint axis-aligned rectangles in Case B are as follows:*

(4-B-1) $(MinB_1 \cap MinB_2) = \phi$ *(intersection of* $MinB_1$ *and* $MinB_2$ *should be empty)*;

(4-B-2) $(S_v \cap (MinB_1 \cup MinB_2)) = \phi)$ *or* $(S_h \cap (MinB_1 \cup MinB_2)) = \phi)$ *(intersection of* $MinB_1$ *and* $MinB_2$ *with* S_h *or* S_v *should be empty).*

Proof. As we have shown $MinB_1 \subseteq B_1$ and $MinB_2 \subseteq B_2$, if $MinB_1$ intersects $MinB_2$, B_1 intersects B_2 and their intersection implies that two disjoint separating rectangles do not exist. For example, in Figure 6, two separating rectangles do not exist, because $MinB_1$ and $MinB_2$ intersect.

Now, we prove correctness of the second condition. Assume that all the conditions of Lemma 1 and the first condition of this lemma hold. Also, assume that the second condition does not hold. Thus, both S_h and S_v are intersected by $MinB_1$ or $MinB_2$. It is clear that if red points of S_h or S_v are in $MinB_1$ or $MinB_2$, then the two point sets are not separable, since $MinB_1 \subseteq B_1$ and $MinB_2 \subseteq B_2$. Thus, assume that all of the red points in S_h and S_v are outside $MinB_1$ and $MinB_2$. According to these assumptions, at least one of S_h and S_v should be monochromatic and each of them should be intersected by $MinB_1$ or $MinB_2$ or both of them. By some simple tests on their location, we see that one of S_h and S_v should be intersected by $MinB_1$ and another one should be intersected by $MinB_2$; consider Figure 7. We claim that at least one blue point should be in the dashed region in this figure. This is because S_h and S_v include all of the red points inside $RCH(Q)$. If we extend $MinB_1$ and $MinB_2$ to cover all of the blue points, they could not cover the blue points inside the dashed region while avoiding the red points. In other words, because of red points on the segment ab, we cannot extend $MinB_1$ to cover the blue points in the dashed region; and because of red points on the segment bc, we cannot extend $MinB_2$ to cover such blue points. Therefore, there are not any two separating rectangles and condition 4-B-2 is another necessary condition for separability.□

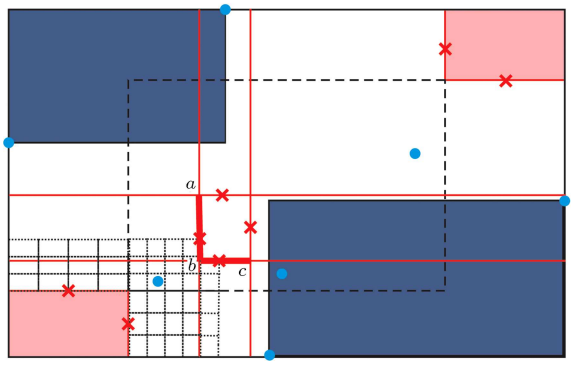

Figure 7. At least one blue point would be in the dashed region if both S_h and S_v intersect $MinB_1$ or $MinB_2$.

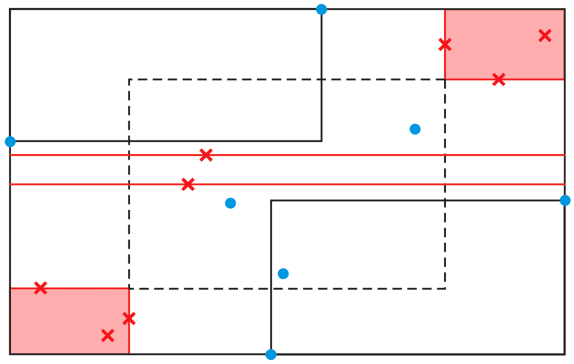

Figure 8. $MinB_1$ and $MinB_2$ when they do not intersect S_h.

Now, we show that all the conditions of Lemma 1 and Lemma 3 are sufficient conditions for separability in Case B.

Observation 1. *If all the conditions of Lemma 1 and Lemma 3 hold, two point sets are separable by two disjoint axis-aligned rectangles.*

Conditions 1 and 4-b-2 imply that there exists a monochromatic vertical or horizontal strip, of which the intersection by $MinB_1$ and $MinB_2$ is empty. Assume that S_h is monochromatic and does not intersect $MinB_1$ and $MinB_2$; consider Figure 8. Since $MinB_1$ and $MinB_2$ share the opposite corners of $R(Q)$, one of them is on one side of S_h and another one is on its another side. Also, recall that all blue points outside B_1B_2 are covered by $MinB_1$ and $MinB_2$. Thus, all of non-covered blue points are inside B_1B_2, of which some are on one side of S_h and another ones are on its another side. Now, we can extend each of $MinB_1$ and $MinB_2$ to cover all of the blue points lying on different sides of S_h and report them as two disjoint axis-aligned separating rectangles. It is sufficient to extend them until they are tangent to red rectangles and S_h. We compute two open rectangles by removing the sides which intersect S_h and red rectangles and consider them as two separating rectangles.

For computing two disjoint axis-aligned separat-

ing rectangles, when S_h and S_v are both empty, if both $R_{\pi/2}(P)$ and $R_{3\pi/2}(P)$ are empty, we can extend the bottom side of $MinB_1$ until it intersects $R(Q)$. If the extended bottom edge does not intersect $MinB_2$, we use the extended bottom edge as S_h. When it intersects $MinB_2$, we must extend the right side of $MinB_1$ until it intersects $R(Q)$ and use it as S_v.

3.2. The algorithm

Algorithm 1 presents the pseudocode of the separability algorithm. In this algorithm, we summarize all sufficient conditions to report the separability rectangles.

Theorem 1. *Deciding on the separability of two disjoint axis-aligned rectangles.*

Proof. At first, all the red points are stored in a 2D range tree data structure in $O(n \log n)$ time [18]. Then, we specify the bottommost, the topmost, the leftmost, and the rightmost points of the blue points in $O(n)$ time. Next, all the extremal points of staircases $S_0(Q), ..., S_{3\pi/2}(Q)$ are stored in four balanced binary search trees $T_1, ..., T_4$, respectively, in $O(n \log n)$ time [18]. $RCH(Q)$ is defined by $T_1, ..., T_4$. Now, we compute time complexity of the other steps of the algorithm.

After the initial step, minimum bounding rectangle $R(Q)$ is computed in constant time. Then, we remove all of the red points outside it in $O(n)$ time. By using data structures $T_1, ..., T_4$, computation of S_h and S_v is done in $O(n \log n)$ time. We can test them if they are monochromatic in $O(n)$ time. We get two extremal points of the minimum and maximum x-coordinates of T_1 and denote them by $minx$ and $maxx$. Then, we pass a vertical ray from $minx$ and a horizontal ray from $maxx$ until they meet. Consider the rectangle that is defined by these rays and $R(Q)$. Among all of the red points in this rectangle, except the points which are in $RCH(Q)$, the closest point to the bottom side of this rectangle and the closest point to the left side of it, define the down side and the left side of $R_0(P)$, respectively. Both of them can be computed in $O(\log^2 n)$ time using the 2D range tree data structure. The right and top sides of $R_0(P)$ are restricted to $R(Q)$. Similarly, we can compute $R_{\pi/2}(P)$, $R_{\pi}(P)$, and $R_{3\pi/2}(P)$.

We can examine whether a corner of the red rectangles is inside $RCH(Q)$ in $O(\log n)$ time by data structures $T_1, ..., T_4$. If two adjacent rectangles are empty, we compute covering rectangles $MinB_1$ and $MinB_2$ in $O(\log n)$ time by $T_1, ..., T_4$ and we test whether they intersect red rectangles in constant time. On the other hand, if two opposite red rectangles are empty, after computation of $MaxB_1$ and $MaxB_2$ in constant time, we compute B_1B_2. Using B_1B_2 and $T_1, ..., T_4$, we can compute $MinB_1$ and $MinB_2$ in $O(\log n)$ time. The intersections of $MinB_1$ and $MinB_2$ with each other and with S_h and S_v are examined in constant time. According to what was mentioned, the total time required to report two disjoint axis-aligned rectangles is $O(n \log n)$.□

4. Separation by two disjoint isothetic rectangles

In the previous section, we introduced an algorithm to report two disjoint axis-aligned separating rectangles. In this section, we show how to separate the bichromatic points by two disjoint isothetic rectangles. Algorithm 1 was extended to relax the constraint of

Input: Bichromatic point sets.
Output: Two disjoint axis-aligned separating rectangles.

1. Compute $R(Q)$, $RCH(Q)$, $R_0(P), ..., R_{3\pi/2}(P)$, S_h, S_v.
2. Check at least one of S_h and S_v to be monochromatic.
3. Check the corners of the red rectangles $R_0(P), ..., R_{3\pi/2}(P)$ not to be inside $RCH(Q)$.
4. Check at least two of the four red rectangles $R_0(P), ..., R_{3\pi/2}(P)$ to be empty.
5. **If** two adjacent red rectangles are empty, **then**
   compute $MinB_1$ and $MinB_2$ and check their intersection with red rectangles to be empty;
6. **else if** two opposite red rectangles are empty, **then**
   a. compute $MinB_1$ and $MinB_2$ and check the intersection of $MinB_1$ and $MinB_2$ with each other to be empty;
   b. check the intersection of $MinB_1$ and $MinB_2$ with S_h or S_v to be empty;
   c. extend each of $MinB_1$ and $MinB_2$ to cover all of the blue points;
7. **end if.**
8. **If** there are not proper $MinB_1$ and $MinB_2$ because of condition vaiolation, **then**
   return null;
9. **else if**
   return $MinB_1$ and $MinB_2$;
10. **end if.**

Algorithm 1. Computing two separating disjoint axis-aligned rectangles.

being axis-aligned. First, we introduce all the possible events that may occur by rotation of the coordinate system. Then, their processing would be explained and finally, our algorithm would be presented.

4.1. Introducing the possible events

By rotating the coordinate system, we have some events as follows:

- **Event 1.** Entrance and exit event that is about red points coming to $R(Q)$ or leaving it: Each red rectangle $R_0(P), ..., R_{3\pi/2}(P)$ may be changed by the entrance (respectively exit) of red points to (respectively from) $R(Q)$. This depends on the position of this red point with respect to $R_0(P), ..., R_{3\pi/2}(P)$. Considering Figure 9, when we rotate $R(Q)$, one red point would enter $R(Q)$ and the separability of two point sets would be damaged.
- **Event 2.** Entrance and exit event that is about red points coming to $RCH(Q)$ or leaving it: When we rotate the coordinate system, one red point may enter $RCH(Q)$ or leave it. So, S_h or S_v may be changed. For example, in Figure 10, when S_h is changed, it would be bichromatic whereas it was monochromatic before the event.
- **Event 3.** The event of changing blue staircases: This event is about the change of extremal points. We know that the intersection of red rectangles and $RCH(Q)$ affects separability. Also, covering rectangles $MinB_1$ and $MinB_2$ in Case A are computed by $RCH(Q)$ sides. Thus, it is necessary to update them and the corresponding data structures after rotating the coordinate system when extremal points have been changed. Figure 11 shows this event. In this figure, one step is removed from $RCH(Q)$ when $RCH(Q)$ is rotated. This event will be explained more in the subsection of processing the events.
- **Event 4.** Changing defining points of red rectangles: When we rotate $R(Q)$, each of the red rectangles would be rotated and each side of them sweeps a part of the plane. Therefore, each of their sides can meet one of the red points and their defining points can be changed. This event will be described in more detail in the next subsection.

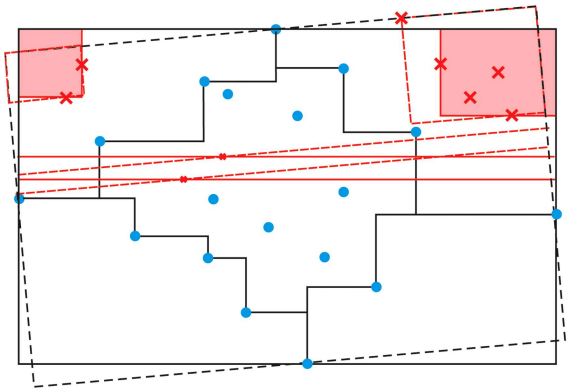

Figure 9. Entrance event of red point coming to $R(Q)$.

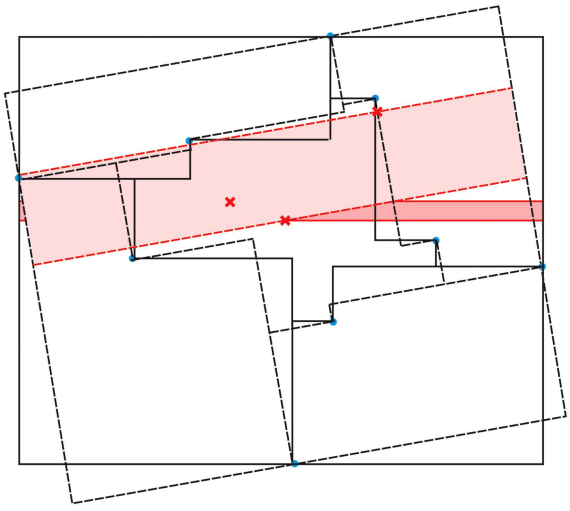

Figure 10. Entrance and exit event of red points coming to $RCH(Q)$ or leaving it.

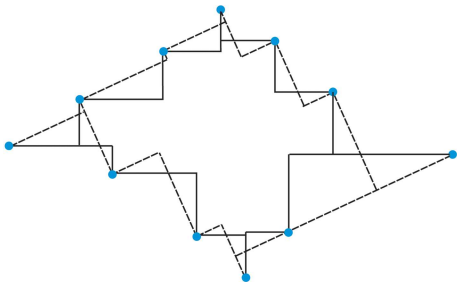

Figure 11. The event of changing blue staircases.

4.2. Processing the events

We describe processing of the events in this subsection.

- **Event 1.** Entrance and exit event that is about red points coming to $R(Q)$ or leaving it: By entering (extracting) a red point into (from) $R(Q)$, it is necessary to update the convex hull of the red points inside the corresponding red rectangle (i.e., one of $CH_0(P), ..., CH_{3\pi/2}(P)$). Event 4 can be predicated by $CH_0(P), ..., CH_{3\pi/2}(P)$. Thus, it may be necessary to update this event in the event queue by changing $CH_0(P), ..., CH_{3\pi/2}(P)$. Also, each of the red rectangles may be changed. The red rectangles are important in steps 3 to 6 of Algorithm 1. Thus, by changing the red rectangles, we must control the separability conditions as follows: We know that one of the necessary conditions for separability is that the corners of the red rectangles should not be inside $RCH(Q)$. Thus, whenever a red rectangle changes, we must update the angles at which its corner would enter $RCH(Q)$ or leave it. In Figure 12(a), the left-bottom corner of $R_0(P)$ would enter $RCH(Q)$ by rotation of the coordinate

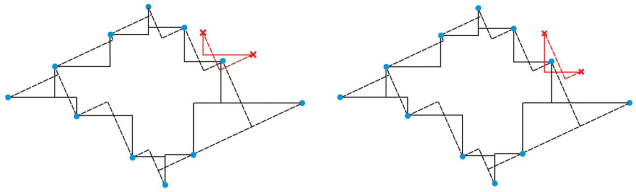

Figure 12. (a) Left-bottom corner of a red rectangle entering $RCH(Q)$ after a rotation. (b) Left-bottom corner of a red rectangle leaving $RCH(Q)$ after a rotation.

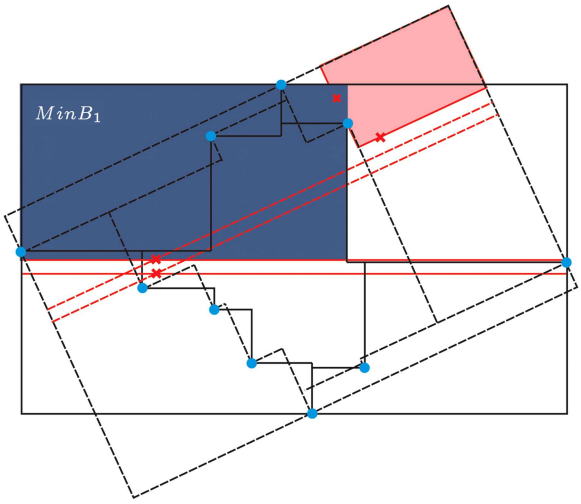

Figure 13. A red rectangle and $MinB_1$ are collinear after a rotation.

system, and in Figure 12(b), this corner would leave $RCH(Q)$.

Now, consider the case in which two adjacent red rectangles are empty. We compute $MinB_1$ and $MinB_2$ according to Algorithm 1 for this angle. If their intersection by red rectangles is not empty, they are not separating rectangles. So, by changing the red rectangles, it is necessary to compute the angle at which the intersection between them would be empty. The intersection between two rectangles would be empty when they are collinear, as shown in Figure 13. Also, if the intersection of $MinB_1$ and $MinB_2$ with red rectangles is empty, they are separating rectangles and we must compute the angle at which they would be collinear with red rectangles. After reaching this angle, they are not separating rectangles, as shown in Figure 14. Thus, by changing the red rectangles or covering rectangles $MinB_1$ and $MinB_2$, it is necessary to compute the angles at which the covering rectangles would be collinear by the red rectangles.

Now, consider the case in which two opposite red rectangles are empty. We know that B_1B_2 can be computed by the sides of red rectangles. Also, $MinB_1$ and $MinB_2$ are computed by all of the blue points outside B_1B_2. So, when each of the red

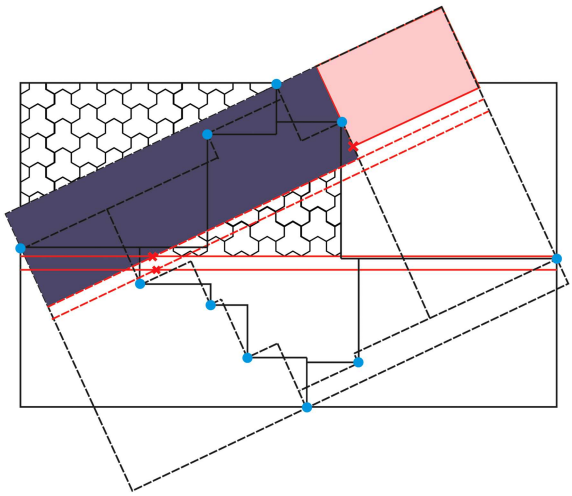

Figure 14. A red rectangle and $MinB_1$ are collinear after a rotation.

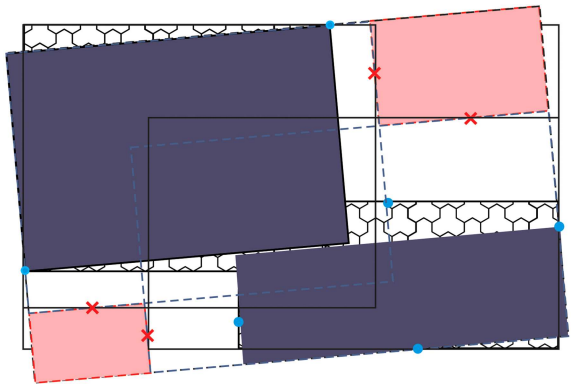

Figure 15. Entrance of a blue point into B_1B_2.

rectangles is changed, it is necessary to compute new B_1B_2 and the angles at which the blue points inside B_1B_2 would be changed. By changing them, $MinB_1$ or $MinB_2$ may be changed. So, the separability of two point sets may be changed. Considering Figure 15, in the initial angle, two point sets are not separable because the intersection between $MinB_1$ and $MinB_2$ is not empty. After that, when a blue point that is outside B_1B_2 comes into B_1B_2, $MinB_2$ would be changed and two point sets would be separable.

- **Event 2.** Entrance and exit event that is about red points coming to $RCH(Q)$ or leaving it: When a red point comes into $RCH(Q)$, it could be outside the red strip S_h or S_v. So, we must compute the new red strip. Also, when a red point comes into $RCH(Q)$ or leaves it, one of the red rectangles would be changed. Thus, this event affects all steps of Algorithm 1. After updating S_h and S_v, it is necessary to check whether the new strip is monochromatic or not. It is clear that when one side of S_h (or S_v) is collinear by one side of $RCH(Q)$, it may be changed to bichromatic from monochromatic or vice versa.

Therefore, we must compute these angles that S_h (or S_v) is changing from bichromatic to monochromatic and vice versa from the current event until the next event. Also, by changing the red rectangles, we do what was mentioned in processing of the first event.

- **Event 3.** Changing of a blue staircase: First, we must update the binary search tree of the changed staircase. Now, we know that all steps of Algorithm 1 are related to $RCH(Q)$. The status of S_h and S_v to be bichromatic or monochromatic, entrance (respectively exit) of the corner of red rectangles to (respectively from) $RCH(Q)$ and computation of $MinB_1$ and $MinB_2$ only depend on $RCH(Q)$. Therefore, we do same as Event 2 for the processing of this event.
- **Event 4.** Change of defining points of red rectangles: Changing defining points of red rectangles is similar to entrance of a red point into $R(Q)$ and affects steps 3 to 6 of Algorithm 1. Thus, we do what was mentioned in the processing of the first event. Now, we compute the total number of these events and their processing time.

Lemma 4. *The number of introduced events is $O(n\alpha(n))$, where $\alpha(n)$ is the inverse of Ackermans function which grows very slowly.*

Proof. van Kreveld et al. showed that the number of Event 1 (entrance and exit event of red points coming to $R(Q)$ or leaving it) is $O(n)$ [2]. Bae et al. showed that the number of Event 3 (change of a blue staircase) is $O(n)$ [16]. Bae's idea can also be used to show the number of Event 2 is $O(n)$. The number of Event 4 is $O(n\alpha(n))$, in which $\alpha(n)$ is the inverse of Ackermans function [9,10].□

Lemma 5. *Processing of each event, when we rotate the coordinate system from 0 to 2π is done in $O(n \log n)$ time.*

Proof. We review the operations that should be done in each occurring event and compute the time required for them.

- **Event 1.** Entrance and exit event of red points coming to $R(Q)$ or leaving it: By entrance of red points to $R(Q)$, we consider the corresponding red rectangle (one of $R_0(P), ..., R_{3\pi/2}(P)$). Then, we compute the convex hull of red points inside the corresponding rectangle and update it in $O(\log n)$ time. When a red point leaves $R(Q)$, the convex hull of the corresponding rectangle can be updated in $O(\log^2 n)$ time [18]. Then, we compute the new red rectangle in constant time. By changing the red points on the sides of the red rectangles, we compute the angles at which the corner of red rectangles would be inside $RCH(Q)$ in $O(n)$ time, using the binary search trees $T_1, ..., T_4$.

  If two adjacent red rectangles are empty, we compute the angles at which new red rectangles and covering rectangles $MinB_1$ and $MinB_2$ would be collinear, in constant time.

  If two opposite red rectangles are empty, we compute the new rectangles $MinB_1$ and $MinB_2$ in $O(\log n)$ time (refer to Theorem 1) and check intersection between each of them and red strips S_h and S_v in constant time. Also, we compute the angles at which the blue points inside B_1B_2 would be changed. These angles can be computed by binary search trees $T_1, ..., T_4$ in $O(n)$ time. So, the total time for the processing of every event of this type is $O(n)$.
- **Event 2.** Entrance and exit event of red points coming to $RCH(Q)$ or leaving it: When this event occurs, we compute the new red strip and check it to be monochromatic in $O(n)$ time. We compute the angles at which the red strip is monochromatic in $O(n \log n)$ time. This time is needed because all of the blue points must be sorted according to their entrance and exit angles of the red strip by rotating the coordinate system. Then, if each of the red rectangles is changed, we do what was mentioned in processing of the first event in $O(n)$ time. Thus, the total time for processing of every event of this type is $O(n \log n)$.
- **Event 3.** Change event of blue staircases: With the occurrence of this event, we must update the corresponding binary search tree which takes $O(\log n)$ time. According to what was mentioned before, we do same as Event 2 for processing of this event. So, the processing time of this event is $O(n \log n)$.
- **Event 4.** Change of defining points of red rectangles: Processing of this event is similar to the first event and it is done in $O(n)$ time. According to what was mentioned, processing time of each event is $O(n \log n)$.□

4.3. The algorithm

In this subsection, we describe the algorithm of computing two disjoint isothetic separating rectangles in all of the orientations. In the first stage, we consider the orientation whose axes are parallel to standard x-axis and y-axis and examine the separability of two point sets by two disjoint axis-aligned rectangles. Thus, in the first stage, we compute $R_0(Q)$, $RCH_0(Q)$, and red rectangles $R_0(P), ..., R_{3\pi/2}(P)$. Furthermore, the convex hull of red points inside each of the red rectangles should be computed. We denote them by $CH_0(P), ..., CH_{3\pi/2}(P)$. Then, we compute all of the events and insert them in the event queue

Input: Blue point set Q and red point set P.
Output: All combinatorially different two separating disjoint isothetic rectangles.

1. Compute $R_0(Q)$, $RCH_0(Q)$, $R_0(P), ..., R_{3\pi/2}(P)$, S_h, S_v.
2. Insert the current state into event queue.
3. Compute the events of types 1 to 4 and insert them into event queue.
4. **For** each event **in** event queue, **do**
   Process it and remove it from the event queue.
5. **End for.**
6. **Return** all combinatorially different two separating disjoint isothetic rectangles.

Algorithm 2. Separating by two disjoint isothetic rectangles.

according to the increasing order of the angles that occur. Computation of these events is done by rotating the coordinate system from 0 to 2π. We also insert the state of current orientation in the beginning of the event queue. Processing of this event is similar to that of the event of type 2. Now, we continuously remove the first event from the event queue and process it, according to what was mentioned in the previous section. Thus, for each pair of consecutive event points, we compute the angles in which there exist two disjoint separating rectangles or such rectangles do not exist.

Algorithm 2 provides the pseudocode of our algorithm for separating bichromatic point sets by two disjoint isothetic rectangles.

Theorem 2. *Realizing separability of two point sets of total size n by two disjoint isothetic rectangles and reporting all of the two separating rectangles are done in* $O(n^2 \log n)$ *time.*

Proof. In the first stage, computation of R_0 (Q), $RCH_0(Q)$, $R0(P), ..., R_{3\pi/2}(P)$, S_h, S_v, $CH_0(P), ..., CH_{3\pi/2}(P)$ is done in $O(n \log n)$ according to what was mentioned in Theorem 1. Then, all the events of type 1 are computed in $O(n \log n)$ time [2]. Also, all the events of types 2 and 3 are computed in $O(n \log n)$ time [16]. In Lemma 3, we saw that the number of these three events is $O(n)$. So, they would be inserted in the event queue in $O(n \log n)$ time. Processing each of them is done in $O(n \log n)$. The number of all the events of type 4 is $O(n\alpha(n))$. Processing each of these events is done in $O(n)$. In fact, $O(\alpha(n))$ is less than 5 for any practical input size n. Thus, processing of all the events is done in $O(n^2 \log n)$ time.

5. Conclusion

We presented an efficient algorithm for reporting the two separating disjoint axis-aligned rectangles of two given point sets P and Q. The algorithm runs in $O(n \log n)$ time. We also presented an $O(n^2 \log n)$ time algorithm for reporting all combinatorially different two separating disjoint isothetic rectangles in all orientations. An interesting problem that can be investigated is finding non-disjoint isothetic rectangles.

References

1. Seara, C. "On geometric separability", Ph.D. Thesis, University Politecnica de Catalunya (2002).
2. Van Kreveld, M., van Lankveld, T. and Veltkamp, R. "Identifying wellcovered minimal bounding rectangles in 2D point data", *25th Eur. Workshop on Comput. Geom.*, Brussels, Belgium, pp. 277-280 (2009).
3. Eckstein, J., Hammer, P., Liu, Y., Nediak, M. and Simeone, B. "The maximum box problem and its application to data analysis", *Comput. Optim. and Appl.*, **23**(3), pp. 85-98 (2002).
4. Liu, Y. and Nediak, M. "Planar case of the maximum box and related problems", *15th Canad. Conf. Comp. Geom.*, Eindhoven, Netherlands, pp. 14-18 (2003).
5. Cortes, C., Diaz-Banez, J.M., Perez-Lantero, P., Seara, C., Urrutia, J. and Ventura, I. "Bichromatic separability with two boxes: A general approach", *J. of Algoritm.*, **64**(2), pp. 79-88 (2009).
6. Cortes, C., Diaz-Banez, J.M. and Urrutia, J. "Finding enclosing boxes with empty intersection", *23rd Eur. Workshop on Comput. Geom.*, Greece, pp. 185-188 (2006).
7. Diaz-Banez, J.M., Seara, C., Sellares, J.A., Urrutia, J. and Ventura, I. "Covering point sets with two convex objects", *21st Eur. Workshop on Comput. Geom.*, Eindhoven, Netherlands (2005).
8. Cabello, S., Diaz-Banez, J.M., Seara, C., Urrutia, J. and Ventura, I. "Covering point sets with two disjoint disks or squares", *Comput. Geom.: Theory and Appl.*, **40**(3), pp. 195-206 (2008).
9. Sheikhi, F., Mohades, A., de Berg, M. and Davoodi, M. "Separating bichromatic point sets by L-shapes", *Comput. Geom.: Theory and Appl.*, **48**(9), pp. 673-687 (2015).
10. Sheikhi, F., de Berg, M., Mohades, A. and Davoodi, M. "Finding monochromatic l-shapes in bichromatic point sets", *22nd Canad. Conf. Comp. Geom.*, Manitoba, Canada, pp. 269-272 (2010).

11. Arkin, E.M., Barequet, G. and Mitchell, J.S.B. "Algorithms for two box covering", *22nd Annu. Symp. on Comput. Geom.*, Arizona, USA, pp. 459-467 (2006).

12. Bespamyatnikh, S. and Segal, M. "Covering a set of points by two axisparallel boxes", *Inf. Process. Lett.*, **75**(3), pp. 95-100 (2000).

13. Datta, A. and Soundaralakshmi, S. "An efficient algorithm for computing the maximum empty rectangle in three dimensions", *Inf. Sci.*, **128**(1), pp. 43-65 (2000).

14. Karmakar, A., Das, S., Nandy, S.C. and Bhattacharya, B.K. "Some variations on constrained minimum enclosing circle problem", *J. of Comb. Opt.*, **25**(2), pp. 176-190 (2013).

15. Mahapatra, P., Goswami, P. and Das, S. "Maximal covering by two isothetic unit squares", *Annual. Canad. Conf. Comp. Geom.*, Montreal, Canada, pp. 169-172 (2008).

16. Bae, S.W., Lee, C., Ahn, H.K., Choi, S. and Chwa, K.Y. "Computing minimum-area rectilinear convex hull and L-shape", *Comput. Geom.: Theory and Appl.*, **42**(9), pp. 903-912 (2009).

17. Ottmann, T., Soisalon-Soininen, E. and Wood, D. "On the definition and computation of rectilinear convex hulls", *Inf. Sci.*, **33**(3), pp. 157-171 (1984).

18. De Berg, M., Cheong, O., van Kreveld, M. and Overmars, M., *Computational Geometry: Algorithms and Applications*, Springer-Verlag, 8th Edn. (2008).

Biographies

Zahra Moslehi received her BSc degree from Yazd University and MSc degree from Amirkabir University of Technology, Tehran, Iran, in 2012, both in Software Engineering. She is currently a PhD candidate in the Department of Electrical and Computer Engineering, Isfahan University of Technology and a visiting graduate researcher in the Computer Science Department of University of Geneva. Her current research interests include computational geometry, machine learning, and pattern recognition.

Alireza Bagheri received his BSc and MSc degrees in Computer Engineering from Sharif University of Technology (SUT), Tehran, Iran. He received his PhD degree in Computer Science from Amirkabir University of Technology (AUT), Tehran, Iran. Currently, he is an Assistant Professor in the Computer Engineering and IT Department at Amirkabir University of Technology (AUT). His research interests include computational geometry, graph Drawing, and graph algorithms.