Largest empty rectangle among a point set

Jeet Chaudhuri,[a] Subhas C. Nandy,[b,*] and Sandip Das [b]

[a] *Alumnus Software Limited, INFINITY, Salt Lake GP, Calcutta 700 091, India*
[b] *Advanced Computing and Microelectronics Unit, Indian Statistical Institute, 203 BT Road, Calcutta 700 108, India*

Abstract

This work generalizes the classical problem of finding the largest empty rectangle among obstacles in 2D. Given a set P of n points, here a maximal empty rectangle (MER) is defined as a rectangle of arbitrary orientation such that each of its four boundaries contain at least one member of P and the interior of the rectangle is empty. We propose a very simple algorithm based on standard data structure to locate a MER of largest area in the plane. The worst-case time complexity of our algorithm is $O(n^3)$. Though the worst-case space complexity is $O(n^2)$, it reserves $O(n \log n)$ space on an average to maintain the required data structure during the execution of the algorithm.

1. Introduction

The problem of recognizing all maximal empty axes-parallel (isothetic) rectangles, commonly known as MER problem, was first introduced in [7]. Given a set P of n points arbitrarily distributed on a 2D plane, a MER is an isothetic empty rectangle which is not contained inside another such rectangle. The objective is to locate all possible MERs. In [7], an algorithm for this problem is proposed with time complexity $O(\min(n^2, R \log n))$, where R denoting the number of reported MERs, may be $O(n^2)$ in the worst case. Later, the time complexity was improved to $O(R + n \log n)$ [1,10]. The algorithms in [2,3] locate the largest empty isothetic rectangle among a point set without

* The preliminary version of this paper appeared in Proc. 19th Conference on Foundation of Software Technology and Theoretical Computer Science, Lecture Notes in Comput. Sci., Vol. 1738, 1999, pp. 34–46.

inspecting all MERs, in time $O(n \log^3 n)$ and $O(n \log^2 n)$, respectively. The MER problem is later generalized among a set of isothetic obstacles [8], and among a set of non-isothetic obstacles [5,9].

In the context of our present work, we need to refer the following problem. Given a set of points P, the location of all/largest empty r-gon whose vertices coincide with the members in P, is studied in [4]. Let $\gamma_r(P)$ denote the number of empty convex r-gons whose vertices coincide with the members in P. The algorithm proposed in [4] runs in $O(\gamma_3(P) + r\gamma_r(P))$ time if $r \geqslant 5$; for $r = 3$ and $r = 4$, it requires $O(\gamma_r(P))$ time. It is also shown that $\gamma_4(P) \geqslant \gamma_3(P) - \binom{n-1}{2}$, which provides a lower bound on the number of empty convex quadrilateral in terms of number of empty triangles. They have also shown that the empty convex polygon having maximum number of sides and vertices coinciding with the points in P can be obtained in $O(\gamma_3(P))$ time. The expected value of $\gamma_3(P)$ is shown to be $O(n^2)$.

This paper outlines a natural generalization of the classical MER problem. Given n points on a 2D plane, a long standing open problem is to locate an empty rectangle of maximum area. Thus the earlier restriction of isotheticity of the MERs is relaxed. This type of problem often arises in different industrial applications where one needs to cut a largest defect-free rectangular piece from a given metal sheet. We adopt a new algorithmic paradigm, called *grid rotation*, to solve this problem. The worst-case time and space complexities of our algorithm are $O(n^3)$ and $O(n^2)$, respectively. But, using a linked list representation of sparse matrices, the space complexity can be reduced to $O(n \log n)$ on an average.

The paper is organized as follows. In Section 2, we describe some important properties of the point set which are helpful for finding the largest MER in arbitrary orientation. We define the concept of prime MER (PMER), which restricts our search space; we will also give a tight combinatorial bound on the number of PMERs. In Sections 3 and 4, we describe our grid rotation technique for identifying the PMERs, and the complexity of our proposed algorithm for this problem. The conclusions on this work and some related open problems are discussed in Section 5.

2. Basic concepts

Let $P = \{p_1, p_2, \ldots, p_n\}$ be a set of n arbitrarily distributed points on a 2D region. Without loss of generality, we may assume that all the points in P lie in the first quadrant of the coordinate system. The coordinate of a point p_i is denoted by (x_i, y_i).

Definition 1. A rectangle (of arbitrary orientation) in the plane is called a MER if it is empty, i.e., not containing any member of P, and no other empty rectangle can enclose it. Thus, each of the four boundaries of a MER must contain at least one point of P.

If any of the boundaries of an empty rectangle does not contain a member of P, then either it is enclosed inside a MER or it is unbounded on that side. In the former case, it is not a MER. In the latter case, such a MER is called unbounded on that particular side. We are interested in locating the largest MER whose each of the four sides is bounded by

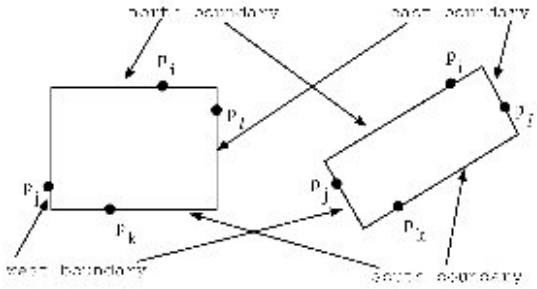

Fig. 1. Definition of a MER.

some point(s) in P. In order to assign some order among the points in four sides of a MER, consider a corner of the MER having maximum y-coordinate. The side (boundary) of the MER incident to that corner and having non-negative slope will be referred as its *north boundary*. The other side adjacent to this corner is the *east boundary*. The *south* and *west* boundaries are defined in an analogous manner. In Fig. 1, p_i, p_j, p_k, and p_ℓ appear on *north, west, south* and *east* boundaries, respectively. Actually, this type of nomenclature is misnomer in the context of non-axis-parallel rectangles, but it will help to explain our method.

Lemma 1. *Given a set of four points in P, if they form a convex quadrilateral, it may generate infinite number of MERs having those four points on its four boundaries, respectively.*

Proof. Let R be an empty convex quadrilateral whose vertices are $p_i, p_j, p_k, p_\ell \in P$. In Fig. 2(a), an example is cited where it can not generate any MER at all. In Fig. 2(b), we demonstrate a situation where MER is possible with those four points on its four boundaries. Let R be such a MER. Now, if we rotate R (as shown in Fig. 2(c)), it will remain empty until we arrive a situation where one of the edges of R contain at least two points of P. It is easy to understand that an infinite number of distinct MERs have been generated during this rotation. □

Lemma 1 tells that, there may exist an infinite number of possible MERs with a set of four points on its four boundaries, respectively. In order to reduce the search space

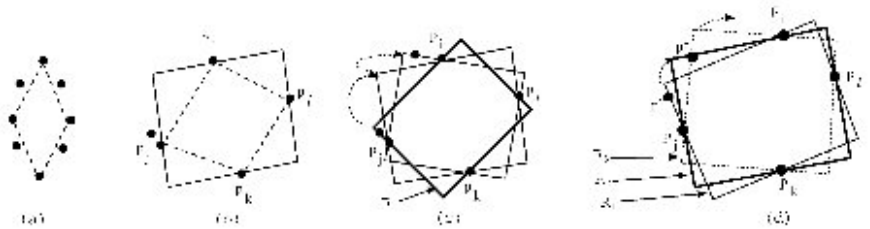

Fig. 2. Proof of Lemma 1.

for locating the largest empty rectangle, we shall introduce the concept of *prime MER* (PMER).

Lemma 2. *Given a fixed angle θ and a pair of points $p_i, p_k \in P$, the MER whose north and south boundaries contain p_i and p_k, respectively, and whose south boundary makes an angle θ with the x-axis, is unique.*

Let us consider a quadruple of four points $\{p_i, p_j, p_k, p_\ell\}$ $(\in P)$ such that they form an empty convex quadrilateral. It is also observed that the rectangle with $\{p_i, p_j, p_k, p_\ell\}$ on its four boundaries, respectively, and the south boundary making an angle θ is a MER. We rotate the rectangle in both clockwise and anti-clockwise directions keeping $\{p_i, p_j, p_k, p_\ell\}$ on its four boundaries until two points appear on any of the boundaries of the rectangle. Let at those instances, the south boundary of the rectangle makes an angle ϕ and ψ with the x-axis. Thus, if we rotate the rectangle beyond the angle ϕ (respectively ψ) in clockwise (respectively anti-clockwise) direction, the rectangle will not remain empty. We define the closed interval $[\phi, \psi]$ as the *maximal interval* with respect to $\{p_i, p_j, p_k, p_\ell\}$.

Lemma 3. *For a given quadruple of four points $\{p_i, p_j, p_k, p_\ell\}$ forming an empty convex quadrilateral, the number of maximal intervals attached with this quadruple may be greater than or equal to* 0.

Proof (by construction). In Fig. 2(a), an example is shown where a quadruple of four points is attached with no maximal interval. In order to show that a quadruple of four points is attached with more than one maximal intervals, let us consider Fig. 2(d). Here, a rectangle (marked as R_1) is shown with p_i, p_j, p_k, p_ℓ on its four boundaries, respectively, and its south boundary makes an angle ϕ_1 with the positive direction of the x-axis. Note that, its west side also touches another point p'. So, the rectangle defined by $\{p_i, p_j, p_k, p_\ell\}$ is empty if its south boundary makes an angle greater than ϕ_1 with the x-axis. Now, we start rotating the rectangle until its north boundary touches a new point p^*. It is marked as R_2 in the same figure; its south boundary makes an angle ψ_1 with the x-axis. If it is further rotated, it will not remain empty (i.e., will contain p^*). Thus $[\phi_1, \psi_1]$ is a maximum interval with respect to $\{p_i, p_j, p_k, p_\ell\}$. We continue rotating, and at some time its west boundary will touch p^* (see the rectangle marked as R_3). If we rotate the rectangle further, it will start to form empty rectangles. Thus it starts creating another maximal interval. □

Definition 2. Consider a set of four points $\{p_i, p_j, p_k, p_\ell\}$ and a maximal interval $[\phi, \psi]$ with respect to this set of four points. A *prime MER* (PMER) is a MER whose area is maximum among the set of MERs whose four boundaries are defined by p_i, p_j, p_k and p_ℓ, respectively, and whose south boundary makes an angle θ with the positive direction of x-axis, where $\theta \in [\phi, \psi]$. We shall refer this PMER using the six tuple $\{p_i, p_j, p_k, p_\ell, \phi, \psi\}$.

Note. Given a set of four points $\{p_i, p_j, p_k, p_\ell\}$ and a maximal interval $[\phi, \psi]$, the corresponding PMER $\{p_i, p_j, p_k, p_\ell, \phi, \psi\}$ can be obtained in $O(1)$ time (see Appendix A).

In [4], it is shown that the number of empty convex quadrilateral $\gamma_4(P)$ in a 2D plane containing n points is $\Omega(n^2)$. But we do not have any knowledge about the worst-case upper bound of $\gamma_4(P)$. Again, each convex quadrilateral may not always produce a PMER, and some times it may produce more than one PMER (see Lemma 3). So, $\gamma_4(P)$ does not give any estimate on the number of PMERs. In the following section, we obtain the worst-case number of PMERs.

2.1. Combinatorial bounds on the number of PMERs

Consider a pair of points $p_i, p_k \in P$. Let L_i and L_k be a pair of parallel lines passing through p_i and p_k, respectively. In order to give an estimate of the worst-case number of PMERs present in the plane, we shall discuss a scheme of generating all possible PMERs with p_i and p_k at their north and south boundaries, respectively, by rotating L_i and L_k around p_i and p_k, respectively, in anti-clockwise direction, and both at same speed. From now onwards, we shall refer the region bounded by L_i and L_k as $corridor_{ik}$. The rotation of L_i and L_k, as mentioned above, will be referred as rotation of $corridor_{ik}$.

Observation 1. *The initial orientation of $corridor_{ik}$, and the schedule of its rotation is decided as follows*:

- *If $x(p_i) > x(p_k)$ and $y(p_i) > y(p_k)$, then initially $corridor_{ik}$ is taken to be horizontal. Its rotation continues until it coincides with the line joining p_i and p_k.*
- *If $x(p_i) < x(p_k)$ and $y(p_i) > y(p_k)$, then initially $corridor_{ik}$ is taken to be horizontal. Its rotation continues until it becomes vertical.*
- *If $x(p_i) < x(p_k)$ and $y(p_i) < y(p_k)$, then initially $corridor_{ik}$ is the line joining p_i and p_k (i.e., a corridor of width zero). The rotation continues until it becomes vertical.*
- *If $x(p_i) > x(p_k)$ and $y(p_i) < y(p_k)$, then no MER with L_i and L_k at its north and south boundaries, respectively, is possible.*

At each position of L_i and L_k, we draw a rectangle whose two parallel sides are aligned with L_i and L_k, respectively, and its diagonal is the line segment $p_i p_k$. This rectangle is defined as the *core* rectangle inside $corridor_{ik}$. Now, if the *core* is non-empty, no MER is possible with L_i and L_k in their present orientation; otherwise a unique MER is possible with its north and south boundaries defined by L_i and L_k, respectively.

We now describe a scheme of generating PMERs by rotating the corridor defined by p_i and p_k. As an initial step, if the *core* is non-empty, we rotate the corridor until it becomes empty. Let the angle of L_i and L_k with the x-axis be ϕ, and the points bounding the west and east sides of the MER be p_j and p_ℓ, respectively. We rotate $corridor_{ik}$ until any of the following situations happen. In Fig. 3, the dotted lines (respectively solid lines) indicate L_i and L_k before (respectively after) the rotation.

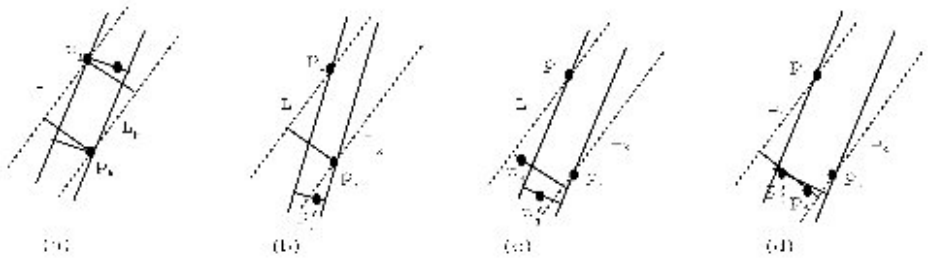

Fig. 3. Rotation of $corridor_{ik}$ and generation of PMER.

(a) A new point appears in the *core* (see Fig. 3(a)).
(b) A new point p'_j appears inside $corridor_{ik}$, and it bounds the west side of a new MER (see Fig. 3(b)), which in turn, will generate another PMER.
(c) The point p_j (which bounds the west side of the current set of MERs) leaves $corridor_{ik}$ (see Fig. 3(c)). Here a new MER emerges. Its west boundary is defined by a point p'_j ($\neq p_j$) inside the corridor.
(d) The set of points inside $corridor_{ik}$ remains same, but a new MER emerges with its west boundary defined by a different point p'_j ($\neq p_j$) inside the corridor (see Fig. 3(d)).
(e) A new point p'_ℓ appears inside $corridor_{ik}$, and it bounds the east side of a new MER as in case (b).
(f) The point p_ℓ (which bounds the east side of the current set of MERs) leaves $corridor_{ik}$. Here a new MER emerges. Its east boundary is defined by a point p'_ℓ ($\neq p_\ell$) inside the corridor.
(g) The set of points inside $corridor_{ik}$ remains same, but a new MER emerges with its east boundary defined by a different point p'_ℓ ($\neq p_\ell$) inside the corridor.

If, after this rotation, the angle of L_i (L_k) with the x-axis is ψ, we report PMER(p_i, p_j, p_k, p_ℓ, ϕ, ψ). In case (a), we continue the rotation of L_i and L_k until *core* becomes empty (similar to the initial step). In all other cases, we continue rotating L_i and L_k with an aim to generate another PMER; here the angle ψ plays the role of ϕ for the next PMER. The process terminates according to Observation 1.

Let m^b_{ik}, m^c_{ik}, m^d_{ik}, m^e_{ik}, m^f_{ik}, and m^g_{ik} be the number of PMERs which have generated due to cases (b)–(g), respectively, during the rotation of $corridor_{ik}$. Now, considering all pairs of points p_i and p_k ($i \neq k$), we have the following lemma.

Lemma 4. (a) $\sum_i \sum_{k\neq i}(m^b_{ik} + m^c_{ik} + m^e_{ik} + m^f_{ik}) = O(n^3)$.
(b) $\sum_i \sum_{k\neq i}(m^d_{ik} + m^g_{ik}) = O(n^3)$.

Proof. Part (a) follows from the fact that, during rotation of the $corridor_{ik}$, a point enters and/or leaves the corridor only once.

In order to prove part (b), we need to consider cases (d) and (g). In case (d) no new point enters or leaves $corridor_{ik}$ during rotation, but the point defining the west boundary of a PMER changes from p_j to p'_j. Note that, the instant of time when such an event is

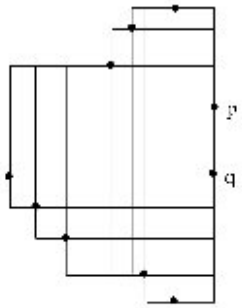

Fig. 4. Proof of Lemma 2(b).

noticed, both p_j to p'_j appeared on the west boundary of a MER. A similar event (i.e., two points p_ℓ and p'_ℓ appear on the east boundary of a MER) is observed when case (g) occurs.

We consider all pairs of points $(p_i, p_k) \in P$ defining the north and south boundaries of the PMERs, and observe the set of PMERs whose west (respectively east) boundary passes through a pair of points (in P). The number of MERs having two distinct points of P on one specific boundary (of those MERs) is $O(n)$ in the worst case. In Fig. 4, a set of MERs is shown with two distinct points p and q ($\in P$) on their east boundary. This amortized analysis shows that the number of times cases (d) and (g) arises during the whole process of generating the PMERs (i.e., considering all pairs of points) may be $O(n^3)$ in the worst case. □

We demonstrate an instance with a total of $n^3/27$ PMERs. Consider three subsets of P, namely A, B and C, each containing $n/3$ points. The distribution of points in each set is as

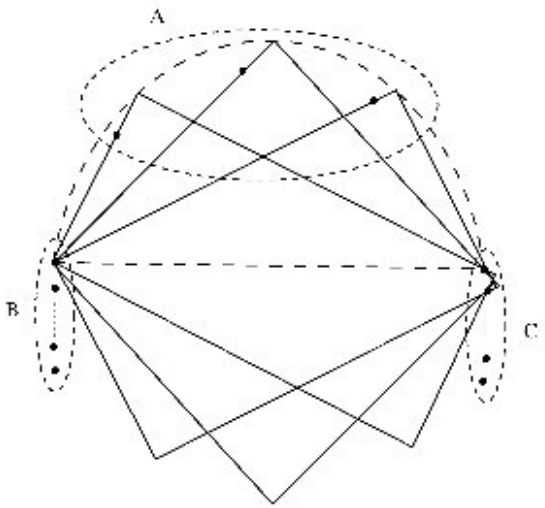

Fig. 5. Justification of the worst-case lower bound on the number of PMERs.

shown in Fig. 5. For each pair of points, one from set B and the other from set C, we may form $n/3$ PMERs with the points in A, as shown in Fig. 5.

Hence we have the following theorem stating the worst-case number of PMERs.

Theorem 1. *The number of PMERs among a set of n points in the plane is* $\Omega(n^3)$ *in the worst case.*

3. Identification of PMERs

In this section, we explain the recognition of all PMERs using a very simple algorithmic technique, called *grid rotation*. Initially, we draw n horizontal lines and n vertical lines through all the members in P. The resulting diagram is a *grid*, but the separation among each pair of horizontal (vertical) lines is not same. For a given point set P, the initial grid diagram is shown in Fig. 6(a). During execution of the algorithm these lines will be rotated, and will no longer remain horizontal/vertical. We shall refer the lines which are initially horizontal, as *red lines*; the lines which are initially vertical, will be referred as *blue lines*. At any instant of time during the execution of algorithm, the angle θ made by each of the red lines with the x-axis, will be referred as the *grid angle* (see Fig. 6(b)).

As mentioned in the proof of Lemma 1, at a particular grid angle, say θ, if a set of four points $\{p_i, p_j, p_k, p_\ell\}$ defines a MER, it will remain valid for some time during the grid rotation, say for an interval $[\theta, \phi]$ of the grid angle. The corresponding entry $\{p_i, p_j, p_k, p_\ell, \theta, *\}$ is created at grid angle θ. We compute the PMER when the grid angle becomes equal to ϕ.

Consider the set of MERs which are *embedded* in the grid, i.e., the set of MERs whose sides are incident to the grid lines. We maintain these MERs in a data structure, called *grid diagram*.

3.1. Data structure

The grid diagram can be maintained using an $n \times n$ matrix, where $n = |P|$. We use two such matrices, called $\mathcal{M}$ and $\mathcal{N}$ during the execution of the algorithm. At any instant of time, each of these matrices stores the set of MERs present on the plane at that particular grid angle. During the grid rotation, when a pair of adjacent red (respectively blue) lines

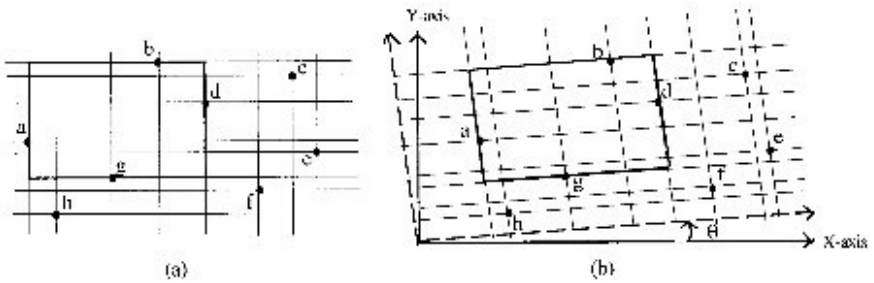

Fig. 6. Demonstration of grid rotation technique using grid diagram.

swap, we use matrix $\mathcal{M}$ (respectively $\mathcal{N}$) to recognize the set of existing MERs that vanishes and the set of MERs that newly emerges.

We sort the points in P in increasing order of their x- and y-coordinates, respectively. Let $P_x = \{p'_1, p'_2, \ldots, p'_n\}$ and $P_y = \{p''_1, p''_2, \ldots, p''_n\}$ denote the same set of points P ordered with respect to their x- and y-coordinates, respectively. Each row (respectively column) of the matrix $\mathcal{M}$ corresponds to an entry in P_y (respectively P_x). Similarly, each row (respectively column) of the matrix $\mathcal{N}$ corresponds to an entry in P_x (respectively P_y). For all points $p \in P$, if $p = p'_k$ and $p = p''_j$ then $\mathcal{M}(j,k)$ and $\mathcal{N}(k,j)$ are set with the value 1. The other entries in $\mathcal{M}$ and $\mathcal{N}$ are initialized to 0.

We now explain the method of representing the *embedded* MERs in the matrix $\mathcal{M}$. The same method will be followed to represent the MERs in the matrix $\mathcal{N}$. As each MER will be present in both the matrices, a pointer, called *self_indicator* attached to each MER in $\mathcal{M}$ points to its own presence in $\mathcal{N}$, and vice versa.

Consider the MER in the grid as shown in Fig. 6(a). It is defined by the points $\{b, a, g, d\} \in P$ at its north, west, south and east sides, respectively. Let $b = p'_{\alpha_n} = p''_{\beta_n}$. In other words, the point b corresponds to the α_nth column and the β_nth row of the matrix $\mathcal{M}$. Similarly, the column (row) indices corresponding to a, g and d are α_w, α_s and α_e (β_w, β_s, and β_e), respectively. Since the objective of our algorithm is to find the largest MER which is bounded by the points of P in its four sides, we store only those MERs in the matrix $\mathcal{M}$ which are bounded by a pair of points (in P) at its north and south boundaries. Each of these MERs is attached with a pair of points which appear on its east and west boundaries, respectively. If such a MER is unbounded to either east or west or both, the corresponding attached point is set to NULL. The reason for storing such an unbounded MER is that, it may eventually be bounded during the rotation. The matrix $\mathcal{N}$ stores the set of MERs whose each member is bounded by a pair of points (in P) at its east and west boundaries; the points in the pair attached to each of these MERs appear on its north and south boundaries, respectively. The MER $\{b, a, g, d\}$, shown in Fig. 6(a), is represented by the point-pair (b, g) (appearing on its north and south boundaries, respectively) in the matrix $\mathcal{M}$; and it is stored in the (β_s, α_n)-th entry of matrix $\mathcal{M}$. The same MER is represented by the point-pair (a, d) (appearing on its east and west boundaries, respectively) in the matrix $\mathcal{N}$; and is stored in the (α_w, β_e)-th entry of the matrix $\mathcal{N}$.

Note that, a MER unbounded in either or both of east and west, is stored in matrix $\mathcal{M}$, but is not stored in matrix $\mathcal{N}$. Similarly, a MER unbounded in either or both of north and south, is stored in matrix $\mathcal{N}$, but is not stored in matrix $\mathcal{M}$.

Observation 2. (i) *Given a fixed grid angle, and a pair of points p_i and p_k, if there exists a MER whose north and south boundaries contain p_i and p_k, respectively, then the points appearing on its east and west boundaries are unique.*

(ii) *Given a fixed grid angle, and a pair of points p_j and p_ℓ, if there exists a MER whose west and east boundaries contain p_j and p_ℓ, respectively, then the points appearing on its north and south boundaries are unique.*

The matrix $\mathcal{M}$ is initialized with the set of all axis-parallel MERs which are present at the grid angle equal to 0. These are obtained by invoking the algorithm presented in [10],

and it requires $O(\mathcal{R} + n \log n)$ time, where $\mathcal{R}$ is the total number of MERs present at that particular grid angle with sides parallel to the coordinate axes.

Note that, for a particular grid angle, the matrix $\mathcal{M}$ is such that

(i) exactly one entry in each row has value 1 and exactly one entry in each column has value 1;
(ii) exactly $\mathcal{R}$ entries have value 2;
(iii) all the entries having value 2 in a row i correspond to the set of MERs with point p_i'' at their south boundaries;
(iv) all the entries having value 2 in a column, say j, correspond to the set of MERs with point p_j' at their north boundaries;
(v) among the non-zero entries in each column, the value 1 appears at the maximum row-index position.

See Fig. 7 for a clear understanding about the matrix $\mathcal{M}$ at a particular grid angle. Similar properties hold for the matrix $\mathcal{N}$ also.

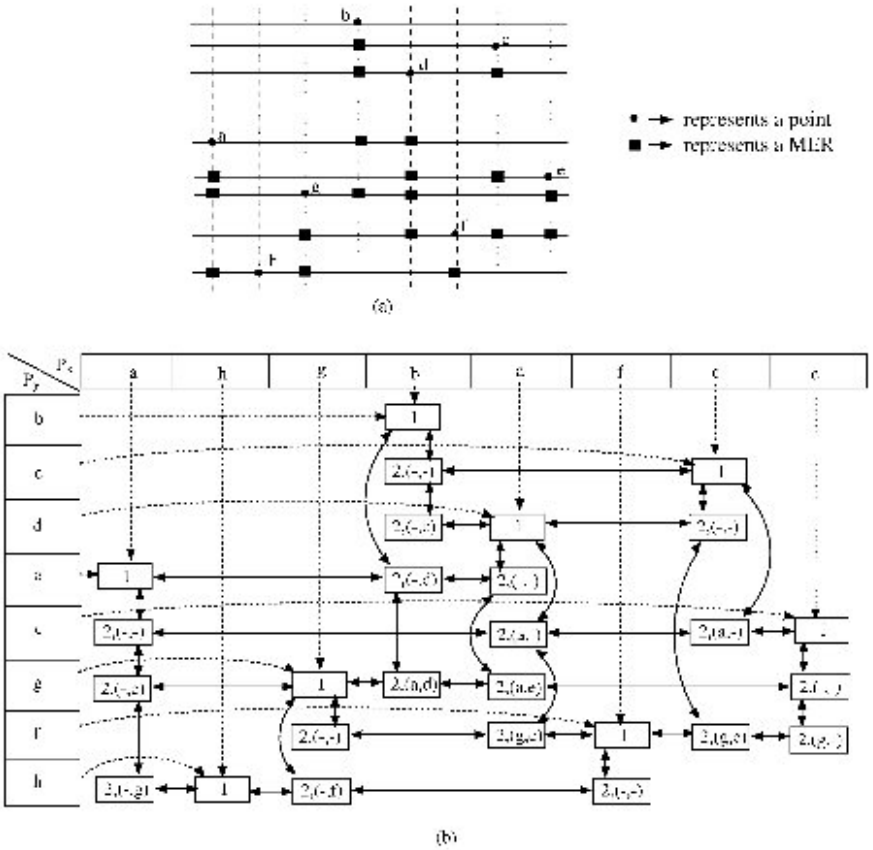

Fig. 7. Sparse matrix representation of matrix $\mathcal{M}$.

Lemma 5 [7]. *At any particular orientation of the grid, the worst- and expected-case number of MERs are $O(n^2)$ and $O(n \log n)$, respectively.*

Thus, we can reduce the space complexity of storing the MERs in the matrices $\mathcal{M}$ and $\mathcal{N}$ by using a suitable linked list representation of sparse matrices [6]. Here, the 0-valued entries in a matrix are absent. During rotation of the grid diagram, the indices of a pair of adjacent rows/columns may be interchanged. The corresponding changes in the matrix entries can be done very easily in our sparse matrix representation.

We use three linear arrays P, P_x, and P_y, where P contains the set of points with respect to their input order. At any instant of time, P_x (respectively P_y) contains the members of P in bottom to top (respectively left to right) order with respect to their corresponding blue and red lines. Initially, when the lines in the two sets are parallel to the y- and x-axis, respectively, the blue lines are ordered from left to right and the red lines are ordered from top to bottom. It is already mentioned that, each element of P_x (respectively P_y) corresponds to a column (respectively row) of the matrix $\mathcal{M}$. Below we explain the linked list representation of matrix $\mathcal{M}$. The matrix $\mathcal{N}$ is represented in the same manner.

Sparse matrix representation of $\mathcal{M}$. Each (non-zero) element of $\mathcal{M}$ consists of the following fields.

(i) A *value* field containing 1 or 2 depending on whether the corresponding entry represents a point in P, or a MER.
(ii) Two pointers $P1$ and $P2$. They indicate two different points in the array P which define the red and blue lines (i.e., the row and column) corresponding to that grid point. If the value field of this entry contains 1, then $P1$ and $P2$ point to the same member of P. If it contains 2, then the points indicated by $P1$ and $P2$ appear at the south and north boundaries of the corresponding MER.
(iii) Two more pointers $P3$ and $P4$. They indicate two different points in P which appear on the east and west boundaries of the MER represented by that element. Again, if the MER represented by an element is unbounded at either east or west or both, the corresponding pointer(s) is (are) set to NULL.
(iv) The grid angle θ where this MER is generated (i.e., inserted in the matrix $\mathcal{M}$) during the grid rotation.
(v) Two pairs of pointers (Q_r^1 and Q_r^2) and (Q_c^1 and Q_c^2). They establish bidirectional links among the neighbors of a matrix element appearing in the same row and in the same column, respectively, as described below.

The (non-zero) matrix elements appearing in a row are connected in a doubly linked list using their Q_r^1 and Q_r^2 pointers.

It is already mentioned that the 1 entry in a column, say j, appears at the maximum row-indexed position. The other non-zero members in that column represent the MERs with point p_j'' at their north boundary. These elements are stored in two doubly linked lists using the pointers Q_c^1 and Q_c^2 as follows:

- *left_list:* the elements such that the MER corresponding to each of them has point on its south boundary to the left of column j in the current orientation of grid diagram, and
- *right_list:* the elements such that the MER corresponding to each of them has point on its south boundaries to the right of column j in the current orientation of grid diagram.
- Q_c^1 and Q_c^2 attached to the element containing '1' in a column, indicate the first element of *left_list* and *right_list*, respectively, in that column.

As mentioned earlier, each element of both the matrices $\mathcal{M}$ and $\mathcal{N}$ is attached with another pointer field, called *self_indicator*. This is used to point its own occurrence in the other matrix, if it is present there.

The sparse matrix data structure ($\mathcal{M}$) for storing the MERs at a particular grid angle is shown in Fig. 7. It needs to mention that, the *left_list* and *right_list* attached to each column are easily understood from the figure.

Each element of P_x and P_y stores the address of the corresponding element in the array P. Apart from that, each entry of P_x and P_y is attached with two sets of three pointers (MQ_1, MQ_2, MQ_3) and (NQ_1, NQ_2, NQ_3). MQ_1 pointer of an element in P_x (P_y) points to the 1 entry in the column (row) of the matrix $\mathcal{M}$ corresponding to that point. The MQ_2 and MQ_3 pointers of an entry in P_y (representing a row of $\mathcal{M}$), point to the address of the left-most and right-most elements in that row. The MQ_2 and MQ_3 pointers of an entry in P_x (representing a column of $\mathcal{M}$), point to the last elements in both the *left_list* and *right_list*, respectively. In Fig. 7(b), MQ_1 pointers of each element in P_x and P_y are shown using dotted lines, but in order to avoid the clumsiness, the MQ_2 and MQ_3 pointers are not shown. The NQ_1, NQ_2, and NQ_3 pointers are set to point the relevant elements in matrix $\mathcal{N}$ in a similar manner.

3.2. Grid rotation

In this subsection, we demonstrate how the grid diagram changes due to the rotation of the grid. During grid rotation a pair of mutually perpendicular lines, passing through each point, are rotated gradually in anti-clockwise direction, and all at the same speed. Let us imagine the MERs *embedded* in the grid to be rotating with the rotation of the grid as shown in Fig. 2(b). As mentioned in the proof of Lemma 1, for a very small rotation of the grid, although the rectangles change in size, their boundary points nevertheless remain same. However, when a pair of adjacent (red/blue) grid lines swap, some rectangles might degenerate, some rectangles might be formed anew, while some may have its bounding vertices changed. These instants are referred as *event points*. At each event point we need to do the following:

- Update the data structure to account for the new set of MERs.
- The rectangles (defined by a specified set of points) which were present in the data structure as MERs prior to the current rotation, and remain MER after the rotation also, do not need any computation.

- For the MERs (defined by a specified set of points) which were present in the data structure, but will not remain present from now onwards, we may need to compute the PMERs as described in Appendix A.

If we gradually rotate the grid by an angle $\pi/2$, we can enumerate all the PMERs that exist in the plane. Our aim is to find the one having maximum area.

3.2.1. Selection of event points

To perform the grid rotation, so that $\mathcal{M}$ and $\mathcal{N}$ are updated properly at each relevant time instant, we need to know the order in which a pair of grid lines of same color swaps. This requires a sorting of the absolute gradients of the lines obtained by joining each pair of points. During grid rotation, we need to stop $O(n^2)$ times when either the *red lines* or the *blue lines* become parallel to any one of those lines. We consider two different sets containing all the lines having positive and negative slopes, respectively. The lines in the first (second) set are sorted in increasing order of the angle θ with the x-axis (y-axis) in anti-clockwise direction. Finally, these two sets are merged to get the ordered set of event points. This needs $O(n^2)$ space for storing the angles of all the lines, and $O(n^2 \log n)$ time for the sorting. But note that, we do not need to store the gradient of all the lines permanently; rather we are satisfied if we get the event points (the angles) in proper order during grid rotation. Below, we describe a method which can generate the event points using $O(n)$ space.

3.2.1.1. A better approach. Let $P^* = \{p_1^*, p_2^*, \ldots, p_n^*\}$ be the set of dual lines corresponding to the points in P. Consider a line $L_{\alpha\beta}$: $y = m_{\alpha\beta}x + c_{\alpha\beta}$ in the primal plane, obtained by joining two points p_α and p_β. In the dual plane, it corresponds to the point $\pi = (m_{\alpha\beta}, c_{\alpha\beta})$, which is the point of intersection of the lines p_α^* and p_β^*. Thus in order to get the lines $L_{\alpha\beta}$ (with $m_{\alpha\beta} > 0$) in increasing order of their gradients, we need to sweep a vertical line $\mathcal{L}$ from $x = 0$ towards right in the dual plane and to report the intersection points among the members of P^* in increasing order of their abscissa. Similarly, the lines $L_{\alpha\beta}$ (with $m_{\alpha\beta} < 0$) are also generated in increasing order of their gradients, by sweeping a vertical line $\mathcal{L}'$ from left to right (starting from $X = -\infty$) in the dual plane and reporting the intersection points among the members of P^* in order of their appearance. The sweeps of $\mathcal{L}$ and $\mathcal{L}'$ are done concurrently. We need to maintain two heaps to obtain the two event points m (> 0) and m' (< 0) (the next point of intersection) to be faced by $\mathcal{L}$ and $\mathcal{L}'$, respectively. Now,

- If $m < |m'|$, then the grid is rotated such that its red lines form an angle $\tan^{-1} m$ with the x-axis of the coordinate system.
- Otherwise, if $m > |m'|$, then we rotate the grid such that its blue lines form an angle $(\tan^{-1} m' - \pi/2)$ with the y-axis of the coordinate system.

The selection of each event point needs $O(\log n)$ time, and the space required for storing the heaps is $O(n)$.

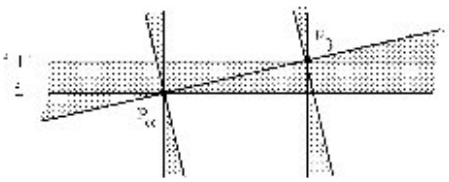

Fig. 8. Empty region—observed while swapping rows i and $i+1$.

3.2.2. Some important properties of grid rotation

Next, we come to the most crucial part of determining the generation of a new set of MERs and consequently updating $\mathcal{M}$ as a pair of grid lines of the same color swap. We need to consider two distinct cases which are caused by the swap of (i) two red lines, and (ii) two blue lines.

We first consider the case when a pair of adjacent rows in $\mathcal{M}$, say i and $i+1$, get swapped due to the swap of a pair of red lines. Let the points attached to these two rows be p_α $(= p''_i)$ and p_β $(= p''_{i+1})$, respectively. After the swap of these two rows in the grid diagram, p_β and p_α will correspond to rows i and $i+1$, respectively. In Fig. 8, the shaded area indicates the region where no point can appear as the line joining the pair of points p_α and p_β has the least gradient among the lines with the unprocessed pairs of points.

Lemma 6. *While processing an event point corresponding to the line $L_{\alpha\beta}$ joining a pair of points (p_α, p_β), p_α to the left of p_β,*

(i) *if the gradient of the line $L_{\alpha\beta}$ is positive then*
   (a) *the MERs whose north or south boundaries contain neither p_α nor p_β, will not be changed with respect to their definition*;
   (b) *the MERs whose south bounding point is p_α, but p_β does not appear on any of its sides, and MERs whose north bounding point is p_β, but p_α does not appear on any of its sides, will not be changed with respect to their definition.*

(ii) *if the gradient of the line $L_{\alpha\beta}$ is negative then*
   (a) *the MERs whose east or west boundaries contain neither p_α nor p_β, will not be changed with respect to their definition*;
   (b) *the MERs whose west bounding point is p_α, but p_β does not appear on any of its sides, and MERs whose east bounding point is p_β, but p_α does not appear on any of its sides, will not be changed with respect to their definition.*

In view of this lemma, we state the following exhaustive set of MERs which may emerge or vanish due to the swap of a pair of rows corresponding to a pair of points p_α and p_β (where p_α is to the left of p_β). A similar set of situations may also arise when a pair of columns swap; we will not mention them explicitly.

All the MERs that vanish due to the swap of two red lines corresponding to p_α and p_β can be classified into one of the following classes.

***A*:** a MER with p_α and p_β on its south and north boundaries, respectively;
***B*:** a set of MERs with p_α and p_β on their south and east boundaries, respectively;

C: a set of MERs with p_β on their south boundary;
D: a set of MERs with p_α and p_β on their west and north boundaries, respectively;
E: a set of MERs with p_α on their north boundary.

The sets of MERs that are generated due to the swap of two red lines corresponding to p_α and p_β can be classified into one of the following classes.

A': a MER with p_α and p_β on its north and south boundaries, respectively;
B': a set of MERs with p_β and p_α on their south and west boundaries, respectively;
C': a set of MERs with p_α on their south boundary. The other boundaries will be newly defined;
D': a set of MERs with p_α and p_β on north and east boundaries, respectively;
E': a set of MERs with p_β on their north boundary. The other boundaries will be newly defined.

Note that, the MER in set A modifies into the MER in set A'.

All the MERs in set B collapse to form members in set C'; in addition, some new MERs may be generated as members in set C', which can be derived by observing few specific members in the set B. Similarly, the members in set C that collapse, result in members of set B' if at all they remain, and conversely every member in set B' results from some member in set C. To be a bit more explicit about the set C of collapsing MERs, ones having their north bounding point to the right of p_α only would still exist and degenerate into members of the set B'. Rest are all destroyed.

Again, the MERs in set D degenerate into MERs in set E'; in addition, some new members in the set E' may also be generated which can be derived by observing few specific members in set D. Similarly, the members in set E that collapse, result in members in set D' if at all they remain, and every member in D' is derived from some member in set E. These observations will guide our actions due to a row swap.

We now highlight the necessary actions when a pair of red lines corresponding to p_α and p_β swap; we also indicate how the creation and deletion of all the MERs are taken care of.

3.3. Updating the grid diagram

Suppose that the line joining (p_α, p_β) is under process. It is having the smallest absolute gradient among the set of unprocessed lines, and its gradient is positive. We now study the effect of rotating the grid so that all red lines become parallel to the line joining (p_α, p_β). Let i and $i+1$ be the rows in $\mathcal{M}$ corresponding to the points p_α and p_β before the rotation; the columns corresponding to p_α and p_β be k and ℓ, respectively. Without loss of generality, assume that p_α is to the left of p_β, i.e., $k < \ell$. After the rotation, p_α and p_β will correspond to rows $i+1$ and i, respectively. But at this stage we like to mention that, the swapping of rows will be done at the end of all other updates on $\mathcal{M}$ which have caused due to the swap of rows i and $i+1$.

During grid rotation, when a new MER emerges it is entered in $\mathcal{M}$, and when an existing MER vanishes, the corresponding PMER is evaluated using the method described

in Appendix A, and the corresponding entry is removed from $\mathcal{M}$. Appropriate updates in the matrix $\mathcal{N}$ are also to be done. From Lemma 6 and succeeding discussions, we have the following results.

Lemma 7. (a) *All the MERs which disappear after processing the line joining p_α and p_β (i.e., due to the swap of two rows corresponding to p_α and p_β) are present in either of the two rows i and $i+1$, and either of the two columns k and ℓ.*

(b) *Similarly, all the MERs that newly emerge after processing the line joining p_α and p_β, will also be inserted in either of the two rows i and $i+1$ and in either of the two columns k and ℓ.*

Below we state the five major steps of processing the line segment (p_α, p_β) with positive gradient. Note that, we do not explicitly create two sets, one for the vanishing MERs (from which PMERs need to be reported), and the other one for the newly emerging MERs (to be inserted in the data structure). The appropriate actions are taken as and when these MERs are encountered.

Step A. The only MER in the set A is the one with p_α and p_β at its south and north boundaries, respectively, before the swap; it is unbounded at its east and west sides. After the rotation, this MER will not exist further. So, $M(i,\ell)$ is to be deleted. But a new MER emerges with p_β and p_α at its south and north boundaries, respectively, which is unbounded in both east and west. This is the only MER in set A'. So, $M(i+1,k)$ is set to 2 (as the rows i and $i+1$ are not yet swapped). Note that, before the deletion of $M(i,\ell)$, it was the first entry in the *left list* of ℓth column. So, in the ℓth column, it is easily reachable in $O(1)$ time. Similarly, after the rotation, $M(i+1,k)$ will be the first entry in the *right_list* of kth column. So, in the kth column, it can also be added in $O(1)$ time. Since this MER is unbounded in east and west sides, the $P3$ and $P4$ pointers attached to it, are set to NULL. No update is necessary in $\mathcal{N}$, since the entry corresponding to $\mathcal{M}(i,\ell)$ was not present in $\mathcal{N}$ as it is unbounded in east and west sides, and $\mathcal{M}(i+1,k)$ will not be stored in $\mathcal{N}$ due to the same reason.

Step B. The set B of MER(s) with p_α and p_β on their south and east boundaries, respectively, before the swap (see Fig. 9(a)), will eventually collapse. So, for each of them the corresponding PMER needs to be computed. This set of MERs are obtained in $\mathcal{M}$ as follows:

B.1 Scan the ith row (corresponding to p_α) from its left end until:
- (i) a rectangle is reached whose east side is not bounded by p_β, or
- (ii) the cell $\mathcal{M}(i,k)$ $(=1)$ is reached.

Each of these entries, excepting the last one, represents a MER in set B.

B.2 Scan the ith row from its right end. The first (non-zero) entry corresponds to the MER in the set A. The second element, if it is not equal to 1 (i.e., the point p_α itself), it corresponds to a MER in set B. In Fig. 9(a), such an entry appears in the column corresponding to the point p_γ.

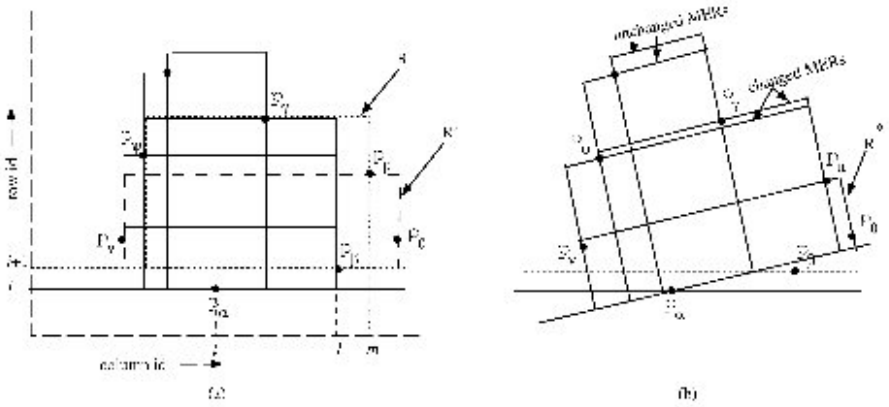

Fig. 9. Illustration of Step B: (a) before rotation, and (b) after rotation.

Thus, location of all elements in set B can be done in $O(|B|)$ time. As mentioned earlier, all the members in set B will contribute a member in the new set of MERs C'. In Step B.3, we explain the necessary modifications that need to be done (in the matrix $\mathcal{M}$) to convert the members in set B to the members in set C', as and when they are encountered. In addition, few more new MERs may appear with p_α at its south boundary after the present grid rotation. In Step B.4, we explain their addition in the matrix $\mathcal{M}$ (as members of C'). The required changes in matrix $\mathcal{N}$ are done in Steps B.5 and B.6.

B.3 Note that, in the $(i+1)$-th row of the matrix $\mathcal{M}$, there exists a MER R with p_γ and p_β at its north and south boundaries, respectively, before the rotation. In Fig. 9(a), R is shown using dotted boundary. If p_μ bounds the east side of R then after the rotation p_μ will bound the east side of all the members in C' (see the changed MERs in Fig. 9(b)). In order to obtain R, one needs to scan the $(i+1)$-th row of the matrix $\mathcal{M}$. Thus, p_μ can be obtained in $O(n)$ time, and all the MERs in C' that are contributed by the elements of B, are generated in $O(|B|)$ time.

B.4 In addition, few new MERs are generated with p_α at its south boundary. Let p_μ corresponds to the column m prior to the present grid rotation. We scan the $(i+1)$-th row (corresponding to p_β) of the matrix $\mathcal{M}$ from the mth column towards right, and generate this new set of MERs as follows:

> Let $R' = M(i+1, m)$ denote a MER with p_μ and p_β at its north and south boundaries, respectively (see Fig. 9(a)). After the rotation, a new MER R^* will be generated with p_μ and p_α at its north and south boundaries, respectively (see Fig. 9(b)). The east side of R^* may be unbounded or bounded by a point (say p_θ) depending on whether R' is unbounded or bounded (by the point p_θ) to its east side. The west side of R^* is either unbounded or is bounded by the point same as that of its preceding entry in C'; this can be settled by observing the rightmost entry in C'. Finally, it is added as the rightmost element in the list attached to the ith row of $\mathcal{M}$; its position in the column of p_μ is just before the element corresponding to the MER R'.

This process of generating new MER is continued by scanning towards the right of the $(i+1)$-th row until a newly generated MER is obtained which is unbounded to its east side. The total time required in this step is $O(n+|C'\backslash B|)$. Here, the $O(n)$ extra time is required for identifying the MER in the $(i+1)$-th row which has p_μ at its north boundary.

B.5 The MERs in set B which are bounded in both sides, are reached in $\mathcal{N}$ using *self_indicator* attached to them. The row index of each MER in C', generated in Step B.3, will remain same as that of the corresponding member in set B; its column index will be changed from ℓ (corresponding to p_β) to m (corresponding to p_μ) after rotation. So, the Q_c^1 and Q_c^2 pointers need to be adjusted to delete this set of MERs from the ℓth column, and to add them in the mth column. Deletion of an entry from a column can easily be done in $O(1)$ time. Regarding the insertion, the members in C' are closer to the '1' entry in the *left_list* of p_μ than its existing elements. So, they can be added in the *left_list* of the '1' entry in the mth column in the reverse order of their generation in $O(|B|)$ time.

B.6 The MERs generated in Step B.4 are considered for insertion in the matrix $\mathcal{N}$ in the reverse order of their generation. If a MER is unbounded in either or both sides, is not inserted in $\mathcal{N}$. Otherwise, let p_θ and p_ν bounds the east and west sides of a MER, say R^*. As, R^* is closest to the '1' entry in the row (respectively column) corresponding to p_ν (respectively p_θ) in $\mathcal{N}$, it can be inserted:
   (i) in the left or the right side of the '1' entry in the row corresponding to p_ν, and
   (ii) as the first element of either *left_list* or *right_list* of the '1' entry in the column corresponding to p_θ.

   Thus, an $O(1)$ time is spent for each MER consider in this step.

Note that, if there exists any MER with p_α at its south boundary but p_β not appearing in any of its sides, it remains unchanged in the data structure during the execution of Step B.

Step C. Next we consider the set of MERs C, each having south boundary containing p_β, but p_α does not appear on any of its boundaries. Due to the rotation, some of them will be truncated by p_α at their west side. The corresponding PMERs are reported and the matrix entries are updated to generate a new set of MERs, referred to as B'. In Fig. 10(a), the possible cases prior to the rotation are shown; the necessary changes after the rotation are demonstrated in Fig. 10(b).

C.1 The entries in row $i+1$ (corresponding to p_β) are considered from extreme right one by one. If the west boundary of such a MER is observed to be to the left of p_α, it will no longer exist after the rotation (in Fig. 10(a), see the MER with p_μ at its north boundary); so the corresponding PMER is reported. Note that, here a new MER is generated (as a member in set B') (see Fig. 10(b)) from the old one by truncating its west side at p_α; the necessary change in the matrix $\mathcal{M}$ is done immediately. The scan continues until a MER is encountered whose west boundary is to the right of p_α, or the cell $\mathcal{M}(i+1,\ell)$ $(=1)$ is reached.

C.2 Next, we check the entries of row $i+1$ from extreme left. All the MERs which appear to the left of $\mathcal{M}(i+1,k)$, i.e., whose north boundaries are defined by points to the

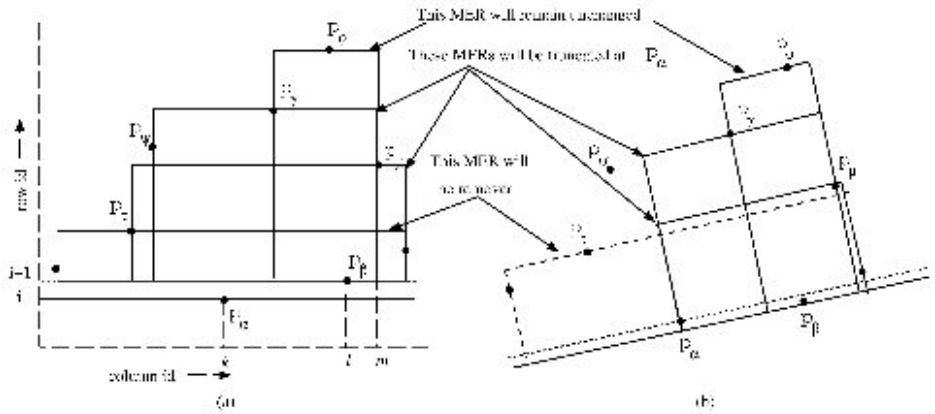

Fig. 10. Illustration of Step C: (a) before rotation and (b) after rotation.

left of p_α, will not remain valid after the current rotation (in Fig. 10(a), see the MER with p_τ at its north boundary). These entries are deleted from the data structure after reporting the corresponding PMERs.

C.3 The search continues along that row past the kth column, to detect the MERs having their west bounding point to the left of p_α. This set of MERs will be truncated by p_α to the west (in Fig. 10(a), see the MER with p_γ at its north boundary) to contribute to the set B'. But prior to the updating of these entries in the data structure, the corresponding PMERs need to be reported. We stop when a MER is encountered whose west boundary is defined by a point to the right of p_α (in Fig. 10(a), see the MER with p_ϕ at the north boundary), or the cell $\mathcal{M}(i+1, \ell)$ $(= 1)$ is reached. Thus $O(|C|)$ time is needed to report all the PMERs corresponding to the members in set C, and $O(|B'|)$ time is needed to generate all the MERs in the set B'.

C.4 The elements in set C are reached using *self_indicator* and are deleted from $\mathcal{N}$. The members in B' are added in the kth row of the matrix $\mathcal{N}$. These newly added entries are closer to the '1' entry in the kth row than all the existing entries in that row. So, they can be added in order of their generation, and in time proportional to their number. The position of these entries in their respective columns will remain same.

Step D. This step is similar to Step B. Here, the set of MERs D having p_α and p_β at their west and north boundaries, respectively, are reached in the matrix $\mathcal{M}$ as follows:

D.1 Traverse the *right_list* of p_β from its beginning until a MER is obtained which is not bounded by p_α at its west side. All these MERs excepting the last one, are the members of the set D. After the current grid rotation, this set of MERs will no longer be bounded by p_α at their west boundaries.

D.2 In addition, the MER corresponding to the second element of the *left_list* of p_β (if it exists) is also bounded by p_α at its west side (as in Step B.2). It is also a member of set D since it will no longer exist after the rotation.

For all these members in D, the corresponding PMERs are reported in $O(|D|)$ time.

D.3 The updated set of MERs E' are obtained by changing the west boundary of each element in D by following the method similar to Step B.3 as described below.

Consider a MER R in the set D. Let its south and east sides are bounded by p_θ and p_η, respectively (see Fig. 11). After rotation, it will not be bounded at the west by p_α. But observe that before the rotation, there exists another MER bounded by p_α on the north and p_θ on the south, and it would also have p_η on its east boundary (in Fig. 11, it is marked as R'). If this MER is bounded by p_δ in the west, then surely the MER R we started with, will have p_δ on its west boundary after the rotation.

The point p_δ is obtained by checking the MERs present in the *right_list* of $M(i,k)$ $(=1$, which corresponds to the point $p_\alpha)$; and it needs $O(n)$ time in the worst case. After the rotation, each element of E' will be bounded at its west by the point p_δ.

D.4 Surely, a new set of MERs will be generated with north side bounded by p_β. For example, see the MER in Fig. 11 whose north and south sides are bounded by p_β and p_δ, respectively. Its west and east bounding points are obtained by scanning the *left_list* and *right_list* of p_α. All such MERs are obtained in a manner similar to Step B.4, and in $O(n + |E' \backslash D|)$ time.

D.5 Each element in set D can be reached in the matrix $\mathcal{N}$ using the *self_indicator* attached to it, and can be deleted in $O(1)$ time. All the MERs in set E' are added in the row corresponding to p_δ of the matrix $\mathcal{N}$. The MERs generated in Step D.3 (i.e., corresponding to the members in D) are closer to the '1' entry in its row (of $\mathcal{N}$) than the existing elements in that row. The position of these entries in their respective columns are obtained as follows:

Consider a member R in E' with p_η and p_δ at its east and west boundaries, respectively, which will be added in the column corresponding to the point p_η. Note that, p_α is above p_η both before and after the rotation. If p_δ is also above p_η then the position of R in the *left_list* of p_η is same as that of the MER with p_η and

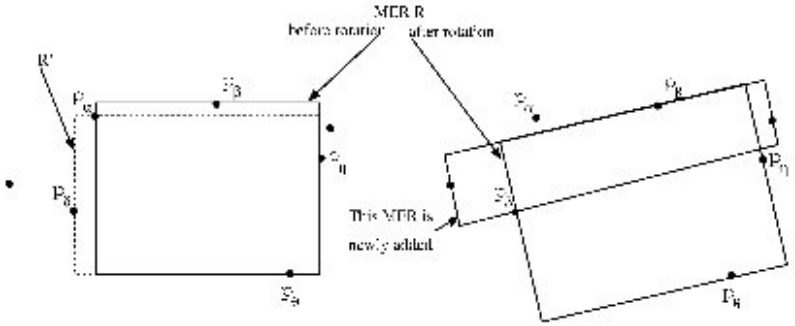

Fig. 11. Illustration of Step D.

p_α at east and west boundary (as a member of D). Otherwise, it will be added as the first element in the *right_list* of p_η.

Thus, each of these MERs can be added in the respective row and column of $\mathcal{N}$ in $O(1)$ time.

Step E. Some of the MERs in set E (i.e., with p_α on their north boundaries), might loose emptiness (as p_β will enter inside those MERs) due to the current rotation. These will either be truncated on the east side by p_β, or simply be destroyed. The new set of MERs D' is obtained using the following procedure which is similar to Step C.

E.1 As in Step C.1, we traverse the *left-list* and *right-list* of the element $M(i,k)$ (the '1' entry corresponding to p_α) separately each from its beginning until:
(i) a MER is encountered whose east side is bounded by a point to the left of p_β, or
(ii) end of the list is reached.

E.2 For each of these MERs, if the point appearing on its south boundary is to the right of p_β, it will no longer exist, and will be deleted from the data structure after reporting the PMER.

E.3 Otherwise, if p_β is to the left of the point bounding the east side of that MER, then the corresponding MER in the set D' is obtained by truncating its east side at the point p_β. So, necessary updates are made in the matrices $\mathcal{M}$ after reporting the PMER. The search continues further downwards in the list to detect MERs having point on their south boundary to the left of p_β.

E.4 The insertion of these MERs in $\mathcal{N}$ are done as in Step C.4.

Thus Step E can be completed in $O(|E| + |D'|)$ time.

Step F. After the computation of the PMERs, and the necessary updates in $\mathcal{M}$ and $\mathcal{N}$, the final step of our algorithm is swapping of rows i and $i+1$ in the matrix $\mathcal{M}$. Surely, this implies the swapping of column i and $i+1$ in the matrix $\mathcal{N}$ also. It can be shown that this task can be completed in $O(n)$ time executing the following substeps.

F.1 p''_i $(= p_\alpha)$ and p''_{i+1} $(= p_\beta)$ are swapped in the array P_y.

F.2 The row-id of p_α and p_β in matrix $\mathcal{M}$ are set to $i+1$ and i, respectively.

F.3 The lists attached to p''_i and p''_{i+1} in matrix $\mathcal{M}$ are swapped. In other words, we traverse row i and $i+1$ simultaneously. If at a particular column, both ith and $(i+1)$-th row has non-zero entries, their Q_c^1 and Q_c^2 pointers are to be adjusted. To be precise, this involves the changing of Q_c^1 and Q_c^2 pointers of two more elements of the same column, one appearing just above $(i+1)$-th row and the other one appearing just below ith row, respectively.

F.4 The column-id of p_α and p_β in matrix $\mathcal{N}$ are set to $i+1$ and i, respectively.

F.5 The lists attached to p''_i and p''_{i+1} in matrix $\mathcal{N}$ are swapped. In other words, we traverse *left-list* (and then *right-list*) of column i and $i+1$ simultaneously. If at a particular row, both ith and $(i+1)$-th column has non-zero entries, their Q_r^1 and Q_r^2 pointers are to be adjusted as in Step F.3.

One crucial point is to be kept in mind while making this update. The steps have to be executed exactly in this order. This is because one step ahead of another may corrupt the values being used by the other. As an example, the updates in row i in Step B depend on the existing entries in row $i+1$. If we execute Step C ahead of B, it is evident that the entries in row $i+1$ get corrupted.

The processing of a line $L_{\alpha\beta}$ (joining p_α and p_β) with negative gradient causes the swap of a pair of rows in the matrix $\mathcal{N}$. The identification of PMERs from matrix $\mathcal{N}$, and updating of $\mathcal{M}$ and $\mathcal{N}$ are done exactly in the same manner as described for swap of a pair of rows in matrix $\mathcal{M}$.

3.4. Correctness of the algorithm

The correctness of our proposed algorithm follows from the following facts:

- We are rotating the grid in one (anti-clockwise) direction.
- Our grid rotation halts at each instance where either a pair of adjacent red grid lines or a pair of blue grid lines swap. These events happen when either red grid lines or blue grid lines are observed to be parallel to the line joining a pair of points in P, and we are considering all the $\binom{n}{2}$ pairs of points in P.
- For all the intermediate grid angles between two consecutive halts of the grid rotation, the identity of each MER (i.e., the quadruple of points defining its boundary) remains same.
- Finally, when a pair of adjacent grid lines swap, we have correctly recognized (i) all the MERs that will no longer exist, and (ii) all the MERs that are newly generated.

The first two facts follow from our processing sequence. The third one follows from Definition 2. In order to prove the fourth one, we consider the swap of a pair of adjacent red grid lines, say i and $i+1$, corresponding to a pair of points, say p_α and p_β, where $p_\alpha = p''_i = p'_k$ and $p_\beta = p''_{i+1} = p'_\ell$. By Lemma 7,

- all the MERs which will no longer exists due to the aforesaid swap, are available in rows i and $i+1$ and columns k and ℓ of the matrix $\mathcal{M}$, and
- all the MERs which are newly generated due to the aforesaid swap, will find their positions in rows i and $i+1$ and columns k and ℓ of the matrix $\mathcal{M}$.

Note that, while swapping of a pair of blue grid lines, the set of MERs which will no longer exist, are to be recognized from matrix $\mathcal{N}$. So, we also need to assure that for all the changes in matrix $\mathcal{M}$, the corresponding changes in matrix $\mathcal{N}$ are done correctly.

In Steps A–E, we have inspected the relevant elements in the aforesaid two rows and two columns to locate the MERs which will no longer exist after the rotation. For each of them, the area of the corresponding PMERs is calculated.

The MERs which emerges after the present rotation, are classified into five categories. In Step A, we have placed the only MER in set A' in its right position in matrix $\mathcal{M}$. Some of the MERs in the other classes are obtained by truncating one of the sides of an already existing MERs (see Steps B.3, C.3, D.3, and E.3). After the necessary modifications, their

positions in matrix $\mathcal{M}$ are adjusted. In addition, some MERs will be formed anew (see Steps B.4 and D.4), which are also identified with reference to some existing MER in $(i+1)$-th row of $\mathcal{M}$. They are also properly positioned in matrix $\mathcal{M}$. The MERs which are present prior to the rotation, and will no longer exist after the rotation, are reached in matrix $\mathcal{N}$ using *self_indicator*, and are deleted. Lemma 7 essentially tells that all the newly generated MERs will be added in the kth and ℓth rows, and ith and $(i+1)$-th column of matrix $\mathcal{N}$. For each of these MERs its position in $\mathcal{N}$ is also successfully obtained.

When a pair of blue lines swap, the relevant MERs can be successfully identified in the matrix $\mathcal{N}$ in an exactly similar manner, and the necessary updates in both the matrices can be done.

4. Complexity analysis

As discussed in the preceding sections, our algorithm consists of two phases, (i) finding the successive event points (grid angles) at which the computation is done during the grid rotation, and (ii) the management of grid diagram during the rotation. The first phase requires $O(n^2 \log n)$ time as shown in Section 3.2.1. Now it remains to analyze the time complexity of the second phase.

The construction of initial grid matrix $\mathcal{M}$ requires $O(n^2)$ time in the worst case. While processing each pair of points (p_α, p_β), it needs to traverse a pair of rows and a pair of columns corresponding to the points p_α and p_β in either of the matrices $\mathcal{M}$ and $\mathcal{N}$ depending on whether the gradient of the line $L_{\alpha\beta}$ (joining p_α and p_β) is positive or negative. The total number of entries encountered during the traversal is $O(n)$ in the worst case. For each MER encountered during the traversal, which will not exist further, the corresponding PMER can be reported in $O(1)$ time. The generation of all new MERs and their insertion in the matrices $\mathcal{M}$ and $\mathcal{N}$ may require $O(n)$ time in total. Finally, the swap of two rows in the matrix $\mathcal{M}$ requires another $O(n)$ time. So, apart from the reporting of the PMERs and generating new MERs, one needs an additional $O(n)$ time for processing each pair of points $p_\alpha, p_\beta \in P$ during the grid rotation. The grid rotation halts $O(n^2)$ time. Thus, we have the final theorem stating the time complexity of our algorithm.

Theorem 2. *The time complexity of our algorithm of recognizing all (and hence the largest) PMER is $O(n^3)$ in the worst case.*

The space required for storing the matrices $\mathcal{M}$ and $\mathcal{N}$ at a particular grid angle is equal to the number of MERs present in the plane at that time. Surely, it may be $O(n^2)$ in the worst case; but it is $O(n \log n)$ on an average [7]. As we are using $O(n)$ space for determining the event points, the average case space complexity of our algorithm is $O(n \log n)$.

5. Conclusion

In this paper, we have considered the problem of locating the largest empty rectangle, of any arbitrary orientation, among a set of points. An algorithmic technique is proposed to solve this problem which inspects all the PMERs present in the plane. One may hope for a faster algorithm without considering all the PMERs.

Acknowledgments

The authors acknowledge Prof. Micha Sharir for suggesting the example in Fig. 5, which proves that the upper bound of the number of PMERs is $\Omega(n^3)$. We thank the referees of the paper for valuable suggestions which improved the presentation of the paper.

Appendix A. Area calculation

Let us consider a set of MERs denoted by six-tuples $\{p_i, p_j, p_k, p_\ell, \phi, \psi\}$. The PMER is a member in this set which has largest area. Let the grid angle corresponding to the PMER be θ. The value of θ is obtained as follows:

At grid angle θ, the lines corresponding to the north and south boundaries of the PMER will be $(y-y_i) = m*(x-x_i)$ and $(y-y_k) = m*(x-x_k)$, respectively, where $m = \tan(\theta)$. The lines at the east and west boundaries will be $(y-y_\ell) = (-1/m)*(x-x_\ell)$ and $(y-y_j) = (-1/m)*(x-x_j)$, respectively. So we have the coordinates of its four corners as follows:

$$\text{north-east:}\quad \left(\frac{m(y_\ell - y_i) + (m^2 x_i + x_\ell)}{m^2+1}, \frac{m(x_\ell - x_i) + (m^2 y_\ell + y_i)}{m^2+1}\right),$$

$$\text{north-west:}\quad \left(\frac{m(y_j - y_i) + (m^2 x_i + x_j)}{m^2+1}, \frac{m(x_j - x_i) + (m^2 y_j + y_i)}{m^2+1}\right),$$

$$\text{south-east:}\quad \left(\frac{m(y_\ell - y_k) + (m^2 x_k + x_\ell)}{m^2+1}, \frac{m(x_\ell - x_k) + (m^2 y_\ell + y_k)}{m^2+1}\right),$$

$$\text{south-west:}\quad \left(\frac{m(y_j - y_k) + (m^2 x_k + x_j)}{m^2+1}, \frac{m(x_j - x_k) + (m^2 y_j + y_k)}{m^2+1}\right).$$

The area of the rectangle is

$$A_\theta = \frac{(m(y_\ell - y_j) + (x_\ell - x_j))((y_k - y_i) + m(x_i - x_k))}{m^2+1}, \quad \phi \leqslant \theta \leqslant \psi.$$

This is a unimodal function in $0 \leqslant \theta \leqslant \pi/2$. Its maximum value can be obtained by solving $\frac{\delta}{\delta\theta} A_\theta = 0$. Now we have

$$\begin{aligned}\frac{\delta}{\delta\theta} A_\theta = {} & \frac{1}{m^2+1}\big[(y_k - y_i)(y_\ell - y_j) + (x_i - x_k)(x_\ell - x_j)\big] \\ & + 2m\big[(x_i - x_k)(y_\ell - y_j) + (y_k - y_i)(x_\ell - x_j)\big] \\ & - m^2\big[(y_\ell - y_j)(y_k - y_i) + (x_\ell - x_j)(x_i - x_k)\big], \quad \text{where } m = \tan(\theta).\end{aligned}$$

The choice of the optimal value of θ^* is as follows:

$$\theta^* = \begin{cases} \phi & \text{if } \frac{\delta}{\delta\theta} A_\theta < 0 \text{ at } \theta = \phi, \psi, \\ \psi & \text{if } \frac{\delta}{\delta\theta} A_\theta > 0 \text{ at } \theta = \phi, \psi, \\ \theta' & \text{if } \frac{\delta}{\delta\theta} A_\theta = 0 \text{ at } \theta = \theta' \in (\phi, \psi). \end{cases}$$

Thus, given the four points and the range of grid angle, θ^* can be computed in constant time.

References

[1] M.J. Atallah, S.R. Kosaraju, An efficient algorithm for maxdominance with applications, Algorithmica 4 (1989) 221–236.
[2] A. Aggarwal, S. Suri, Fast algorithm for computing the largest empty rectangle, in: Proc. 3rd Annual ACM Symp. on Computational Geometry, 1987, pp. 278–290.
[3] B. Chazelle, R.L. Drysdale, D.T. Lee, Computing the largest empty rectangle, SIAM J. Comput. 15 (1986) 300–315.
[4] D.P. Dobkin, H. Edelsbrunner, M.H. Overmars, Searching for empty convex polygons, in: Proc. 4th Annual ACM Symp. on Computational Geometry, 1988, pp. 224–228.
[5] K. Daniels, V. Milenkovic, D. Roth, Finding the largest area axis-parallel rectangle in a polygon, in: Computational Geometry: Theory and Applications, Vol. 7, 1997, pp. 125–148.
[6] D.E. Knuth, Data Structure, in: The Art of Computer Programming, Vol. 1, Addison–Wesley, 1973.
[7] A. Naamad, D.T. Lee, W.L. Hsu, On the maximum empty rectangle problem, Discrete Appl. Math. 8 (1984) 267–277.
[8] S.C. Nandy, B.B. Bhattacharya, S. Ray, Efficient algorithms for identifying all maximal empty rectangles in VLSI layout design, in: Proc. FSTTCS-10, in: Lecture Notes in Comput. Sci., Vol. 437, Springer, 1990, pp. 255–269.
[9] S.C. Nandy, A. Sinha, B.B. Bhattacharya, Location of largest empty rectangle among arbitrary obstacles, in: Proc. FSTTCS-14, in: Lecture Notes in Comput. Sci., Vol. 880, Springer, 1994, pp. 159–170.
[10] M. Orlowski, A new algorithm for largest empty rectangle problem, Algorithmica 5 (1990) 65–73.