Recognition of Largest Empty Orthoconvex Polygon in a Point Set

Subhas C. Nandy* Krishnendu Mukhopadhyaya* Bhargab B. Bhattacharya*

Abstract

An algorithm for computing the maximum area empty isothetic orthoconvex polygon among a set of n points in a rectangular region, is presented. The worst case time and space complexities of the proposed algorithm are $O(n^3)$ and $O(n^2)$ respectively.

1 Introduction

The problem of finding an empty convex k-gon of maximum area or perimeter amidst a point set [4] has several applications. A survey paper [2] has been published very recently, which elaborates several optimization issues related to this problem. In VLSI layout design, document image processing and shape description, isothetic polygons play a major role. A polygon is said to be *isothetic* if its sides are parallel to coordinate axes. The problem of identifying the largest empty isothetic rectangle among a set of points has been studied extensively. The best known algorithm for this problem runs in $O(n \log^2 n)$ time [1]. The same time complexity holds if the obstacles are arbitrary polygons [6, 9]. Recently it is shown that the largest isothetic rectangle inside a simple polygon can be obtained in $O(n \log n)$ time [5].

In isothetic domain, the generalization of this problem is recognizing the largest empty orthoconvex polygon. An isothetic polygon is said to be *orthoconvex* if the intersection of the polygon with a horizontal or a vertical line is a single line segment. Orthoconvexity has importance in robotic visibility, and also its discrete variant appears in other areas like digital geometry [3] and discrete tomography [8]. Datta and Ramkumar [7] proposed algorithms for recognizing largest empty orthoconvex polygon of some specified shapes amidst a 2D point set. These include (i) L-shape, (ii) cross shape, (iii) point visible, and (iv) edge visible polygons. The time complexity of these algorithms are all $O(n^2)$. Another variation in this class of problems is recognizing the largest empty staircase polygon among point and isothetic polygonal obstacles, which can also be solved in $O(n^2)$ time and space [10]. But, to the best of our knowledge, the problem of recognizing an empty orthoconvex polygon of arbitrary shape maximizing area/perimeter is not studied yet. In this paper, we propose an algorithm of recognizing an empty orthoconvex polygon of maximum area, that runs in $O(n^3)$ time using $O(n^2)$ space.

*Indian Statistical Institute, Kolkata - 700 108, India

2 Priliminaries

Let $\mathcal{R}$ be a rectangular region containing a set of n points $P = \{p_1, p_2, \ldots, p_n\}$. We will assume a coordinate system with bottom and left boundaries of $\mathcal{R}$ as the x- and y-axes respectively. The coordinates of a point α are denoted as $(x(\alpha), y(\alpha))$. We assume that the points in P are in general positions, i.e., for every two points p_i and p_j , $x(p_i) \neq x(p_j)$ and $y(p_i) \neq y(p_j)$. Henceforth, we shall use H_i and V_i to denote a horizontal and a vertical line passing through the point p_i.

Definition 1 *An isothetic curve is a rectilinear path consisting of alternately horizontal and vertical line segments. An isothetic curve is a monotonically rising staircase (R-stair) if for every pair of points α and β on the curve, $x(\alpha) \leq x(\beta)$ implies $y(\alpha) \leq y(\beta)$. Similarly, for every pair of points α and β on a monotonically falling staircase (F-stair), we have $x(\alpha) \leq x(\beta)$ implies $y(\alpha) \geq y(\beta)$. An isothetic polygon is a region bounded by a closed isothetic curve.*

Definition 2 *An isothetic polygon Π is said to be orthoconvex if for any horizontal or vertical line ℓ, the intersection of Π with ℓ is a line segment of length greater than or equal to 0. In other words, ℓ either intersects no edge or exactly two edges of Π.*

An orthoconvex polygon is *empty* if it does not contain any member of P in its interior. Our objective is to identify the largest empty orthoconvex polygon in $\mathcal{R}$.

Definition 3 *An empty orthoconvex polygon Π is said to be maximal empty orthoconvex polygon (MEOP) if there exists no other empty orthoconvex polygon Π' that properly encloses Π.*

It is easy to observe that an $MEOP$ is bounded by two R-stairs $R_{t\ell}$ and R_{br}, and two F-stairs F_{tr} and $F_{b\ell}$, where $R_{t\ell}$ spans from the left boundary to the top boundary, R_{br} spans from the bottom boundary to the right boundary, F_{tr} spans from the top boundary to the right boundary and $F_{b\ell}$ spans from the left boundary to the bottom boundary of $\mathcal{R}$, and each concave vertex of these stairs must coincide with a member in P (see Figure 1). It may be observed that an R-stair (or an F-stair) may degenerate to a corner point of $\mathcal{R}$.

The number of maximal empty staircase polygons amidst a point set of size n may be exponential in n; however, the maximum-area empty staircase polygon can be computed in $O(n^2)$ time [10]. Since a maximal empty staircase polygon is an $MEOP$ as well, the same

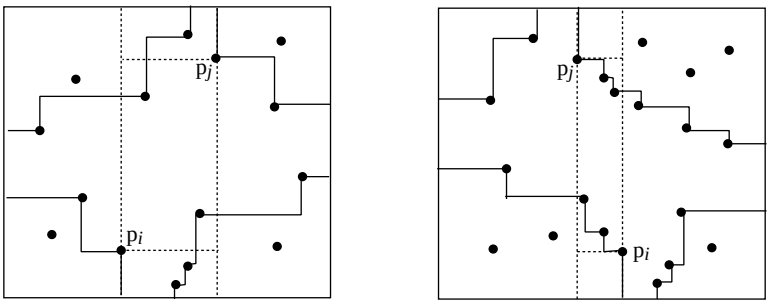

Figure 1: Orthoconvex polygons

combinatorial explosion holds for $MEOP$ also. We now present a polynomial time algorithm for computing the maximum-area $MEOP$.

3 Algorithm

We shall consider all possible pairs of points $p_i, p_j \in P$, and identify the maximum area $MEOP$ with $p_i \in F_{b\ell}$ as the closest point of the bottom boundary of $\mathcal{R}$, and $p_j \in F_{tr}$ as the closest point of the top boundary of $\mathcal{R}$. The points p_i and p_j are said to be the bottom-pivot and top-pivot respectively, and the corresponding $MEOP$ is denoted by $MEOP(p_i, p_j)$. We will use S to denote the vertical slab bounded by V_i and V_j. The projections of a point $p_k \in S$ on V_i, V_j, H_i and H_j are denoted by q_k, q'_k, r_k and r'_k respectively. We now separately consider two cases: (i) $x(p_i) < x(p_j)$, and (ii) $x(p_i) > x(p_j)$.

In Case (i), the lines V_i and V_j split the point set P into three parts, P_1, P_2 and P_3, where P_1 and P_3 are the set of points to the left of V_i and to the right of V_j respectively, and the points in P_2 lie inside the vertical slab S. If V_i hits the top and bottom boundaries of $\mathcal{R}$ at t_1 and b_1 respectively, and V_j hits the top and bottom boundaries of $\mathcal{R}$ at t_2 and b_2 respectively. Now, the portion of the $MEOP$ inside the vertical slab S, denoted by $M_2(b_1, t_2)$, is an empty staircase polygon with diagonally opposite corners b_1 and t_2 among the points in P_2. The two stairs of $M_2(b_1, t_2)$ are parts of the rising stairs R_{br} and R_{tl} respectively. If R_{tl} hits V_i at q_α, then the portion of the $MEOP$ to the left of V_i, denoted by $M_1(q_\alpha)$, is an empty edge-visible polygon with base $[p_i, q_\alpha]$ among the points in P_1 such that every point inside the polygon is visible from its base $[p_i, q_\alpha]$. Similarly, if R_{br} hits V_j at q'_β then $M_3(q'_\beta)$ is an empty edge-visible polygon with base $[p_j, q'_\beta]$ among the points in P_3.

In Case (ii), V_j is to the left of V_i. Here M_2 is an empty staircase polygon from p_i to p_j, and these are the parts of F_{bl} and F_{tr} of the $MEOP$ respectively. If F_{bl} (resp. F_{tr}) hits V_j (resp. V_i) at q'_α (resp. q_β), then $M_1(q'_\alpha)$ (portion to the left of V_j) is an edge-visible polygon with base $[q'_\alpha, p_j]$, and $M_3(q_\beta)$ (portion to the right of V_i) is an edge-visible polygon with base $[q_\beta, p_i]$. After fixing p_i and p_j as the bottom-pivot and top-pivot respectively, we need to choose M_1, M_2 and M_3 such that the sum of areas of these three polygons is maximum among all such polygons. We shall describe our algorithm for Case (i) only. Case (ii) can easily be handled using a similar method. For Case (i), we explain the method of computing the desired M_1 and M_2. The computation of M_3 is the same as that of M_1.

3.1 Computation of M_1

Let us consider a point $p_i \in P$. Let $P_1 = \{p_k | x(p_k) < x(p_i)\}$ and $Q = \{p_k | x(p_k) > x(p_i)$ and $y(p_k) > y(p_i)\}$. Q includes the top-right corner of $\mathcal{R}$, and $|Q| = m + 1$. Let $q_0, q_1, q_2, \ldots, q_m$ denote the projections of the points in Q on the vertical line V_i in decreasing order of their y-coordinates. We create an array $EVL(p_i)$ whose elements are the maximum area empty edge-visible polygon $M_1(q_k)$ with $[p_i, q_k]$ as the *base* for all $k = 0, 1, 2, \ldots, m$.

We use vertical line sweep among the points in P_1 starting from the position of V_i to create a height-balanced binary tree $\mathcal{T}$. Its each node v is represented as a 5-tuple $(I, x\_val, y\_val, \Delta, \delta)$. I is the base of the edge-visible polygons attached to node v. (x_val, y_val) is the point where the node v is generated, and Δ contains the area of the largest edge visible polygon rooted at that node. The Δ parameters are computed in two passes. In the forward pass during the sweep, the Δ parameter of a node contains the area of the edge visible polygon that is computed so far at that node. At the end of the sweep, a backward pass is executed from the leaf level of $\mathcal{T}$ up to its root, and Δ value of each node is properly set. The δ value of all nodes are 0 at the time of creation of $\mathcal{T}$; it will be set and used during the computation of $M_1(q_k)$ for different q_k.

Creation of $\mathcal{T}$

The root r of $\mathcal{T}$ corresponds to the entire interval $I = [p_i, q_0]$; its x_val and Δ parameters are set to $x(p_i)$ and 0 respectively. A vertical line sweep is performed from $x = x(p_i)$ towards left. When a point $p = (x(p), y(p)) \in P_1$ is faced by the sweep line, the leaf nodes in $\mathcal{T}$ are searched. If $y(p)$ lies in the interval $[\alpha, \beta]$ of a node $v = ([\alpha, \beta], \mu, \nu, \Delta, \delta)$, we compute

$\Delta^* = \Delta + (\mu - x(p)) \times (\beta - \alpha)$. Next, we create two children of v, namely $v_a = ([\alpha, y(p)], x(p), y(p), \Delta^*, 0)$ and $v_b = ([y(p), \beta], x(p), y(p), \Delta^*, 0)$. Finally, the backward pass is executed from leaves towards the root in post-order manner to set the Δ values as described above.

Computation of $M_1(q_k)$

$M_1(q_0) = \Delta$ attached to the root node r. While processing q_k, we assume that q_{k-1} is already processed. We start scaning from the root of $\mathcal{T}$. At a particular node $v = ([\alpha, \beta], \mu, \nu, \Delta(v), \delta(v))$ on the search path, one of the following two situations may happen: (i) $\nu \geq y(q_k)$ and (ii) $\nu < y(q_k)$.

In Case (i), we compute $A = (x(p_i) - \mu) \times (y(q_k) - y(q_{k-1}))$. The area A is to be subtracted from all the edge-visible polygons stored in the the right-child v_b of the node v. We subtract A from $\Delta(v_b)$. Without entirely traversing the subtree rooted at v_b, we add A in $\delta(v_b)$. The motivation is that, while processing some other q_ℓ, if the subtree rooted at v_b is traversed, $\delta(v_b)$ will be subtracted from the Δ value of those nodes. The search proceeds towards the left child of the node v. At each move from a node v to its children v', $\delta(v)$ is subtracted from $\Delta(v')$, and added to $\delta(v')$, and then $\delta(v)$ is set to 0.

In Case (ii), the edge visible polygon in the left child v_a of the node v does not exist; so we delete the subtree rooted at v_a and the node v also; the search proceeds towards the right child of v. The propagation of δ is to be performed at each step, but the computation of excess area A is to be performed when Case (i) arises. Next, a backward pass is needed to set the Δ field of all the nodes in the updated $\mathcal{T}$. Finally, $M_1(q_k)$ is set with the Δ value of the root of the updated $\mathcal{T}$.

Lemma 1 *The computation of $M_1(q_k)$ for all $k = 0, 1, \ldots, m$ needs $O(n^2)$ time.*

Proof. The creation of $\mathcal{T}$ needs $O(n \log n)$ time. The lemma follows from the fact that the combinatorial complexity of $M_1(q_k)$ is $O(n)$ for all $k = 0, 1, \ldots, m$. □

Similarly, with each point $p_i \in P$, an array $EVR(p_i)$ is attached. If the projections of the points $\{p_k | x(p_k) < x(p_i)$ & $y(p_k) < y(p_i)\}$ are denoted by $q'_0, q'_1, q'_2, \ldots, q'_{m'}$, then $|EVR(p_i)| = m' + 1$, and the content of its k-th element is the largest empty edge-visible polygon $M_3(q'_k)$ with base $[p_i, q'_k]$ among the points to the right of V_i.

Lemma 1 says that $EVL(p_i)$ and $EVR(p_i)$ for all $i = 1, 2, \ldots, n$ can be created in $O(n^3)$ time.

3.2 Computation of M_2

Consider the processing of a pair of points $p_i, p_j \in P$ satisfying $x(p_i) < x(p_j)$. Let R_{ij} be the rectangle with p_i and p_j at its diagonally opposite corner, and P_2^* be the set of points in P that lie in R_{ij}. Here M_2 can be split into three parts: the L_polygons inside the slab S below H_i and above H_j, and the empty staircase polygon $MESP(p_i, p_j)$ from p_i to p_j inside R_{ij}. The objective is to choose the staircase polygon such that the sum of its area along with the area of the corresponding L_polygons in S and the edge-visible polygons M_1 and M_3 on two sides of S is maximum.

Computation of L-polygons

Let $r_1, r_2, \ldots, r_m$ be the projections of the points in R_{ij} on H_i in the increasing order of their x-coordinates, and r_{m+1} is the intersection of H_i and V_j. We execute a horizontal line sweep among the points in S from the floor of $\mathcal{R}$ up to H_i to compute the area of the maximal empty L-polygons $LB(r_k)$ for $k = 1, 2, \ldots, m+1$. The upper stair of $LB(r_k)$ is an L-path with p_i at its corner, and the lower stair is a staircase path from b_1 to r_k. The L-polygons $LA(r_k), k = 0, 1, 2, \ldots, m$ above H_j are computed in an exactly similar manner; here r_0 is the intersection point of V_i and H_j. This needs $O(n)$ time in the worst case.

Computation of the staircase polygon

We now describe the last step of our algorithm for computing the maximal empty staircase polygons $MESP(p_i, p_j)$ considering the area of the coresponding L-polygons and edge-visible polygons such that the total area of $MEOP(p_i, p_j)$ is maximum. Let G be a directed graph with vertices corresponding to the points in R_{ij} and edges $\{e_{k\ell} = (p_k, p_\ell) | p_k, p_\ell \in P_2^*$ with $x(p_k) < x(p_\ell)$ and $y(p_k) < y(p_\ell)\}$. Any path from p_i to p_j in G corresponds to the lower stair of an $MESP(p_i, p_j)$; but there are different choices of the upper stairs corresponding to the same lower stair. The problem of computing the maximum area $MESP(p_i, p_j)$ can be formulated as finding the maximum weighted path in an weighted directed graph, called the staircase graph [10],.

Definition 4 *[10] Let (p_a, p_b) be an edge of G. The point $(x(p_a), y(p_b))$, where the vertical line V_a abuts the horizontal line H_b, is called the footprint of p_b contributed by p_a, and is denoted by b_a. The footprint of the point p_i is p_i itself. We use $FP(p)$ to denote the set of footprints of the point $p \in P_2^*$.*

Definition 5 *[10] The staircase graph $SG = (V, E)$ is a weighted digraph with nodes $V = \cup_{p_a \in P_2^*} FP(p_a)$. A footprint $b_a \in FP(p_b)$ has a directed edge to a footprint $d_c \in FP(p_d)$ if (p_b, p_d) is an edge in G, and the upper stair of the L_polygon$[b_a, p_d]$ meets the horizontal line H_d at the footprint d_c. The weight of the edge (b_a, d_c), denoted by $w(b_a, d_c)$ is equal to the area of the L_polygon$[b_a, p_d]$.*

A path in SG corresponds to a unique staircase polygon from p_i to p_j. Assuming $|P_2^*| = m$, the worst-case number of vertices and edges in SG are $O(m^2)$ and $O(m^3)$ respectively, and the maximum weighted path in SG can be found in $O(m^3)$ time.

In our problem, computing the maximum area empty staircase polygon among the points in P_2 will not suffice. Suppose $MEOP(p_i, p_j)$ consists of a staircase polygon $MESP(p_i, p_j)$, that has edges (p_i, q_α) along V_i, $(p_j, q_{\alpha'})$ along V_j, (p_i, r_β) along H_i and $(p_j, r_{\beta'})$ along H_j, then it includes (i) an edge-visible polygon with base (p_i, q) to the left of V_i, (ii) an edge visible polygon with base (p_j, q') to the right of V_j, (ii) an L-polygon with base (p_i, r) and (iv) an L-polygon with base (p_j, r'). Let q, q', r and r' correspond to $p_\alpha, p_{\alpha'}, p_\beta, p_{\beta'} \in P_2$ respectively. Thus, in order to compute the $MEOP$ of maximum area, we need to modify the weight of some edges of the graph SG as follows, and then compute the maximum weighted path in the graph SG.

For each α such that $p_\alpha \in P_2$, change the weight of its each outgoing edge e to $w(e) + area(M_1(p_i, q_\alpha))$.

For each edge $e' = (p_i, \beta_{k'})$, change the weight of e' to $w(e') + area(LB(p_i, r_\beta))$.

For each edge α' such that $p_{\alpha'} \in P_2$, if there exists an edge e^{**} from a footprint of α' to a footprint of p_j, then change its weight to $w(e^{**}) + area(M_3(p_j, q'_{\alpha'}))$.

For each incoming edge e' on $j_{\beta'}$), change the weight of e' to $w(e') + area(LA(p_j, r_{\beta'}))$.

Theorem 2 *The largest $MEOP$ among a set of n points can be computed in $O(n^5)$ time and $O(n^2)$ space.*

In [10], it is also shown that the geometric properties of the problem can be exploited to design an algorithm for computing the empty staircase polygon of maximum area in $O(|P_2|^2)$ time and space. Here the results of procesing a point p_j for computing the $MESP(o, p_j)$ are used to compute $MESP(o, p_k)$, where $x(p_k) > x(p_j)$ and $y(p_k) > y(p_j)$; The point o is the bottom left corner of the rectangular floor. We will use the same principle to reduce the time complexity of the problem of computing the maximum area $MEOP$ to $O(n^3)$.

4 Further improvement

We will fix a point $p_i \in P$, and consider all $p_j \in P$ with $x(p_j) > x(p_i)$ and $y(p_j) > y(p_i)$. We also use the the notion of complete processing of a point p_j [10]. A point p_j is said to be *completely processed* if all the edges incident to p_j in the graph G are processed. When a point p_j is completely processed, the weights of different paths in the staircase graph (SG) with bottom-pivot and p_i and p_j respectively, are available at the footprints of p_j. Each of these polygons has included the corresponding $M_1(p_i, q_\alpha)$ and $LB(p_i, r_\beta)$ for some appropriate $p_\alpha, p_\beta \in S$. In order to get the maximum area $MEOP$ with a $MESP(p_i, p_j)$, we need to add the area of the appropriate $M_3(p_j, q_{\alpha'})$ and $LA(p_j, r_{\beta'})$, where $p_{\alpha'}, p_{\beta'} \in S$. We can compute the array L containing $LA(p_j, r_k)$ for all the points $p_k \in S$ above H_j in $O(n)$ time. The values of $area(M_3(p_j, q_k))$ are all available in the array $EVR(p_j)$. Now we can use $EVR(p_j)$, L, and the area attached to the different footprints of p_j to compute the largest $MEOP(p_i, p_j)$ in $O(n)$ time.

We process the points above H_i in S by sweeping a horizontal line upwards. After complete processing of p_j, we compute $MEOP(p_i, p_j)$ as described above, and then process the outgoing edges of p_j in G. Thus, we have the following theorem:

Theorem 3 *The largest $MEOP$ among a set of n points can be computed in $O(n^3)$ time and $O(n^2)$ space.*

Proof. For a fixed p_i, the generation of footprints for all $p_j \in P$ with $x(p_j) > x(p_i)$ and $y(p_j) > y(p_i)$ needs $O(n^2)$ time [10]. The additional time required to process each $p_j \in P$ is $O(n)$. Since we need to fix each p_i, the time complexity result follows.

The space complexity result follows from the fact that the preprocessed arrays $M_1(p_i)$ and $M_3(p_i)$ are of size $O(n)$ in the worst case. Moreover while processing a point p_i, the number of footprints generated is $O(n^2)$ in the worst case. These need to be stored during the processing of p_i. □

References

[1] A. Aggarwal and S. Suri, *Fast algorithms for computing the largest empty rectangle*, in Proc. 3rd. Annual Symp. Comp. Geom., pp. 278-290, 1987.

[2] C. Audet, P. Hansen and F. Messine, *Extremal problems for convex polygons*, J. of Global Optimization, vol. 38, pp. 163-179, 2007.

[3] A. Biswas, P. Bhowmick and B. B. Bhattacharya, *Finding the orthogonal hull of a digital object*, Proc. IWCIA, LNCS-4958, pp. 124-135, 2008.

[4] J. E. Boyce, D. P. Dobkin, R. L. Drysdale, and L. Guibas, *Finding extremal polygons*, Proc. 14th Annual ACM Symp. on Theory of comput., pp. 282 - 289, 1982.

[5] R. P. Boland and J. Urrutia, *Finding the largest axis aligned rectangle in a polygon in $o(n \log n)$ time*, Proc. Canad. Conf. in Comp. Geom., pp. 41-44, 2001.

[6] K. Daniels, V. Milenkovic and D. Roth, *Finding the largest area axis-parallel rectangle in a polygon*, Computational Geometry: Theory and Applications, vol. 7, pp. 125-148, 1997.

[7] A. Datta and G. D. S. Ramkumar, *On some largest empty orthoconvex polygons in a point set*, Proc. FSTTCS, LNCS 472, pp. 270-285, 1990.

[8] G. T. Herman and A. Kuba (eds.), Advances in Discrete Tomography and its Applications, Birkhauser, 2007.

[9] S. C. Nandy, A. Sinha and B. B. Bhattacharya, *Location of the largest empty rectangle among arbitrary obstacles*, Proc. FSTTCS, LNCS 880, pp. 159-170, 1994.

[10] S. C. Nandy and B. B. Bhattacharya, *On finding an empty staircase polygon of largest area (width) in a planar point-set* Computational Geometry: Theory and Applications, vol. 26, pp. 143-171, 2003.