Computational Geometry: Theory and Applications, 7:125-148, 1997.

Finding the Largest Area Axis-Parallel Rectangle in a Polygon*

Karen Daniels † *Victor Milenkovic*‡ *Dan Roth*§
Harvard University,
Division of Applied Sciences,
Center for Research in Computing Technology,
Cambridge, MA 02138.

JUNE 1995

Abstract

This paper considers the geometric optimization problem of finding the Largest area axis-parallel Rectangle (LR) in an n-vertex general polygon. We characterize the LR for general polygons by considering different cases based on the types of contacts between the rectangle and the polygon. A general framework is presented for solving a key subproblem of the LR problem which dominates the running time for a variety of polygon types. This framework permits us to transform an algorithm for orthogonal polygons into an algorithm for nonorthogonal polygons. Using this framework, we show that the LR in a general polygon (allowing holes) can be found in $O(n \log^2 n)$ time. This matches the running time of the best known algorithm for orthogonal polygons. References are given for the application of the framework to other types of polygons. For each type, the running time of the resulting algorithm matches the running time of the best known algorithm for orthogonal polygons of that type.

A lower bound of time in $\Omega(n \log n)$ is established for finding the LR in both self-intersecting polygons and general polygons with holes. The latter result gives us both a lower bound of $\Omega(n \log n)$ and an upper bound of $O(n \log^2 n)$ for general polygons.

Keywords: rectangles, geometric optimization, fast matrix searching.

*An earlier version of this paper appeared in the proceedings of the Fifth Annual Canadian Conference on Computational Geometry, 1993.

†This research was funded by the Textile/Clothing Technology Corporation from funds awarded to them by the Alfred P. Sloan Foundation.

‡This research was funded by the Textile/Clothing Technology Corporation from funds awarded to them by the Alfred P. Sloan Foundation and by NSF grants CCR-91-157993 and CCR-90-09272.

§Supported by NSF grants CCR-89-02500 and CCR-92-00884 and by DARPA AFOSR-F4962-92-J-0466.

1 Introduction

The problem of finding the Largest area axis-parallel Rectangle (LR) inside a general polygon[1] of n vertices is a geometric optimization problem in the class of polygon *inclusion* problems [7]. Define $Inc(\mathcal{P}, \mathcal{Q}, \mu)$: Given $P \in \mathcal{P}$, find the μ-largest $Q \in \mathcal{Q}$ inside P, where $\mathcal{P}$ and $\mathcal{Q}$ are families of polygons, and μ is a real function on polygons such that:

$$\forall Q,\, Q' \in \mathcal{Q}, \qquad Q' \subseteq Q \quad \Rightarrow \quad \mu(Q') \leq \mu(Q).$$

Our problem is an inclusion problem where $\mathcal{Q}$ is the set of axis-parallel rectangles, $\mathcal{P}$ is the set of general polygons, and μ gives the area of a rectangle.

This rectangle problem arises naturally in applications where a quick internal approximation to a polygon is useful. It is needed, for example, in the industrial problem of laying out apparel pattern pieces on clothing "markers" with minimal cloth waste [22, 23] (see Section 6).

1.1 Related Work

Despite its practical importance, work on finding the LR has been restricted to orthogonal polygons[2] [4, 20, 30] and, recently, convex polygons [5] (see Figure 1). Amenta [5] has shown that the LR in a convex polygon can be found in linear time by phrasing it as a convex programming problem. For a constrained type of orthogonal polygon, Aggarwal and Wein [4] give a $\Theta(n)$ time algorithm for finding the LR using the monotonicity of an area matrix associated with the polygon.

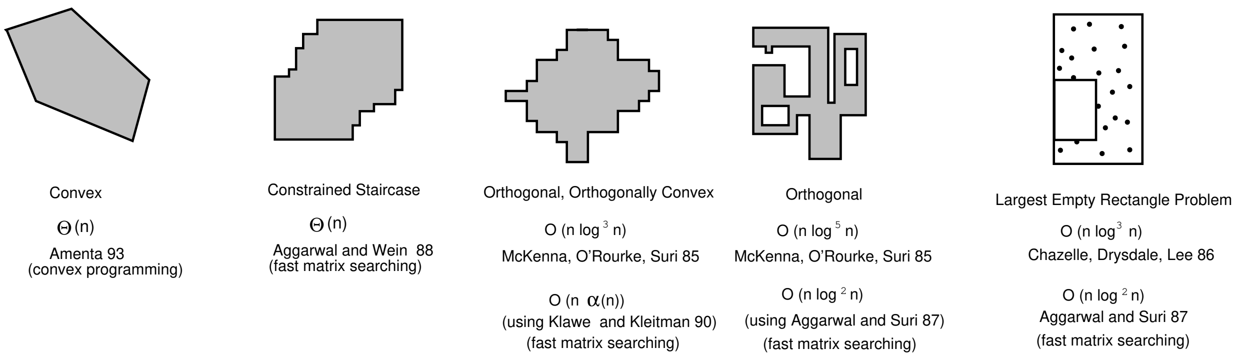

Figure 1: Related Work

McKenna *et al.* [20] use a divide-and-conquer approach to find the LR in an orthogonal polygon in $O(n \log^5 n)$ time. For the merge step at the first level of divide-and-conquer, they obtain an orthogonal, vertically separated, horizontally convex, polygon[3]. At the second level, their merge step produces an orthogonal, orthogonally convex polygon[4], for which they solve the LR problem in $O(n \log^3 n)$ time. They also establish a lower bound of time in $\Omega(n \log n)$ for finding the LR in orthogonal polygons with degenerate holes, which implies the same lower bound for general polygons with degenerate holes.

McKenna *et al.* note, without giving details, that the LR in an orthogonal polygon can also be found using the more complicated $O(n \log^3 n)$ time divide-and-conquer algorithm of Chazelle *et al.*[9] for the largest empty rectangle (LER) problem. The LER problem is stated as follows: given a rectangle containing a set

[1]A general polygon is a polygonal region in the plane with an arbitrary number of components and holes. A rectangle is *inside* if it is a subset. The rectangle can share part of its boundary with the polygon's.

[2]We use O'Rourke's definition: "An orthogonal polygon is one whose edges are all aligned with a pair of orthogonal coordinate axes, which we take to be horizontal and vertical without loss of generality" [25]. In the context of this paper, this might be called an *axis-parallel* polygon.

[3]The boundary of a *vertically separated* polygon consists of two chains which extend from the highest point of the polygon to the lowest point and which are on opposite sides of some vertical line. A *horizontally convex* polygon contains every horizontal line segment whose endpoints lie inside the polygon. For a *vertically separated, horizontally convex* polygon, the two chains are y-monotone.

[4]An *orthogonally convex* polygon is both horizontally and vertically convex. This class contains the class of convex polygons.

S of n points, find the largest area rectangular subset, with sides parallel to those of the original rectangle, whose interior contains no points from S ([9, 24, 2]). Chazelle *et al.* observe that the running time of the merge step of their algorithm is dominated by the largest empty corner rectangle (LECR) problem: given two subsets S_{left} and S_{right} of S, find the largest rectangle containing no point of S which has lower-left corner in S_{left} and upper-right corner in S_{right}. The fastest solution to LECR is Aggarwal and Suri's $O(n \log n)$ time algorithm, which they present as part of an $O(n \log^2 n)$ time solution to the LER problem [2]. Their LECR algorithm relies on fast searching of area matrices.

We observe that a speed-up in the LECR algorithm automatically improves the running time for finding the LR in an orthogonal polygon. This speed-up occurs because a fast LECR algorithm implies a fast algorithm for the Largest Corner Rectangle (LCR) in an orthogonal, vertically separated, horizontally convex polygon[5]. Computing the LCR, in turn, dominates the running time of the LR problem for orthogonal, vertically separated, horizontally convex polygons. Finally, as we have previously stated, this special case is required for the merge step of a divide-and-conquer algorithm for general orthogonal polygons. Thus the $O(n \log n)$ time algorithm for LECR yields an $O(n \log^2 n)$ time algorithm for finding the LR in an orthogonal polygon.

The $O(n \log^3 n)$ time algorithm of [20] for orthogonal, orthogonally convex polygons can also be improved by applying recent results in fast matrix searching. Aggarwal and Suri [2] note that, for the LECR problem which can be associated with the vertices of this type of polygon, there is a corresponding area matrix whose maximum can be found in $O(n \log n)$ time by decomposing it into a set of simpler area matrices. They note in [3] that Klawe and Kleitman's results [17] for this simpler type of matrix imply $O(n\alpha(n))$ search time for the more complex matrix, where $\alpha(n)$ is the slowly growing inverse of Ackermann's function. It is easy to see that this yields $O(n\alpha(n))$ time for finding the LR in an orthogonal, orthogonally convex polygon.

Melissaratos and Souvaine [21] use the visibility techniques of [15] to solve several geometric optimization problems. In particular, they find the largest triangle contained in a polygon in $O(n^4)$ time by considering the types of contacts between the polygon and the triangle. A similar approach can be applied to the LR problem by using the concept of *rectangular visibility*[6] [27], but this leads to an $O(n^5)$ algorithm, which is much slower than the $O(n \log^2 n)$ one we propose in this paper.

Another possible approach to the LR problem involves Voronoi diagrams, but it is unlikely to produce an algorithm faster than $O(n \log^2 n)$. Chew and Drysdale [10] discuss using a Voronoi diagram of a point set, based on a convex distance function, to find the associated largest empty convex shape. Chazelle *et al.* [9], in their work on the largest empty rectangle problem, cite the use of a Voronoi diagram in the L_∞ or L_1 metric [19, 16] to find the largest empty axis-parallel square for a point set. Aurenhammer [6] notes that a transition from squares to rectangles is complicated because the distance function depends on the aspect ratio of the rectangle, which is unknown. Chazelle *et al.* [9] use a Voronoi-like diagram to solve the LECR problem. However, this approach is slower than Aggarwal and Suri's LECR algorithm [2], which is based on fast matrix searching. In order to use Voronoi diagrams to solve the problem treated in this paper, one would need a generalized Voronoi diagram, often called the *medial axis* [26, 18] of a polygon. Since the fastest algorithm for the largest empty rectangle problem (for point sets) is not based on Voronoi diagrams, we doubt that using such a generalized Voronoi diagram would yield an algorithm faster than the $O(n \log^2 n)$ one we present in this paper.

No published algorithm is known for finding the LR in a general non-orthogonal polygon with (non-degenerate) holes, nor has a lower bound tighter than $\Omega(n)$ been established.

1.2 Overview

We present the first algorithmic results for general polygons with holes: an $O(n \log^2 n)$ time algorithm. We also prove a lower bound for this type of polygon of time in $\Omega(n \log n)$. The divide-and-conquer approach used for finding LRs in orthogonal polygons is applicable to non-orthogonal polygons, but it is a challenge to deal with the special cases of the LR problem that arise during the merge step. As is the case for

[5]The LCR of an orthogonal polygon is the largest area rectangle with diagonally opposite corners on the boundary of the polygon. Our definition of LCR for non-orthogonal polygons is somewhat more specific (see Section 3).

[6]Overmars and Wood [27] define rectangular visibility as follows: "Given a set of points S in the plane, a point p is said to be rectangularly visible from a point q with respect to S if and only if there exists an orthogonal rectangle R that contains both p and q, but no other point of S." We use a slightly less restrictive version of rectangular visibility (see Section 3.2).

orthogonal polygons, the running time is dominated by the LCR (largest corner rectangle) problem for vertically separated, horizontally convex polygons. Unfortunately, for non-orthogonal polygons, it is not so easy to reduce the LCR problem to a LECR problem. For this reason, we present a general framework which can be used to transform LCR problems for several types of non-orthogonal polygons into LCR problems for "partially orthogonal" polygons. The framework shows how to modify a LECR algorithm to solve these special LCR problems. This framework allows us to achieve the same LR time bounds for the non-orthogonal case as has already been achieved for the corresponding orthogonal case. In this paper, we apply the framework to vertically separated, horizontally convex polygons to obtain an $O(n \log^2 n)$ time algorithm for general polygons. In [12] we apply it to find the LR of: an xy-monotone polygon[7] in $\Theta(n)$ time, an orthogonally convex polygon in $O(n\alpha(n))$ time and a horizontally (vertically) convex polygon in $O(n\alpha(n) \log n)$ time.

Our paper is organized as follows. In Section 2 we characterize the LR for general polygons by considering different cases based on the types of contacts between the rectangle and the polygon. In Section 3 we present a general framework for solving the 2-contact case of the LR problem, which dominates the running time for a variety of polygon types[8]. The framework involves transforming the polygon, via vertex projection and inner orthogonal approximations, into a "partially orthogonal" polygon for which we can solve the associated LCR problem by solving a modified LECR problem. The LECR problem is solved efficiently using fast matrix searching techniques from the literature.

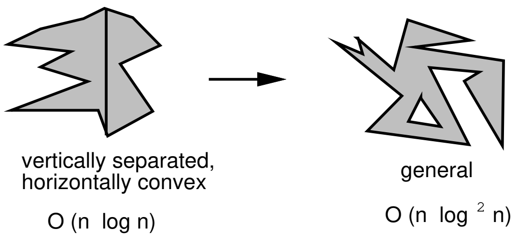

Figure 2: Algorithmic Results

Section 4 presents an $O(n \log^2 n)$ time divide-and-conquer algorithm for finding the LR in a general polygon with holes. Our 2-contact framework is applied to solve the 2-contact case for a vertically separated, horizontally convex polygon. This type of polygon arises in the merge step, and finding its LCR and LR dominates the running time of the divide-and-conquer algorithm. We show that its LR can be found in $O(n \log n)$ time. This $O(n \log n)$ algorithm uses the results of Aggarwal and Suri [2, 3] for the LECR problem. Our running time results are summarized in Figure 2.

In Section 5 we prove a lower bound of time in $\Omega(n \log n)$ for finding the LR in both self-intersecting polygons and general polygons with holes. The latter result gives us both a lower bound of $\Omega(n \log n)$ and an upper bound of $O(n \log^2 n)$ for general polygons with holes. It uses symbolic perturbation to extend the $\Omega(n \log n)$ lower bound of McKenna *et al.* for orthogonal polygons with degenerate holes. The proof for self-intersecting polygons involves a reduction from MAX-GAP. This $\Omega(n \log n)$ lower bound clearly demonstrates that the LR *inclusion* problem is harder than the corresponding smallest rectangle *enclosure* problem, which has a trivial linear time algorithm.

Section 6 discusses LR applications.

2 Characterizing the LR

In this section we characterize the LR contained in a general polygon P by considering different cases based on the types of contacts between the LR and the boundary of P. We outline a naive algorithm for finding

[7]A simple polygon consisting of two xy-monotone chains is an xy-monotone polygon. A chain is xy-monotone if it is monotone with respect to both the x and y axes. A chain is monotone with respect to a line l if a line orthogonal to l intersects the chain in exactly one point [28].

[8]The 2-contact case is equivalent to a constant number (eight) of LCR problems.

the LR based on this characterization. Others have used contact classification for algorithmic development (see, for example, [21, 20, 13]).

2.1 Types of Contacts

Intuitively, if an axis-parallel rectangle is inside P, it has four degrees of freedom (parameters) and can "grow" until each of its four sides is stopped by contact with the boundary of P. Contacts between the rectangle and P are of two types: 1) a side of the rectangle with a vertex of P, and 2) a corner of the rectangle with an edge of P. In order to discuss the first type, we require the notion of a *reflex extreme* vertex, introduced in [29].

Definition 2.1 *A vertex v of P is a* **vertical reflex extreme vertex** *if* exterior(P) *has a local vertical line of support at v: for some epsilon > 0, the vertical line segment of length ϵ and with midpoint v is a subset of P (boundary plus interior). A* **horizontal reflex extreme vertex** *is defined similarly.*

For type 1 contacts, a reflex extreme vertex of P touches a side of the rectangle and stops growth in one direction; we call this a *reflex contact*. Each reflex contact can remove one degree of freedom. Two reflex contacts with adjacent sides of the rectangle fix a corner of the rectangle. For type 2 contacts, a corner of the rectangle touches an edge of P forming an *edge contact*.

2.2 A Determining Set of Contacts

Definition 2.2 *A set of contacts C is a* **determining set of contacts** *if the LR R satisfying C has finite area and if the LR R' satisfying any proper subset $C' \subset C$ has greater or infinite area.*

For example, a set of four reflex contacts, one on each side of the rectangle, is a determining set. Note: A determining set determines the area of the LR, but it does not necessarily determine a *unique* rectangle or LR.

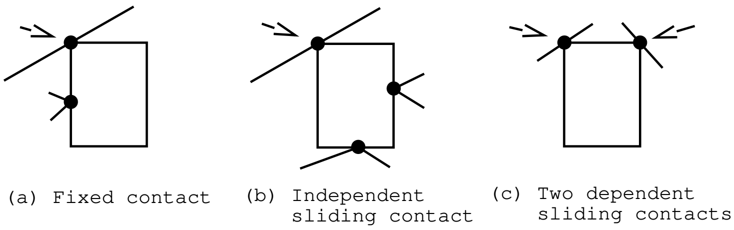

Figure 3: Edge Contact Types for a Determining Set

Within a determining set, we distinguish between two different subtypes of edge contacts. An edge contact is *fixed* if the set of constraints uniquely determines the point of contact with the rectangle. Otherwise, it is a *sliding contact.* Note that we are considering here the set of *all* rectangles which satisfy the determining set of contacts, and we are not just considering rectangles of maximal area.

A *fixed contact* can arise when there is no freedom to slide along an edge because a reflex contact fixes a coordinate. For example, in Figure 3(a), the reflex contact of the determining set fixes the x-coordinate of the edge contact, which completely determines the location of the edge contact. If an edge contact has an adjacent side which has either a reflex or fixed contact, then the edge contact must also be a fixed contact.

Two sliding edge contacts are *dependent* if the position of one determines the position of the other; otherwise they are *independent.* An independent sliding contact requires that the two adjacent sides of the rectangle do not have any contact with P (see Figure 3(b)). A sliding contact adjacent to another sliding contact is dependent, because the two contacts must share a coordinate (see Figure 3(c)).

2.3 Maximization Problems

Here we examine maximization problems associated with certain determining sets of contacts. Finding the LR associated with a determining set of contacts requires solving a maximization problem if the set contains a sliding contact. For a given set of contacts, the number of degrees of freedom is the number of undetermined parameters of the rectangle. Degrees of freedom within a determining set can arise only from sliding contacts because any other degree of freedom would result in a rectangle of infinite area, and therefore the contacts would not form a determining set. It follows that if a determining set consists of only reflex or fixed edge contacts, no maximization is required. For each independent sliding contact in the set, we can parameterize the associated edge. The maximization problems can then be classified based on the number of parameters.

2.3.1 1-Parameter Problems

The set of 1-parameter maximization problems can be further subdivided according to the number of dependent sliding contacts.

The Basic 1-Parameter Problem: The simplest 1-parameter problem involves no dependent sliding contacts, just a single independent one. This is the basic 1-parameter problem, and it arises when one corner of the LR has a sliding contact and the diagonally opposite corner is fixed. The basic 1-parameter problem can be solved by parameterizing the edge associated with the sliding contact and maximizing a quadratic in one variable. This can be solved in O(1) time.

Two Dependent Sliding Contacts: If there are exactly two dependent sliding contacts in a determining set, then these contacts are at the endpoints of one edge of the rectangle, and there is a reflex contact with the opposite edge of the rectangle. W.l.o.g. these are the top and bottom edges with y-coordinates y and y', as shown in Figure 4(a). To find the LR, we parameterize edge $\overline{p_2 p_1}$ by t, yielding a quadratic in t to maximize (see [12] for details).

(a) Pair of Constrained Sliding Contacts (b) Triple of Constrained Sliding Contacts

Figure 4: 1-Parameter Problems with Two and Three Dependent Sliding Contacts

Three Dependent Sliding Contacts: The case of three dependent sliding contacts is depicted in Figure 4(b). It is dealt with in a manner similar to the case of two dependent sliding contacts. See [12] for details.

2.3.2 The 2-Parameter Problem

There is only one type of 2-parameter problem. It has two independent sliding contacts. The following lemma allows us to reduce a 2-parameter problem to a set of 1-parameter problems.

Lemma 2.1 *Let e_1 and e_2 be non-intersecting line segments. Consider the set of empty axis-parallel rectangles which have diagonally opposite corners on e_1 and e_2. There is a largest area rectangle in this set with at least one corner at an endpoint of e_1 or e_2.*

Proof: Parameterize the positions of the corners of the rectangle on e_1 and e_2. The area of the rectangle is a quadratic function of the two parameters. It is easily shown that the graph of the quadratic over the patch $[0,1] \times [0,1]$ is a saddle surface, and therefore the maximum is achieved along the boundary of the patch. The boundary corresponds to the subset of rectangles which have at least one corner at an endpoint of either e_1 or e_2.

■

Having established that there exists a LR with a corner at a vertex in this case, we can find it by considering, in turn, each of the four endpoints of e_1 and e_2, solving the associated 1-parameter problems, and then comparing the four resulting 1-parameter LR areas.

2.4 Characterization Theorem

To characterize the LR, we examine the possible determining sets of contacts. By enumerating the reflex contacts between the LR and P, we derive the set of five cases shown in Figure 5.

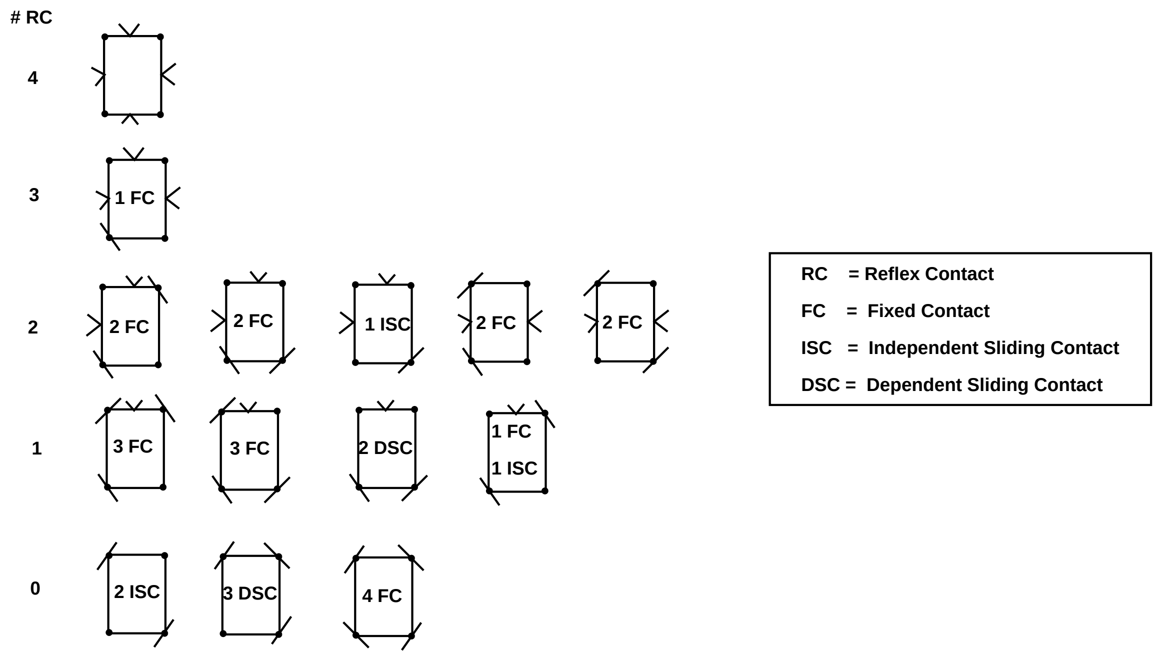

Figure 5: The Determining Sets of Contacts for the LR

Theorem 2.2 *The determining set of the LR of a general polygon P conforms (up to symmetry) to one of the five cases in Figure 5.*

Proof: The proof is a straightforward examination of cases. See [12] for details. ■

Corollary 2.3 *Given a determining set C for a LR of a general polygon, it follows that $2 \leq |C| \leq 4$.*

Based on the above characterization, we can find the LR in a general polygon by finding the LR under the constraints of each of the five cases and selecting the largest one. It is easy to show that, for each determining set of contacts in Figure 5, the LR can be found in constant time. A naive LR algorithm can use this result and find the LR in each case for all possible determining sets for P. These can be identified using an algorithm with up to four nested loops, one for each element of the determining set. For each LR candidate, we can check if it is empty (i.e. contains no point from the boundary of P in its interior) in $O(n)$ time. We conclude:

Theorem 2.4 *The LR of an n-vertex general polygon can be found in $O(n^5)$ time.*

In the remainder of the paper we show how to use the LR characterization combined with fast matrix searching to develop a more sophisticated approach to this problem which yields an $O(n \log^2 n)$ time algorithm.

3 A General Framework for the 2-Contact Case

We compute the LR for a general polygon with holes using divide-and-conquer. The divide-and-conquer algorithm must find the LR in a polygon P, which is a subset of the general polygon, and which is of the following type: the boundary of P consists of two y-monotone chains V and E on opposite sides of a vertical line. Recall from Section 1.1 that we call this a *vertically separated, horizontally convex* polygon (see Figure 2). By Corollary 2.3, the algorithm must consider the 2, 3, and 4-contact cases in order to find the LR in P. Of these, the 2-contact case dominates the running time. This section gives a framework for creating algorithms for the 2-contact case for classes of polygons with y-monotone chains (including the class of vertically separated, horizontally convex polygons). We call this the *2-contact framework*. Section 4 applies the 2-contact framework to create a 2-contact LR algorithm for vertically separated, horizontally convex polygons and then gives the divide-and-conquer LR algorithm for general polygons. In [12] we apply the 2-contact framework to create 2-contact LR algorithms for other types of polygons and then give LR algorithms for these types. For all of these polygon types, the 2-contact framework yields an algorithm which has the same order running time as the fastest LR algorithm for the *orthogonal* version of that type.

The 2-contact case for polygons with y-monotone chains V and E involves finding the largest rectangle which is inside P and which has one corner on V and the diagonally opposite corner on E. By Lemma 2.1, one of the corners of the largest 2-contact rectangle is at a vertex of either V or E. Furthermore, there are four choices for which corner of the rectangle has this corner-vertex contact. We refer to each of eight possibilities as a Largest Corner Rectangle (LCR) problem. In what follows, we treat only the LCR problem for which the lower-left corner of the rectangle is at a vertex of V and the upper-right corner is on an edge of E. We call such a rectangle a *vertex-edge rectangle* for V and E. This definition of LCR is analogous to Chazelle's definition of the LECR.

In Section 1.1, we mentioned that Chazelle *et al.* use a divide-and-conquer strategy to solve the LER (Largest Empty Rectangle) problem for a set of points. For their algorithm, the most difficult subcase is the LECR (Largest Empty Corner Rectangle) problem: given a set S of points and given two subsets S_{left} and S_{right} of S, find the largest rectangle containing no point of S in its interior which has lower-left corner in S_{left} and upper-right corner in S_{right}. Ideally, we would like to solve the LCR problem for P, V and E in the following manner. Set $S = \text{vertices}(P)$, $S_{\text{left}} = \text{vertices}(V)$, and $S_{\text{right}} = \text{vertices}(E)$. Then find the LECR of S, S_{left}, and S_{right}. Unfortunately, there are two ways in which the LECR can fail to be the LCR. First, some edge of P might intersect the interior of the LECR. Second, the actual LCR might have its upper-right corner in the middle of an edge of E, not at a vertex.

Fortunately, for a variety of polygon types, it is possible to reduce the problem of computing the LCR to that of computing the LECR. The reduction involves several steps, and these steps constitute our *2-contact framework* for solving 2-contact LR problems. Section 3.1 gives a high level description of the 2-contact framework, and Sections 3.2 through 3.4 give specific details.

3.1 High Level Description of the 2-Contact Framework

This section gives a high level description of the 2-contact framework. The "user" of the framework must provide a linear-time transformation of P, V, and E into P', V', and E' which satisfies certain properties. The framework specifies the properties (which amount to the notion of creating "partially orthogonal" P', V', and E'). The "user" must also provide a LECR algorithm with certain properties that the framework also specifies. The framework shows how to create an algorithm for the LCR R' of P', V', and E'. Because of the properties, R' is at least as large as the LCR of P, V, and E, and $R' \subseteq P$.

Notice that the framework does *not* necessarily create an algorithm for the LCR of P, V, and E. The rectangle R' is an LCR of P', V', and E', but it might be a 3-contact or 4-contact rectangle inside P. Nevertheless, an algorithm for R' is sufficient. Recall that the overall goal is to compute the LR of P. When the LR algorithm "checks" the 2-contact case, it is acceptable for the LCR algorithm to find a rectangle inside P that is *larger* than the largest 2-contact rectangle.

Thus, the framework takes a transformation and a LECR algorithm as "input" and creates a LCR algorithm as "output." To do this, the framework defines a second transformation that consists of adding a new vertex at the midpoint of every edge of E'. This transforms P' into P'' and E' into E''. The framework also defines a measure η for the size of corner rectangles which have diagonally opposite corners in vertices(V') and vertices(E''). The framework modifies the user-supplied LECR algorithm by substituting a call to η whenever the LECR algorithm computes the area of a corner rectangle.

Here, in broad detail, is the LCR algorithm that the framework creates. The algorithm first transforms P, V, and E, into polygon P' and chains V' and E' of P' using the transformation provided by the "user". Next, it applies the second transformation, yielding P'' and E''. It calls the modified LECR algorithm to compute the rectangle R'' which maximizes η over all rectangles with one corner in vertices(V'), the diagonally opposite corner in vertices(E''), and which contains no element of vertices(P'') in its interior. It applies a constant-time transformation to convert R'' into R', the LCR of P', V', and E'.

The framework deals with both of the ways in which the output R'' of the LECR algorithm could fail to solve the LCR problem: it might not be inside P and it might not have the largest area. First, the properties which the user-supplied transformation of P to P' must satisfy guarantee that R'' is inside P' and P. Second, R'' might not have the correct area, but $\eta(R'')$ is equal to the area of the LCR R' of P', V', and E', and R'' can be transformed into R'.

Section 3.2 defines the three properties that P', V', and E' must have in order to apply the 2-contact framework. It defines η, and proves that the LECR R'' for the vertices of P'', V', E'', and the measure η can be transformed into the LCR R' of P', V', and E' which, in turn, is inside P and is at least as large as the LCR of P, V, and E. Section 3.3 considers the question of transforming a LECR algorithm into a LECR algorithm for measure η by substituting the η function for the area function in the implementation of the LECR algorithm. This section defines a property called *total monotonicity* and proves that, if both V and E are y-monotone, then both the area function and the η function are totally monotone. It then observes that if the proof of correctness of the LECR algorithm only depends on the total monotonicity of the area function, then the proof will still work if η is substituted for area. This "meta-theory" is a general scheme for transforming algorithms and proofs. However, the only way to be really sure that the proof "only depends" on total monotonicity is to substitute η for area and recheck the proof. Section 3.4 observes that if the number of vertices of P', V', and E' are linear in the number of vertices in P, then the algorithm for the LCR of P', V', and E' has the same order running time as the corresponding LECR algorithm.

3.2 Properties and the LR Measure

This section defines three properties of P', V', and E' with respect to P, V, and E. It also defines a size measure η for rectangles with opposite corners at vertices of V', and E''. Recall that E'' has vertices at the midpoints of edges of E'. We call these added vertices *special vertices.* For a rectangle whose corner is at a vertex of E', η is the area function, but for a rectangle whose corner is at a special vertex, η has a different value defined below. We prove that the LECR R'' for the vertices of P', V', and E'' has the property that $\eta(R'')$ is greater than or equal to the area of the LCR of P, V, and E. We also show how to generate a rectangle R' inside P with this area.

The following are the three properties that P', V', and E' must satisfy with respect to P, V, and E.

Property I: Polygonal regions P and P' satisfy $P' \subseteq P$ and each vertex-edge rectangle for P, V, and E is a vertex-edge rectangle for P', V', and E'. (Even if the upper-right corner of the rectangle is at a vertex of E, we still consider it to be a vertex-edge rectangle.)

Property II: For every vertex $v \in V'$ and every edge $e \in E'$: if any point q in the interior of e is rectangularly visible[9] from v inside P', then the entire edge e is rectangularly visible from v.

Property III: If vertex $v \in V'$ and a point $q \in E'$ are rectangularly visible with respect to vertices(P'), then v and q are rectangularly visible with respect to P'.

Adding a "special" vertex at the midpoint of every edge of E' does not alter the satisfaction of Properties I-III. Therefore P', V', and E'' also satisfy Properties I-III. We introduce a new measure η on corner rectangles of V', and E''. For rectangles which have a corner at a special vertex, this measure differs from the area function.

[9]Two points p and q are *rectangularly visible* [27] inside polygonal region P if $\mathrm{rect}(pq) \subseteq P$, where $\mathrm{rect}(pq)$ is defined to be the axis-parallel rectangle with diagonal pq.

Definition 3.1 (LR Measure) *The* **LR measure** η *of rectangle* $\mathrm{rect}(vw)$*, for vertices* $v \in V'$ *and* $w \in E''$*, is defined as follows. If* w *is not a special vertex, it is* $\mathrm{area}(\mathrm{rect}(vw))$*. If* w *is a special vertex,* $\eta(\mathrm{rect}(vw))$ *is the area of the LR for vertex* v *and the edge* $e \in E'$ *containing* w*.*

For a given v and e, the LR can clearly be computed in constant time (see Section 2.3).

Lemma 3.1 *Let polygon* P' *and* y*-monotone chains* V' *and* E' *satisfy Property I with respect to* P*,* V*, and* E*. Then the LCR for* P'*,* V'*, and* E' *lies inside* P*, and it is at least as large as the LCR for* P*,* V*, and* E*.*

Proof: Let R be the LCR for P, V, and E, and let R' be the LCR for P', V', and E'. By Property I, every vertex-edge rectangle for P, V, and E is a vertex-edge rectangle for P', V', and E'. Therefore $\mathrm{area}(R') \geq \mathrm{area}(R)$. Since $P' \subseteq P$, $R' \subseteq P$. ■

Lemma 3.2 *Let polygon* P' *and* y*-monotone chains* V' *and* E' *satisfy Properties II and III. Let* E'' *and* η *be as defined above. Then the LECR for* $\mathrm{vertices}(P')$*,* $\mathrm{vertices}(V')$*,* $\mathrm{vertices}(E'')$*, and measure* η *can be transformed in constant time into a LCR for* P'*,* V'*, and* E'*.*

Proof: Let R' be a LCR for P', V', and E', and let R'' be the LECR for $\mathrm{vertices}(P')$, $\mathrm{vertices}(V')$, $\mathrm{vertices}(E'')$, and measure η.

We first prove the following claim: $\eta(R'') = \mathrm{area}(R')$.

$\eta(R'') \geq \mathrm{area}(R')$: Let vq be the diagonal of R', where q lies on edge e of E'. If q is an endpoint of e, then $\mathrm{area}(R') = \eta(R')$ by the definition of η. Also, $\eta(R'') \geq \eta(R')$ because R'' is the LECR under the measure η. Thus, $\eta(R'') \geq \mathrm{area}(R')$. If q is not an endpoint of e, let w be the special vertex of e. By Property II, since v can see q, v can see w. Since $\eta(\mathrm{rect}(vw))$ is equal to the area of the largest vertex-edge rectangle for v and e, $\eta(\mathrm{rect}(vw)) \geq \mathrm{area}(R')$. Again, since R'' is the LECR, $\eta(R'') \geq \eta(\mathrm{rect}(vw))$, and thus $\eta(R'') \geq \mathrm{area}(R')$.

$\mathrm{area}(R') \geq \eta(R'')$: It suffices to show that, given R'', we can construct $\mathrm{rect}(vq)$, where $\mathrm{area}(\mathrm{rect}(vq)) = \eta(R'')$, where v is a vertex of V', and where q lies on an edge of E'. Let vw be the diagonal of R''. If w is not special, then let $q = w$. If w is special, then, by Property II, since v can see w, v can see all of edge e containing w, and thus v can see q where vq is the diagonal of the LR for v and e. Thus, $\mathrm{area}(\mathrm{rect}(vq)) = \eta(R'')$. Finally, we observe that $\mathrm{area}(R') \geq \mathrm{area}(\mathrm{rect}(vq))$ since R' is the LCR and thus is at least as large as any other vertex-edge rectangle. By transitivity, $\mathrm{area}(R') \geq \eta(R'')$. Putting this together with the inequality from the previous paragraph shows that $\mathrm{area}(R') = \eta(R'')$, which establishes the claim.

The last paragraph of the proof of the claim implies that we can construct a vertex-edge rectangle $\mathrm{rect}(vq)$ in constant time such that $\mathrm{area}(\mathrm{rect}(vq)) = \eta(R'')$. The claim itself establishes $\eta(R'') = \mathrm{area}(R')$, and therefore $\mathrm{area}(\mathrm{rect}(vq)) = \mathrm{area}(R')$. This means that $\mathrm{rect}(vq)$ is either *the* LCR of P', V', and E' or at least *an* LCR. Thus, we can construct R' from R'' in constant time. ■

3.3 Total Monotonicity of the LR Measure

Lemmas 3.1 and 3.2 reduce the LCR problem to a LECR problem with the notion of "size" given in Definition 3.1. A LECR algorithm clearly depends on the notion of "size": the algorithm of Aggarwal and Suri for largest perimeter is very different from their algorithm for largest area [2, 3]. Furthermore, our notion of size does *not* possess the property required for the class of inclusion problems defined in Section 1:

$$\forall Q,\, Q' \in \mathcal{Q}, \qquad Q' \subseteq Q \;\;\Rightarrow\;\; \eta(Q') \leq \eta(Q).$$

Here we assume $\mathcal{Q}$ is the set of all corner rectangles with lower-left corner in $S_{\mathrm{left}} = \mathrm{vertices}(V')$ and upper-right corner in $S_{\mathrm{right}} = \mathrm{vertices}(E'')$. However, η does possess this property if Q is the set of all *empty* corner rectangles with respect to $S = \mathrm{vertices}(P')$.

Fortunately, for polygons with y-monotone chains which we consider here and in [12], almost any known algorithm for the LECR can be used to compute the LECR by merely substituting size for area in the algorithm.

Suppose we number the vertices of V' by decreasing y-coordinate. Similarly, we number the vertices of E''. It is standard to define an *area matrix*[10] M whose entry m_{ij} contains the size of the rectangle with

[10] *Size matrix* would be the more general term.

lower-left corner at vertex $v_i \in \text{vertices}(V')$ and upper-right corner at vertex $w_j \in \text{vertices}(E'')$. The LECR algorithms we use here and in [12] only require that M satisfies a certain *monotonicity* property. Of course, some entries in the area matrix are invalid because v_i and w_j are not rectangularly visible. However, the only property which the algorithms really depend on is the *total monotonicity* property for legal[11] 2×2 minors of the matrix. We define this property and show that M defined using the LR measure of Definition 3.1 satisfies it.

Definition 3.2 ([1]) *M is* **totally monotone**[12] *if, for every $i < i'$ and $j < j'$ corresponding to a legal 2×2 minor, $m_{i'j'} > m_{i'j}$ implies $m_{ij'} > m_{ij}$.*

Lemma 3.3 *If an increasing index in M corresponds to decreasing y-coordinates of the associated vertices, then M defined by the LR measure is totally monotone.*

Proof: It suffices to assume that $w_j \in \text{vertices}(E'')$ and $w_{j'} \in \text{vertices}(E'')$ are special vertices, since we can consider an ordinary vertex to represent an edge of zero length with a special vertex equal to the ordinary vertex. Let e_j and $e_{j'}$ be the edges containing w_j and $w_{j'}$, respectively. Let the LR vertices for the pairs (v_i, e_j), $(v_{i'}, e_j)$, $(v_i, e_{j'})$ and $(v_{i'}, e_{j'})$ be p_1, p_2, p_3, and p_4, respectively. Let A, B, C, D, E, and F be the areas of $\text{rect}(v_i p_1)$, $\text{rect}(v_i p_3)$, $\text{rect}(v_{i'} p_2)$, $\text{rect}(v_{i'} p_4)$, $\text{rect}(v_i p_2)$, and $\text{rect}(v_{i'} p_3)$, respectively. To show monotonicity, it suffices to show that: $B > A \Rightarrow D > C$. To show this we need the intermediate result: $B > E \Rightarrow F > C$.

$$\begin{aligned}
B &> E \\
(p_{3x} - v_{ix})(p_{3y} - v_{iy}) &> (p_{2x} - v_{ix})(p_{2y} - v_{iy}) \\
p_{3x}p_{3y} + v_{ix}(p_{2y} - p_{3y}) &> p_{2x}p_{2y} + v_{iy}(p_{3x} - p_{2x})
\end{aligned}$$

For the next step we need the following: $v_{iy} \geq v_{i'y}$, $p_{2y} \geq p_{3y}$, $v_{i'x} \geq v_{ix}$, and $p_{3x} \geq p_{2x}$. The y-coordinate inequalities are direct consequences of the y-monotonicity of vertex and edge chains. By assumption, we are dealing with a valid 2×2 minor of the matrix. Therefore, there are four distinct empty rectangles which have a lower left vertex in the set $\{v_i, v_{i'}\}$ and an upper right vertex in the set $\{w_j, w_{j'}\}$. By Properties II and III, there are twelve empty rectangles which have a lower left vertex in the set $\{v_i, v_{i'}\}$ and an upper right vertex in the set $\{p_1, w_j, p_2, p_3, w_{j'}, p_4\}$ (although they are not necessarily distinct since p_1 could equal w_j, for instance). The two x-coordinate inequalities must hold in order that rectangles $\text{rect}(v_i p_3)$ and $\text{rect}(v_{i'} p_2)$ be empty.

$$\begin{aligned}
p_{3x}p_{3y} + v_{i'x}(p_{2y} - p_{3y}) &> p_{2x}p_{2y} + v_{i'y}(p_{3x} - p_{2x}) \\
(p_{3x} - v_{i'x})(p_{3y} - v_{i'y}) &> (p_{2x} - v_{i'x})(p_{2y} - v_{i'y}) \\
F &> C
\end{aligned}$$

The definition of a LR implies that: $A \geq E$ and $D \geq F$. $B > A$ and $A \geq E \Rightarrow B > E \Rightarrow F > C$. $D \geq F$ and $F > C \Rightarrow D > C$. Therefore $B > A \Rightarrow D > C$. ∎

Corollary 3.4 *If V' and E' are y-monotone, then M defined by the LR measure is totally monotone.*

Proof: P' satisfies the condition of Lemma 3.3. ∎

3.4 LCR Running Time

Based on Lemmas 3.1 and 3.2 and Corollary 3.4, we make the following claim, which is used in Section 4 and in [12] to establish running times for solving the 2-contact case in a variety of types of polygons.

[11] A legal 2×2 minor contains only entries corresponding to empty rectangles.

[12] In the literature, the term **totally monotone** refers to a matrix which has the total monotonicity property and no illegal entries. Matrices which have the property but have some illegal entries are sometimes referred to as **monotone** (e.g. monotone-single-staircase, monotone-double-staircase). This is confusing, since monotonicity is also presented in the literature as a weaker condition than total monotonicity: a matrix for which the column of the row maximum moves to the right as i increases is monotone, and it is totally monotone only if all 2×2 minors also possess this property. Our terminology removes this confusion.

Claim 3.5 *For an n-vertex polygon P with y-monotone chains V and E, if the following two conditions hold, then the LCR can be found in the same asymptotic running time as the LECR algorithm.*

- *An $O(n)$ vertex polygon P' and with y-monotone chains V' and E' satisfying Properties I-III can be produced from P, V, and E in $O(n)$ time.*
- *The LECR algorithm for $S = \text{vertices}(P')$, $S_{\text{left}} = \text{vertices}(V')$, and $S_{\text{right}} = \text{vertices}(E'')$ depends only on the total monotonicity of the matrix associated with the size measure.*

As stated previously, E'' is E' with a special vertex added at the midpoint of each edge of E'.

In general, we produce P', V', and E' from P, V, and E by projecting vertices of P and replacing some edges of P by inner orthogonal approximations. For this, we assume $O(n)$ preprocessing time to construct horizontal and vertical visibility maps for P. Projection and orthogonalization can be done in $O(n)$ time and add only a linear number of vertices to P. The LECR algorithms we use here and in [12] depend only on the total monotonicity of the area matrix. To obtain the desired running time in each case for finding the LCR, it therefore suffices to construct P', V', and E', show that they satisfy Properties I-III, and establish the running time of the appropriate LECR algorithm.

Implementation Note: We can avoid the use of special vertices if we accept a more complicated definition of the LR measure. For entry m_{ij}, let e be the edge whose upper endpoint is w_j. If all of e is rectangularly visible to v_i, $\eta(\text{rect}(v_i w_j))$ is the area of the LR whose lower-left corner is at v_i and upper-right corner is on e. Otherwise, $\eta(\text{rect}(v_i w_j)) = \text{area}(\text{rect}(v_i w_j))$.

4 LR Algorithm for General Polygons

This section develops an $O(n \log^2 n)$ time divide-and-conquer algorithm for finding the LR in a general polygon with holes. First, the 2-contact framework is applied in Section 4.1 to solve the 2-contact case for a vertically separated, horizontally convex polygon. This type of polygon arises in the merge step of the divide-and-conquer algorithm. Next, we show in Section 4.2 that the LR of a vertically separated, horizontally convex polygon (see Page 1 for the definition) can be found in $O(n \log n)$ time. This $O(n \log n)$ algorithm uses the results of Aggarwal and Suri [2, 3] for the LECR problem. Finally, Section 4.3 gives the full algorithm for general polygons with holes.

4.1 LCR of a Vertically Separated, Horizontally Convex Polygon

In this section, we apply the 2-contact framework to solve the 2-contact case for a vertically separated, horizontally convex polygon. Suppose V is the left chain and E is the right chain. Recall that V and E are y-monotone. Section 4.1.1 gives the transformation from P, V, and E to P', V', and E' that is required by the framework (Section 3.2), and Section 4.1.2 provides the required LECR algorithm. By Claim 3.5, the transformation and the LECR algorithm are all we need to create a LCR algorithm for vertically separated, horizontally convex polygons.

4.1.1 Construction of P', V', and E'

We produce P', V', and E' using the following set of projections followed by orthogonalization (see Figure 6). From each vertex in V, project a vertical ray upwards, adding vertices where the rays hit V, as shown in Figure 6(a). For each vertex in V (including new ones) project a horizontal ray rightward, as in Figure 6(b). Add vertices to E where these rays hit. For each vertex of E (including new ones) project up, adding new vertices, as in Figure 6(c). Now, replace each edge of modified V and E which has positive slope by its inner orthogonal approximation to produce P', V', and E'. This process adds a linear number of new vertices. The final result is illustrated in Figure 6(d).

Proof of Property I: Edges of P which have negative slope are not orthogonalized, so a vertex-edge rectangle of P is also a vertex-edge rectangle of P'. We need only show that if it is empty in P it is empty in P'. Suppose vertex-edge rectangle $\text{rect}(vq)$ is empty in P, but not empty in P'. Then $\text{rect}(vq)$ must contain a vertex c of the orthogonalized upper-left (w.l.o.g.) boundary of P', but it does not intersect edge ab of P,

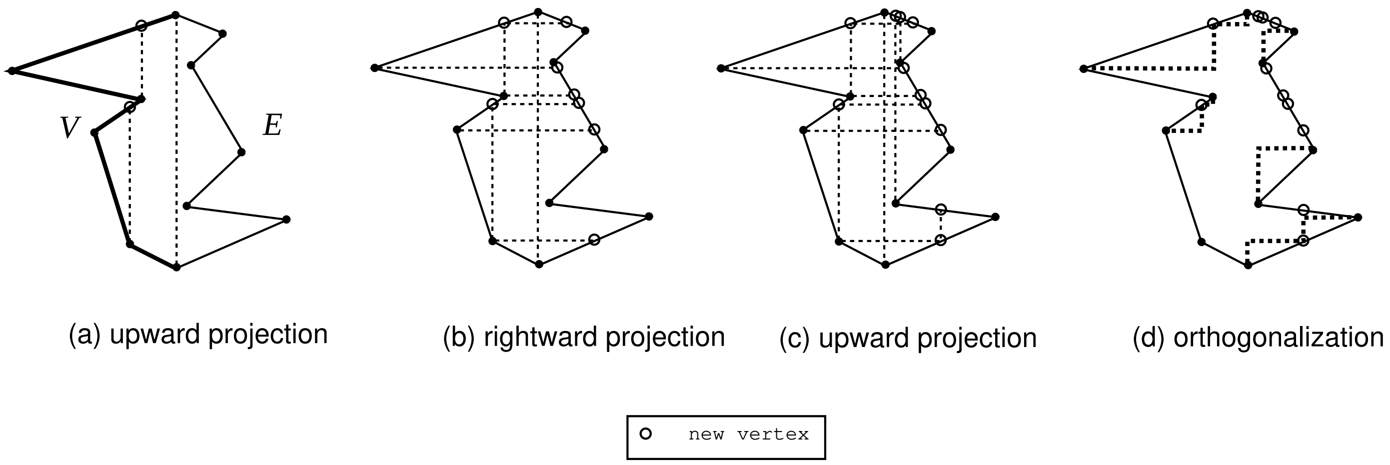

Figure 6: P' Construction for Two Chains of a Vertically Separated, Horizontally Convex Polygon

where acb is the inner orthogonal approximation of ab. Therefore $a_x < v_x < c_x$. But this cannot be because v was projected upwards, so there should be a vertex between a and b with x-coordinate v_x.

Proof of Property II: Let v be a vertex of V' and let q and q' be two vertices on the same edge e of E'. We need to show that if $\text{rect}(vq)$ is empty, then $\text{rect}(vq')$ is also. We consider first the case in which q is below q'. Assume $\text{rect}(vq)$ is empty but $\text{rect}(vq')$ contains a vertex c of the upper-left chain. As we shift q towards q', the top edge of $\text{rect}(vq)$ moves upwards and must hit c before q reaches q'. But $c_y = a_y$, where acb is the inner orthogonal approximation of ab, and a was projected rightwards onto the edge chain. Therefore q hits a projected vertex before it reaches q', contradicting the assumption that they were on the same edge. Now consider the case in which q is above or at the same height as q'. Assume $\text{rect}(vq)$ is empty but $\text{rect}(vq')$ contains a vertex c of the lower-right chain. As we shift q towards q', the right edge of $\text{rect}(vq)$ moves rightwards and must hit c before q reaches q'. But $c_x = b_x$, where acb is the inner orthogonal approximation of ab, and b was projected upward to E'. Therefore q hits a projected vertex before it reaches q', contradicting the assumption that they were on the same edge.

Proof of Property III: The proof of Property I guarantees that rectangle $\text{rect}(vw)$ does not intersect the inner orthogonal approximation of any positive slope subedge of P. It remains to show that no edge with negative slope can cut across $\text{rect}(vw)$. A negative slope edge cannot cross the vertical line separating L and R, so it must cross either the upper-right or lower-left corner, contradicting the y-monotonicity of one of the chains.

4.1.2 LECR Algorithm

Aggarwal and Suri have an $O(n \log^2 n)$ algorithm for the LER (Largest Empty Rectangle) problem for points[2, 3]. They use a divide-and-conquer approach which partitions the set S of points into two subsets S_{left} and S_{right} about a vertical line, and recursively finds the LER in S_{left} and S_{right}. The merge step requires finding the LECR (Largest Empty Corner Rectangle) whose lower-left corner is in S_{left} and upper-right corner is in S_{right}. They solve the LECR problem in $O(n \log n)$ time by forming an area matrix whose legal 2×2 minors are monotone and by applying fast searching techniques to this matrix. Their proof of this LECR algorithm relies only on the monotonicity property. We reimplement this LECR algorithm by substituting our LR measure η (see Definition 3.1) instead of the area measure. We run this modified algorithm on inputs $S_{\text{left}} = \text{vertices}(V')$ and $S_{\text{right}} = \text{vertices}(E'')$.

By Claim 3.5, the transformation of the previous section and the modified LECR algorithm yield an $O(n \log n)$ time algorithm for finding the LCR in a vertically separated, horizontally convex polygon.

4.2 LR of a Vertically Separated, Horizontally Convex Polygon

Theorem 4.1 *The LR in an n-vertex vertically separated, horizontally convex polygon (or horizontally separated, vertically convex polygon) can be found in* $O(n \log n)$ *time.*

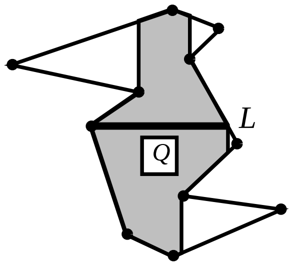

Figure 7: Orthogonally Convex Polygon at Merge Step

Proof: We treat the vertically separated, horizontally convex polygon P, w.l.o.g. The 2-contact case consists of eight LCR subcases. The results of Section 4.1 show that each LCR problem can be solved in $O(n \log n)$ time. The remaining cases involve either 3 or 4 contacts. We claim these cases can be solved by an $O(n \log n)$ time divide-and-conquer algorithm[13].

Lemma 4.2 *The 3 and 4-contact LRs for an n-vertex vertically separated, horizontally convex polygon P can be found in* $O(n \log n)$ *time.*

Proof: We use a divide-and-conquer algorithm which, at each step, partitions the vertex set using a horizontal line L into two sets, each of size at most $\lfloor n/2 \rfloor + 4$. We determine the endpoints of L in linear time by examining all the edges of the polygon. We construct the polygon above L and the polygon below L in linear time by walking around the boundary of P. Then we recursively find the 3 and 4-contact LRs above L and the 3 and 4-contact LRs below L. The merge step requires that we find the 3 and 4-contact LRs intersecting L. Let R_L denote the larger of the two 3 and 4-contact LRs intersecting L. Let Q be the largest polygon inside P and containing L that is monotone with respect to L (see Figure 7).

Lemma 4.3 $R_L \subseteq Q$*. Furthermore, Q is orthogonally convex, can be constructed in $\Theta(n)$ time, and has* $O(n)$ *vertices.*

Proof of Lemma: Any axis-parallel rectangle R intersecting L must be such that each point $p \in R$ is vertically visible to L; hence $R_L \in Q$. To construct Q we first supplement the vertices of P with the extra points obtained from the precomputed vertical visibility map. We then claim that two simple traversals of P suffice to construct Q. Let l be the left endpoint of L, and r the right endpoint. The first traversal is counterclockwise from l to r to construct the bottom portion of Q; the second is clockwise from l to r to build the top part. We begin the counterclockwise traversal by following the downward projection of l until it hits the boundary of P. Then we follow the boundary of P unless we encounter either 1) a reflex extreme vertex that is supported from the right by a vertical line, 2) a vertex which is the bottom endpoint of a vertical visibility line emanating from a reflex extreme vertex that is supported from the left by a vertical line, or 3) the x value of r. In case (1), we follow the visibility line downwards to the boundary of P. In case (2) we follow it upwards to the associated reflex extreme vertex. In case (3), we proceed to r, and terminate the traversal. Constructing the top part of Q is similar.

The visibility map introduces at most one new vertex for each vertex of P, so Q has $O(n)$ vertices. We visit each vertex at most once during each sweep, so the algorithm requires $\Theta(n)$ time. To show Q is orthogonally convex, let b and t be the lowest and highest points (respectively) on P that are visible to L. The counterclockwise sweep builds an xy-monotone path from l to b and from b to r. Similarly, the clockwise sweep builds an xy-monotone path from l to t and from t to r. Since the result is a polygon consisting of four xy-monotone chains, such that $l_x \leq b_x \leq r_x$ and $l_x \leq t_x \leq r_x$, it is orthogonally convex.

This completes the proof of Lemma 4.3. ■

[13]Note that it is also possible to solve the 3 and 4-contact case in $O(n)$ time using a sweep-line algorithm, but that does not improve the overall running time.

The 3 and 4-contact LRs in the orthogonally convex polygon Q can be found in $\Theta(n)$ time using a sweep-line algorithm. The algorithm is essentially the same as that used by McKenna *et al.*[20] to obtain the same time bound for orthogonal, orthogonally convex polygons. Details appear in [12].

Now we argue that if the LR in P intersects L and is a 3 or 4-contact LR, it is also a 3 or 4-contact LR in Q. This is because, if a rectangle r has at least three contacts with P, it has at least three contacts in Q.

The running time of the algorithm therefore satisfies the recurrence $T(n) \leq 2T(\lfloor n/2 \rfloor + 4) + \Theta(n)$, which gives an $O(n \log n)$ algorithm for finding the 3 and 4-contact LRs.

This completes the proof of the Lemma 4.2. ■

This completes the proof of Theorem 4.1. ■

4.3 LR of a General Polygon with Holes

Theorem 4.4 *The LR in an n-vertex general polygon can be found in* $O(n \log^2 n)$ *time.*

Before giving the proof, we discuss a difficulty which arises in constructing a partitioning line for a divide-and-conquer algorithm for finding the LR in a general polygon. If the polygon did not have holes, we could apply a corollary of Chazelle's polygon-cutting theorem [8] to find a single vertical line segment within P which partitions the boundary of P into two pieces, each containing less than $2n/3$ vertices. Because we allow holes, we cannot subdivide the boundary of P into two pieces using a single vertical line segment; we must partition it using multiple line segments. Let L be a vertical line which partitions the vertices of P into two sets, each of size roughly $n/2$, and suppose L is partitioned into k pieces $L_1, L_2, \ldots, L_k$ by the interior of the polygon. We want to split P into left and right subpolygons P_{left} and P_{right}, recursively find the LR in each subpolygon, and then perform a merge step in which we find the LR intersecting L_i, for $1 \leq i \leq k$. However, in such an approach, the fact that the endpoints of L_i are not vertices of P means we add $2k$ vertices each time we recurse.

McKenna *et al.* [20] observed that, if P is an orthogonal polygon with holes, one need not add $2k$ new vertices if the following technique is used. Before the start of the divide-and-conquer algorithm, preprocess P so that all vertical projections (internal to P) of vertices of P are vertices. At each step of the divide-and-conquer algorithm, construct a trapezoid Q_i corresponding to each L_i as follows. L_i intersects two edges of P; these edges are vertically visible from each other. Because of the preprocessing, the left endpoints of these edges can be joined by a vertical line segment l_i, and their right endpoints can be joined by a vertical line segment r_i[14]. Segments l_i and r_i contain only points which are internal to or on the boundary of P. Let Q_i be the (empty) trapezoid bounded on the left by l_i and on the right by r_i (see Figure 8), and let $Q = \cup Q_i$. McKenna *et al.* observe that, if the LR does not intersect L_i, then it does not contain any point in the interior of Q_i. This allows them to redefine P_{left} and P_{right} to be completely disjoint by removing Q from consideration. Unfortunately, their observation about Q_i does not hold in the non-orthogonal case. We overcome this in the proof below by considering rectangles which cross either l_i or r_i and finding the LR in Q_i.

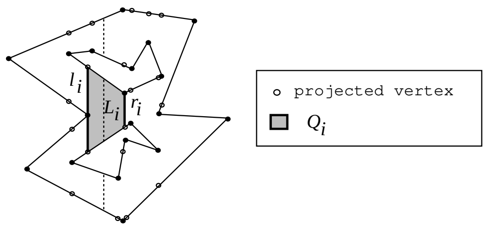

Figure 8: Construction of Trapezoid Q_i

[14] In a degenerate case when vertices of P lie on L_i, one can treat this as if $L_i = l_i$, and arbitrarily choose Q_i to be the trapezoid to the right of the partitioning line segment L_i. It is easy to show that this implies that no more than 1/2 the vertices of P on L end up on the boundary of P_{left}. Therefore, $|P_{\text{left}}| \leq 3n/4$.

Proof: We preprocess P to construct horizontal and vertical visibility maps and to add the internal vertical projections of vertices. Our divide-and-conquer algorithm partitions the vertices of P (both original and vertical projections) at each step using a vertical line L into two sets, each of size at most $\lceil n/2 \rceil$. Suppose that L is partitioned into k pieces $L_1, L_2, \ldots, L_k$ by the interior of the polygon. For $1 \leq i \leq k$, we define l_i, r_i, and Q_i as above. As before, let $Q = \cup Q_i$, and construct subpolygons P_{left} and P_{right} of $P \setminus Q$ to the left and right of L such that they do not share any vertices and each has no more than $\lceil n/2 \rceil$ vertices. We recursively find the LR in P_{left} and P_{right}. In the merge step, for $1 \leq i \leq k$, we find the LR in Q_i, the LR of P which intersects l_i, and the LR of P which intersects r_i.

To show that this algorithm finds the LR of P, we argue as follows. If the LR does not intersect the interior of Q, then it lies either in P_{left} or P_{right}, so, at the divide step, we can recursively find the LR in P_{left} and P_{right}. If the LR intersects the interior of Q, we can find it during the merge step as follows. If the LR lies entirely within Q, we can find it by finding the LR in each Q_i. If the LR is not entirely within Q, it must cross some l_i or r_i.

We now show that the algorithm requires $\mathrm{O}(n \log^2 n)$ time. First, we note that an $\mathrm{O}(n \log n)$ sweep algorithm suffices for constructing the visibility maps and projecting the vertices. This need only be done once before the start of the divide-and-conquer algorithm, and the maps can be updated in linear time at each step. We can determine the endpoints of L_i, $1 \leq i \leq k$, in linear time by examining all the edges of the polygon and using the vertical visibility map. Because we have the visibility maps, P_{left} and P_{right} can be constructed in $\mathrm{O}(n)$ time. P_{left} and P_{right} each have size $\leq \lceil n/2 \rceil$. Since Q_i is a trapezoid, the LR in Q_i can be found in $\mathrm{O}(1)$ time, so the LR in Q can be found in $\mathrm{O}(n)$ time. We describe below how to find the LR intersecting l_i, $1 \leq i \leq k$, in a total of $\mathrm{O}(n \log n)$ time. The technique for r_i is the same.

Lemma 4.5 *For an n-vertex polygon P, the LR which intersects l_i, for $1 \leq i \leq k$, can be found in a total of $\mathrm{O}(n \log n)$ time.*

Proof of Lemma: Let H_i be the largest polygon in P which is horizontally visible from l_i. Let H_i have n_i vertices.

Claim 4.6 *The LR which intersects l_i is a subset of H_i. Furthermore, H_i is a vertically separated, horizontally convex polygon, and can be constructed in $\mathrm{O}(n_i)$ time.*

Proof of Claim: The proof is similar to the proof of Lemma 4.3. We use the horizontal visibility map and two traversals to construct H_i. H_i is a vertically separated, horizontally convex polygon because each traversal builds a chain that is monotone with respect to the vertical line l_i.

This establishes Claim 4.6. ∎

Claim 4.7 $\sum_{i=1}^{k} n_i \in \mathrm{O}(n)$.

Proof of Claim: The horizontal visibility map on P partitions the interior of P into a set of trapezoids T. Let $T_i \subseteq T$ be the set of trapezoids that contains a point in the interior of l_i; hence $O(n_i) \in O(|T_i|)$. Consider i, j such that $i \neq j$ and l_j is to the right of l_i (w.l.o.g.). The only points which the interior of l_i sees to its right (and to the left of L) are points in Q_i. Since the interior of Q_i is empty, this means that no point in the interior of l_j is horizontally visible from a point in the interior of l_i. Therefore, l_i and l_j cannot share a trapezoid. Thus, each trapezoid in T is associated with at most one i, and therefore $\sum_{i=1}^{k} \mathrm{O}(|T_i|) \in \mathrm{O}(|T|) \in \mathrm{O}(n)$.

This establishes Claim 4.7. ∎

By Theorem 4.1, we can find the LR in H_i in $\mathrm{O}(n_i \log n_i)$ time. Combining this result with Claim 4.7 implies that we can find the LR which intersects l_i, for $1 \leq i \leq k$, in a total of $\mathrm{O}(n \log n)$ time, which establishes Lemma 4.5. ∎

Lemma 4.5 implies that the merge step can be performed in $\mathrm{O}(n \log n)$ time. This yields the following recurrence: $T(n) \leq 2T(\lceil n/2 \rceil) + \mathrm{O}(n \log n)$, which gives $\mathrm{O}(n \log^2 n)$ time for the combined algorithm.

This completes the proof of Theorem 4.4. ∎

Note: In a degenerate case when vertices of P lie on L_i (see footnote on Page 14), the recurrence becomes $T(n) \leq T(n_1) + T(n_2) + \mathrm{O}(n \log n)$, where $n_1 + n_2 = n$ and $n_1, n_2 \leq \lceil 3n/4 \rceil$, which still has the solution $\mathrm{O}(n \log^2 n)$.

This completes the presentation of the LR algorithm for general non-orthogonal polygons with holes. The reader is referred to [12] for LR algorithms for other types of polygons.

5 Lower Bounds

Here we establish lower bounds of time in $\Omega(n \log n)$ for finding the LR in both self-intersecting polygons and general polygons with holes. The latter result gives us both a lower bound of $\Omega(n \log n)$ and an upper bound of $\mathrm{O}(n \log^2 n)$ for general polygons with holes.

These lower bounds contrast with the $\Theta(n)$ time result achievable for the corresponding enclosure problems [15].

5.1 Self-Intersecting Polygons

We prove a lower bound of time in $\Omega(n \log n)$ for finding the LR in a self-intersecting polygon.

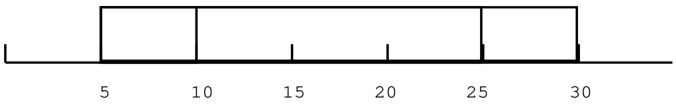

Figure 9: Orthogonal self-intersecting polygon constructed for input = 10, 5, 30, 25

Theorem 5.1 *Finding the LR in an n-vertex self-intersecting polygon requires time in $\Omega(n \log n)$ in both the linear and algebraic decision tree models.*

Proof: We reduce the MAX-GAP problem[16] [4] to the LR problem for self-intersecting orthogonal polygons. Consider an instance of MAX-GAP: given a set of n real numbers $x_1, x_2, ..., x_n$, we must find the maximum difference between two consecutive numbers in the sorted list. We construct from this set, in linear time, a self-intersecting orthogonal polygon of unit height as follows: each x_i in the sequence corresponds to a rectangle $r_i = [(x_1, 0), (x_1, 1), (x_i, 1), (x_i, 0)]$. We start the construction from $(x_1, 0)$, complete the degenerate rectangle r_1, then construct $r_2, \ldots, r_n$ (as shown in Figure 9). This construction results in a self-intersecting polygon, with the property that the area of the LR included in it is the solution to the corresponding MAX-GAP problem, thus proving the theorem. ■

5.2 General Polygons with Holes

McKenna *et al.* [20] have given a lower bound of time in $\Omega(n \log n)$ for finding the LR in a general polygon with degenerate (zero area) holes. Aggarwal and Suri [2] have given the same lower bound for LER. Using symbolic perturbation [14, 31], both of these can be extended to lower bounds on the computation of the LR in a general polygon. For the degenerate case, McKenna *et al.* use a reduction from the *even distribution problem*: given a set of n real numbers $x_1, x_2, x_3, \ldots, x_n$ (not sorted), determine if there exist adjacent x_i and x_j in the sorted list such that $x_j - x_i > 1$. Their reduction involves the construction of a long horizontal rectangle with vertical "slits" at each x_i. These slits can be thought of as degenerate rectangular holes. Given a slit $(x_i, y_b)(x_i, y_t)$ we can "expand" it to a rectangle with diagonal $(x_i, y_b)(x_i + \epsilon, y_t)$ where $\epsilon > 0$. Of course, if we choose ϵ greater than the value of the minimum gap between points (possibly another $\Omega(n \log n)$ problem), then neighboring slits will overlap and the polygon will be self-intersecting; in effect, we have to know the minimum gap in order to compute the maximum gap.

[15] It is interesting to note that the dual problems of largest empty circle and smallest enclosing circle for a set of points also have different lower bounds. The largest empty circle can be constructed in $\Theta(n \log n)$ time, and the smallest enclosing circle can be found in $\Theta(n)$ time [28].

[16] In both the linear and algebraic decision tree models (if not enhanced to include floor and ceiling functions), MAX-GAP has a lower bound of $\Omega(n \log n)$.

Symbolic perturbation rescues us from this chicken and egg problem by allowing ϵ to remain unevaluated until after we have run the LR algorithm. Given an algorithm for computing the LR of a polygon with non-degenerate holes, we modify the way the algorithm evaluates and tests the sign of arithmetic expressions. Since some of the inputs involve ϵ, the arithmetic expressions of the modified algorithm are polynomials in ϵ. For these, the modified algorithm computes the sign by taking the sign of the first (lowest degree in ϵ), non-zero coefficient. We observe that:

- there exists a value of ϵ such that the signs computed by the modified algorithm equal the signs computed by the unmodified algorithm on this value of ϵ (this is the basic theory of symbolic perturbation);
- the running time of the modified algorithm is a constant times the running time of the unmodified algorithm on that value of ϵ.

Hence, any algorithm for the LR in a general polygon can be used to test even distribution via a linear time reduction. Hence the construction of the LR in a general polygon has an $\Omega(n \log n)$ lower bound. We could have also reduced the LER problem to the LR problem by replacing every point in the LER instance by a square of size ϵ.

6 Applications

When a polygon is nearly rectangular, the LR provides a good inner approximation. Many LR applications have surfaced in our automatic marker-making project for the apparel industry. We briefly describe two of them in this section. The goal of our project is to automate the task of laying out polygonal apparel pattern pieces on a rectangular sheet of cloth of fixed width and minimal length [22, 23]. In the apparel industry, this layout is called a *marker*.

Pants markers consist of large *panel* pieces and smaller *trim* pieces. We have a heuristic method that does a good job placing the larger panel pieces [23]. We use LRs during the trim placement stage. Figure 10 shows a rectangular marker with the large panels already placed. The smaller trim pieces to the left of the marker rectangle must be placed in the *gaps* of unused material between adjacent panels. We compute the LR of each trim piece and use that inner approximation as part of our algorithm that decomposes the gaps into smaller, more manageable regions [11]. The decomposition algorithm is part of software which we have licensed to a CAD firm in the apparel industry. We have also considered computing the LR of each gap region and then packing the nearly rectangular trim pieces into the LRs using techniques from the rectangle packing domain. We do not currently use this strategy in our trim placement heuristic.

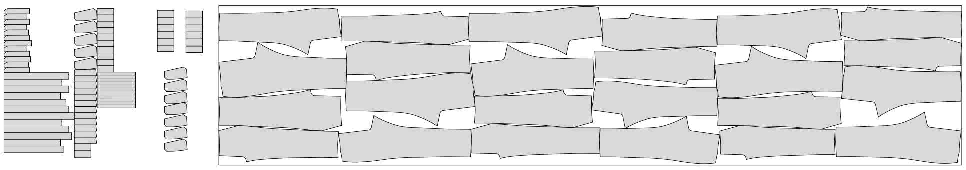

Figure 10: Pants Marker with Placed Panels and Unplaced Trim

7 Conclusion

We have presented the first algorithmic result for finding the LR in non-orthogonal general polygons with holes: an $O(n \log^2 n)$ time algorithm. We have also established a lower bound of time in $\Omega(n \log n)$ for this type of polygon. In this paper and in [12] we have shown, for a variety of non-orthogonal polygons, that the LR can be found in the same asymptotic running time as the best algorithms for their orthogonal counterparts.

Pursuing more efficient algorithms for finding the *exact* LR is certainly one direction for future work. Another direction of practical importance is to find a fast *approximate* LR algorithm. Such an algorithm would be very helpful in our applications.

This paper has described a general mechanism for developing LR algorithms for non-orthogonal polygons. The mechanism has three key components: 1) the idea of "determining sets" of contacts, used to characterize the LR for a general polygon with holes, 2) identifying the determining set of contacts corresponding to the one subproblem which dominates the running time for finding LRs in a variety of types of polygons, and 3) a general framework for solving the dominant subproblem using a new notion of rectangle size. The framework involves creating a partially orthogonal polygon to which we apply a known algorithm for solving the LECR problem. To develop an LR algorithm, we solve the key subproblem using our framework and then solve the remaining subproblems. There may be other classes of polygons, in addition to the ones we examine here and in [12], that are amenable to this general method.

It is interesting that in order to solve the LR problem we need a notion of rectangle size which does not possess the following important property held by both area and perimeter for rectangles: $\forall Q, Q' \in \mathcal{Q}, Q' \subseteq Q \Rightarrow \eta(Q') \leq \eta(Q)$. We think it might be useful in other instances to consider such nonstandard size measures.

Acknowledgments

The authors gratefully acknowledge the helpful comments made by Pankaj Agarwal, Zhenyu Li, Marios Mavronicolas, and Binhai Zhu. We also acknowledge the background information provided by David Dobkin and Joseph O'Rourke, and the unpublished manuscript provided by Alok Aggarwal.

References

[1] A. Aggarwal, M. M. Klawe, S. Moran, P. Shor, and R. Wilber. Geometric applications of a matrix-searching algorithm. *Algorithmica*, 2:195–208, 1987.

[2] A. Aggarwal and S. Suri. Fast Algorithms for Computing the Largest Empty Rectangle. In *Proceedings of the 3rd Annual ACM Symposium on Computational Geometry*, pages 278 – 290, 1987.

[3] A. Aggarwal and S. Suri. Fast Algorithms for Computing the Largest Empty Rectangle. Personal communication, 1992.

[4] A. Aggarwal and J. Wein. *Computational Geometry Lecture Notes for MIT 18.409*. 1988.

[5] N. Amenta. Largest Volume Box is Convex Programming. Personal communication, 1992.

[6] F. Aurenhammer. Voronoi Diagrams - A Survey of a Fundamental Geometric Data Structure. *ACM Computing Surveys*, 23(3):345–406, 1991.

[7] J. S. Chang and C. K. Yap. A Polynomial Solution for the Potato-peeling Problem. *Discrete and Computational Geometry*, 1:155–182, 1986.

[8] B. Chazelle. A theorem on polygon cutting with applications. In *Proceedings of the 23rd IEEE Symposium on the Foundations of Computer Science*, pages 339–349, 1982.

[9] B. Chazelle, R. L. Drysdale III, and D. T. Lee. Computing the largest empty rectangle. *SIAM Journal on Computing*, 15:300–315, 1986.

[10] L. P. Chew and R. L. Drysdale III. Voronoi Diagrams Based on Convex Distance Functions. In *Proceedings of the 1st ACM Symposium on Computational Geometry*, pages 235–244, 1985.

[11] K. Daniels and V. J. Milenkovic. Limited Gaps. In *Proceedings of the 6th Canadian Conference on Computational Geometry*, pages 225–230, 1994.

[12] K. Daniels, V. J. Milenkovic, and D. Roth. Finding the Largest Rectangle in Several Classes of Polygons. Technical Report 22–95, Center for Research in Computing Technology, Division of Applied Sciences, Harvard University, 1995.

[13] N. DePano, Y. Ke, and J. O'Rourke. Finding Largest Inscribed Equilateral Triangles and Squares. In *Proceedings of the 25th Allerton Conference on Communications, Control, and Computing*, pages 869–878, 1987.

[14] H. Edelsbrunner and E. P. Mücke. Simulation of Simplicity: A technique to cope with degenerate cases in geometric algorithms. *ACM Transactions on Graphics*, 9:66–104, 1990.

[15] L. Guibas, J. Hershberger, D. Leven, M. Sharir, and R. E. Tarjan. Linear-Time Algorithms for Visibility and Shortest Path Problems Inside Triangulated Simple Polygons. *Algorithmica*, 2:209–233, 1987.

[16] F. Hwang. An $O(n \log n)$ algorithm for rectilinear minimal spanning trees. *Journal of the ACM*, 26:177–182, 1979.

[17] M. M. Klawe and D. J. Kleitman. An Almost Linear Time Algorithm for Generalized Matrix Searching. *SIAM Journal of Discrete Mathematics*, 3(1):81–97, 1990.

[18] D. Lee. Medial axis transformation of a planar shape. *IEEE Transactions on PAMI*, 4:363–369, 1982.

[19] D. Lee and C. Wong. Voronoi Diagrams in L_1 (L_∞) Metrics With 2-Dimensional Storage Applications. *SIAM Journal on Computing*, 9(1):200–211, 1980.

[20] M. McKenna, J. O'Rourke, and S. Suri. Finding the largest rectangle in an orthogonal polygon. In *Proceedings of the 23rd Allerton Conference on Communication, Control, and Computing*, pages 486–495, 1985.

[21] E. Melissaratos and D. Souvaine. On Solving Geometric Optimization Problems Using Shortest Paths. In *Proceedings of 6th Annual ACM Symposium on Computational Geometry*, pages 350–359, 1990.

[22] V. J. Milenkovic, K. Daniels, and Z. Li. Automatic Marker Making. In *Proceedings of the 3rd Canadian Conference on Computational Geometry*, 1991.

[23] V. J. Milenkovic, K. Daniels, and Z. Li. Placement and Compaction of Nonconvex Polygons for Clothing Manufacture. In *Proceedings of the 4th Canadian Conference on Computational Geometry*, pages 236–243, 1992.

[24] A. Naamad, W. L. Hsu, and D. T. Lee. On the maximum empty rectangle problem. *Discrete Applied Mathematics*, 8:267–277, 1984.

[25] J. O'Rourke. *Art Gallery Theorems and Algorithms.* Oxford University Press, 1987.

[26] J. O'Rourke. *Computational Geometry in C, draft.* Cambridge University Press, 1993.

[27] M. H. Overmars and D. Wood. On rectangular visibility. *Journal of Algorithms*, 9:372–390, 1988.

[28] F. Preparata and M. Shamos. *Computational Geometry: An Introduction.* Springer-Verlag, New York, 1985.

[29] S. Schuierer, G. J. E. Rawlins, and D. Wood. A Generalization of Staircase Visibility. In *Proceedings of the 3rd Canadian Conference on Computational Geometry*, pages 96–99, 1991.

[30] D. Wood and C. K. Yap. The Orthogonal Convex Skull Problem. *Discrete and Computational Geometry*, 3:349–365, 1988.

[31] C. K. Yap. A geometric consistency theorem for a symbolic perturbation scheme. *Journal of Computer and System Sciences*, 40:2–18, 1990.