ON ENVELOPES OF ARRANGEMENTS OF LINES†

D. Eu, E. Guévremont* and G.T. Toussaint

School of Computer Science
McGill University
Montréal, Québec, Canada

*School of Computing Science
Simon Fraser University
Burnaby, B. C., Canada

Abstract

The envelope of an arrangement of lines is the polygon consisting of the finite length segments that bound the infinite faces of the arrangement. We study the geometry of envelope polygons (simple polygons that are the envelope of some arrangement). We show that envelope polygons are L-convex and derive several geometric properties of envelopes. Given an envelope polygon P of n vertices, we show how to sort its edges by slope in O(n) time (for unrestricted simple polygons this problem has complexity Ω(n log n)). Using this result, we give a linear time procedure to verify if a given polygon is an envelope polygon. We introduce a hierarchy of classes of arrangements of lines based on the number of convex vertices of their envelopes. In particular, we look at a class called *sail arrangements*, which we prove has properties that enable us to solve a number of problems optimally. Given a sail arrangement consisting of n lines (and $O(n^2)$ vertices), we show how the prune-and-search technique can be used to determine all the convex vertices of its envelope in O(n) time. This implies that the intersection point with minimum or maximum x-coordinate, the diameter and the convex hull of sail arrangements (problems that also have Ω(n log n) complexity for arbitrary arrangements) can be found in O(n) time. We show, in spite of this, that the problem of constructing the *full* envelope of a sail arrangement still has a lower bound of Ω(n log n). We also examine the existence of hamiltonian circuits through the intersection points of a non-trivial subclass of sail arrangements. Finally, we establish an Ω(n log n) time lower bound for the problem of constructing a hamiltonian circuit through the vertices of an arrangement of n lines, where only the vertices where a turn is made need be output.

---

†Research supported by NSERC grant # OGP0009293 and FCAR grant # 93-ER-0291.

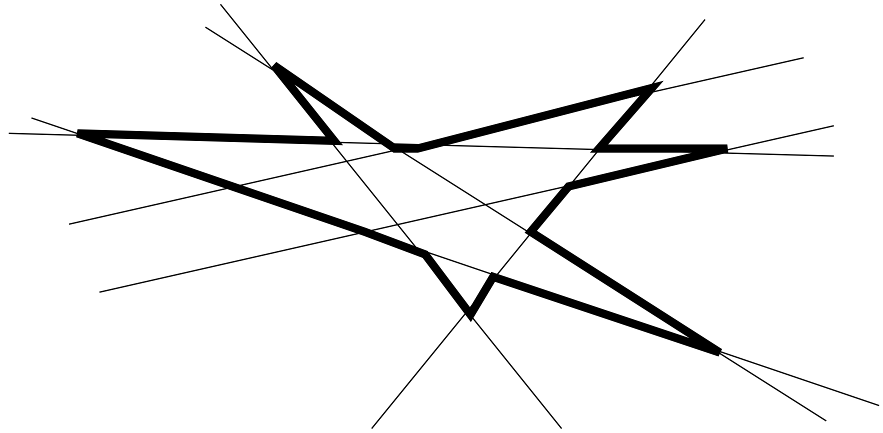


Figure 1.1. The envelope of an arrangement of lines

1. Introduction

The *arrangement* of a set of n distinct lines in the plane ($\Re^2$) is the partition of the plane induced by the lines into $O(n^2)$ vertices, $O(n^2)$ edges and $O(n^2)$ faces[1]. The *vertices* are the intersection points of the lines, the *edges* are the connected components of the lines when the vertices are removed and the *faces* are the connected components of $\Re^2$ when the lines are removed. The *envelope* of an arrangement A, denoted as E(A), is the polygon formed by the union of all the bounded edges of the unbounded faces of A, where adjacent segments that belong to the same line are collapsed into a single edge. One can imagine taking the arrangement A, removing segments that extend to infinity, and tracing around the outer boundary of the resulting figure to form the envelope (see figure 1.1 for an example of an envelope). In 1985, Ching and Lee [CL85] established an $\Omega(n \log n)$ lower bound for the problem of computing the envelope of an arrangement of n lines under the algebraic tree model of computation [BO83]. Since then, Suri [Su85], Vegter [Ve87] and Keil [Ke91] have contributed algorithms that achieve this bound. It is worth noting that Keil's simple and elegant algorithm [Ke91] runs in O(n) time if the n lines of the arrangement are given in sorted order of slope.

In this paper, we explore the geometry of envelopes. The analysis of morphological properties of arrangements of lines (also referred to as line patterns) is of considerable interest to geographers, nuclear physicists and urban planners among others (see [To91] for references). In studying envelopes, which contain a great deal of information about arrangements, we hope to gain a better understanding of the morphology of arrangements and perhaps stimulate new ideas of research on the subject to the benefit of the aforementioned specialists. At the same time we contribute to the field of computational geometry by introducing and studying a new class of polygons; that of envelope polygons (simple polygons which are the envelopes of arrangements). We show that properties of envelopes allow us to solve some problems for certain classes of arrangements of

---

1. When there is no ambiguity in the context, we will also use the expression *arrangement of lines* (by abuse of notation) to also mean the set of lines that induces the arrangement.

lines more efficiently than for arbitrary arrangements.

In section 2 we prove several fundamental properties of envelope polygons. These properties are useful in proving subsequent results. Then, we examine the relationship between the class of envelope polygons and some other well-known classes of polygons: convex, star-shaped and L-convex polygons.

In section 3, we present an algorithm to sort by slope in O(n) time the n edges of an envelope polygon (the problem is Ω(n log n) for arbitrary polygons). This algorithm, together with the above mentioned algorithm of Keil [Ke91] is used in section 4 to recognize in linear time if a given polygon is an envelope or not. Given an arbitrary polygon P of n edges, we run our edge sorting algorithm on it (it is guaranteed to always terminate in O(n) time). If the lines (edges) of the output are not sorted, the input could not have been an envelope polygon. Otherwise, we run Keil's algorithm on the sorted set of lines A to construct the envelope E(A). We can then easily check in linear time that P=E(A), the condition for a polygon to be an envelope. Our sorting algorithm is closely related to the well-known Graham scan [Gr72]. This characterization of envelope polygons concludes the first part of this paper.

In section 5, we study envelopes from a different perspective. We show that given an arrangement of lines for which we know some property of its envelope, we can solve some problems more efficiently than for arbitrary arrangements. In fact, we introduce a hierarchy of classes of arrangements of lines based on the number of convex vertices of their envelopes. This approach has proven to be a productive exploratory strategy in the field of computational geometry as witnessed in [ET89], where much progress was made as a result of defining a hierarchy of polygons that possess more structure than arbitrary simple polygons.

In section 6, we show that a certain class of arrangements, which we call *sail arrangements* (the envelope of a sail arrangement has exactly three convex vertices), has properties that allow us to determine the convex vertices of sail envelopes in O(n) time given a sail arrangement of n lines. Consequently, we can solve several other problems regarding the O(n^2) intersection points of a sail arrangement in O(n) time. In particular we can find the intersection point with minimum or maximum x-coordinate as well as the diameter and convex hull of the intersection points. These problems were shown, in [CSSS89] and [CL85] respectively, to have Ω(n log n) lower bounds for arbitrary arrangements under the algebraic tree computation model [BO83]. We show in section 7, in spite of this, that even if we know the convex vertices of the envelope of a sail arrangement, computing the complete envelope remains Ω(n log n), improving on Ching and Lee's result, which holds only for arbitrary arrangements ([CL85]).

In section 8, we show that there are classes of arrangements which have properties that do not hold for arbitrary arrangements. In particular, we look at the problem of computing a hamiltonian circuit through the vertices of an arrangement of lines. It was shown by Hazel Everett that not all arrangements of lines admit a hamiltonian cycle; refer to figure 8.1 for an illustration of a counterexample. We show that nevertheless, there exists a non-trivial subclass of sail arrangements which always admits a hamiltonian cycle. Furthermore, we exhibit an O(n log n) time algorithm to compute such a cycle through the vertices of an arrangement of n lines (only the vertices where a turn is made are output). In section 9, we show that even if a hamiltonian cycle through the vertices of an arrangement of n lines can be specified in o(n log n) space (by only listing the vertices where a turn is made), it must take Ω(n log n) time to compute it. This proves the optimality of the algorithm we give in section 8.

2. Properties of Envelopes

A simple polygon P is an *envelope polygon* if there exists an arrangement of lines A such that P=E(A). It is useful, when discussing envelope polygons, to use the following notation. The *induced arrangement* of a simple polygon P, denoted as IA(P), is the arrangement induced by the set of lines obtained by extending the edges of P to lines. We assume that P is in general position in the sense that no two lines of IA(P) are identical. First we give a characterization of envelope polygons.

Proposition 1. A simple polygon P is an envelope polygon if and only if P=E(IA(P)).

Proof: ("Only if" implication) Suppose that simple polygon P is an envelope polygon. Let A denote the arrangement of lines that yields P, i.e. such that E(A)=P. For every edge e of P, there is a line L of IA(P) colinear to e, hence IA(P) $\subset$ A. Now suppose that there is a line L of A not in IA(P). L is not colinear to any edge of P and therefore does not contribute any line segment to E(A). L can be removed from A without affecting E(A). It follows that E(IA(P)) = E(A), hence that P=E(IA(P)). The "if" implication of the proposition is obvious.♣

This characterization of envelope polygons will be useful in determining if a given polygon is an envelope polygon (section 4). We begin this section by introducing some basic terms we will be using throughout the paper and by uncovering some simple and basic combinatorial and geometric properties of envelope polygons.

Let A = $\{L_0,L_1,...,L_{n-1}\}$ be an arrangement of n lines. We denote by $I(L_i,L_j)$ the intersection of two non-parallel lines L_i and L_j. We classify the vertices of arrangement A as follows. Vertex p = $I(L_i,L_j)$ (i, j $\in$ [0,n-1]) is said to be *extreme* on L_i if every vertex (other than p) lying on L_i lies on one side of p. The vertex p is said to be *critical* if it is extreme on both L_i and L_j; it is *interior* if it is not extreme on either L_i or L_j. Two non-parallel lines in an arrangement are said to be *adjacent* if they are neighbors in the list of the lines of the arrangement sorted by slope. We now characterize the convex vertices of envelope polygons with the following lemma.

Lemma 2.1. Let P be an envelope polygon. A vertex of P is convex if and only if it is a critical vertex of IA(P).

Proof: We first prove the "only if" part of the lemma. Let P = $[p_0,p_1,...,p_{n-1}]$ be an envelope polygon and let p_i be a convex vertex of P. Let e_i and e_{i+1} be the edges that are incident on p_i. Let L_i and L_{i+1} be the lines extended through e_i and e_{i+1} respectively. If p_i is not critical, then p_i is not extreme on at least one of L_i and L_{i+1}. Without loss of generality, suppose that p_i is not extreme on L_i in IA(P). Then there is a vertex p_j of IA(P) on L_i such that p_i lies between p_{i-1} and p_j on L_i. The line segment

$[p_{i-1},p_j]$ must lie in P, by the definition of an envelope. Therefore, p_i cannot be a convex vertex, a contradiction. Hence p_i must be critical.

To see that the "if" part of the lemma holds let p_i be a critical vertex and suppose that it is concave. Let edges e_i and e_{i+1} be incident on p_i. We extend e_i and e_{i+1} to obtain L_i and L_{i+1} respectively. The vertex chain $C = [p_{i+1},p_{i+2},...,p_{i-1}]$ closes P. By the Jordan curve theorem, L_i and L_{i+1} must each intersect the boundary of P on C at extreme points say, p_a and p_b respectively. The colinearity and orientation of p_{i-1},p_i,p_a and p_{i+1},p_i,p_b imply that p_i is not extreme on either L_i or L_{i+1}. Therefore p_i is not critical, a contradiction.♣

Corollary 2.1.1. If P is an envelope polygon then the convex vertices of P are the intersection points of adjacent lines in the list of lines of IA(P) given in sorted order of slope.

Proof: Ching and Lee [CL85] and independently Atallah [At86] showed that the critical vertices of an arrangement are a subset of the intersections of adjacent lines.♣

In [Su85], Suri shows that the envelope of an arrangement of n lines has at most 4n-2 edges. We can improve this bound by noticing that the recent results of [BEPY91] for the horizon theorem also give tight bounds on the complexity of an envelope.

Lemma 2.2. Let P be an envelope polygon such that IA(P) consists of n lines ($n \geq 3$). Then P has at most 3.5n vertices or edges, and this bound is tight up to a constant.

Proof: Envelopes are projectively equivalent to the well-studied horizons or zones of lines in an arrangement of lines. Specifically, the envelope of a set of lines is the zone or horizon of the line at infinity. Thus, the recent bound for the horizon theorem [BEPY91] also gives an upper bound of 3.5n vertices on the complexity of envelope polygons. We present the following example due to Jorge Urrutia (figure 2.1) to show that the bound is tight up to a constant.♣

We now examine the relationship between the class of envelope polygons and some other well-known classes of polygons. In particular, we establish that envelope polygons are L-convex. We strengthen this result by first showing that non-trivial envelopes are not convex and that envelopes need not be star-shaped.

Lemma 2.3. If P is an envelope polygon of more than four vertices then P is not convex.

Proof: (by contradiction) Suppose that $P = [p_0,p_1,...,p_{n-1}]$ is a convex envelope polygon of n (> 4) vertices. Then P must have three consecutive edges, no two of which are parallel. Let e_i, e_{i+1} and e_{i+2} be three such edges. By lemma 2.1, p_i, p_{i+1}, p_{i+2} and p_{i+3} are critical vertices in IA(P) since they are convex in P. Then since e_i and e_{i+2} are not parallel, it follows that e_i and e_{i+2} must intersect each other at their endpoints. Therefore, p_i and p_{i+3} are the same vertex, contradicting the assumption that P has more than four vertices.♣

Corollary 2.3.1. The only convex envelope polygons are triangles and parallelograms; they are the envelopes of trivial arrangements.

Given a simple polygon P, we say that two points x and y in P are *visible* if the line segment

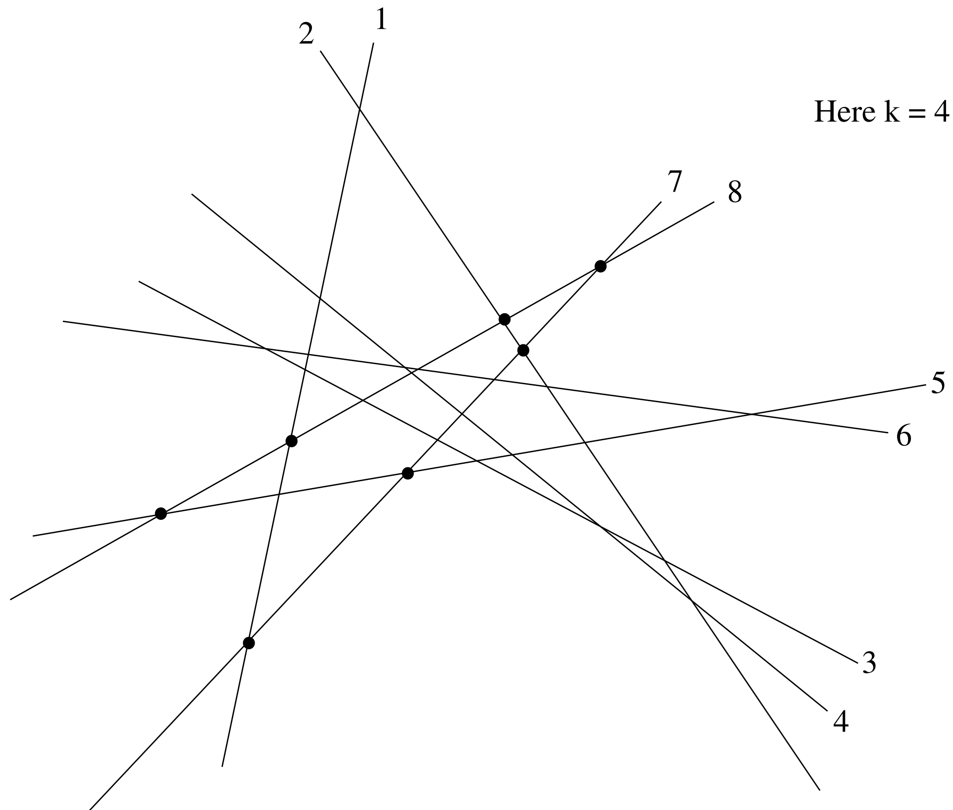


Figure 2.1. Urrutia's example of an arrangement of n lines whose envelope has 3.5n-10 vertices (n even). Lines 2k-1 and 2k add 7 new vertices to the envelope (k>2).

[x,y] lies in P. A simple polygon P is *star-shaped* if there is a point x in P such that for every point y in P, x and y are visible. In [Za75], Zaslavsky conjectured that envelope polygons are star-shaped; the class of star-shaped polygons subsumes the class of convex polygons. We submit the counter-example below (figure 2.2) due to Vegter [Ve87] to show that this is not the case. We show, however, that envelope polygons are L-convex.

A simple polygon P is *L-convex* if for every two points x and y in P, there exists a point q in P such that the pairs (x,q) and (q,y) are each visible. Before we show that envelope polygons are L-convex, we first recall the following fact, a sub-case of a result that can be found in [LPSSSSTWY88].

Lemma 2.4. A simple polygon P is L-convex if and only if for every two *vertices* of P there is a point q visible to both vertices.

We can now state the following.

Lemma 2.5. If P is an envelope polygon, then P is L-convex.

Proof: By lemma 2.4, it is sufficient to show that every two vertices of P are visible from a point inside P. Let p_a and p_b be two arbitrary vertices of P. If p_a and p_b are on the same line in IA(P), then they are visible. Otherwise, there is an intersection point q between a line L_a supporting p_a and a line L_b supporting p_b in IA(P). The path $[p_a,q]$, $[q,p_b]$ is interior to P since P is an envelope.♣

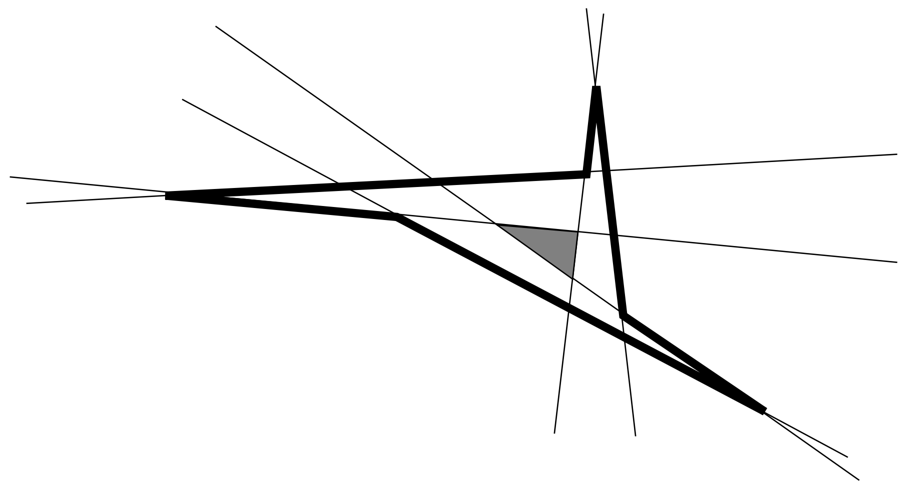


Figure 2.2. An envelope polygon which is not star-shaped.

Whereas envelope polygons have not been studied much, L-convex sets have received considerable attention [HV49]. Properties of L-convex polygons have been exploited in [EAT83] to obtain efficient algorithms for solving a variety of geometric problems. Since envelopes are L-convex, properties of L-convex polygons are useful for answering questions about envelope polygons. In particular, this result implies for example that there is a very simple O(n) time algorithm to triangulate an envelope polygon of n vertices [EAT83].

3. Sorting the Edges of an Envelope Polygon in Linear Time

In this section, we consider the problem of sorting, by slope, the edges of a given simple polygon of n vertices. We start by showing that this problem is Ω(n log n) even for a very restrictive class of polygons, namely, monotonic convex-fans (and hence also for arbitrary simple polygons). A simple polygon P is *convex-fan* if it is star-shaped from a convex vertex; the class of convex-fan polygons was introduced by O'Rourke [OR87]. In spite of the fact that the class of monotonic convex-fan polygons has a non-empty intersection with the class of envelope polygons, we will show that we can sort, by slope, the n edges of an envelope polygon in O(n) time.

Our proof of the lower bound requires the following definitions. A polygonal chain $C=[r_u, r_{u+1},...,r_v]$ is said to be *monotone* with respect to a straight line L if the perpendicular projections of the vertices of C on L are ordered as $(r_u,r_{u+1},...,r_v)$ (note that we do not allow consecutive vertices of C to be projected on the same point on L). A polygon P is *monotone* if it can be split in two chains monotone with respect to a common line.

Lemma 3.1. Given a monotonic convex-fan polygon P of n vertices, it takes Ω(n log n) time to sort the edges of P in order of slope under the algebraic tree model of computation.

Proof: (By a reduction from Sorting) Let S = $\{x_0,..., x_{n-1}\}$, a set of positive integers, be the input to the Sorting problem. We first show how to build in linear time a comb-like polygonal chain from S (refer to figure 3.1 for an illustration of the construction). Output the point (0,1), and for every x_i, produce two edges, one of slope x_i ($\neq 0$) and one of slope -1, intersecting on x=1. To create a convex-fan polygon P, add a vertex v far enough to the right on y=0. By this construction the polygon P is simple, monotonic (with respect to the y-axis), a convex-fan (from v), and produced from S in O(n) time. Obtaining the edges of this polygon sorted by order of slope sorts the x_i's and hence takes Ω(n log n) time under the algebraic tree model of computation.♣

Define *orientation(L)* as the smallest angle through which L must be rotated clockwise about a point on L so that L is parallel to the x-axis. Note that this definition of orientation is equivalent to the definition of slope. We use it to simplify presentation by avoiding the use of negative slopes. Suppose that we are given an envelope polygon P of n vertices. We show how the edges of P can be sorted in O(n) time by order of orientation. Our strategy is to represent P as the intersection of two unbounded regions D and U that we define as follows. Choose an edge e of P and let L denote the line colinear with e (assume without loss of generality that L is the x-axis). Then let D be the union of P with the closed half-plane below L and let U be the union of P with the closed half-plane above L (refer to figure 3.2 for an illustration of region D).

In what follows, we concentrate on sorting the edges of the polygonal chain delimiting the unbounded region D: $[r_0=(-\infty,0),r_1,\ldots,r_{m-1},r_m=(\infty,0)]$ (r_1 and r_{m-1} are the two extreme vertices of IA(P) on L); sorting the edges of the boundary of U is similar. To sort the edges of the boundary of D (which are edges of P) we use two Graham scans [Gr72], one clockwise (cw) from r_0, procedure **cwScan()**, and one counterclockwise (ccw) from r_m, procedure **ccwScan()**. The edges output by each scan will be stored in order of orientation in linked lists Q_1 and Q_2 respectively; they can easily be merged in linear time afterwards. Q_1 and Q_2 together hold almost all the edges of the boundary of D. The only edges that our algorithm will not detect are those which are parallel to L. However, they can clearly be detected and inserted in our sorted list of edges in O(n) time once L is known. Henceforth, we assume that all edges parallel to e are colinear with L to simplify presen-

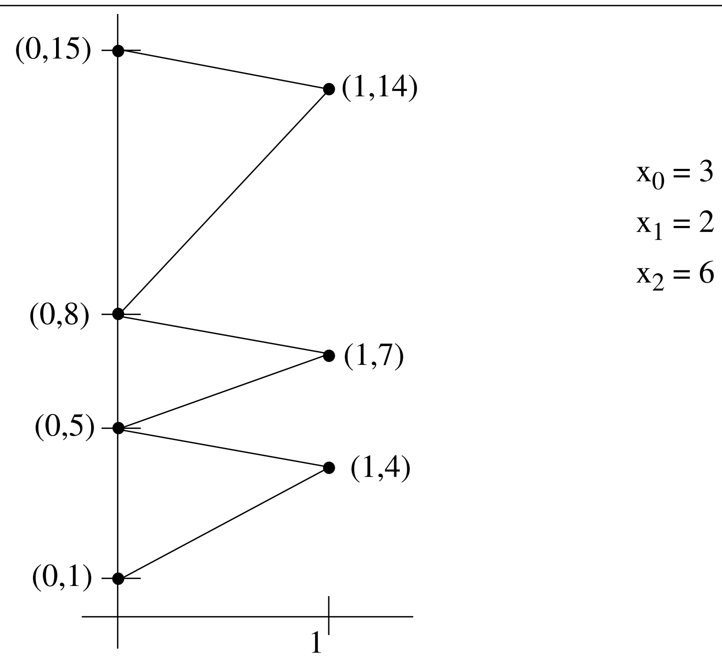


Figure 3.1. Constructing the comb-like polygonal chain in the proof of lemma 3.1

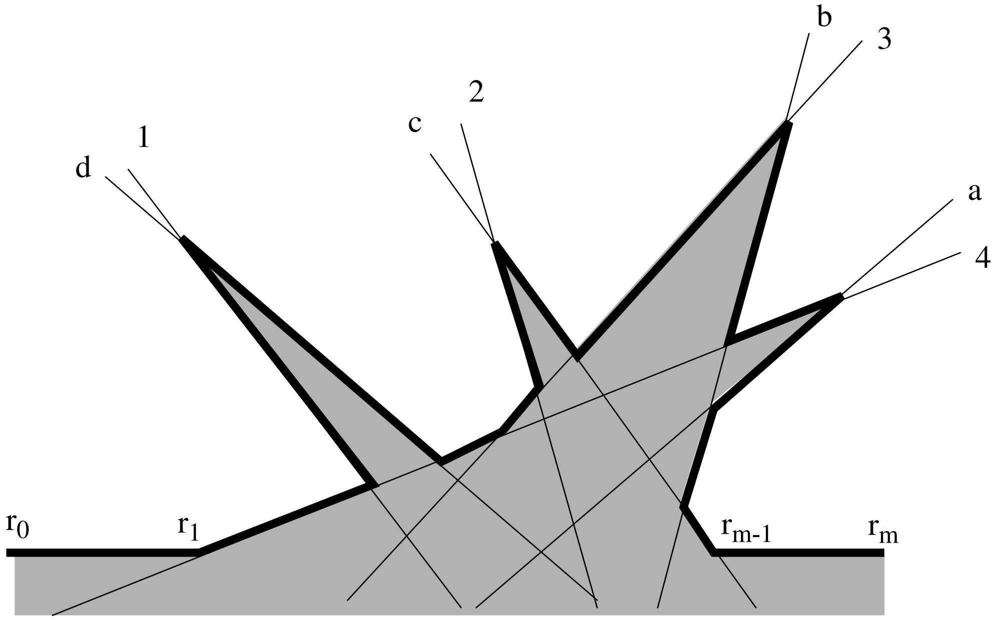


Figure 3.2. The unbounded region D (shaded).
cwScan outputs lines 1, 2, 3, 4. ccwScan outputs lines a, b, c, d.

tation. We now present procedure cwScan(), that we run clockwise on D, starting from r_0; procedure ccwScan() is similar, and therefore will be omitted.

Procedure `cwScan(D,r0)` /* in clockwise direction*/

{Input: A polygonal chain D and vertex r_0, obtained from an envelope polygon P as described above}

{Data Structure: A vertex stack with pointers *top* and *next* to the top two elements of the stack.}

{Output: A list Q_1 which is a subset of the edges of D in sorted order of orientation.}

begin

push(r_0);

push(r_1);

$r_{current} \leftarrow r_2$;

Repeat

  If (*next*, *top*, $r_{current}$) form a left turn or are colinear

  then push($r_{current}$); $r_{current} \leftarrow r_{current+1}$; /* move forward */

  else /* the 3 points form a right turn */

    /* Insert an edge of D in Q_1 when it is popped from the stack */

    if ([*next*, *top*] is an edge of D) **then** insert [*next*, *top*] in Q_1;

    pop(); /* backtrack */

Until ($r_{current} = r_m$)

end {of **Procedure `cwScan`**}

Consider figure 3.2, an example of the polygonal chain D obtained as described above. It shows the order in which the lines are output by procedures cwScan (1, 2, 3, 4) and ccwScan (a, b, c, d). We recall the following fact, proven by Graham [Gr72].

Lemma 3.2. The procedure cwScan() performs at most O(m) steps given an arbitrary polygonal chain of m vertices as input.

We say that an edge $e_i = [r_i,r_{i+1}]$ of D is *visited* by the scan if at one point in the algorithm r_i and r_{i+1} are consecutive on top of the stack. We say that an edge is *popped* by the scan if it was visited and one of its endpoints is now popped from the stack. We now prove the correctness of the algorithm when it is given as input D= $[r_0,r_1,...,r_m]$ obtained from an envelope, as described above. Let L_i denote the line colinear with edge $e_i=[r_i,r_{i+1}]$ of D and let s denote the segment $[r_1,r_{m-1}]$ on L (the line used to split P in two). The proof that the scan correctly sorts some of the edges of an envelope polygon is based on the following property: if P is an envelope and L_i is a line of IA(P), then the intersection of P and L_i is a unique connected line segment (let $s_i=[f_i,g_i]$ denote that segment on line L_i) i.e. every edge e_i of D not parallel to L must satisfy the following two properties: (1) the extension of e_i towards L remains in D and (2) the extension of e_i away from L crosses out of D and then intersects no line of D (without loss of generality, suppose f_i is above L; then clearly g_i must lie below L— the x-axis).

A simple polygon P is *edge-visible* from an edge e of P if for every point x in P, there is a point y on e such that x and y are visible. Property (1) above of the unbounded region D is similar to edge-visibility from a line at infinity. Edge-visibility was introduced by Toussaint and Avis in [TA82], where they show that a Graham scan gives an easy algorithm to triangulate edge-visible polygons; our algorithm is closely related to theirs. We will use some of their results in our proof of correctness.

The proof of correctness of the sorting procedure will follow from lemmata 3.5 and 3.7 which establish that every edge of D (not parallel to L) is visited either by the clockwise or the counterclockwise scan, and that the edges, in both cases, are popped by the scan in order of orientation. The following two technical lemmata, which exploit properties (1) and (2) above, are useful in establishing the correctness of the procedure. Refer to figure 3.3 for an illustration of their statements.

Lemma 3.3. Let e_i be an edge of D not parallel to L. Then no vertex of the clockwise chain from r_1 to f_i lies to the right of L_i and no vertex of the clockwise chain from f_i to r_{m-1} lies to the left of L_i.

Proof: By properties (1) and (2) above, $s_i=[f_i,g_i]$ is contained entirely in D (f_i lies above L). The clockwise chain from r_1 to r_{m-1} (entirely above or on L) can only cross L_i at f_i. It follows that no vertex of the chain from r_1 to f_i can lie to the right of L_i and that no vertex of the chain from f_i to r_{m-1} lies to the left of L_i.♣

Lemma 3.4. If edge $e_i=[r_i,r_{i+1}]$ (not parallel to L) lies on the clockwise chain from r_1 to f_i, then r_{i+1} lies further from L than r_i. If e_i lies on the clockwise chain from f_i to r_{m-1}, then r_i lies further from L than r_{i+1}.

Proof: By contradiction. Suppose that edge $e_i=[r_i,r_{i+1}]$ lies on the clockwise chain from r_1 to f_i and assume that r_i lies further from L than r_{i+1} on L_i. By lemma 3.3 and by the definition of D, no vertex of the chain from r_1 to f_i lies to the right of L_i or below L. In particular, it must hold for the chain from r_1 to r_i. That chain together with $I(L,L_i)$ forms a closed polygon containing r_{i+1} (on $[r_i,I(L, L_i)]$). Since s_i lies entirely inside D (properties (1) and (2)), then the chain from r_{i+1} to f_i must intersect with the chain from r_1 to r_i, contradicting the simplicity of the clockwise chain from r_1 to r_{m-1}. The symmetric statement follows from the same argument.♣

The following lemma uses lemma 3.3 and lemma 3.4 to establish that the edges of D not parallel to L are all visited and popped by the clockwise or by the counterclockwise scan. These facts establish that Q_1 and Q_2 together contain all the edges of D (except those colinear with L).

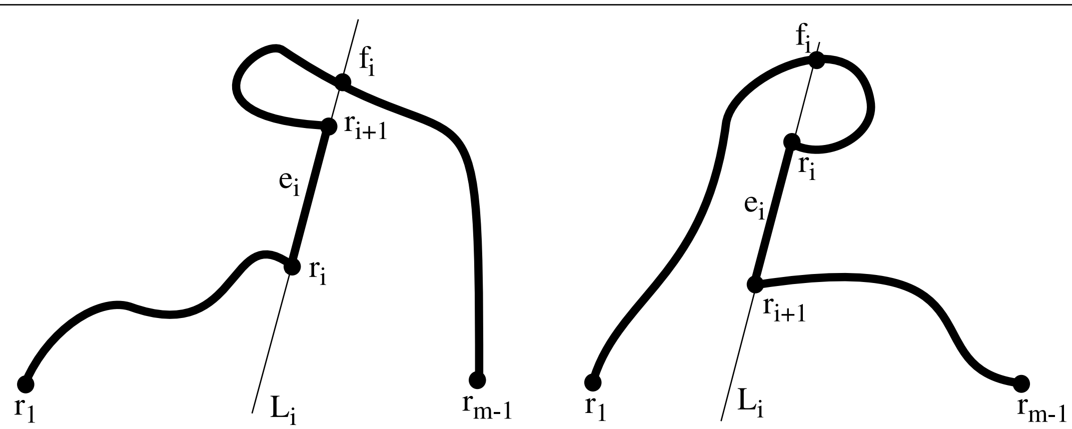
Lemma 3.5. Let $e_i=[r_i,r_{i+1}]$ be any edge of D (not parallel to L). Then e_i is visited and popped by the scan started clockwise from r_0 or by the scan started counterclockwise from r_m.

Proof: Suppose, without loss of generality that e_i lies on the clockwise chain from r_1 to f_i. We know from lemma 3.3 that no vertex of the chain from r_1 to f_i lies to the right of L_i, hence that e_i lies on the convex hull of the chain from r_1 to r_{i+1}. Since r_{i+1} lies above r_i on L_i (lemma 3.4) then when r_i is pushed on the stack, (*next*, *top* $= r_i$, $r_{current} = r_{i+1}$) also forms a left turn (*next* must be on the chain from r_1 to r_{i-1} since the scan progresses in clockwise order from r_0) i.e. e_i is visited by a clockwise scan from r_0. Edge e_i will be popped when $r_{current}$, for the first time, will be on the chain from f_i to r_{m-1}. If e_i lies on the clockwise chain from r_1 to f_i, then by a similar argument it is visited and popped by a counterclockwise scan from r_m.♣

To complete the proof of correctness, we argue that in the scan, the edges are inserted in Q_1 in order of orientation. The following lemma and its corollaries will serve that purpose.

Lemma 3.6. When the (cw or ccw) scan runs on D, then at every step the content of the stack forms a simple convex polygonal chain monotone (non-decreasing) with respect to the y-axis.

Proof: By property (1) above, the polygon D is edge-visible from a line at infinity. i.e. every vertex



e_i lies on the chain from r_1 to f_i. e_i lies on the chain from f_i to r_{m-1}.

Figure 3.3. Illustrating the statements of lemmata 3.3. and 3.4.

of D can be joined to the line $y = -\infty$ by a segment that lies entirely inside D. The statement of the lemma therefore follows directly from lemma 1 in [TA82].♣

Corollary 3.6.1. The content of the stack forms a convex chain of edges sorted in increasing order of orientation after each loop.

Proof: Follows directly from the fact that the stack chain is made of left turns and that it is monotone with respect to the y-axis.♣

Corollary 3.6.2. If edge $e_i=[r_i,r_{i+1}]$ of D lies on the clockwise chain from f_i to r_m, then e_i is not visited by a cwScan. If e_i lies on the clockwise chain from r_1 to f_i, then e_i is not visited by ccwScan.

Proof: Suppose that edge $e_i=[r_i,r_{i+1}]$ lies on the clockwise chain from f_i to r_m. By lemma 3.4, r_i lies further from L than r_{i+1}, hence has greater y-coordinate. If e_i is visited by a clockwise scan then r_{i+1} lies above r_i on the stack, contradicting lemma 3.6. The argument is the same if e_i lies on the clockwise chain from r_1 to f_i (with respect to a ccw scan).♣

We now complete the proof of correctness with the following.

Lemma 3.7. The edges visited by the procedure cwScan are inserted in Q_1 in decreasing order of orientation. The edges visited by ccwScan are inserted in Q_2 in increasing order of orientation.

Proof: Suppose that we run cwScan on D in a clockwise direction, starting from r_0. We show that all edges popped (hence inserted in Q_1) before a given edge $e_i=[r_i,r_{i+1}]$ is popped have greater orientation than e_i. First consider the edges popped after e_i has been visited and before e_i is popped (hence while e_i is on the stack). Such an edge e_j is visited after e_i since it is popped before, hence it is on top of e_i in the stack when e_j is visited. By corollary 3.6.1, e_j has greater orientation than e_i.

Now suppose, for the sake of contradiction, that an edge $e_j=[r_j,r_{j+1}]$ popped before e_i is visited has smaller orientation. Let r_k be the vertex which pops e_j (r_1, r_j, r_{j+1}, r_k, r_i, r_{i+1}, r_{m-1} are in clockwise order on D). We use the following three facts to derive a contradiction: (1) the chain from r_k to r_{i+1} cannot lie to the left of L_j (r_k lies to the right of L_j since (a) r_{j+1} lies above r_j on L_j (corollary 3.6.2 and lemma 3.4) and (b) (*next* = r_j, *top* = r_{j+1}, $r_{current} = r_k$) must form a right turn to pop r_{j+1}. The fact follows from lemma 3.3). (2) the chain from r_k to r_{i+1} cannot lie to the right of L_i (follows from corollary 3.6.2 and lemma 3.3 since e_i is visited). (3) r_{i+1} must lie above r_i on L_i (follows from corollary 3.6.2 and lemma 3.4).

Line L must lie below both e_j and e_i (it lies below all of the chain from r_1 to r_{m-1}). Since we assumed that L_j has a smaller orientation than L_i, we conclude from facts (1) and (2) above that e_i and e_j lie below $I(L_i,L_j)$ (i.e. e_i lies on $[I(L_i,L), I(L_i,L_j)]$ and e_j must lie on $[I(L_j,L),I(L_i, L_j)]$). We thus obtain a triangle $T= (I(L_i,L), I(L_j,L), I(L_i,L_j))$ lying above L. Since P is an envelope, no vertex of P can lie inside T and no edge of P (hence of D) can intersect with the interior of T. The chain from r_k to r_{i+1} cannot lie inside T and must therefore lie on L_i, to the right of L_j, following facts (1) and (2). But since the clockwise polygonal chain from r_1 to r_{m-1} is simple, then r_i must lie above r_{i+1} on L_i, contradicting fact (3). Hence e_j must have greater orientation than e_i. The symmetric argument applies for a counterclockwise scan.♣

Theorem 3.1. The edges of an envelope polygon P of n vertices can be sorted in order of orientation in O(n) time.

Proof: The proof of correctness of our algorithm follows from the previous discussion. Splitting P in D and U can be done in O(n) time since P is stored in a doubly-linked list. Running the scan twice on both D and U also takes linear time (lemma 3.2). It follows from lemma 3.5 and lemma 3.7 that all the edges of D (U) (except those colinear with L) are sorted by the scan. Inserting the edges colinear with L and merging four sorted lists can also be done in O(n) time.♣

4. Recognizing an Envelope Polygon in Linear Time

We show in this section how, given an arbitrary simple polygon of n vertices, we can determine in O(n) time if it is an envelope polygon. For this purpose, we rely both on the algorithm for sorting the edges of envelope polygons given in section 3 and on Keil's algorithm for computing in O(n) time the envelope of a set of lines given in sorted order of orientation [Ke91].

Algorithm Recognize_Envelope(P)

{Input: An arbitrary simple polygon P stored in a doubly-linked list}

{Output: Whether or not P is an envelope.}

begin

<u>Attempt to sort the edges of P by order of orientation:</u>

Choose any edge e of P and find the two vertices of P, r_1 and r_{m-1}, which are extreme on L, the line colinear to e.

Split P in two parts: F (from r_1 to r_{m-1} clockwise) and G (from r_{m-1} to r_1 clockwise).

For F (similarly for G):

$Q_1 \leftarrow$ cwScan(F,r_0).

$Q_2 \leftarrow$ ccwScan(F,r_m).

Verify that the edges of Q_1 and Q_2 are sorted by orientation.

If not **then return**(NO) /* P cannot be an envelope */

else merge Q_1 and Q_2 and add to them the edges colinear with L.

Q $\leftarrow$ merge the lists obtained from F and G together.

A $\leftarrow$ The lines obtained from Q in sorted order of orientation.

<u>Check if P = E(A):</u> /* the edges of P were correctly sorted by orientation */

Q $\leftarrow$ Compute the envelope of A using Keil's algorithm [Ke91].

Compare P and Q to see if they are identical:

Assume that the vertices of P and Q are given in clockwise order.

Pick any vertex of P and find if there is a vertex of Q with which it corresponds.

If yes, scan the vertices of P and Q in order, checking that each pair of vertices matches.

If the vertices of both polygons correspond pairwise **then Return**(Yes).

else Return(No).

end {of **Algorithm Recognize_Envelope**}

Theorem 4.1. Algorithm Recognize_Envelope determines in O(n) time if a simple polygon is an envelope.

Proof: Note first that if P is an envelope polygon, the two polygonal chains F and G correspond to D and U as defined in the previous section. Now suppose that the edges of P get sorted in O(n) time (by theorem 3.1, this is always the case if P is an envelope polygon). The correctness of Keil's algorithm [Ke91] implies that Q, the envelope of A, is correctly computed in O(n) time. By lemma 2.3, Q has O(n) vertices and so can clearly be compared with P in O(n) time. By proposition 1 (section 2), P is an envelope polygon if and only if P = Q.

If we determine from Q_1 and Q_2 that the edges of P are not correctly sorted by the algorithm (by lemma 3.2 our sorting procedure is guaranteed to terminate in O(n) time in any case), then the input polygon cannot be an envelope (following theorem 3.1), and so is rejected by the algorithm.♣

5. Sail Arrangements

We say that a line of an arrangement A is *exterior* if it contributes at least one edge to E(A). More generally, we say that a line of A is *k-exterior* if it contributes exactly k distinct edges to E(A). We also introduce a similar convention for arrangements of lines. We say that an arrangement A is *exterior* if all the lines of the arrangement are exterior and we say that A is *k-exterior* if (i) it is exterior, (ii) each line of A contributes at most k edges to E(A) and (iii) at least one line is k-exterior. See figure 5.2 for an example of a 1-exterior arrangement of lines.

The fact that an arrangement is exterior is useful in the study of arrangements. Since every line of the arrangement contributes to the envelope, properties of the envelope can be used to induce an "ordering" of the lines of the arrangement. It follows from the definition (of exterior) that an arrangement A is exterior if and only if A = IA(E(A)). This implies that there exists a one-to-one correspondence between exterior arrangements and their envelopes.

The notion of *exteriority* is one of two important notions that we will use. The second one is that of general position. An arrangement of lines A is said to be *simple* if no point of the plane belongs to more than two lines of A, i.e., if no three lines are concurrent [Ro88]. An arrangement of lines A is said to be in *general position* if (i) it is simple and (ii) no two lines of A are parallel. It is quite common in computational geometry to assume that the lines of the arrangement are in general position (e.g. [Ed89]). This, in general, is to avoid exceptions on important properties of arrangements. Some authors also use general position to simplify presentation. In this paper, the assumption that we need is that no three lines of an arrangement A intersect on the boundary of E(A). This is to make sure that the extreme vertices on the lines of the arrangements are well-defined as the intersection of exactly two lines. We will assume general position in order to satisfy this requirement and simplify exposition of the essential ideas.

We define E_c as the class of exterior arrangements in general position whose envelope is a simple polygon containing exactly c convex vertices. In particular, we study the class E_3, which we also call the class of *sail arrangements*. The name *sail* stems from the fact that simple polygons of exactly three convex vertices are known as *sail polygons*. Sail polygons have found applications in the design of efficient algorithms for intersecting convex polygons and triangulating point sets [To85].

We now present a property of simple polygons that is useful in characterizing the class of sail arrangements (refer to figure 5.1 for an illustration). Suppose that $C = [p_u, p_{u+1}, ..., p_v]$ is a clockwise vertex chain of a simple polygon P. Let the interior of the polygon be the region to the right of C as we move from p_u to p_v on C. Let R_u be the directed line colinear to $[p_{u+1}, p_u]$ and let R_{i+1} be the directed line obtained by rotating R_i counterclockwise about p_{i+1} until it is colinear to $[p_{i+2}, p_{i+1}]$, $i \in \{u, ..., v\text{-}2\}$. Let α_{i+1} be the angle induced by the rotation we make to obtain R_{i+1} from R_i. Define $Ang(C) = \Sigma\, \alpha_i$, $i \in \{u+1, ..., v\text{-}1\}$. If we let the directed line R become successively colinear to R_u, $R_{u+1}, ..., R_{v-1}$ (by rotating R *counterclockwise* about successive p_i, $i \in \{u+1, ..., v\text{-}1\}$) then we say that R *travels* on C in *clockwise* fashion about P. If R travels on C from $[p_{v-1}, p_v]$ to $[p_u, p_{u+1}]$ in counterclockwise fashion, then R should be rotated in clockwise fashion. Note that the value of Ang(C) is independent of the direction in which R travels.

Similarly, given a simple polygon P, let Ang(P) be the sum of the angles made by the directed line R as it travels in clockwise order once around P (as described above) until it has rotated about every vertex of P. We show that for every simple polygon P, $Ang(P) = (c\text{-}2)\pi$, where c is the number of convex vertices of P. We will see later how to use this fact in characterizing sail arrangements.

Lemma 5.1. Let P be a simple polygon P of n vertices, c of which are convex. Then $Ang(P) = (c\text{-}2)\pi$.

Proof: Let β_i denote the internal angle induced by vertex p_i of simple polygon P. Remember that we defined α_i above as the angle induced by rotating R_{i-1} (colinear to $[p_i, p_{i-1}]$) counterclockwise about p_i until it is colinear to R_i (colinear to $[p_{i+1}, p_i]$). R_{i-1} can be pointing in two possible directions: toward p_{i-1} or towards p_i. In both cases, α_i will be the same. If p_i is a convex vertex of P, then α_i is the same as β_i. If p_i is a reflex vertex (hence the corresponding internal angle is strictly greater than π) then α_i consists of $(\beta_i\text{-}\pi)$. Thus, for simple polygon P, $Ang(P) = (n\text{-}2)\pi$ - (#reflex vertices)π = (n-#reflex vertices-2)π = $(c\text{-}2)\pi$.♣

We can now obtain the following result, useful for small values of c.

Corollary 5.1.1. Let P be a simple polygon P of n vertices, c of which are convex. Then the lines of IA(P) can be sorted in O(cn) time.

Proof: By letting a directed line R travel once around P, we create c-2 lists of edges sorted by ori-

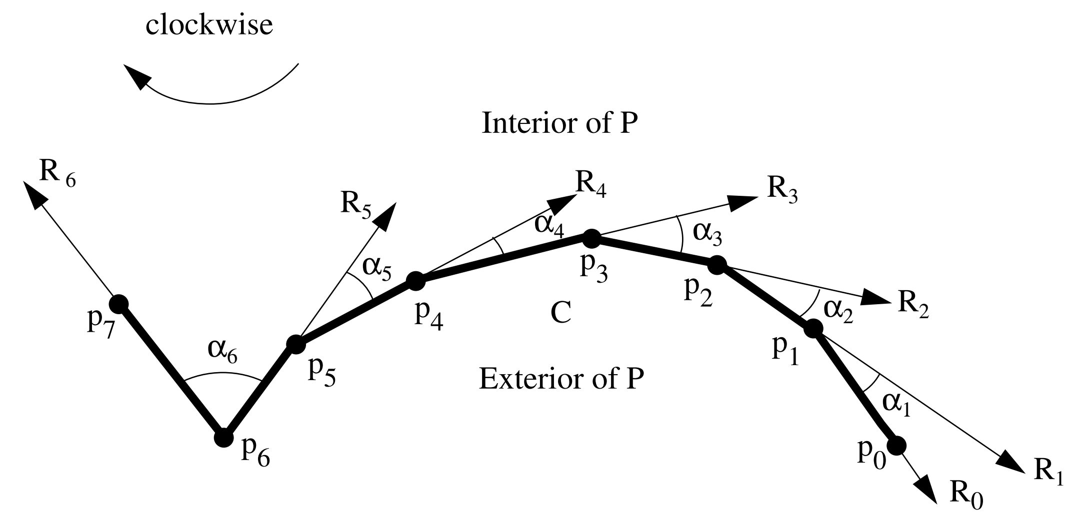


Figure 5.1. The walk of directed line R on a simple polygonal chain C.

entation. The lists have a total of n edges, so merging them can be accomplished in O(cn) time.♣

This property of simple polygons allows us to characterize sail arrangements as follows: we observe that the class of sail arrangements is equivalent to another class of arrangements which we call class $\tilde{A}$. We define *class* $\tilde{A}$ to be the set of arrangements A in general position that have the property that the vertices of A are vertices of E(A) if and only if they are determined by adjacent lines (in the list of lines sorted by orientation). The properties of arrangements in class $\tilde{A}$ (and hence, of sail arrangements) are exploited in section 6, where we determine in O(n) time all the convex vertices of the envelope of a given sail arrangement. The following lemma, along with lemma 5.1, is used in the proof of theorem 5.1.

Lemma 5.2. Class $\tilde{A} \subset$ 1–exterior arrangements.

Proof: Let arrangement A $\in$ class $\tilde{A}$. It follows from the definition of class $\tilde{A}$ that A must be exterior. Since no two lines are parallel in A (A is in general position), exactly two lines are adjacent to every line L in A, and each of those two neighbors of L intersects L at exactly one point. Therefore, a line in A contributes exactly two vertices, hence, exactly one edge to E(A). This forces A to be 1-exterior. We can see that class $\tilde{A}$ is a proper subset of 1–exterior arrangements with the example in figure 5.2. Here $A = \{L_0,L_1,L_2,L_3,L_4,L_5,L_6\}$ is a 1-exterior arrangement of seven lines. The indices of the lines L_i $(0 \le i \le 6)$ correspond to an increasing order of orientation. Consider the points $p_1 = I(L_1,L_4)$, $p_2 = I(L_2,L_5)$ and $p_3 = I(L_3,L_6)$. They are generated by non-adjacent lines but are vertices of the envelope.♣

Theorem 5.1. An arrangement of lines A is in class $\tilde{A}$ if and only if A is a sail arrangement.

Proof: ("If" implication) Suppose that we are given a sail arrangement A. Since E(A) has exactly three convex vertices, it follows from lemma 5.1 that $Ang(E(A)) = \pi$. i.e. as a directed line R travels around E(A), it will rotate through an angle of exactly π and R becomes successively colinear with the lines of A in order of orientation. Since A is exterior (by definition of a sail), every line of A will be met by R. It follows that the intersection point of adjacent lines (in order of orientation) is a vertex of E(A) and that conversely, a vertex of E(A) is the intersection of adjacent lines. i.e. A is in class $\tilde{A}$.

("Only if" implication, proof by contradiction) Suppose that we are given an arrangement A in class $\tilde{A}$ and that E(A) is a simple polygon of more than three convex vertices. By lemma 5.1, Ang(E(A)) > π. Let directed line R travel clockwise around E(A) starting at edge e_0. Let e_i be the first edge that causes the angle covered by R to exceed π. Then e_i has an orientation that lies in between that of some edges e_j and e_{j+1} on the chain from e_0 to e_i, $0 < j < i-1$, so e_j and e_{j+1} are not adjacent in the list of lines sorted by orientation. Clearly, the vertex p_j that is common to both e_j and e_{j+1} is not the intersection of adjacent lines of A, contradicting the assumption that A is in class $\tilde{A}$. We conclude that E(A) must have exactly three convex vertices. Since by definition arrangements in class $\tilde{A}$ are in general position and by lemma 5.2 they are 1-exterior, we conclude that A must be a sail arrangement.♣

6. Recognizing the Critical Vertices of Sail Arrangements

Let X,Y and Z be the three convex vertices of the envelope of a sail arrangement A (E(A) is a sail polygon). Then combinatorially, we obtain four possible geometric situations.

(i) [X,Y], [Y,Z] and [X,Z] are all edges of E(A). E(A) is a triangle.

(ii) Two of [X,Y], [Y,Z] and [X,Z] are edges of E(A). E(A) has one concave chain of vertices.

(iii) One of [X,Y], [Y,Z] and [X,Z] is an edge of E(A). E(A) has two concave chains of vertices.

(iv) None of [X,Y], [Y,Z] and [X,Z] are edges of E(A). E(A) has three concave chains of vertices.

We further classify sail arrangements as k-sail, $0 \le k \le 3$, where k indicates the number of concave chains in the envelope. Thus, case (i) above describes a 0-sail, case (ii) a 1-sail, case (iii) a 2-sail and case (iv) a 3-sail. Figure 6.1 gives examples of the four subclasses of sail arrangements.

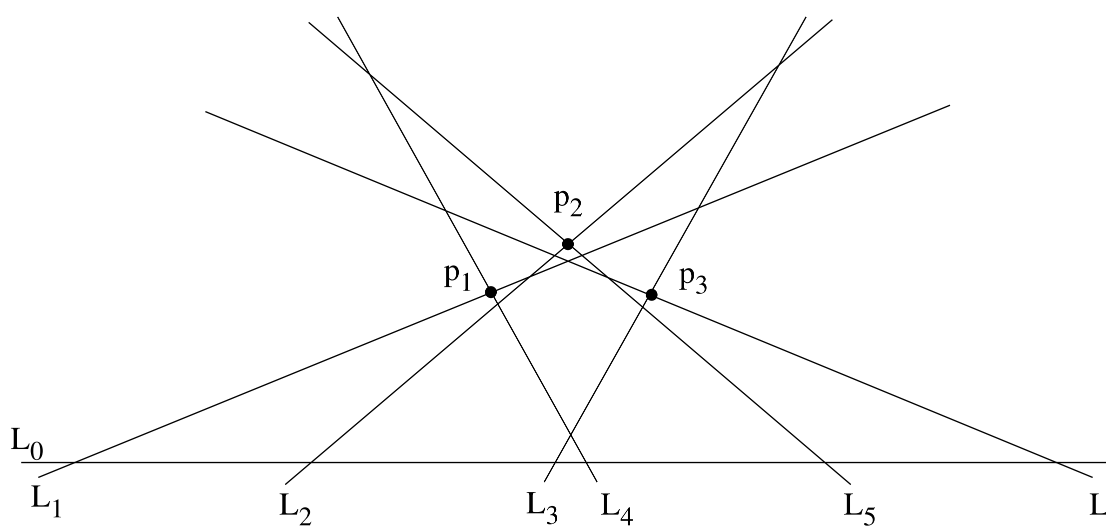


Figure 5.2. A 1-exterior arrangement which is not in class $\tilde{A}$.

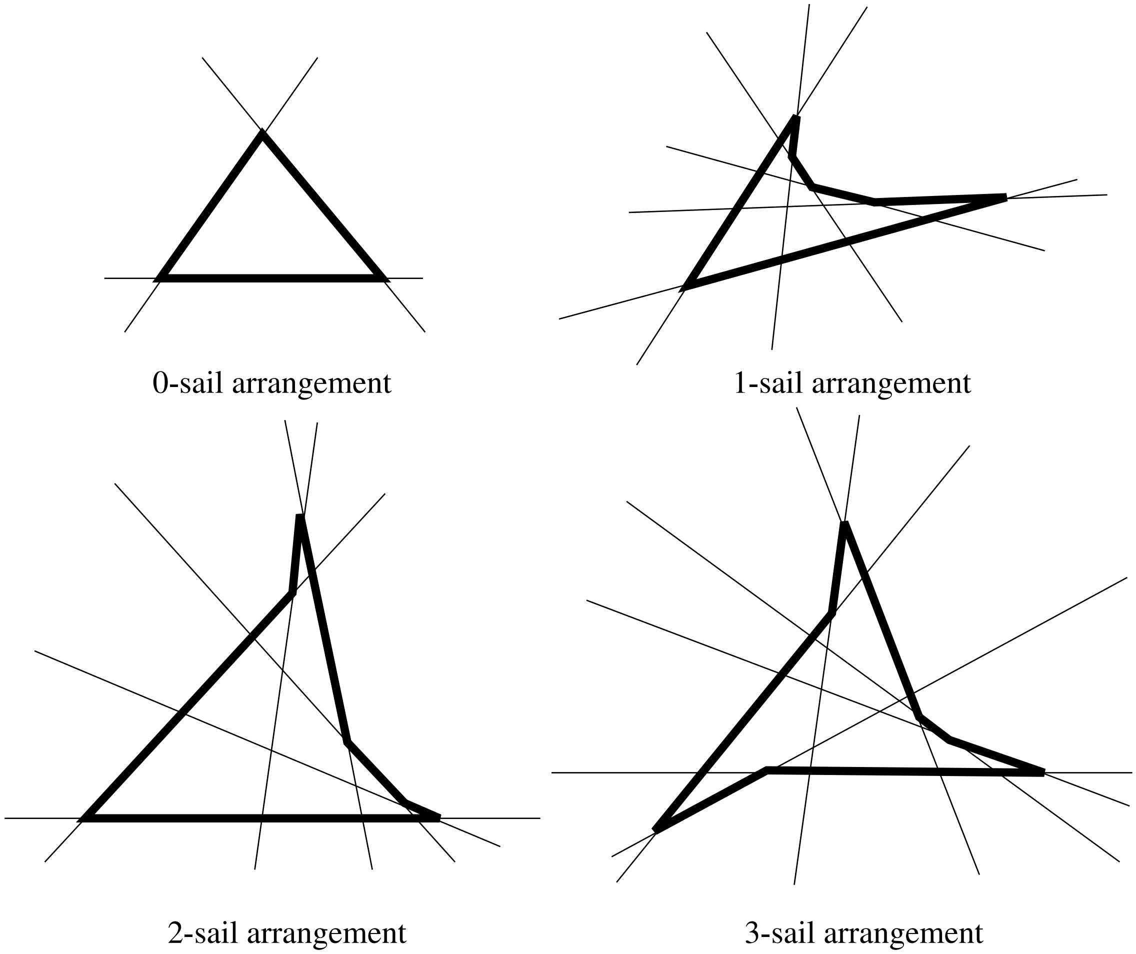


Figure 6.1. The family of sail arrangements.

Owing to the trivial nature of 0-sails, subsequent discussion shall omit this subclass.

Now we ask the following question. Given a sail arrangement of n lines A, can the three convex vertices X, Y and Z of the resulting envelope be found in linear time (i.e. can we determine which pairs of lines in A realize the three convex vertices of E(A))? Finding X,Y and Z in linear time would allow us to compute in O(n) worst-case time the diameter and the convex hull of the $O(n^2)$ intersection points of the arrangement. Furthermore, we would be able to compute the vertices of A with minimum or maximum x-coordinate in linear worst-case time. All these problems have been shown to have Ω(n log n) worst-case lower bounds for arbitrary arrangements [CL85], [CSSS89], although O(n) *expected* time algorithms have recently been discovered [DT93]. It turns out that we can indeed find the convex vertices of a sail arrangement in O(n) worst-case time. In what follows, we show how to find X,Y and Z in O(n) time for each sail subclass. We later show that we can determine in O(n) time the subclass to which a sail arrangement A belongs.

First we define a few terms. A line segment that joins two critical vertices of an arrangement is said to be a *critical edge*. We say that a line of A is a *critical line* if it is colinear with a critical edge. Thus, a k-sail arrangement A has (3 - k) critical edges ($0 \leq k \leq 3$). We shall frequently make use of the following straightforward linear-time procedures. Procedure **Compute_Extreme(A,L)** computes for a line L in an arrangement A the two lines that determine the extreme vertices on L. Procedure **Critical_Line(A,L)** determines whether or not the given line L in A is critical.

Lemma 6.1. On a critical line L of sail arrangement A, the order of the vertices defines an orientation ordering of the remaining n-1 lines of A.

Proof: Let X,Y and Z be the three critical vertices of A. The convex hull of the vertices of A is the triangle ΔXYZ. Since L is critical, two of the three critical vertices of A lie on L, say X and Y. [X,Y] is a critical edge of E(A). Assume, without loss of generality, that [X,Y] lies on the x-axis, X is to the left of Y on L and Z lies above the x-axis. Suppose that there exist lines L_i and L_j such that orientation(L_i) < orientation(L_j) but I(L,L_j) lies to the left of I(L,L_i) on L. Then I(L_i,L_j) is forced to lie below the x-axis and thus, outside of the convex hull of the vertices of A, a contradiction.♣

We can find the three convex vertices X,Y and Z of a 1-sail arrangement A in O(n) time with the following algorithm:

Algorithm One_Sail(A)

{Input: A 1-sail arrangement A of n lines.}

{Output: The three pairs of lines that determine the three convex vertices X,Y and Z.}

begin

L ← Any line of A

L_a,L_b ← Compute_Extreme(A,L)

L_c,L_d ← Compute_Extreme(A,L_a)

L_e,L_f ← Compute_Extreme(A,L_b)

We now have the four intersection points: I(L_a,L_c), I(L_a,L_d), I(L_b,L_e) and I(L_b,L_f).

if L is critical

then determine which three of the four points are colinear and delete the point that is not critical

else determine which two of the four points are identical and delete one of them.

Return the three pairs of lines that determine the three remaining intersection points.

end {of **Algorithm One_Sail**}

Proof of correctness: By definition, a 1-sail envelope has two critical edges which have as end-points the three convex vertices X,Y and Z. Without loss of generality, let the two critical edges be E_{XY} = [X,Y] and E_{YZ} = [Y,Z]. Note that the intersection points of E_{XY} (E_{YZ}) with every other line in A are extreme (lemma 6.1). Now, two cases arise when a line L is chosen. Either (i) the chosen line L is colinear to neither E_{XY} nor E_{YZ} or (ii) the chosen line L is colinear to either E_{XY} or E_{YZ}.

Suppose that the first case is true. Then it follows that lines L_a and L_b are colinear to E_{XY} and E_{YZ}. Computing lines L_c and L_d for L_a gives I(L_a,L_c) and I(L_a,L_d), which must be critical vertices. Similarly, computing lines L_e and L_f for L_b gives I(L_b,L_e) and I(L_b,L_f), which also must be critical vertices. But since Y is common to both E_{XY} and E_{YZ}, two of the points $I(L_a,L_c)$, $I(L_a, L_d)$, $I(L_b,L_e)$ and $I(L_b,L_f)$ must identify vertex Y, meaning that one of the four computed vertices is redundant.

If the second case is true, then without loss of generality, assume that L is colinear to E_{XY}. Then either L_a or L_b is colinear to E_{YZ}. Again without loss of generality, suppose that L_a is colinear to E_{YZ}. L_a will identify the two critical vertices Y and Z. When we compute the extreme vertices for L_b, we obtain one vertex which is X (= $I(L,L_b)$) and the other which is a non-critical vertex on E_{YZ}, say p. The vertex p lies in between Y and Z on E_{YZ} and hence, is non-critical. The algorithm uses procedures Compute_Extreme and Critical_Line which have O(n) time complexity. Finally, the deletion of the redundant or non-critical vertex takes constant time.♣

We can combine the above results with the technique of prune and search to find the three convex vertices X, Y and Z of a 2-sail arrangement of n lines in O(n) time. We find X,Y and Z with procedure Find_Sail_Tip and algorithm Two_Sail. By definition, a 2-sail arrangement A has exactly one critical line. Procedure Find_Sail_Tip accepts the sole critical line L and finds the convex vertex that does not lie on L. This is done by applying the prune and search technique. Algorithm Two_Sail finds the sole critical line L (and the two convex vertices that make L critical) and then applies the recursive procedure Find_Sail_Tip, with L as input, to find the third convex vertex of E(A).

Procedure Find_Sail_Tip(A,L,n)

{Input: A critical line L in a 2-sail arrangement A = $\{L_0,L_1,...,L_{n-1}\}$ of n lines.}

{Output: The two lines that determine the convex vertex of E(A) that does not lie on L.}

{Assume that L lies on the x-axis and that n is even to simplify presentation}

begin

$L_a \leftarrow$ the line $L_i \in A - \{L\}$ such that $(n-2)/2$ intersection points lie on either side of $I(L,L_i)$ on L

$L_b \leftarrow$ Extreme line on L_a other than L

if (L_a is extreme on L_b) **then return**(L_a and L_b)

else **begin**

$A' \leftarrow$ {all lines $L_i \in A$ such that $I(L,L_i)$ lies

in between $I(L,L_a)$ and $I(L,L_b)$ on L} $\cup$ $\{L, L_a, L_b\}$

return Find_Sail_Tip(A´,L,||A´||)

end

end {of **Procedure Find_Sail_Tip**}

Proof of correctness: Let L_a be any line $\in A - \{L\}$. Then we can determine the line L_b other than L such that $I(L_a,L_b)$ is extreme on L_a using procedure Compute_Extreme(). If $I(L_a,L_b)$ is also extreme on L_b then by definition $I(L_a,L_b)$ is critical, in which case we are done. Otherwise, from theorem 5.1 it follows that as we walk around E(A), we meet the lines of A in sorted order of orientation. By lemma 6.1, the order of the intersection points on L also define an orientation ordering of the lines of A, the same as E(A). Now note that L_a and L_b are exterior on different concave chains of E(A). It follows that the two lines that we seek have corresponding intersection points that lie in between $I(L,L_a)$ and $I(L,L_b)$ on L.

Suppose now that we choose L_a such that there are (n-2)/2 intersection points on either side of $I(L_a,L)$ on L. We can do this in O(n) time using the k-selection algorithm of Blum et al. [BFPRT73]. L_b lies on one of these sides and so we can “throw away” ((n-2)/2 + c) lines with each call to procedure Find_Sail_Tip (c depends on the relative position of $I(L_b,L)$ on L). So we obtain the following recurrence relation: T(n) = T(n - ((n-2)/2 + c)) + O(n) = T(n/2) + O(n) = O(n). Therefore procedure Find_Sail_Tip runs in O(n) time.♣

Algorithm Two_Sail(A, n)

{Input: A 2-sail arrangement A of n lines.}

{Output: The three pairs of lines that determine the three convex vertices X,Y and Z.}

begin

Find the sole critical line of A and call it L.

L_a,L_b ← Compute_Extreme(A,L) {Determine the critical vertices on L}

L_c,L_d ← Find_Sail_Tip(A,L,n) {Find the third convex vertex Z}

Return $\{(L,L_a),(L,L_b),(L_c,L_d)\}$

end {of **Algorithm Two_Sail**}

Proof of correctness: We only need to show that the first two convex vertices X and Y that define a critical edge of A are correctly determined. Pick an arbitrary line L. Either the line L that we choose is critical or one of the two lines which give extreme vertices on L, $L_a$and L_b, must be critical. We can use procedure Critical_Line to determine which of L, L_a or L_b is critical, and we relabel the critical line L. All procedures used have linear time complexity so algorithm Two_Sail has O(n) time complexity.♣

We can apply the previous results for 2-sails to obtain a divide and conquer algorithm for 3-sails. Given a 3-sail arrangement of lines, we first select an arbitrary line L. From this line, we obtain the two lines L_a and L_b that determine the extreme vertices on L. We show that L_a and L_b define three orientation ranges that separate the given arrangement of lines into three 2-sail arrangements. We first present the divide and conquer algorithm and then give a proof of correctness.

Algorithm Three_Sail(A)

{Input: A 3-sail arrangement A of n lines.}

{Output: The three pairs of lines that determine the three convex vertices X,Y and Z of E(A).}

begin

$L \leftarrow$ Any line of A /* without loss of generality, assume that orientation(L) = 0 */

$L_a, L_b \leftarrow$ Compute_Extreme(A,L) /* Assume that orientation(L_a) > orientation(L_b) */

Split the lines of A into three subsets A_1, A_2 and A_3 where

$A_1 = \{$lines $L_i \in$ A such that orientation $(L_b) \le$ orientation$(L_i) \le$ orientation$(L_a)\} \cup \{L\}$

$A_2 = \{$lines $L_i \in$ A such that orientation$(L_i) \ge$ orientation$(L_a)\} \cup \{L\}$

$A_3 = \{$lines $L_i \in$ A such that orientation $(L_i) \le$ orientation$(L_b)\}$

$L_c, L_d \leftarrow$ Find_Sail_Tip(A_1,L,$\|A_1\|$)

$L_e, L_f \leftarrow$ Find_Sail_Tip(A_2,L,$\|A_2\|$)

$L_g, L_h \leftarrow$ Find_Sail_Tip(A_3,L,$\|A_3\|$)

Return $\{(L_c,L_d),(L_e,L_f),(L_g,L_h)\}$

end {of **Algorithm Three_Sail**}

Proof of correctness: We assume that L is parallel to the x-axis (if not we can rotate the lines of A appropriately). Thus, we can think of L as having orientation(L) = 0 or π. Assume orientation(L_a) > orientation(L_b). Also, let Q be the list of lines of A sorted on increasing orientation starting from orientation 0. Without loss of generality, let X, Y and Z, the three critical vertices of A, appear in clockwise order on E(A), with Y being the sole critical vertex lying above L (X and Z lie on or below L).

First note that L, L_a and L_b are exterior on different concave chains of E(A). Therefore, it follows from theorem 5.1 that there exists two lines between L (orientation 0) and L_b in Q that determine a convex vertex of E(A), X, two lines between L_b and L_a in Q that determine a second convex vertex of E(A), Y, and two lines between L_a and L (orientation π) in Q that determine the third convex vertex of E(A), Z. We split A into three subsets A_1, A_2 and A_3, such that each subset contains the lines that form one of the critical vertices.

Let A_1 be defined as in the algorithm. Let C_1 be the vertex chain $[I(L,L_b),...,Y,..., I(L,L_a)]$ on E(A). Since A is in class $\tilde{A}$ (theorem 5.1), a line $L_i \in$ A-{L} is in A_1 if and only if it is exterior on C_1. Hence, C_1 is a portion of E(A_1) that lies on or above L. Now we show that no vertex of A_1 lies below L and hence that A_1 is a 2-sail arrangement. Suppose that there exists a vertex p of A_1 below L. E(A_1) must contain p, so there exists a critical vertex of A_1 below L. This critical vertex is the intersection of adjacent lines in A_1 (corollary 2.1.1), and must therefore lie on C_1 since A is in class $\tilde{A}$, a contradiction. Hence, all of the vertices of A_1 on L are extreme and thus L is critical. Therefore, A_1 is a 2-sail arrangement. By similar arguments, A_2 and A_3 are also 2-sails having L as their sole critical line, unless of course they contain only the two desired lines, which would be the case if one of the critical vertices were to lie on L. So we can apply algorithm Find_Sail_Tip to A_1, A_2 and A_3 with L as the sole critical line to obtain Y = (L_c,L_d), Z = (L_e,L_f) and X = (L_g,L_h). All of the operations have O(n) time complexity, so algorithm Three_Sail() exhibits O(n) time complexity.♣

Thus far we are able to find X, Y and Z given that A is k-sail ($0 \le k \le 3$). The following theorem establishes that we need only know that A is a sail arrangement.

Theorem 6.1. Given $A = \{L_0, L_1, ..., L_{n-1}\}$, a k-sail arrangement of n lines ($0 \le k \le 3$), we can determine the value of k and the three convex vertices of A in O(n) time.

Proof: Let L be any line in A and L_a and L_b be the lines in A such that $I(L,L_a)$ and $I(L,L_b)$ are the extreme vertices on L. We can use procedure Critical_Line to determine whether or not L, L_a and L_b are critical. The number of critical lines tells us the number of convex vertices in E(A) and hence, by definition of sail arrangements, the value of k. These tests are sufficient since all critical lines in A (if they exist) are identified by computing the extreme vertices of any line in A. For example, if L was found to be non-critical, then A can be only be 1, 2 or 3 -sail and $L_a$and L_b can be the only candidates for critical lines since they are the extreme lines on L. Once we know the value of k, we can invoke the appropriate algorithm to find the three critical vertices of A in O(n) time.♣

Corollary 6.1.1. The diameter, the convex hull and the points with minimum or maximum x-coordinate of the $\Theta(n^2)$ vertices of a sail arrangement of n lines can be computed in O(n) time.

7. The Complexity of Constructing the Envelope of a Sail Arrangement

In [CL85], Ching and Lee established a lower bound of $\Omega(n \log n)$ for the problems of computing the diameter, convex hull and envelope of an arbitrary arrangement of lines. We have introduced and studied sail arrangements with the hope that characterizations of a non-trivial class of arrangements would allow us to sharpen these lower bounds. Theorem 6.1 implies that given a sail arrangement of n lines, the convex vertices can be determined in O(n) time. This allows us to compute the diameter and convex hull of the arrangement in O(n) time even though there are $\Theta(n^2)$ intersection points. We show here that the $\Omega(n \log n)$ lower bound, however, still holds for envelope construction of sail arrangements. This implies that the additional information and structure derived from sail arrangements is insufficient for solving the envelope construction problem in time $o(n \log n)$ and thus improves on Ching and Lee's result for arbitrary arrangements.

Theorem 7.1. Given a 1-sail arrangement A of n lines, the complexity of computing E(A) is $\Omega(n \log n)$ under the algebraic tree model of computation.

Proof: (By reduction from Sorting) Let $S = \{x_1, ..., x_n\} \cup \{x_0 = 0\}$ be the input to the Sorting problem; assume that the x_i's are distinct and refer to figure 7.1 for illustration. Let C be the half-parabola defined by the set of points on the curve $y = x^2$ with positive x-coordinate. For every x_i, compute L_i, the unique line tangent to C with slope x_i (let $A = \{L_i\}_{i=0,...,n}$).

We now show that A thus constructed is a 1-sail arrangement of lines. Without loss of generality, assume that $\text{slope}(L_0) \le \text{slope}(L_1) \le \text{slope}(L_i) \le \text{slope}(L_{n-1}) \le \text{slope}(L_n)$, i = 2,...,n-2. Let $X = I(L_{n-1},L_n)$, $Y = I(L_0,L_n)$ and $Z = I(L_0,L_1)$. Notice the following two facts: (i) C is monotone in-

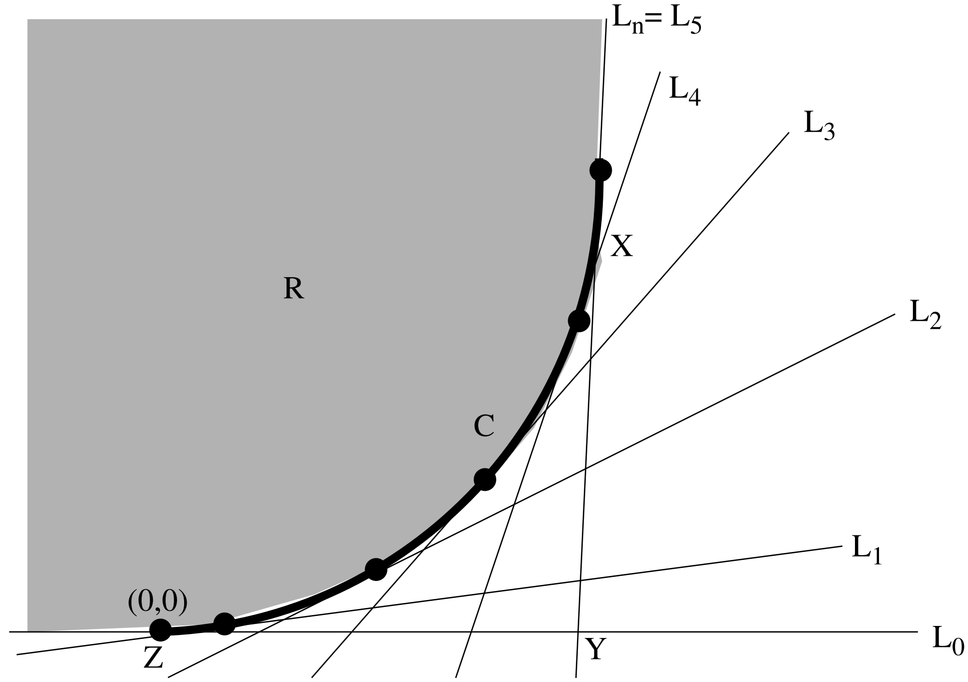


Figure 7.1. Constructing a 1-sail arrangement of lines from a set of integers.

creasing with respect to both L_0 and L_n (we naturally extend the definition of *monotone* from a polygonal chain to a smooth curve) and (ii) the order of the intersection points of the lines of A with C define on C a slope ordering of the lines of A. From these two facts, it follows that the order of the intersection points on L_0 (and on L_n) also define a slope ordering of the lines of A; the same ordering as on C. Therefore no two lines of A intersect below L_0 or to the right of L_n. i.e. X, Y and Z are all critical vertices. It also follows that no critical vertex can lie on L_i, i= 2,...,n-2 (the extreme vertex of L_i on L_0 (or L_n) is not extreme on L_0 (or L_n) because of the slope ordering). We can therefore conclude that E(A) has exactly three convex vertices, X, Y and Z.

To show that A is a 1-sail, we still have to demonstrate that A is exterior and that the lines of A are in general position. First consider the region R to the left of L_n, above L_0 and to the left of C. Because the lines of A are tangent to C, it follows that (a) R lies to the left of every line of A (and so does not intersect with the interior of E(A)) and that (b) every line of A has a point (the tangent) on the boundary of R. From this, we conclude that A is exterior.

No two lines of A are parallel since we assumed that the x_i's are distinct. Since each line of A intersects C in a unique point, no three lines of A can intersect in a common point, establishing that the lines of A are in general position.

We have thus shown that A, constructed in O(n) time from S, is a 1-sail arrangement. Following theorem 5.1, in traversing E(A) in clockwise order, the lines of A are met in sorted order of slope, which is the order of the x_i's. Hence we can solve the Sorting problem by reducing it in linear time to the problem of constructing the envelope of a 1-sail arrangement. Since Sorting is Ω(n log n) under the algebraic tree model of computation, the result follows.♣

8. Hamiltonian Circuits

An arrangement of lines A is said to be *hamiltonian* or to admit a *hamiltonian circuit* if there exists a closed path through some of the bounded segments of A which visits every vertex of A exactly once. We say that a hamiltonian circuit of an arrangement of lines is *stream-lined* if only the vertices where a turn is made in the circuit are specified. Everett showed with the example illustrated in figure 8.1 that there exist arrangements of six lines that are not hamiltonian. One can observe that the critical vertices of the arrangement, adjacent to only two bounded segments, force every hamiltonian circuit to follow those two bounded segments. Therefore there is no way for any circuit to reach the vertex p without visiting vertices which have already been forced to be visited. We conclude that this arrangement does not admit any hamiltonian circuit. It is easy to construct, for any odd value of n ($n > 6$), an arrangement of n lines which does not admit any hamiltonian circuit; this is done by maximizing the number of critical vertices in the arrangement— the envelope has the shape of a "star."

We now ask if there are classes of arrangements which always admit a hamiltonian circuit. We show in this section that every 1-sail arrangement of lines is hamiltonian. We also give an $O(n \log n)$ time procedure to produce a stream-lined hamiltonian circuit from a 1-sail arrangement of n lines. Our argument exploits a property of 1-sail arrangements that we establish with the following lemma.

Lemma 8.1. Let $A = \{L_0, L_1, ..., L_{n-1}\}$ be a 1-sail arrangement of n lines. Then for each line L of A, the ordering of the n-1 vertices on L corresponds to an orientation ordering of the lines of A.

Proof: (By induction on n) First note that for every arrangement of four lines (only one combinatorial possibility) the vertices are ordered as required. As our inductive hypothesis, assume that the statement of the lemma holds for every 1-sail arrangement of n-1 lines, and suppose that we are given a 1-sail arrangement A of n lines. Let X, Y and Z be the three convex vertices of E(A), such that [X,Y] and [Y,Z] are the two critical edges of E(A). Let L_0 (L_{n-1}) be the critical line through Y and Z (X and Y). L_0 and L_{n-1} are adjacent in the list of lines of A sorted by orientation since Y is a critical vertex. To simplify presentation, assume without loss of generality (by rotating A) that L_0 (L_{n-1}) has the smallest (largest) orientation of A, and that the indices of the other lines give their order of orientation. Thus $Z = I(L_0, L_1)$ and $X = I(L_{n-1}, L_{n-2})$.

Let A' be the arrangement of n-1 lines obtained by removing critical line L_{n-1} from A. We first show by contradiction that A' is a 1-sail arrangement. The portion of E(A) from $I(L_{n-2}, L_{n-3})$ to $I(L_0, L_{n-2})$ that contains Z is not affected by the removal of L_{n-1} and therefore is part of $E(A')$. If the interior of triangle $\Delta = (X, Y, I(L_0, L_{n-2}))$ contains no vertex of A' (and hence of A) then the segment $[I(L_{n-2}, L_{n-3}), I(L_0, L_{n-2})]$ closes $E(A')$ and A' is a 1-sail arrangement (since $E(A')$ is a 1-sail polygon). Suppose then that there exists at least one vertex of A' in the interior of triangle Δ. Let $p = I(L_i, L_j)$ be the first such vertex in the interior of Δ that is met by L_{n-1} as we imagine rotating it clockwise about X. Then the triangle $(p, I(L_{n-1}, L_i), I(L_{n-1}, L_j))$ is empty and therefore L_i and L_j must be adjacent on L_{n-1}. Since the order of the vertices of A on L_{n-1} define an orientation ordering for the lines

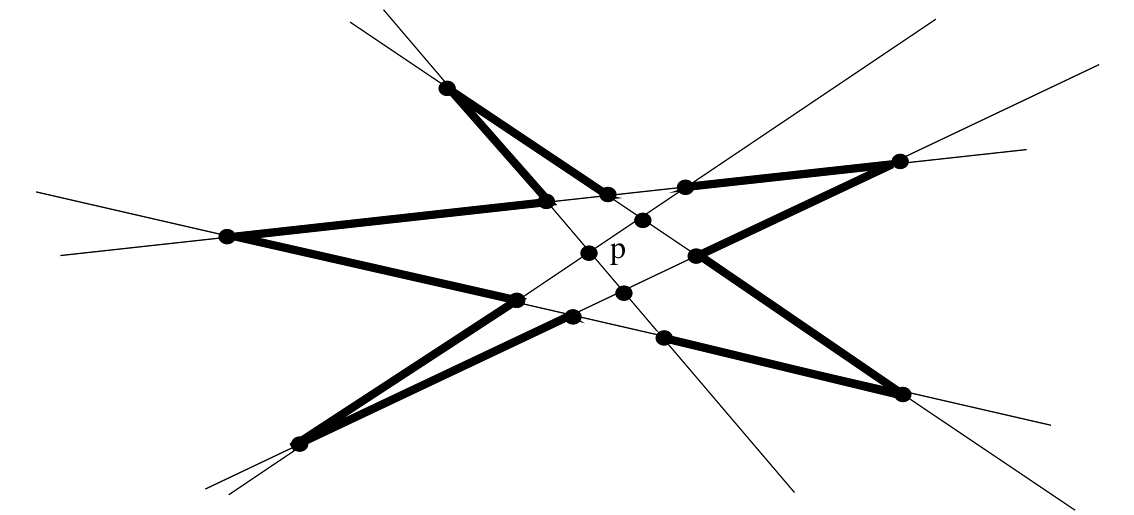


Figure 8.1. Everett's example of an arrangement of 6 lines that is not hamiltonian.

of A (by lemma 6.1), it follows that L_i and L_j are adjacent lines in A. Vertex p= $I(L_i,L_j)$ is therefore a vertex of E(A), since $A \in$ class $\tilde{A}$, contradicting the assumption that p lies in the interior of triangle Δ. Therefore Δ must be empty and A' must be a 1-sail arrangement of n-1 lines, for which the statement of the lemma holds by the inductive hypothesis.

We now show that re-inserting the line L_{n-1} preserves the slope ordering in A. Since the line L_{n-1} is a critical line, the ordering of the vertices on L_{n-1} corresponds to an orientation ordering of the lines in A (by lemma 6.1). Now it suffices to show that for every L_i, the intersection point $I(L_i, L_{n-1})$ respects the ordering of the vertices on L_i. Because line L_{n-1} has the greatest orientation in A and $I(L_i, L_{n-1})$ is extreme on L_i (for $i \in \{0,...,n-2\}$), it follows that the position of $I(L_i,L_{n-1})$ on L_i corresponds to the position of L_{n-1} in the order of the lines of A sorted by orientation. The statement of the lemma therefore holds for A, a 1-sail arrangement of n lines.♣

The following algorithm uses the above property to trace a hamiltonian circuit through the vertices of a 1-sail arrangement A. See fig. 8.2 for an illustration of the resulting hamiltonian circuit. The vertices of A at which left or right turns are made are indicated by the dots. The direction of the traced hamiltonian circuit is indicated with arrows.

Algorithm One_Sail_Hamiltonian(A)

{Input: A 1-sail arrangement $A = \{L_0,L_1,...,L_{n-1}\}$ of n lines.}

{Output: A stream-lined hamiltonian circuit denoted as a list H of vertices of A}

begin

Sort the lines by increasing orientation. Re-label the lines appropriately.

Use algorithm One_Sail to find the three convex vertices X, Y and Z of E(A).

Rotate the lines such that $X = I(L_0, L_1)$, $Y = I(L_0,L_{n-1})$ and $Z = I(L_{n-1},L_{n-2})$

$H \leftarrow Y$

$H \leftarrow Z$

for i = n-2 down to 2 by 2

begin

$H \leftarrow I(L_1,L_i)$ {Circuit passes through vertices on L_i in decreasing order of orientation}

if $(i - 1 \geq 2)$ **then** $H \leftarrow I(L_1,L_{i-1})$ {Move to adjacent line with smaller orientation}

if $(i - 1 > 2)$ **then** $H \leftarrow I(L_{i-2},L_{i-1})$ {Circuit passes through vertices on L_{i-1} in increasing order of orientation}

end

$H \leftarrow X$

Return H

end {of **Algorithm One_Sail_Hamiltonian**}

Proof of correctness: Map the vertices of A onto an nxn integer grid G in the following manner. Map each vertex $I(L_i,L_j)$ of A, $i < j$, to the entry of G with row i and column j (refer to figure 8.2). Two vertices that are row or column neighbors in G share an edge in G and this edge corresponds to the line segment that is bounded by the two corresponding vertices of A and lies on a line of A. Note that each line segment $[I(L_i,L_{i+1}),I(L_{i+1},L_{i+2})]$ $(i \in \{0,...,n-3\})$ on the concave chain of E(A) corresponds to the two connected edges in G $[(i,i+1),(i+1,i+1)]$ and $[(i+1,i+1),(i+1,i+2)]$ (these connected edges in G are indicated with dashed lines). Note also that the line L_0 corresponds to row 0 in G while the line L_{n-1} corresponds to column n-1 in G. A line $L_i \in A - L_0 - L_{n-1}$ corresponds

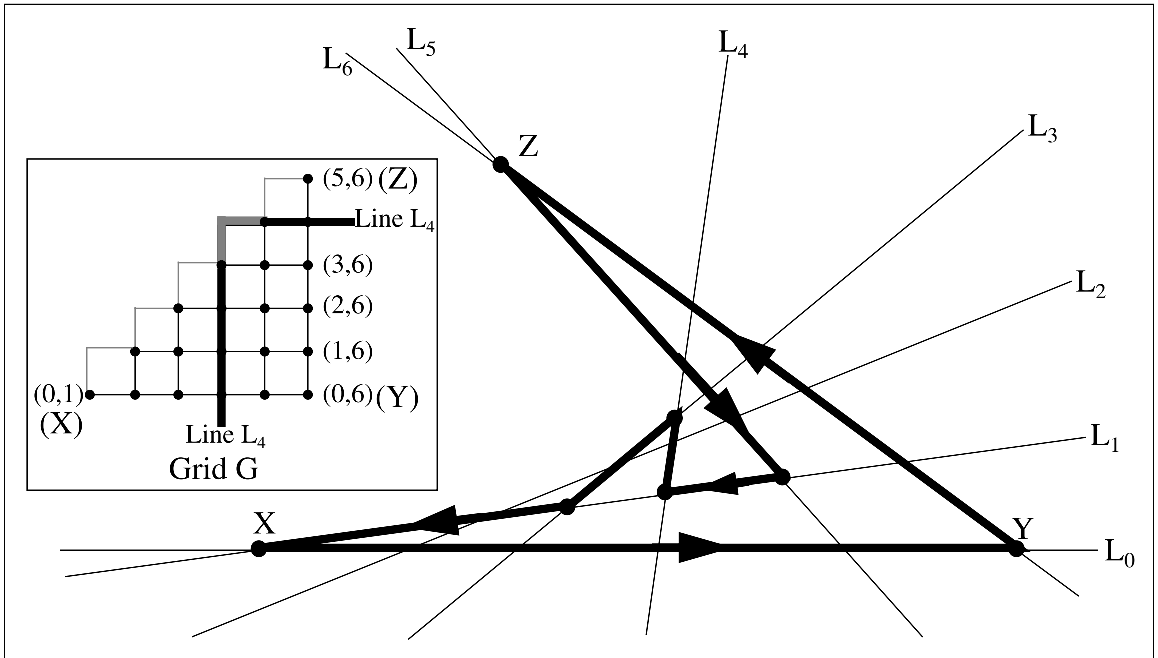


Figure 8.2. Tracing a hamiltonian circuit through the vertices of A of a 1-sail.

to the path in G from (0,i) to (i,i) to (i,n-1) (see figure 8.2 for an example).

On each line representation in G, the order of coordinate points of G corresponds to the orientation ordering of the lines in A. The property of 1-sails shown in lemma 8.1 allows us to define this mapping. Tracing a hamiltonian circuit through the vertices of A is then transformable to tracing a hamiltonian circuit through the mapped coordinates in G. Algorithm One_Sail_Hamiltonian outputs a hamiltonian circuit in G and thus, a hamiltonian circuit through the vertices of A as required. Even though there are $O(n^2)$ vertices in A (and hence, $O(n^2)$ mapped coordinates in G), we need only specify the vertices of A (mapped coordinates in G) at which the circuit makes a left or right turn. Since there are n lines in A, there are at most (n-1) + (n-2)/2 vertices in our stream-lined hamiltonian circuit. We know at which vertices these turns are made if the lines are sorted. There are n lines so we can perform the traversal in O(n) time. The complexity of our algorithm is bounded, however, by the fact that we first sort the lines. Therefore, we can find a stream-lined hamiltonian circuit for a 1-sail arrangement of n lines in O(n log n) time.♣

9. Complexity of Computing Hamiltonian Circuits

In the previous section, we showed that every 1-sail arrangement of lines admits a hamiltonian circuit which can be specified by giving the $\Omega(n)$ vertices at which the circuit makes a turn (even though the circuit must pass through the $\Theta(n^2)$ vertices of the arrangement). We also gave an O(n log n) time algorithm to output a stream-lined hamiltonian circuit. A natural question is whether one can do better. In this section, we prove that computing the hamiltonian circuit of a 1-sail arrangement of lines must take Ω(n log n) time thus establishing the optimality of our algorithm. Our proof of based on the following observation.

Lemma 9.1. Given an arbitrary arrangement A of n lines, every hamiltonian circuit of the arrangement must visit the vertices of E(A) in order.

Proof: By contradiction. Given arrangement A, suppose that there exists a hamiltonian circuit of A, $(\ldots,v_i,\ldots,v_j,\ldots,v_{i+1},\ldots,v_k,\ldots)$ such that v_i, v_{i+1}, v_j and v_k are vertices of E(A) and such that $[v_i, v_{i+1}]$ is an edge of E(A) i.e., v_i and v_{i+1} do not appear in order in the hamiltonian circuit, in spite of the fact that they are consecutive on E(A). Then either the path between v_j and v_{i+1} (not through v_i) splits the graph in two non-empty subgraphs, one containing v_i and one containing v_k, or the path from v_i to v_k through v_{i+1} is not simple. Given these conditions, no hamiltonian circuit can include both v_i and v_k, contradicting the assumption that the arrangement has a hamiltonian circuit.♣

We obtain our lower bound proof by a reduction from a variant of sorting, a problem we call **k-Subset Sorting**: the input is a set S of n integers and the output is any subset of n/k elements of S in sorted order.The lower bound of Ω(n log n) can be proven using standard arguments.

Lemma 9.2. The k-Subset Sorting problem has a complexity of Ω(n log n) under the algebraic tree model of computation, for any fixed value of k.

Given n positive integers, the input to 3-Subset Sorting, we construct in linear time a 1-sail arrangement of lines exactly as that in the proof of Theorem 7.1: given x_i, let L_i be the unique line of slope x_i tangent to the curve $y = x^2$. Observe that p_i, the intersection of L_i with the parabola will lie on an edge of the concave chain of the envelope of the arrangement thus created, but that it is not a vertex of the arrangement (and therefore not listed in any hamiltonian circuit of the arrangement). Moreover, since we are considering stream-lined hamiltonian circuits, we are not guaranteed that all the vertices of the arrangement are specified by the circuit.

Nevertheless, we show how given any stream-lined hamiltonian circuit of such a 1-sail arrangement we can output n/3 of the p_i's (hence the x_i's) in sorted order of x-coordinate, thereby solving the 3-Subset Sorting problem, which has an $\Omega(n \log n)$ time complexity.

Accordingly, consider the edges of the arrangement that lie on the concave chain of the (1-sail) envelope. Partition the concave chain into groups of 3 consecutive edges (determined by 4 consecutive vertices of the envelope). Let $\{e_j = [v_j,v_{j+1}], e_{j+1}= [v_{j+1},v_{j+2}], e_{j+2} = [v_{j+2},v_{j+3}]\}$ be one such group. Suppose that one of these 3 edges is contained in an edge e of the hamiltonian circuit. Then e contains exactly one p_i (edges of the circuit are colinear to the lines of the arrangement), which we can output, given e, in O(1) time. If none of those 3 edges is contained in an edge of the hamiltonian circuit, imagine removing the edges e_j, e_{j+1} and e_{j+2} from the arrangement (they are not used) and refer to figure 9.1. Vertices v_{j+1} and v_{j+2} must now have degree 2, so the hamiltonian circuit must make turns at v_{j+1}, v_{j+2} and v, the common neighbor of v_{j+1} and v_{j+2}. The vertices v_{j+1}, v, v_{j+2} are consecutive in the stream-lined hamiltonian circuit. We can therefore retrieve the edge $[v_{j+1},v_{j+2}]$ of the envelope of the arrangement, and output its corresponding p_i.

Algorithm Output_p_i(H)

{Input: A stream-lined hamiltonian circuit $H = (h_1,...,h_m)$ through the vertices of a 1-sail arrangement obtained as described above.}

{Output: A sequence of at least n/3 of the p_i's in sorted order of x-coordinate.}

begin

for j= 1,...,m by 2 **do**

  if edge $[h_j,h_{j+1}]$ or edge $[h_{j+1},h_{j+2}]$ contains a p_i, **then** output it

  else if edge $[h_j,h_{j+2}]$ contains a p_i, **then** output it.

end {of **algorithm Output_p_i**}

Proof of correctness: Detecting if an edge e contains a p_i can be done in O(1) time by extending e to a line L_e, computing the intersection point p of L_e with the parabola (if it exists), and verifying that p lies on e. Point p will lie on e only if e is colinear with one of the lines of the arrangement (i.e. only if L_e is tangential to the parabola). It follows from the previous discussion that at least n/3 of the p_i's are output by the algorithm. Since we know by lemma 9.1 that the vertices of the envelope of the arrangement are visited in order, we conclude that the p_i's are also output in sorted order of x-coordinate. The algorithm terminates in O(m) time.♣

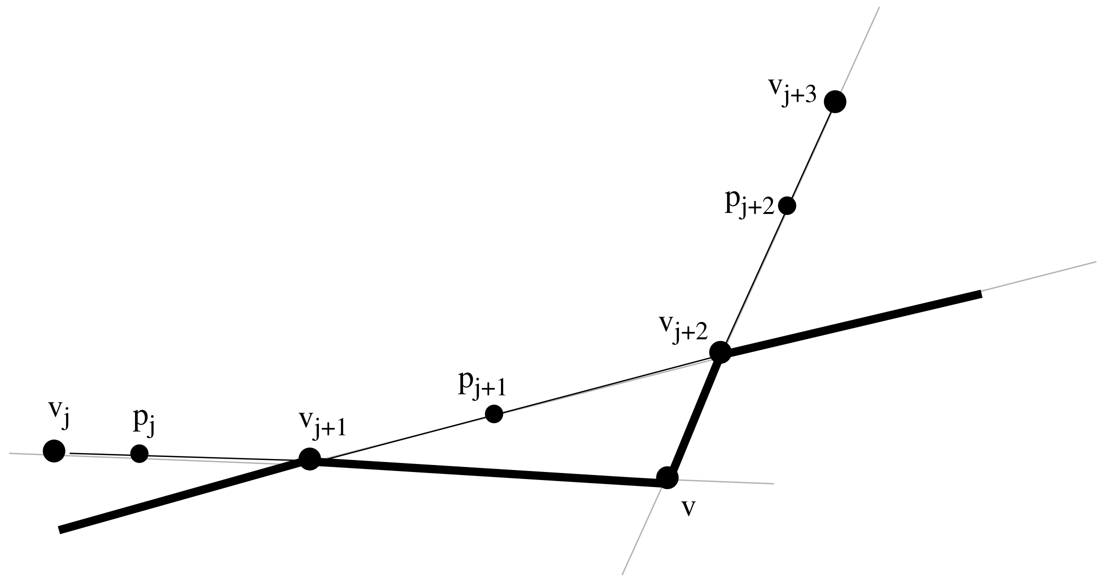


Figure 9.1. Outputting at least one out of every three p_i.

We can now prove the following.

Theorem 9.1. Given a 1-sail arrangement A of n lines, the complexity of computing a stream-lined hamiltonian circuit through the vertices of A is Ω(n log n) under the algebraic tree model of computation.

Proof: We showed in the previous discussion how to reduce 3-Subset Sorting to our problem. The steps are as follows:

1. Given $\{x_1,..., x_n\}$, produce the lines of the 1-sail arrangement A.
2. Compute a stream-lined hamiltonian circuit of size m through the vertices of A.
3. From the hamiltonian circuit, output at least n/3 of the x_i's in sorted order.

Step 1 is accomplished in O(n) time, while step 3 is done in O(m) time using algorithm Output_p_i. We only need to show that step 2 must take Ω(n log n) time. Since we solve 3-Subset Sorting, the total time of the procedure is Ω(n log n).

$(a_1 n + c_1) + t(n) + (a_2 m + c_2) \geq (a_3 n \log n + c_3)$ for $n \geq n_o$

Since the hamiltonian circuit is the output of step 2 (and t(n) is the dominant term on the LHS), we must have $t(n) \geq m$.

$(a_2+1) t(n)+ c_2 \geq (a_4 n \log n + c_4)$ for $n \geq n_o$, which implies that $t(n) \in \Omega(n \log n)$.♣

10. Conclusion

We have shown that envelope polygons are L-convex and hence that properties of L-convex polygons are useful in studying envelopes. By combining several properties of envelopes, we have also shown that it can be determined in O(n) time if a given simple polygon of n vertices is an envelope. From this, we can ask whether problems which have superlinear lower bounds or no known tight upper bounds for general polygons can be solved in linear time for envelope polygons. A few examples of such open problems include finding the longest or shortest diagonal of an envelope, or computing the geodesic diameter and the geodesic center of an envelope.

We introduced a hierarchy of classes of arrangements of lines based on the number of convex vertices of their envelopes. In particular, we looked at a class called sail arrangements: given a sail arrangement A of n lines, we can find the three convex vertices of E(A) and therefore the convex hull, the diameter and the points with minimum or maximum x-coordinate of the arrangement in O(n) time. Curiously, however, the complexity of constructing the remainder of the envelope of a sail arrangement remains Ω(n log n). We also showed that 1-sail arrangements admit hamiltonian circuits, and that a stream-lined hamiltonian circuit of a 1-sail arrangement of n lines can be computed in time Θ(n log n).

It remains, however, an open problem as to whether or not 2 and 3 -sail arrangement graphs admit hamiltonian circuits. Owing to the well defined geometrical structure of sail arrangements, we conjecture that 2 and 3 -sail arrangements are hamiltonian. Another open problem is to see if we can determine in O(n) time if a given arrangement of lines is a sail or not. Finally, it would be of interest to study further the hierarchy E_c, especially for values of c greater than 3, or to define additional new classes of arrangements for which interesting results can be obtained.

Acknowledgments

The results of sections 3 and 4 were obtained as a result of collaboration between Jack Snoeyink and the second author. The proof of lemma 2.6 is due to Patrice Belleville. The proof of lemma 9.2 is due to Bruce Reed. We would like to thank Pinaki Mitra, Prosenjit Bose and Tom Shermer for helpful discussions related to this paper. Finally, we thank the referees for their helpful suggestions.

References

[At86] Atallah M.J., "Computing the convex hull of line intersections," *Journal of Algorithms*, Vol. 7, 1986, pp. 285-288.

[BEPY91] Bern M., Eppstein D., Plassman P., Yao F."Horizon theorems for lines and polygons," *DIMACS Series in Discrete Mathematics and Computer Science*, No. 6, 1991, pp. 45-66.

[BFPRT73] Blum M., Floyd R.W., Rivest V.R., Tarjan, R.E., "Time bounds for selection," *Journal of Computer and System Sciences*, Vol. 7, No. 4, 1973, pp. 448-461.

[BO83] Ben-Or M., "Lower bounds for algebraic computation trees," *Proc. 15th ACM Sympos. on Theory of Comput.*,1983, pp. 80-86.

[CL85] Ching Y.T., Lee D.T., "Finding the diameter of a set of lines," *Pattern Recognition*, Vol. 18, 1985, pp. 249-255.

[CSSS89] Cole R., Salowe J., Steiger W., Szemerédi E., "An optimal-time algorithm for slope selection," *SIAM Journal on Computing*, Vol. 18, No. 4, 1989, pp. 792-810.

[DT93] Devroye, L. and Toussaint, G. T., "Convex hulls for random lines," *Journal of Algorithms,* vol. 14, 1993, pp. 381-394.

[Ed87] Edelsbrunner H., "Algorithms in combinatorial geometry," *Springer-Verlag*, 1987.

[EAT83] ElGindy H., Avis D., Toussaint G.T., "Applications of a two-dimensional hidden-line algorithm to other geometric problems," *Computing*, Vol. 31, 1983, pp. 191-202.

[ET89] ElGindy H., Toussaint G.T., "On geodesic properties of polygons relevant to linear time triangulation," *The Visual Computer*, Vol. 5, 1989, pp. 68-74.

[Gr72] Graham R., "An efficient algorithm for determining the convex hull of a planar set," *Information Processing Letters*, Vol. 1, 1972, pp. 132-133.

[HV49] Horn A., Valentine F.A., "Some properties of L-sets in the plane," *Duke Mathematics Journal*, Vol. 16, 1949, pp. 131-140.

[Ke91] Keil, M., "A simple algorithm for determining the envelope of a set of lines," *Information Processing Letters*, Vol. 39, 1991, pp. 121-124.

[LPSSSSTWY88] Lenhart W., Pollack R., Sack J.R., Seidel R., Sharir M., Suri S., Toussaint G.T., Whitesides S., Yap C.K., "Computing the link diameter of a simple polygon," *Discrete and Computational Geometry*, Vol. 3, No. 3, 1988, pp. 281-293.

[OR87] O'Rourke J., "Art gallery theorems and algorithms," *Oxford University Press*, 1987.

[Ro88] Roudneff, J.-P., "Arrangements of lines with a minimum number of triangles are simple," *Discrete & Computational Geometry*, vol. 3, No. 2, 1988, pp. 97-102.

[Su85] Suri S., "Computing the envelope of a set of lines," *Unpublished manuscript,* Johns Hopkins University, 1985.

[TA82] Toussaint G.T., Avis D., "On a convex hull algorithm for polygons and its application to triangulation problems," *Pattern Recognition*, Vol. 15, No. 1, 1982, pp. 23-29.

[To85] Toussaint G.T.,"A simple linear algorithm for intersecting convex polygons," *The Visual Computer*, Vol. 1, 1985, pp. 118-123.

[To91] Toussaint G.T., "Computing morphological properties of arrangements of lines," *Proc. Conf. Society for the Advancement of Materials Process Engineering,* Japan 1991, Dec. 11-14, 1991, Tokyo.

[Ve87] Vegter G., "Computing the bounded region determined by finitely many lines in the plane," *Technical Report 87-03,* Dept. of Mathematics and Computing Science, Rijkuniversiteit Groningen, 1987.

[Za75] Zaslavsky T., "Facing up to arrangements: face-count formulas for partitions of space by hyperplanes," *Mem. Amer. Math. Soc.*, Vol. 154, 1975.