Discrete Comput Geom 3:349-365 (1988)

Discrete & Computational
Geometry

The Orthogonal Convex Skull Problem*

Derick Wood[1] and Chee K. Yap[2]

[1] Department of Computer Science, University of Waterloo, Waterloo, Ontario, Canada N2L 3G1

[2] Department of Computer Science, Courant Institute of Mathematical Sciences, New York University, 251 Mercer Street, New York, NY 10012, USA

Abstract. We give a combinatorial definition of the notion of a simple orthogonal polygon being k-concave, where k is a nonnegative integer. (A polygon is orthogonal if its edges are only horizontal or vertical.) Under this definition an orthogonal polygon which is 0-concave is convex, that is, it is a rectangle, and one that is 1-concave is orthoconvex in the usual sense, and vice versa. Then we consider the problem of computing an orthoconvex orthogonal polygon of maximal area contained in a simple orthogonal polygon. This is the orthogonal version of the potato peeling problem. An $O(n^2)$ algorithm is presented, which is a substantial improvement over the $O(n^7)$ time algorithm for the general problem.

1. Introduction

The general problem is the following: given a simple polygon, find a maximal area convex polygon contained in it. Goodman [4] called this the "potato peeling problem;" he also posed the question of finding a finite algorithm for the problem. Independently, [10] considered the problem, dubbing it the "convex skull problem." Recently, [2] and [1] gave a finiteness criterion for the maximal area convex polygon contained in a given polygon with n vertices and solved the problem in $O(n^7)$ time. In this paper we investigate the orthogonal versions of this problem. Note that the finiteness of the convex skull problem in the orthogonal case follows immediately from the definitions. Figure 1 gives an orthogonal polygon that we

* The work of the first author was supported under a Natural Sciences and Engineering Research Council of Canada Grant No. A-5692 and the work of the second author was partially supported by NSF Grants Nos. DCR-84-01898 and DCR-84-01633.

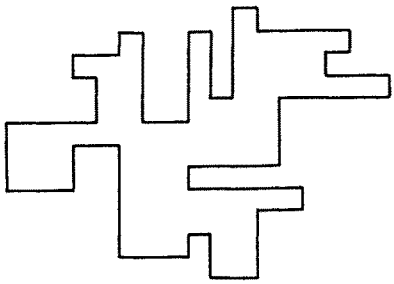

Fig. 1. An orthogonal polygon.

can peel to obtain orthoconvex polygons in various ways; see Fig. 2 for two example peelings.

The paper is organized as follows. We begin, in Section 2, by providing a definition of k-concave orthogonal polygons, where a 0-concave polygon is a rectangle and a 1-concave polygon is an orthoconvex orthogonal polygon. This is followed, in Sections 3 and 4, by discussions of two simpler problems, finding a maximal area enclosed rectangle in a simple orthogonal polygon and a maximal area enclosed 1-convex orthogonal polygon in a Manhattan skyline, respectively.

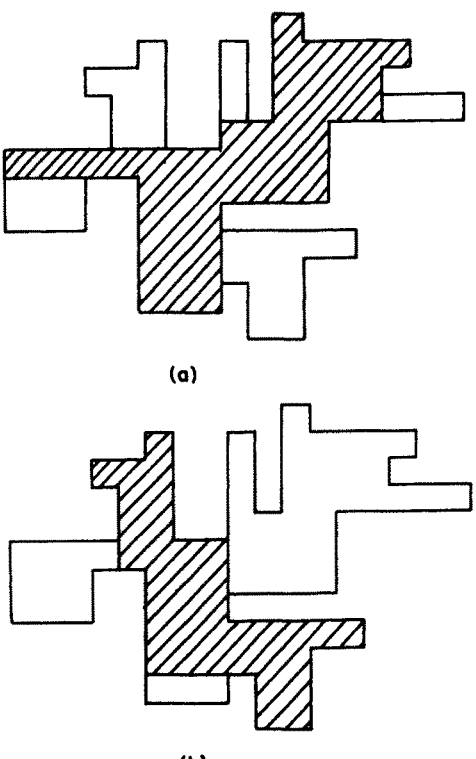

Fig. 2. Two peelings of the example polygon.

In Sections 5 onward, we then consider the general problem. First, in Section 5, we provide a simple algorithm that runs in $O(n^5 \log n)$ time. Second, in Section 7, we improve this result with a more complex algorithm that is based on the approach taken in Section 4. Both algorithms make use of the $O(n \log n)$ algorithm, described in Section 6, that computes a maximal staircase. The computation of maximal staircases is of independent interest.

2. k-Concavity for Orthogonal Polygons

We assume that the plane has associated with it a fixed cartesian coordinate system. An (*orthogonal*) *polygon* is a polygon whose edges are parallel to the two axes. Thus, vertices are at 90 and 270° corners. An orthogonal polygon is *simple* if it has no holes and its edges are nonintersecting. Since we only deal with simple orthogonal polygons in the remainder of this paper, we refer to them simply as polygons.

The usual definition of convexity clearly implies that rectangles are the only orthogonal polygons that are convex. However, less restrictive definitions are also known: for instance, following [6] define an orthogonal polygon to be *orthoconvex* if its intersection with any horizontal or vertical line is either a line segment or is empty. We provide an alternative approach to defining convexity by quantifying the "degree of concavity." We define the notion of "k-concavity," for $k = 0, 1, 2, \ldots$, such that rectangles correspond precisely to 0-concave polygons, and orthoconvex polygons to 1-concave polygons. We remark that a completely different approach to quantifying concavity is used in [9] where the area of "concave pockets" of a polygon is compared with the total area of the polygon; in contrast, our approach is strictly combinatorial. Our approach is similar to that of Sack (see [8] and [7]). We will, therefore, omit proofs of the stated results in this section. The interested reader is referred to [11], [8], and [7].

Let P be a simple polygon represented by its vertices $(v_0, v_1, \ldots, v_{n-1}, v_n)$ on its boundary path, listed in clockwise order. In this list, $v_0 = v_n$. We assign a *turning number* τ_i to each v_i such that $\tau_0 = 0$ and

$$\tau_{i+1} = \begin{cases} \tau_i + 1 & \text{if } v_i \text{ is a right turn,} \\ \tau_i - 1 & \text{if } v_i \text{ is a left turn.} \end{cases}$$

In particular, if P is a rectangle the sequence of turning numbers is $(\tau_0, \ldots, \tau_4) = (0, 1, 2, 3, 4)$. As another example, suppose P is the U-shaped polygon displayed in Fig. 3. Then the turning sequence is $(\tau_0, \ldots, \tau_8) = (0, 1, 2, 1, 0, 1, 2, 3, 4)$. Clearly, the turning numbers as defined depend on the particular choice of initial vertex v_0. One invariant property of these turning numbers is:

Lemma 2.1. $\tau_n = 4$.

We now turn to another invariant of these numbers. For $i = 1, \ldots, n-1$, we say i is a *local maximum* (resp. *minimum*) if $\tau_i > \tau_{i-1} = \tau_{i+1}$ (resp. $\tau_i < \tau_{i-1} = \tau_{i+1}$). We

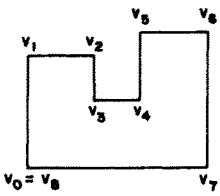

Fig. 3. A U-shaped polygon.

extend this definition to $i = 0$ by defining 0 to be a local maximum (resp. minimum) if $\tau_0 > \tau_1 = \tau_{n-1} - 4$ (resp. $\tau_0 < \tau_{n-1} - 4$). Note that this definition makes sense because of the previous lemma. Clearly, the number of maxima and the number of minima are equal. Also P is a rectangle if and only if there are no maximas. Let I be the set of integers of the form

$$\tau_i - \tau_j - 4\delta \qquad (j < i),$$

where i is a local maximum, j is a local minimum, and δ $(j < i)$ is equal to 1 or 0 depending on whether $j < i$. If P is a rectangle then I is defined to be the empty set.

Lemma 2.2. *The set I is an invariant of the polygon P.*

In view of this lemma, we may define the *concavity* of P to be

$$\begin{cases} \max\{|m| : m \in I\} & \text{if} \quad I \neq \varnothing, \\ 0, & \text{otherwise.} \end{cases}$$

By definition, the concavity of a rectangle is 0. A polygon is *k-concave* if it has concavity at most k. (Remark: a more refined measure of the concavity of P is to take two parameters, the minimum and maximum integers in I.)

We now characterize the k-concave polygons for $k \leq 2$. A rectangular polygon is naturally represented by four parameters:

$$(x_{\min}, x_{\max}, y_{\min}, y_{\max}).$$

We call the rectangle *semi-infinite* if exactly one of these four parameters is infinite. So semi-infinite rectangular polygons are essentially three-sided figures. The relatively easy proof of the following theorem is omitted.

Theorem 2.3.

(a) *The 0-concave polygons are precisely the rectangles.*
(b) *The 1-concave polygons are precisely the orthoconvex polygons.*
(c) *The 2-concave polygons are precisely those connected polygons each having the form*

$$R - \textit{interior}\left(\bigcup_{i=1}^{k} S_i\right), \qquad k \geq 0,$$

where R is a rectangle, and the S_i's are semi-infinite rectangles whose interiors are pairwise disjoint.

Note that strictly speaking, the rectangle R in (c) of the theorem is not necessary: 2-concave polygons can also be characterized as the bounded connected sets that are complements of a finite union of semi-infinite rectangles. Figure 1 illustrates a 2-concave polygon.

3. Maximal Rectangles

It should be clear from the definition that the concavity of P can be determined in linear time. Furthermore, the largest 0-concave polygon, that is, rectangle, contained in P can be found in time $O(n \log^3 n)$ using the algorithm of [3] or in time $O(n \log^5 n)$ using the elegant divide-and-conquer algorithm of [5]. The more interesting question is to ask for the largest 1-concave polygon contained in P. This will be the subject of the rest of this paper.

4. The Manhattan Skyline Problem

To begin, we consider a simple case that will give some insight into the general problem. Let P be an orthogonal polygon that is a histogram or a "Manhattan skyline." See Fig. 4 for an example of such a polygon. It is not hard to verify that P is 2-concave (using our characterization in the last theorem). A polygon Q is a *subpolygon* of a polygon P, if $Q \subseteq P$, when considered as point-sets.

Theorem 4.1. *A largest 1-concave subpolygon of a Manhattan skyline can be found in $O(n \log n)$ time and $O(n)$ space.*

Proof. Partition the skyline P into rectangles by introducing a minimal number of horizontal line-segments. It is easy to characterize these segments—each segment may be regarded as the extension of some horizontal edge of the skyline. It is easy to see that there are $O(n)$ horizontal segments and hence $O(n)$ rectangles (called *slabs*) in the partition. We can form a *slab tree* where each node corresponds to a slab, the root of the tree corresponding to the (unique) bottommost

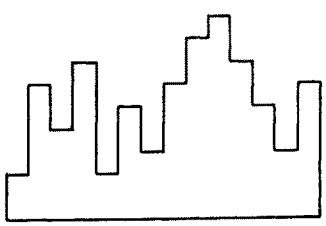

Fig. 4. A Manhattan skyline.

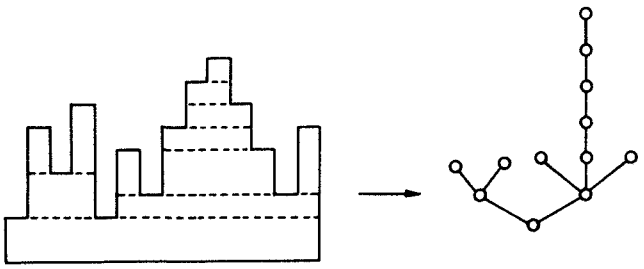

Fig. 5. Segmenting the skyline.

slab and the children of a node v consists of all those nodes whose slab sits directly on the slab corresponding to v. (Figure 5 should make the intent clear.)

We associate a "weight" with each node of the tree: the weight of a leaf is the area of the slab it represents. The weight of each internal node is the sum of the area of its own slab and the maximum of the weights of its children. The weight of the root gives the maximum area of a convex subset of P. The computational effort can be seen to be $O(n \log n)$ and the space requirement is $O(n)$. □

5. Maximal 1-Concave Subpolygons

For brevity, a *subpolygon* of a polygon always means a 1-concave (or orthoconvex) subpolygon of the polygon. (For emphasis we may still refer to a "1-concave (or orthoconvex) subpolygon.") A subpolygon is *maximal* if it is not properly contained in another subpolygon, and it is *maximum* if no other subpolygon has a strictly larger area. (Thus "maximal" means locally maximum). We now prove two simple properties of maximal subpolygons that can be exploited algorithmically.

Let $D \in \{\mathrm{N}, \mathrm{S}, \mathrm{E}, \mathrm{W}\}$ be any of the four compass directions, and let e be any edge of a polygon P. If the vector normal to e and outward pointing from P is in the direction D, then we say e is a *D-edge.* Two directions C and D are called *opposite* if either $\{C, D\} = \{\mathrm{N}, \mathrm{S}\}$ or $\{C, D\} = \{\mathrm{E}, \mathrm{W}\}$; if they are not opposite directions, then they are *adjacent* directions. For adjacent directions $C, D \in \{\mathrm{N}, \mathrm{S}, \mathrm{E}, \mathrm{W}\}$, the common endpoint of a C-edge and a D-edge is called a *CD-corner.* Note that there are no CD-corners if C and D are opposite; conversely if C and D are adjacent, then each polygon has at least one CD-corner. (Note: It is possible for edges of all four directions to meet at a point. In this case we may want to define the notion of CD-corners more carefully; this is not needed here and we leave it to the reader.) A corner is *convex* if a sufficiently small disk about it intersects the polygon in a convex set. For example, the Manhattan skyline in Fig. 4 has 1 S-edge, 14 N-edges, 8 W-edges, and 7 E-edges; it also has 8 convex NW-corners and 7 nonconvex NW-corners, and a unique convex SW-corner.

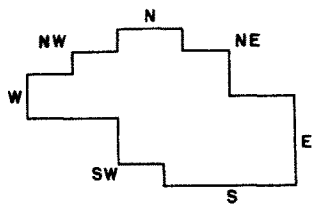

Fig. 6. A 1-concave polygon with empty SE-staircase.

We call an edge *e extremal* if, in a clockwise transversal of the boundary of P, the turning number increases both times that we pass the endpoints of e. If e is an extremal E-edge, we call it an "easternmost" edge; and similarly for "northernmost," etc. For instance, the Manhattan skyline of Fig. 4 has five northernmost edges and one extremal D-edge for each of the other three directions D. It is not hard to verify that a polygon is 1-concave if and only if it has a unique extremal D-edge for each $D \in \{N, S, E, W\}$. If e_N and e_E are the extremal N- and E-edges of a 1-concave polygon then the boundary from e_N to e_E is monotonically nondecreasing in the southern and eastern directions. We call this portion of the boundary (not including e_N and e_E) the *NE-staircase.* Note that the NE-staircase is empty if and only if the northernmost and easternmost edges meet, if and only if the NE-corner is unique. We can define the SW-staircase, etc., similarly. Therefore, the boundary of a 1-concave polygon can be decomposed into the following parts: the four extremal edges, and between any two adjacent extreme edges, the "staircase" connecting them. Figure 6 illustrates these definitions. Two line segments are said to *overlap* if their intersection is a line segment of positive length.

Lemma 5.1. *Let C be a maximal (1-concave) subpolygon of P and e be any edge of C.*

1. *Then the edge e must overlap some edge of P.*
2. *Let e overlap edge f of P; then e is a D-edge of C if and only if f is a D-edge of P.*
3. *If e is extremal, then e is contained in some (not necessarily extremal) D-edge of P.*

Proof. The proof of this lemma is easy: (2) is immediate. If either (1) or (3) is violated, we can extend C to a strictly larger subpolygon. □

This simple lemma can be the basis of an $O(n^5 \log n)$ algorithm for finding a largest 1-concave subpolygon of a polygon P: for each quadruple (e_N, e_S, e_E, e_W) of edges of P, where each e_D is a D-edge of P, find in $O(n \log n)$ time the maximal 1-concave subpolygon C of P such that the extremal D-edge of C is contained in e_D, for $D \in \{N, S, E, W\}$; this is done by constructing in $O(n \log n)$ time the "best" staircase between e_C and e_D for each pair of adjacent directions C, D. This can be done using the plane-sweep technique; the details are given

in the next section. In Section 7 we improve this simple solution to the skull problem.

6. Computing the Maximal Staircase

Let P be a polygon, e_N be an extremal N-edge of P, and e_W be an extremal W-edge of P. Let S be a staircase joining e_W and e_N, an (e_W, e_N)-staircase. Note that there may be no such staircase. S is said to be *maximal*, if there is no other (e_W, e_N)-staircase having a point to the northwest of S. See Fig. 7 for an illustration of this concept. Hence, if there is an (e_W, e_N)-staircase, then there is a unique maximal one.

To compute the maximal staircase we use the plane-sweep approach. A horizontal line is swept upward from the bottommost endpoint of e_W until it reaches e_N. From this viewpoint, candidate partial staircases from e_W appear, disappear, are modified, or are unaffected when edges of P are met. In a maximal staircase each horizontal edge must meet an N-edge of P and each vertical edge must meet a vertical edge of P. If this were not so, an edge of the staircase could be moved either farther west or north. Some of the difficulties in finding the maximal staircase are to be seen in Fig. 8. Nonconvex parts of the polygon become *obstacles* to the tracing out of the maximal staircase. For this reason we may have more than one candidate partial staircase at each position of the sweepline. These are shown as dashed lines in Fig. 8. Fortunately, from the sweepline's point of view each candidate staircase is a point and each obstacle is an interval. Vertical edges have no effect on them, but new horizontal edges do. We enumerate the situations that occur on meeting a horizontal edge H below.

Case 1. H is an S-edge.

(a) It equals the interval of an obstacle, that is, the obstacle is a hole. Remove the staircase to its immediate right, if there is a staircase to its left; see Fig. 9(a).

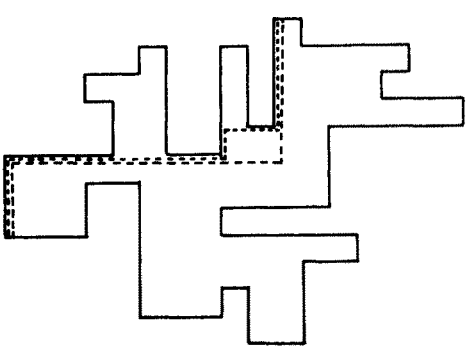

Fig. 7. A maximal staircase.

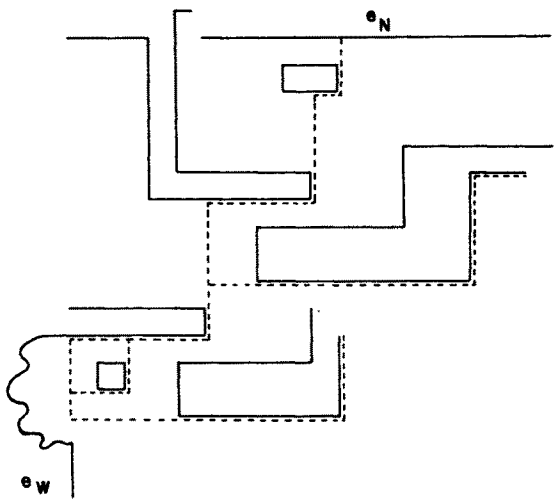

Fig. 8. The difficulty of finding a maximal staircase.

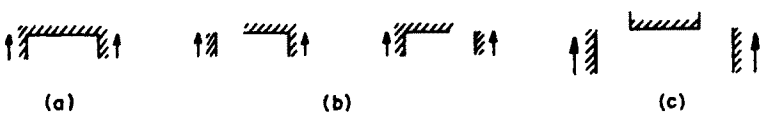

Fig. 9. On meeting an S-edge.

(b) It adjoins an obstacle's interval. This causes the obstacle to shrink, but it has no effect on the neighboring staircases; see Fig. 9(b).

(c) It falls within an obstacle's interval. This causes the obstacle to split into two obstacles, but it has no further effect; see Fig. 9(c).

Case 2. H is an N-edge.

(a) It is disjoint from all obstacles; hence, it creates a new obstacle. The staircase immediately to its left (if there is one) is copied. The copy follows H to its right endpoint and up the vertical edge that is to be found there; see Fig. 10(a).

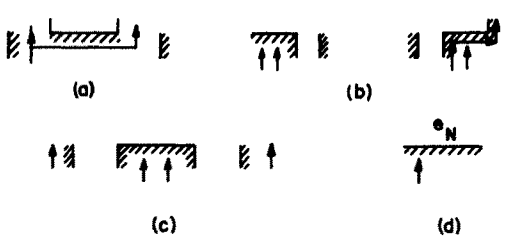

Fig. 10. On meeting an N-edge.

(b) It adjoins an obstacle's interval. This causes the obstacle to expand. If it is to the left of the obstacle, it terminates any staircases that hit it. If it is to the right, then all staircases that hit it are terminated, except for the leftmost one. This latter staircase follows H to its right endpoint and up the vertical edge that is to be found there; see Fig. 10(b).

(c) It joins two obstacles into one; see Fig. 10(c). Any staircases that hit it are terminated.

(d) H is e_N. If there are staircases that hit it, take the leftmost one; see Fig. 10(d).

We can keep track of obstacles, that is intervals, and staircases, that is, points, using two balanced search trees. The case analysis above demonstrates that we need to insert and delete intervals, perform intersection queries with intervals, insert and delete points, and perform range queries with points. Since, the obstacle's intervals are disjoint we can represent them by their endpoints in a search tree. Similarly, we can represent staircases as points in a second search tree. This ensures logarithmic performance, if the trees are balanced. There are two remaining subtleties. First, it appears that arbitrarily many staircases can be terminated in Case 2. However, because there can be at most n staircases introduced (one for each horizontal edge), there can be at most n staircases terminated during the algorithm. This contributes $O(n \log n)$ time to the running cost. Second, each partial staircase needs to be kept in its entirety. This appears, at first glance, to require $O(n^2)$ space; however, we keep only one copy of common portions of staircases. This ensures that $O(n)$ space is sufficient for all staircases.

There remain a number of technical details that are left to the interested reader, for example, the algorithm has to take into account the obstacles that already exist when the plane sweep begins.

To summarize, we have:

Theorem 6.1. *Given a polygon P with a total of n edges and two edges e_W and e_N in P, the maximal (e_W, e_N)-staircase can be found in $O(n \log n)$ time and $O(n)$ space.*

7. Algorithm for Maximal 1-Concave Subpolygon

Let P be an arbitrary but fixed polygon in this discussion. We now describe an algorithm to compute the area of a maximal area 1-concave subpolygon of the polygon P. It should be relatively straightforward to convert our description to an algorithm that actually computes a maximal area 1-concave subpolygon.

As in Section 4, we introduce a minimal number of horizontal line segments sufficient to partition P into rectangles that we call *slabs* (of P). A slab B is *immediately below* another slab B' if the S-edge of B' overlaps the N-edge of B. The reflexive transitive closure of this relation is the *below* relation. The inverse of the "(immediately) below" relation is the "(immediately) above" relation. Again, we can form a directed graph with nodes representing the slabs and directed edges from each slab to those slabs immediately below it. This digraph

is a generalization of the slab tree previously defined for the Manhattan skyline. Our algorithm processes one slab at a time, starting from slabs that have no slabs above them, in the order of any topological sort of the digraph.

A simple but useful concept is that of "illumination." Let X be any set of points within P. A point p is said to be *illuminated by* X (*from below*) if there exists a point $q \in X$ vertically below p such that the line-segment $[p, q]$ lies in P. We can similarly talk of p being illuminated from the west if q is west of p, etc. Typically, $X = B$ is a slab and the part illuminated by X from below is thus a Manhattan skyline, denoted M_B, with $B \subseteq M_B$ as its "base." Now let B be fixed and consider a point $p \notin B$ in the boundary of M_B. If p is not in the boundary of P then p must be in some eastern or western edge of M_B. A maximal connected set of such points forms a line-segment called a *window* of M_B. The window is either eastern or western depending on the edge of M_B that contains it. Let w be any window. A subpolygon C is said to *abut* a window w if the easternmost or westernmost edge of C overlaps w. The nonilluminated component of C that abuts a window w is called the *pocket outside the window* w.

Let $M \subseteq P$ be a Manhattan skyline obtained by illumination from its base B. We will associate an *induced slab tree* T_B with M where the induced slab tree is similar to, but not necessarily identical to, the slab tree associated with M in Section 4. More precisely, T_B is induced by P in the sense that each slab of T_B has the form $M \cap S$, where S is a slab of P. Therefore T_B is in general a refinement of the slab tree defined for M in Section 4, since what would be a single slab in the slab tree may be split into a number of slabs in the induced slab tree. It is convenient to speak of a node of T_B interchangeably with the slab it represents. With each node v in T_B, we associate the following values

$$\alpha(v), \beta(v),$$

and

$$\varepsilon_i(v), \omega_i(v) \qquad (i = 0, 1, 2, 3)$$

defined as follows:

1. An *eastern flank* of v is an orthoconvex subpolygon $C \subseteq M$ such that C has unique NW-, SW-, and SE-corners, with the NW-corner lying in the ray projecting vertically downward from the SE-corner of v. $\varepsilon_0(v)$ is the area of the largest eastern flank of v. Note that the largest eastern flank of v is unique. See Fig. 11 for an example of an eastern flank.

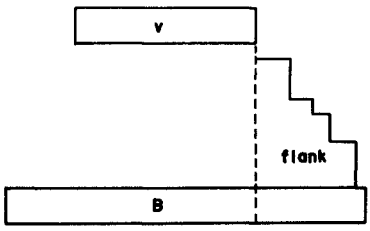

Fig. 11. An eastern flank.

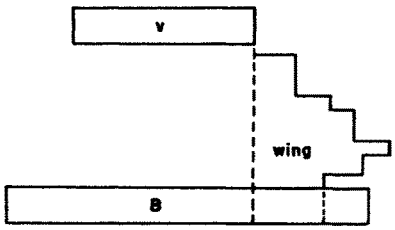

Fig. 12. An eastern wing.

2. An *eastern wing* of v is an orthoconvex subpolygon $C \subseteq P$ such that C has unique NW- and SW-corners and has its NW-corner lying in the ray projecting vertically downward from the NE-corner of v. $\varepsilon_1(v)$ is the area of the largest eastern wing of v. Unlike the flank above, C need not be contained in M. Note that since an eastern flank is also an eastern wing, we have $\varepsilon_0(v) \leq \varepsilon_1(v)$. This is illustrated in Fig. 12.
3. Suppose v abuts an eastern window w. A *hidden eastern wing* of v is an eastern wing that is additionally restricted to lie within the pocket outside w. Now define $\varepsilon_2(v)$ as the area of the largest hidden eastern wing of v; if the window w does not exist then define $\varepsilon_2(v) = 0$. Clearly, $\varepsilon_2(v) \leq \varepsilon_1(v)$. This is illustrated in Fig. 13.
4. An *eastern chamber* of v is an orthoconvex subpolygon $C \subseteq P$ such that C has a unique SW-corner and its westernmost edge contains the eastern edge of v. $\varepsilon_3(v)$ is the area of the largest eastern chamber of v. Note that a chamber is 1-concave and it is necessarily contained in a pocket (and thus completely hidden); again $\varepsilon_3(v) = 0$ if v does not abut an eastern window. This is illustrated in Fig. 14.
5. $\omega_i(v)$, for $i = 0, \ldots, 3$, is the western analogue of $\varepsilon_i(v)$.
6. $\beta(v)$ is the area of the "rectangle below v," that is, the unique largest rectangle in M whose N-edge coincides with the N-edge of v; see Fig. 15.
7. For lack of a better word, an orthoconvex subpolygon C is called *v-centered* if it contains the rectangle R below v and $C - R$ has two components that are either both wings or a wing and a chamber. Note that the components can be empty and it is necessarily the case that one component is eastern and the other western. Finally, $\alpha(v)$ is the area of a largest v-centered orthoconvex subpolygon.

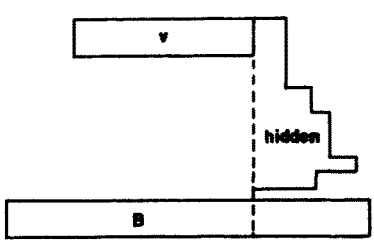

Fig. 13. A hidden eastern wing.

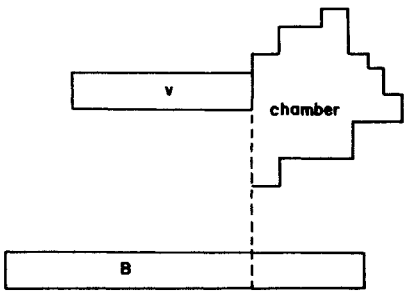

Fig. 14. An eastern chamber.

A *stage* of the algorithm is defined as the period during which a particular slab is being processed. At each stage of the algorithm we have a set $\mathcal{B}$ of *base slabs* of P such that

(a) every slab processed up to this point is above some slab in $\mathcal{B}$ and
(b) for every unprocessed slab B that is ready to be processed next, if B' is immediately above B, then $B' \in \mathcal{B}$.

(Note: The slabs in $\mathcal{B}$ need not be pairwise incomparable with respect to the "above/below" relation. To see this, suppose B in $\mathcal{B}$ has two slabs B' and B'' immediately below it. After processing B' we now have B' in $\mathcal{B}$ but we still cannot remove B from $\mathcal{B}$ until B'' is processed.) For each base slab B we store the induced slab tree T_B. At each node v in T_B we have the values ε_i, ω_i, etc., described above. We claim:

Lemma 7.1. *By the time all the slabs of P are processed, the area of a maximal area orthoconvex subpolygon of P is equal to the largest $\alpha(v)$ encountered over the course of the algorithm.*

Proof. Suppose P^* is a maximal area orthoconvex subpolygon. Consider those points of P^* that are illuminated by the southernmost edge of P^*. Among these, the points with the highest altitude form a horizontal line segment σ on the boundary of P^*. Define R to be the largest rectangle in P^* whose N-edge coincides with σ. Let B be the slab of P whose S-edge contains the S-edge of R. Consider the induced slab tree T_B constructed during the algorithm: if v is the node of T_B

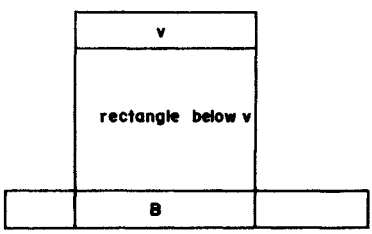

Fig. 15. Rectangle below v.

whose N-edge coincides with the N-edge of R (note that v is well defined) then we see that P^* is indeed a v-centered polygon. So, by definition, $\alpha(v)$ is the area of P^*. □

It remains to show how the values described above can be maintained. The set of values $\{\varepsilon_0(v), \omega_0(v), \beta(v): n \in T_B\}$ are straightforward to compute in linear time from T_B. Just do a horizontal sweep from the base of the Manhattan skyline upward. Consider how to determine $\varepsilon_0(v)$. Assume the sweepline is at level $y = y_0$. If the line $y = y_0$ intersects the Manhattan skyline in $k > 0$ sections then we must maintain m different values corresponding to k possible forms of (staircase defining) the eastern flank. Next, when the sweepline rises to the next level, the number of sections will increase from k to some $k' \geq k$. There are also k' new values of ε_0 to be computed, but they are easily done using the k previous values. Note that $\varepsilon_i(v)$ and $\omega_i(v)$, for $i = 2, 3$, are values that depend only on the pockets outside the windows of the Manhattan skyline: for this reason we call these the *hidden values.* Assuming these are (recursively) available, we can compute the rest of the values as shown next:

Lemma 7.2. *Let T_B be the induced slab tree rooted at B. If the values $\varepsilon_2(v)$, $\varepsilon_3(v)$, $\omega_2(v)$, and $\omega_3(v)$, for each node v in T_B, are available, then all the values for ε_1, ω_1, and α can be computed in $O(n)$ time.*

Proof. The value $\varepsilon_i(v)$ (and $\omega_1(v)$ similarly) can be computed by the following formula:

$$\varepsilon_1(v) = \max\{\varepsilon_0(v), \varepsilon_2(v), \varepsilon_1(v') + \delta(v)\},$$

where v' is the node immediately below v, $\delta(v)$ is the area of the (possibly degenerate) rectangle whose NW-corner (resp. NE-corner) coincides with the SE-corner (resp. NE-corner) of v (resp. v') and whose S-edge lies in the S-edge of B. (If $v = B$, then v' is undefined, but we let $\varepsilon_1(v')$ be 0 in the formula.) To justify the formula, note that the maximum eastern wing is either fully illuminated ($\varepsilon_1 = \varepsilon_0$) or is contained in the eastern pocket abutting v ($\varepsilon_1 = \varepsilon_2$) or is entirely south and east of the SE-corner of v. This last case gives a recursive expression in terms of $\varepsilon_1(v')$ and the area $\delta(v)$ described above.

The formula for $\alpha(v)$ is

$$\alpha(v) = \beta(v) + \max\{\omega_1(v) + \varepsilon_2(v), \omega_2(v) + \varepsilon_1(v), \omega_1(v) + \varepsilon_1(v)\}.$$

It follows almost immediately from the definition of $\alpha(v)$. Again we have three cases for the maximum v-centered subpolygon P^*: the first case is where the northernmost edge of P^* is in an eastern chamber of v, the second case is where the northernmost edge is in a western chamber, and the last case is where the northernmost edge is (partly) illuminated. □

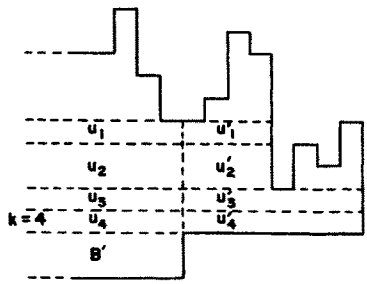

Fig. 16. Formation of a new eastern window.

Our next task is to show how to update the hidden values ε_i, ω_i, for $i = 2, 3$, assumed in the above lemma. Updating is necessary whenever the windows of the Manhattan skyline change, more precisely, whenever a new window is formed. Suppose B' is the new base being processed in the current stage.

(a) The current slab B' is immediately below a single base $B \in \mathcal{B}$. Then a new eastern window is formed if and only if the eastern edge of B is to the east of the eastern edge of B'. For simplicity, let us first assume that no new western window is formed. Let w be the new eastern window (w is aligned with the eastern edge of B'). Let $v_1, \ldots, v_k$ be the slabs of T_B that intersect w, where v_i is immediately above v_{i+1} and $v_k = B$. Let w split v_i into a western slab u_i and an eastern u_i' (see Fig. 16). The tree T_B is then split into a western T and an eastern T'. Note that the new slab tree T_B' is just T appended with a new root B'; also T' will be useful for the computation to be described next but it can be discarded after that. It is not hard to see that using any reasonable representation (such as pointers for edges) for T_B, we can construct T and T' from T_B in linear time. Clearly, to compute the hidden values of T it is sufficient to compute $\varepsilon_2(u_i)$ and $\varepsilon_3(u_i)$, for $i = 1, \ldots, k$.

Consider the Manhattan skyline corresponding to T' (this is just the part illuminated by u_k'). For each v in T', the values $\varepsilon_i(v)$, for $i = 2, 3$, are intact from T_B. If we "seal" off the window w (that is, treat w as if it were part of the boundary of P), then the values of $\omega_i(v)$, for $i = 2, 3$, are also available for all $v \in T'$, because these values are the original values plus the area of a suitable rectangle. By the previous lemma, the other values ($\alpha(v)$, $\varepsilon_0(v)$, $\varepsilon_1(v)$, etc.) can be computed from ε_i and ω_i, for $i = 2, 3$, in linear time. It is easy to see that

$$\varepsilon_2(u_i) = \beta(u_i') + \varepsilon_1(u_i').$$

To determine $\varepsilon_3(u_i)$ we proceed as follows. For each $v \in T'$, it is easy to see that the largest western flank P of v abuts the window w. Furthermore, the northern endpoint of $P \cap w$ is the NW-corner of some u_j', for $j = 1, \ldots, k$; we say v *belongs to* u_j' in this case. If v has an empty western flank then v is one of the u_j' and we say v belongs to itself. We claim that $\varepsilon_3(u_i)$ is the maximum value of the expression

$$\omega_0(v) + \beta(v) + \max\{\varepsilon_1(v), \varepsilon_3(v)\}, \tag{1}$$

where v ranges over the nodes belonging to u'_j for some $j \leq i$. (Note that ε_1 and ε_2 are intact.) To see this, let P^* be a maximum eastern chamber of u_i. The southernmost edge of P^* is contained in the southern edge of u'_k. As in the proof of Lemma 6, consider the points illuminated by the southernmost edge of P^*: the subset σ of these points with the highest altitude forms a horizontal segment. Let v be the node of T' whose northern edge is σ (this is well defined). Clearly, v belongs to some u'_j with $j \leq i$. The area of P^* is given by equation (1) for this choice of v, proving the claim. Computationally, it is easy to partition the nodes of T' according to the u'_i that they belong to. Then for each i, determine the largest value of expression (1) among those v belonging to some $j \leq i$. All this can be done in linear time.

This completes the discussion of updating the hidden values of each slab v in the case where one new window is formed when we process B'.

(b) The current slab B' is immediately below a single base slab B in $\mathcal{B}$, but now the E-edge of B' is west of the E-edge of B, and the W-edge of B' is east of the W-edge of B (so a new eastern and new western window are formed). Now we have to split the slab tree T_B into three trees T'', T, and T' corresponding to the eastern, central, and western portion of the split, respectively. We can apply the previous method using T' to compute the ε_i values for the new eastern window, and T'' for the western window.

(c) The current slab B' is immediately below a single base slab B in $\mathcal{B}$, but its N-edge contains the S-edge of B. This case is also trivial.

(d) The current slab B' is not below any other slabs. This case is trivial.

(e) The current slab B' is immediately below two or more base slabs in $\mathcal{B}$. At most two windows are formed and these can be treated as in (a).

We have now completely described the algorithm and conclude with our main result.

Theorem 7.3. *A maximal area 1-concave subpolygon of a simple orthogonal polygon with n vertices can be found in $O(n^2)$ time.*

Proof. Using the above algorithm, there are $O(n)$ slabs to process and each slab can be processed in linear time. The algorithm as described only keeps track of the area but it is not hard to modify them to remember the actual subpolygon with the remembered area. Alternatively, we can proceed in two stages as follows: in the first stage, proceed as described except that we also keep track of the extremal N-, E- and W-edge of the various maximum 1-concave polygons (flanks, wings, etc.) connected with each of the numbers $\varepsilon_i(v)$, etc. At the end of the first stage we have the extremal edges of the maximum subpolygon. It is now quite easy to find the connecting staircases between adjacent extremal edges in $O(n \log n)$ time. □

Finally, if we consider the convex skull problem for nonsimple polygons, we find that the above approach still works. Indeed, the only time our algorithm must take account of holes is when finding a maximal staircase, see Section 6.

However, even there the algorithm performs correctly within the same time bounds. Therefore, we conclude that the following theorem holds.

Theorem 7.4. *A maximal area 1-concave subpolygon of an orthogonal polygon with n vertices can be found in $O(n^2)$ time.*

References

1. J. S. Chang and C. K. Yap. A polynomial solution for potato-peeling problem. *Discrete Comput. Geom.*, **1**:155–182, 1986.
2. J. S. Chang and C. K. Yap. A polynomial solution to potato-peeling and other polygon inclusion and enclosure problems. In *Proceedings of the 25th Annual Symposium on the Foundations of Computer Science*, pp. 408–416, 1984.
3. B. Chazelle, R. L. Drysdale, and D. T. Lee. Computing the largest empty rectangle. *SIAM Journal on Computing*, **15**:300–315, 1986.
4. J. Goodman. On the largest convex polygon contained in a non-convex region. *Geometrica Dedicata*, **11**:75–78, 1981.
5. M. McKenna, J. O'Rourke, and S. Suri. Finding the Largest Rectangle in an Orthogonal Polygon. Technical Report JHU/EECS-85/09, Electrical Engineering and Computer Science, The Johns Hopkins University, 1985.
6. Th. Ottmann, E. Soisalon-Soininen, and D. Wood. On the definition and computation of rectilinear convex hulls. *Information Sciences*, **33**:157–171, 1984.
7. J.-R. Sack. Rectilinear Computational Geometry. Technical Report SCS-TR-54, School of Computer Science, Carleton University, 1984.
8. J.-R. Sack. A simple hidden-line elimination algorithm for rectilinear polygons. In *Proceedings of the 20th Annual Allerton Conference on Communication, Control, and Computing*, pp. 437–446, 1983.
9. J. Sklansky. Measuring concavity on a rectangular mosaic. *IEEE Transactions on Computers*, **21**:1355–1364, 1972.
10. T. C. Woo. The Convex Skull Problem. Technical Report TR 86-31, Department of Industrial and Operations Engineering, University of Michigan, Ann Arbor, MI, 1986.
11. D. Wood and C. K. Yap. The Orthogonal Convex Skull Problem. Technical Report CS-86-57, Department of Computer Science, University of Waterloo, 1986.

Received May 31, 1985, and in revised form July 10, 1987.