JOURNAL OF ALGORITHMS 7, 258–269 (1986)

An Optimal Algorithm for Finding Minimal Enclosing Triangles

JOSEPH O'ROURKE, ALOK AGGARWAL, SANJEEV MADDILA, AND MICHAEL BALDWIN

Department of Electrical Engineering and Computer Science, The Johns Hopkins University, Baltimore, Maryland 21218

Received June 20, 1984

Klee and Laskowski's $O(n \log^2 n)$ algorithm for finding all minimal area triangles enclosing a given convex polygon of n vertices is improved to $\Theta(n)$, which is shown to be optimal both for finding all minima and for finding just one.

1. INTRODUCTION

The problems of circumscribing and inscribing convex polygons with, respectively, minimum and maximum area k-gons have been studied extensively in the recent past because of their applications to robotics and collision avoidance problems [CY, DA]. In particular, Klee and Laskowski [KL] have given an $O(n \log^2 n)$ algorithm for finding all local minima (with respect to area) among the triangles that contain a given convex n-gon P. (A triangle T is a local minimum if there exists some $\varepsilon > 0$ such that area of $T' \geq$ area of T for each triangle T' that is at a Hausdorff distance less than ε from T.) The strength of their paper lies in establishing an elegant geometric characterization of these minima, which permits the avoidance of brute-force optimization. They show that although there may be infinitely many local minima, these fall into at most n equivalence classes, each of which is a "segment" of triangles having the same area. Their algorithm computes all the local minima in $O(n \log^2 n)$ time. Selecting the global minima from these can then be accomplished in an additional $O(n)$ time.

Klee and Laskowski find each local minimum afresh, without using any information gleaned from finding previous local minima. Inspired by Toussaint's "rotating caliper" algorithms [T], we show that it is possible to move from one local minimum to the "next" in an orderly fashion,

0196-6774/86 $3.00

achieving a linear-time algorithm.[1] This is obviously asymptotically optimal for finding *all* minima, as there could be as many as $n/3$ of them (e.g., for a regular n-gon). We show that it is also optimal for the problem of finding just *one* global minimum.

First, we review results from Klee and Laskowski's paper, and then present two key lemmas in Section 3 that establish the possibility of moving from one minima to another quickly. In Section 4 we present some technical lemmas needed to justify the algorithm, which is described in Section 5 and proved correct and optimal in Section 6.

2. Klee and Laskowski's Results

Throughout the paper, we will use the following notation. The convex n-gon to be enclosed is P. The enclosing triangle has sides A, B, and C, with vertices α, β, γ opposite these sides; a, b, and c will be points of sides A, B, and C, or sometimes points of the polygon near those sides. Vertices of the polygon will be referred to by their indices, which increase clockwise.

We will state three simplified versions of Klee and Laskowski's geometric characterizations without proof.

THEOREM 1 (Klee). *If T is a local minimum among triangles containing P, then the midpoint of each side of T touches P.*

Klee has established a much stronger version of this theorem, generalized to arbitrary dimensions and arbitrary convex enclosing bodies [K].

We will say that a triangle side S is *flush* with an edge e of P (or just flush with P) if $S \supseteq e$. The second characterization we need is implied by a more precise result from [KL]:

THEOREM 2 (Klee and Laskowski). *If T is a local minimum among triangles containing P, then at least one side of T is flush with P.*

We will use the convention throughout the remainder that side C is the one guaranteed flush by this theorem.

Finally, we introduce Klee and Laskowski's notion of *low* and *high* (again in simplified form), the key to their algorithm. Let $h(p)$ be the height of p above the line determined by side C. Then fixing C induces a partition of the vertices of P into a *left chain*, consisting of all those vertices p for which $h(p) \leq h(p+1)$, where $p+1$ is the next vertex clockwise from p, and a *right chain*, consisting of all the remaining vertices. Let a be

[1] Another linear algorithm has been proposed [DB], but it does not find a minimum in all cases [O].

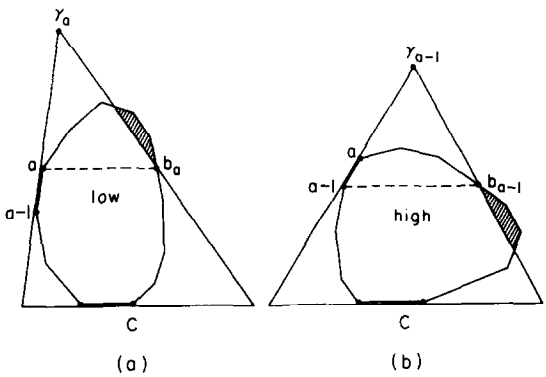

FIG. 1. In (a), edge $[a-1, a]$ is low; (b) edge $[a-1, a]$ is high.

(the index of) a vertex on the left chain, $a-1$ the previous vertex counterclockwise, A the line flush with the edge $[a-1, a]$, γ_p the point on A such that $h(\gamma_p) = 2h(p)$, and finally, for any point a on the left chain, let b_a be the point on the right chain with $h(b_a) = h(a)$.

DEFINITION. Edge $[a-1, a]$ is *low* if $\gamma_a b_a$ intersects P above b_a, *high* if $\gamma_{a-1} b_{a-1}$ intersects P below b_{a-1}, and *critical* if neither low nor high.

These definitions are illustrated in Fig. 1. Following [KL], we define a circumscribing triangle to be *P-anchored* if one edge is flush and the other two edges touch P at their midpoints; our convention will be that C is the flush edge. A P-anchored triangle is not necessarily a local minimum, but every local minimum is P-anchored.

THEOREM 3. [Klee and Laskowski]. *In order of increasing height from C, both the left and right chains consist of a sequence of low edges, followed by at most two critical edges, followed by a sequence of high edges. For each flush C, a P-anchored triangle exists. If ABC is a P-anchored with C flush, then the midpoints of sides A and B either lie on critical edges, or on a vertex between a low and a high edge.*

Klee and Laskowski use the notions of low and high to search for the critical edges via binary search. Each of $\log n$ probes on the left chain requires $\log n$ probes on the right chain to determine high or low status according to the definition. Thus, for a given C edge, they can zero in on the midpoints in $\log^2 n$ time, yielding an $O(n \log^2 n)$ algorithm overall.

The next section will show how to eliminate one of the binary searches, and succeeding sections will eliminate the other.

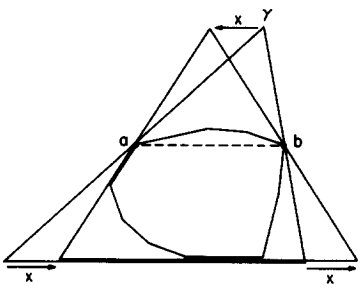

FIG. 2. A second edge of a P-anchored triangle may be made flush by moving γ.

3. TWO FLUSH EDGES AND INTERSPERSING

LEMMA 1. *For any P-anchored triangle T, there always exists another equal area P-anchored T' within the same segment of T (and therefore a representative of the same equivalence class) that has at least two of its edges flush with P.*

Proof. Let C be the flush edge of T guaranteed by the definition of P-anchored, as illustrated in Fig. 2. Assume that neither edge A nor B is flush, and let them contact P at vertices a and b. Now move the vertex γ of T leftwards and parallel to C, while maintaining the contacts at a and b. It is easy to see that the base of the triangle remains the same length, and as its height does not change either, the area remains fixed throughout the movement. Move γ until either side A or B becomes flush with P. This triangle is T'. Q.E.D.

Our algorithm examines "all" P-anchored triangles by examining the segment endpoint representatives guaranteed by this lemma.

The key to our algorithm is the following "interspersing" lemma. If x and y are two points of P, we use the notation (x, y) to indicate the open chain of points from x clockwise to y, and $[x, y]$ the closed chain.

LEMMA 2. *Let $T = ABC$ be a P-anchored triangle flush on side C with a and b the midpoints of sides A and B, and c the clockwise endpoint of the flush edge. Let C' be tangent to P within the chain (c, a). Then if $T' = A'B'C'$ is a P-anchored triangle flush on C', with A' and B' midpoints a' and b', and c' the clockwise endpoint on the flush edge, then $b' \in (b, c')$ and $a' \in (a, b')$.*

Proof. By the naming convention, the midpoints occur in clockwise sequence a', b', c'. Suppose, for the sake of contradiction, that $b' \notin (b, c')$ and/or $a' \notin (a, b')$. Then, we show that $A'B'C'$ cannot be a P-anchored triangle by considering the following three cases:

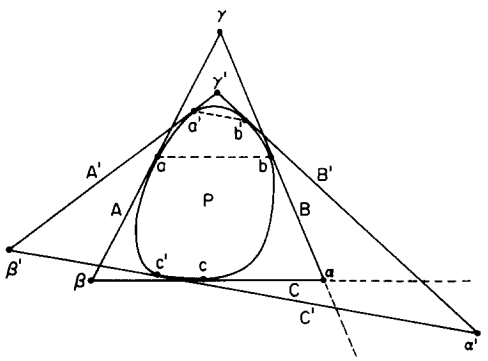

FIG. 3. Case 2.2 of the interspersing lemma.

Case 1. $b' \in [c', a]$. Then, $a' \in (c', b')$ so that a', b', and c' all belong to the left chain determined by C. Thus the angles between C' and A' and between A' and B' are both at least $\pi/2$ so that $A'B'C'$ does not even form a triangle.

Case 2. $b' \in [a, b]$. Then, we establish the lemma by considering two subcases:

Case 2.1. The intersection point of B' and C', α', lies on the same side of B as P. Since $a' \in [c', b']$, the triangle $A'B'C'$ lies entirely to one side of P, contradicting circumscription.

Case 2.2. α' lies on the side of B opposite P (see Fig. 3). Since $A'B'C'$ is a P-anchored triangle, segment $a'b'$ is parallel to C'. Consequently, $a' \in (a, b')$, implying that γ' lies in the interior of the triangle $a\gamma b$. Now, since C' is rotated downwards with respect to C, it is clear that α' lies below C. But then we have that both γ' is below γ and α' is below α, which implies that the midpoint b' of $\gamma'\alpha'$ is below the midpoint b of $\gamma\alpha$. But this contradicts our assumption that $b' \in (a, b)$.

Case 3. $b' \in (b, c')$. Then $a' \notin (a, b')$ (otherwise the lemma is satisfied), so $a' \in [c', a]$. Then we have both c' and a' in $[c, a]$, which is identical to Case 2, with Case 2 c relabeled as b, and b relabeled as a, and so on. Q.E.D.

We can now sketch the operation of a simple $O(n \log n)$ algorithm. First, a single P-anchored triangle is obtained by Klee and Laskowskis' algorithm, and a second edge is made flush as in Lemma 1. Label these triangle edges C and A in clockwise order. The algorithm will advance C to be flush with the next clockwise edge of P, and search for new contact

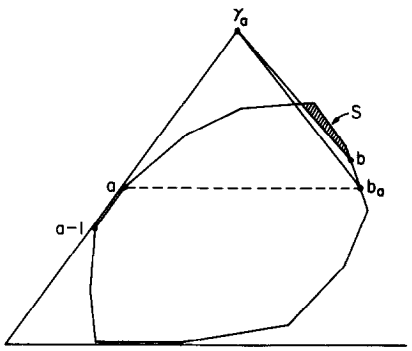

FIG. 4. Edge $[a-1, a]$ is low by Lemma 3.

points for sides A and B. Lemma 2 guarantees that these new contact points need only be searched for in a clockwise direction. Lemma 1 permits us to only consider flush contacts for A. These two immediately yield an $O(n \log n)$ algorithm as follows. After advancing C, determine whether $[a-1, a]$ is low in $O(\log n)$ time using Klee and Laskowski's binary search procedure. If it is low, then advance a. Repeat until the edge behind a is no longer low, at which point it is known to be critical or high. Output the triangle and continue advancing C. This avoids the binary search on the A side, but maintains it on the B side. In the next section we will show how to avoid a binary search on the B side.

4. LOW / HIGH DETERMINATION

We present two lemmas establishing sufficient conditions for concluding that edges are low or high.

LEMMA 3. *If $h(b) > h(a)$ and $\gamma_a b$ cuts P above or is tangent to b, then edge $[a-1, a]$ is low.*

Proof. Let S be the subset of P that lies above $\gamma_a b$; see Fig. 4. Since $h(b) > h(a)$, $h(b) > h(b_a)$. Therefore S also lies above $\gamma_a b_a$, and so this line cuts P above b_a. By the definition of low, edge $[a-1, a]$ is low. Q.E.D.

LEMMA 4. *If $h(b) > h(a)$ and $\gamma_a b$ cuts P below b, then edge $[b-1, b]$ is high.*

Proof. Assume that $b-1$ is on the right chain; otherwise $[b-1, b]$ is high trivially. Consider the line B' determined by $[b-1, b]$, and define γ'_b

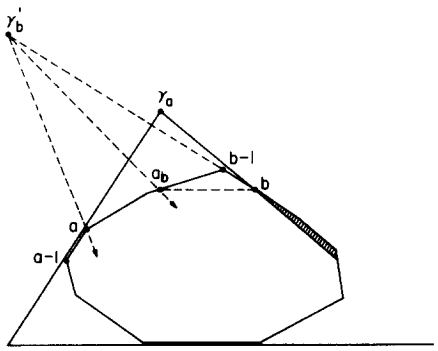

FIG. 5. Edge $[b-1, b]$ is high by Lemma 4.

to be the point on B' with $h(\gamma_b') = 2h(b)$. Then γ_b' must be to the left of the line determined by $[a-1, a]$, since (1) $h(b) > h(a)$, and so $h(\gamma_b') > h(\gamma_a)$, and (2) the slope of B' is lower than that of $\gamma_a b$; see Fig. 5. Now, it is clear that the line $\gamma_b' a$ intersects P below a. This implies, by an argument similar to that used in the preceding lemma, that $\gamma_b' a_b$ intersects P below a_b, where a_b is the point on the left chain with $h(a_b) = h(b)$. But this is precisely the definition of what it means for edge $[b-1, b]$ to be high. Q.E.D.

The import of these lemmas is that it is sometimes possible to determine low/high status information for an edge on the left chain *without* examining vertices at the same height on the right chain, and vice versa, despite the fact that "low" and "high" are defined in terms of such vertices.

5. THE ALGORITHM

Toussaint's "rotating caliper" algorithm for finding minimal enclosing rectangles decides which of the four contact points is the next to advance clockwise; in general one contact moves and three remain fixed between one local minimum and the next [T]. Our algorithm does not attempt to make such a "minimal" movement, but rather advances the C side to be flush with each edge of the polygon in turn with a **for** loop, regardless of whether C is the "next" to touch. The algorithm thus searches for P-anchored triangles, a superset of the local minima.

With the **for** loop, vertex pointers a and b are advanced clockwise by three consecutive **while** loops. The first advances b until it is on the right

chain; the advancement of c by the **for** loop may have redefined the chains so that b is on the left chain. The second **while** loop advances a or b according to circumstances dictated by Lemmas 3 and 4. The third **while** takes over when a critical or high edge has been found for the A side; it advances b until tangency is achieved, and adjusts if side A cannot be flush. Finally, the area of the triangle is computed.

```
{Notation:
        A, B, C are triangle sides.
        a, b, c are indices to polygon vertices on sides A, B, C.
        + 1 advances clockwise; −1 counterclockwise.
        h(p) is the height of point p from the line determined by side C.
        γ_p is the point on side A with h(γ_p) = 2h(p).
}
a ← 2
b ← 3
for c = 1, ..., n do
begin
    {Advance b to right chain.}
    while h(b + 1) ≥ h(b) do b ← b + 1

    {Move a if low, and b if high.}
    while h(b) > h(a) do
        if γ_a b intersects P below b
            then b ← b + 1
            else a ← a + 1
    {[a − 1, a] is now critical or high.}

    {Search for the B tangency.}
    while γ_b b intersects P below b and h(b) ≥ h(a − 1) do b ← b + 1

    if γ_b b intersects P above b or h(b) < h(a − 1)
        then set side B flush with [b − 1, b], and
            if midpoint of B < h(a − 1)
                then set side A to have midpoint a − 1
    else side B is determined by γ_b b
    {All three sides are now determined.}

    Compute area of ABC.

end
```

The algorithm has been implemented and tested. An example showing several iterations is shown in Fig. 6.

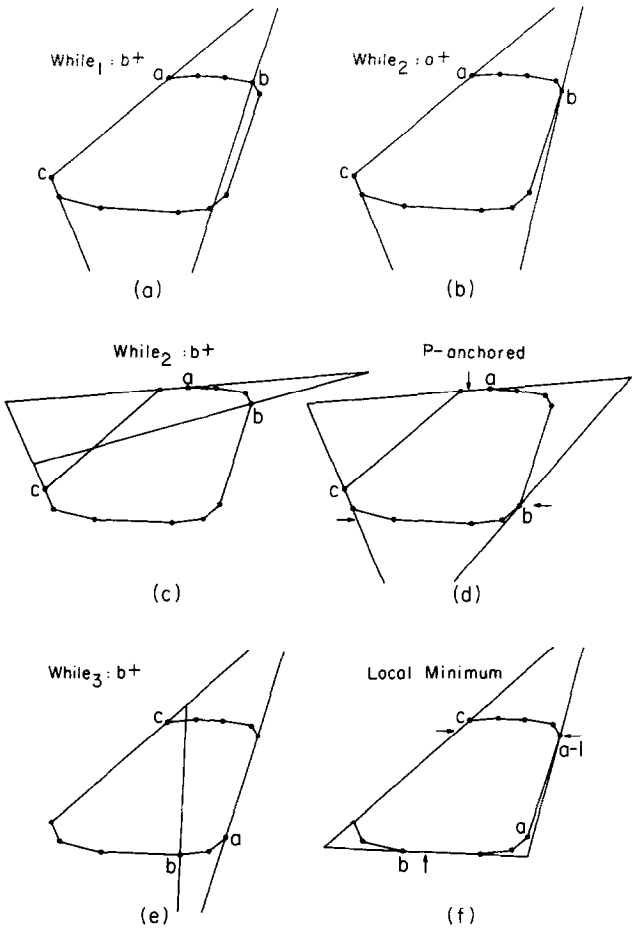

FIG. 6. Snapshots of the execution of the algorithm. Each panel shows which **while** loop is active and the advance decision taken (a + abbreviates $a \leftarrow a + 1$). Panels (a)–(d) represent consecutive iterations of the algorithm. In (d), a P-anchored triangle is found, but it is not a local minimum since the midpoint of side C (indicated by an arrow) falls outside the flush edge $[c-1, c]$. Panels (e) and (f) show two later consecutive iterations. The P-anchored triangle found in (f) is a local minimum because the midpoints touch; this triangle is also the global minimum for this polygon.

6. CORRECTNESS AND OPTIMALITY

THEOREM 4. *The algorithm correctly finds all locally minimal triangles enclosing an* n-gon *in* $\Theta(n)$ *time.*

Proof. Theorem 3 guarantees that a P-anchored triangle exists for each iteration of the **for** loop. If a P-anchored triangle is found in one iteration, Lemma 2 guarantees that we need only search clockwise for the next

P-anchorage when c advances. Certainly this is true at the first iteration, when $c = 1$, $a = 2$, and $b = 3$. Lemma 1 guarantees that we will not miss any P-anchored triangles by restricting the search to those with both the C and A sides of the triangle flush. The first **while** loop is unproblematical, as b clearly must be on the right chain, and moving b to the vertex of maximum height cannot move it past a critical edge. (See Fig. 6a.)

The second **while** loop is the trickiest, but Lemma 3 guarantees that it only advances a when $[a - 1, a]$ is known to be low, and that it only advances b when $[b - 1, b]$ is known to be high. Thus this loop cannot move either a nor b past a critical edge. We now establish that at the end of this second **while** loop, the edge $[a - 1, a]$ is not-low.

Define t as the most clockwise point of P with the property that $\gamma_a t$ supports P at t from above; for a given a, t is the B side tangency. We claim that, if b is advanced by the second **while** loop, the relationship $h(b) \geq h(t)$ will thenceforth be maintained by the loop. Whenever the second loop advances b, t is below b since $\gamma_a b$ intersects P below b (and b is on the right chain), and the advance of b never moves beyond tangency. (See Figs. 6c and d.) Whenever the second loop advances a, t moves only clockwise. (See, e.g., Figs. 6b and c.) This establishes that t is always equal or clockwise of b after b's first advance.

We establish that $[a - 1, a]$ is not low at the end of the second **while** by contradiction, considering two cases.

Assume first that $[a - 1, a]$ is low after the loop finishes, and that the loop *has* advanced b. Because it has finished, $h(b) \leq h(a) = h(b_a)$; because $[a - 1, a]$ is low, $\gamma_a b_a$ cuts P above b_a. This implies that $\gamma_a b$ cuts P above b. But then b is below its tangency point t, contradicting the relationship we showed above: $h(b) \geq h(t)$.

Second, assume that $[a - 1, a]$ is low after the loop, and that b was *not* advanced by the loop. Then either b was moved by the first **while**, or it has not been moved at all during this iteration of the **for** loop. If it was moved by the first **while**, then it is at maximum height from C, and it is clearly impossible for $h(a) \geq h(b)$ (the exit condition for the second **while**) and $[a - 1, a]$ still be low. So it must be that b has never been moved in this iteration of the c loop. In this case Lemma 2 tells us that the ultimate resting place for b is clockwise to its current position. But if $h(a) \geq h(b)$ and $[a - 1, a]$ is low, then the midpoint of A and therefore B must lie above a and therefore above b, a contradiction.

Since $[a - 1, a]$ is not-low, it must be critical or high. Thus it is justified to cease advancement of a. We now consider the operation of the third **while** loop and the remaining code at the end of the **for** loop in both the critical and high cases.

If $[a - 1, a]$ is critical, then since it is not low, $\gamma_a b_a$ does not cut above, and, since it is not high, $\gamma_{a-1} b_{a-1}$ does not cut below. This implies that

there is some b_t with $h(a-1) \leq h(b_t) \leq h(a)$ such that $\gamma_{b_t} b_t$ is tangent at b_t (this is equivalent to Proposition 2.10 in [KL]). Therefore the third **while** will either find b_t if it is a vertex of P, or go one vertex beyond it clockwise if b_t lies on the interior of an edge of P. In the later case, the **if** statement following sets B flush with $[b-1, b]$. In either case, a correct P-anchored representative is found.

Now consider the case when $[a-1, a]$ is high. Here it is not possible for the A edge to be flush; rather the A edge must touch only at $a-1$, which is the vertex separating low from high. Since $[a-1, a]$ is high, $\gamma_{a-1} b_{a-1}$ intersects P below b_a. Thus the first part of the third **while** condition will be true when $h(b) \geq h(a-1)$. The loop is therefore exited with $h(b) < h(a-1)$, and the following **if** sets B flush with $[b-1, b]$, thereby straddling the height of $a-1$, and sets A so that $a-1$ is its midpoint (see Fig. 6f). This is indeed a representative P-anchored triangle in this case.

Finally, the area of the P-anchored triangle ABC is computed. Although the triangle is not necessarily a local minimum (e.g., the midpoint of C may not lie on $[c-1, c]$; see Fig. 6d), no minima will be missed.

It is obvious that the algorithm is linear: it only increments a, b, and c, and therefore completes within $3n = \Theta(n)$ steps. Q.E.D.

THEOREM 5. *$\Omega(n)$ is a lower bound on any algorithm that finds at least one globally minimal area triangle.*

Proof. Suppose there is a sublinear algorithm that finds a global minimum. Consider the operation of this algorithm on a regular convex n-gon P. P has a global minimum corresponding to each edge flush with C. Since it is sublinear, there is at least one vertex x that the algorithm never examines, and which consequently is not chosen as a flush edge in the output. Modify P to P' by moving x along a line orthogonal to $[x-1, x+1]$, to x' within the triangle $(x-1, x, x+1)$. Now the algorithm must output the same answer for P' as it did for P, since it is "unaware" of the modification. We claim, however, that P' has just two global minima, both involving the modified vertex x as the endpoint of a flush edge.

The situation is illustrated in Fig. 7. $\alpha\beta\gamma$ is one of the global minima for P. Rotate side C about $x-1$ to become flush with $[x-1, x']$, forming the triangle $\alpha'\beta'\gamma$. Our claim is that this triangle has smaller area than $\alpha\beta\gamma$. Note that the midpoint c of C falls in the middle of $[x-1, x]$, since $\alpha\beta\gamma$ is a minimum and since P is regular. Thus $[\beta, x-1]$ is longer than $[x-1, \alpha]$. Thus if we add a segment $\beta' y$ parallel to $\alpha\alpha'$ as illustrated, the triangles $(x-1, y, \beta')$ and $(x-1, \alpha, \alpha')$ are congruent. Therefore the rotation of C has reduced the triangle area. A symmetrical argument holds for the flush edge $[x', x+1]$.

Therefore, the output of the assumed algorithm on P' cannot be a global minimum, and so the assumed sublinear algorithm is incorrect. Q.E.D.

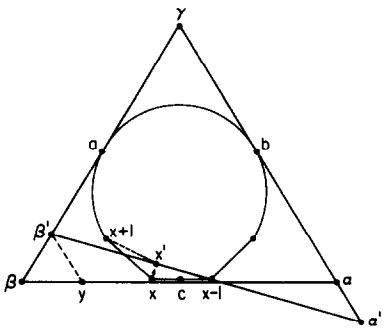

FIG. 7. Modifying x changes the global minimum.

7. DISCUSSION

The presented algorithm is typical of many linear geometric algorithms in being less than transparent. This obscurity makes it difficult to extend the method beyond the particular problem considered. Nevertheless, it is possible that a similar approach may be applicable to the problem of finding minimal convex k-gons circumscribing an n-gon [CY, DA].

ACKNOWLEDGMENTS

This research was partially supported by NSF Grants MCS-83-04780 and DCR-83-51468.

REFERENCES

[CY] J. S. CHANG AND C. K. YAP, A polynomial solution for potato-peeling and other polygon inclusion and enclosure problems, *in* "Proceedings of Foundation of Computer Science," pp. 408–416, IEEE Computer Society, West Palm Beach, Fla., 1984.

[DA] N. A. A. DEPANO AND A. AGGARWAL, Finding restricted k-envelopes for convex polygons, *in* "Proc. 22nd Allerton Conf. on Comm., Contr., and Comp.," pp. 81–90, Univ. of Illinois, Urbana, 1984.

[DB] D. DORI AND M. BEN-BASSAT, Circumscribing a convex polygon by a polygon of fewer sides with minimal area addition, *Comput. Vision Graphics Image Process.* **24** (1983), 131–159.

[K] V. KLEE, Facet centroids and volume minimization, *in* "Fejes Tóth Festschrift," 1986, to appear.

[KL] V. KLEE AND M. C. LASKOWSKI, Finding the smallest triangles containing a given convex polygon, *J. Algorithms* **6** (1985), 359–375.

[O] J. O'ROURKE, Counterexamples to a minimal circumscription algorithm, *Comput. Vision Graphics Image Process.* **30** (1985), 364–366.

[T] G. T. TOUSSAINT, Solving geometric problems with the "rotating calipers," *in* "Proceedings of IEEE MELECON 83," Athens, Greece, May 1983.