1983

A Linear Time Algorithm for the Minimum Area Rectangle Enclosing a Convex Polygon

Dennis S. Arnon

John P. Gieselmann

Report Number:
83-463

A LINEAR TIME ALGORITHM FOR THE MINIMUM AREA RECTANGLE ENCLOSING A CONVEX POLYGON

by

Dennis S. Arnon
Computer Science Department
Purdue University
West Lafayette, Indiana 47907

John P. Gieselmann
Computer Science Department
Purdue University
West Lafayette, Indiana 47907

Technical Report #463
Computer Science Department - Purdue University

December 7, 1983

ABSTRACT

We give an $O(n)$ algorithm for constructing the rectangle of minimum area enclosing an n-vertex convex polygon.

Keywords: computational geometry, geometric approximation, minimum area rectangle, enclosing rectangle, convex polygons, space planning.

1. Introduction. The problem of finding a rectangle of minimal area that encloses a given polygon arises, for example, in algorithms for the cutting stock problem and the template layout problem (see e.g [4] and [1]). In a previous paper, Freeman and Shapira [2] show that the minimum area enclosing rectangle for an arbitrary (simple) polygon P is the minimum area enclosing rectangle R_P of its convex hull. They give an algorithm which first constructs the convex hull of the input polygon P, then constructs R_P in time $O(n^2)$, where n is the number of vertices of the hull.

In this paper we give an $O(n)$ algorithm for constructing the rectangle of minimum area enclosing an n-vertex convex polygon. If our algorithm is combined with an $O(s)$ algorithm for finding the convex hull of an s-vertex simple polygon (e.g [5] or [3]), one has an $O(s)$ algorithm for finding the minimum area enclosing rectangle of an s-vertex simple polygon. We remark that our algorithm resembles, and in fact was inspired by, an algorithm in [6], p. 79, for finding the antipodal pairs of vertices of a convex polygon.

Section 2 has background material, Section 3 gives the algorithm and proves the bound on its computing time, and the Appendix has implementation notes.

2. Background. A *line of support* of a simple polygon P [6] is a line L meeting the boundary of P, such that P lies entirely on one side of L. It is well known (e.g [7], p. 8) that a convex polygon possesses exactly two lines of support having a given slope.

By an *enclosing rectangle* of a polygon we mean a rectangle whose interior contains the interior of the polygon. An *encasing rectangle* is an enclosing rectangle R of minimal area among enclosing rectangles having sides with the same slopes as the sides of R (Figure 1).

Given any pair of parallel lines, and a second pair of parallel lines perpendicular to the first, the four lines have exactly four intersections, which are the vertices of a rectangle. Clearly an encasing rectangle of a convex polygon is the rectangle formed by two

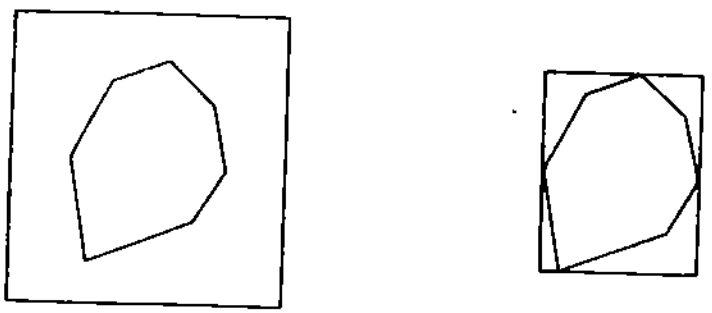

Figure 1 - Enclosing rectangle (left) and encasing rectangle (right)

perpendicular pairs (L_1,L_2), (L_3,L_4) of parallel lines of support of the polygon (Figure 2).

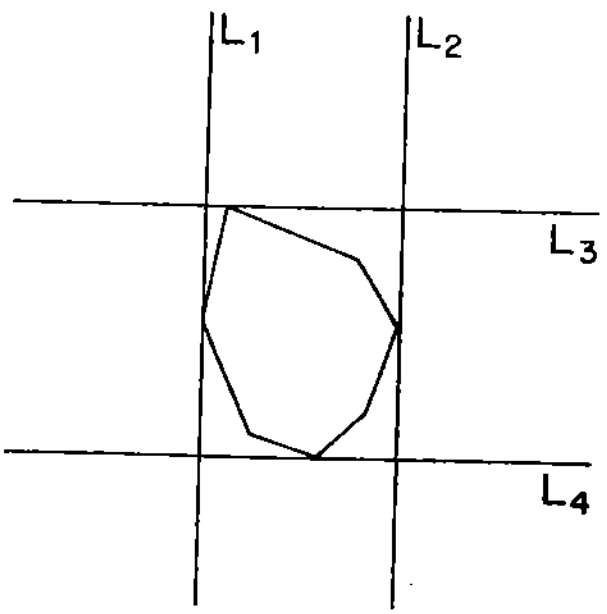

Figure 2 - Lines of support forming encasing rectangle

For each edge of a convex polygon, there is a unique encasing rectangle collinear with that edge, namely the rectangle formed by the line of support L_1 collinear with the edge, the unique line of support L_2 parallel to L, and the unique pair (L_3,L_4) of lines of support perpendicular to L_1 (Figure 3).

Our algorithm relies on:

THEOREM 2.1. *([2]) A rectangle of minimum area enclosing a convex polygon has a side collinear with some edge of the polygon.*

We summarize the algorithm. We will say that a rectangle having a side collinear with an edge of a polygon is "anchored" on that edge. From Theorem 2.1, it follows that we can find the minimum area enclosing rectangle for a convex polygon by finding the minimum of the areas of the encasing rectangles anchored on the edges of the polygon. For each edge, we determine the distance between the two lines of support of the polygon parallel to that edge (one of these lines is collinear with the edge), and the

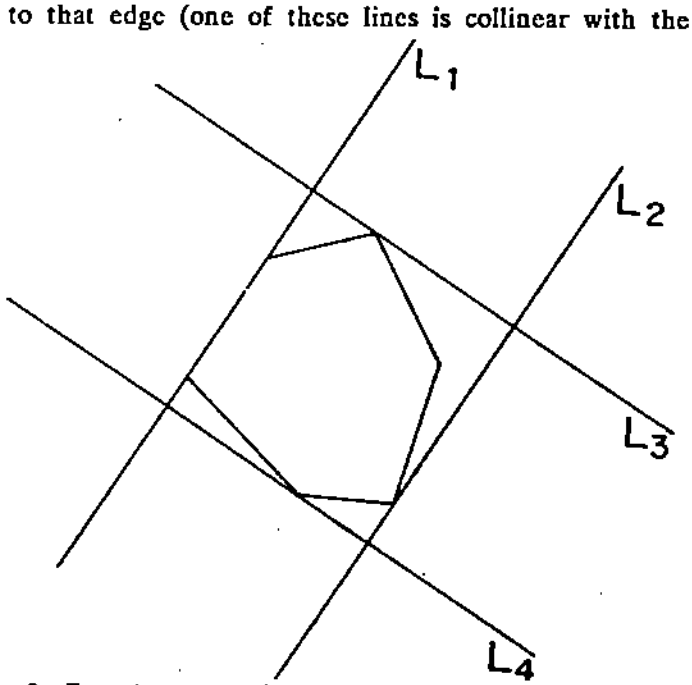

Figure 3 - Encasing rectangle collinear with edge of convex polygon

distance between the two lines of support perpendicular to the first pair. The area of the encasing rectangle anchored on this edge is the product of these distances. If we know one vertex on each line of support, and the slope of the edge, then the distances between the two pairs of lines of support can be computed in constant time. We determine a vertex on each line of support by keeping track of the angle of rotation, from the current edge, that we must go through to reach other edges, and observing that the four lines of support are respectively rotated through angles of 0, $\frac{\pi}{2}$, π, and $\frac{3\pi}{2}$ radians from the current edge.

3. The Algorithm.

ALGORITHM MINIMUM_AREA_RECTANGLE

Input: a convex polygon P.
Output: The area A of a rectangle of minimum area enclosing P, and
an edge of P at which such a rectangle can be anchored.

{ Beginning of comments.

We assume that P is given in *standard form* ([6]): (the x and y coordinates of) its vertices $\{v_i\}$ $(1 \leq i \leq n)$ are listed in counterclockwise order, all vertices are distinct, no three consecutive vertices are collinear, and the vertex with smallest y-coordinate is listed first (if there is a tie, choose the one with least x-coordinate).

indices i, j, k, m below are mod n. α, β, γ, and δ are variables; π is the constant 3.14159...

We use the following (constant-time) functions:

PERPDIST(x_1, y_1, x_2, y_2, s)
{ returns the perpendicular distance from the line through (x_1,y_1)
with slope s to the line through (x_2, y_2) with slope s.}

ANGLE(i)
{returns the counterclockwise angle of rotation (in radians) between the
edge (v_{i-1},v_i) and the edge (v_i,v_{i+1}).

End of comments.}

{Initialization}
$\alpha := 0.0$; $j := 2$; $\beta :=$ ANGLE(2);
$k := 2$; $\gamma := \beta$; $m := 2$; $\delta := \beta$;

{Find area of encasing rectangle anchored on each edge.}
for i := 1 to n do begin

{ Find angle of rotation of next edge of polygon}
if $i > 1$ then $\alpha := \alpha$ + ANGLE(i);

{ Find a vertex on first perpendicular line of support}
while $\beta < \alpha + \frac{\pi}{2}$ do begin
$j := j + 1$; $\beta := \beta$ + ANGLE(j);
end;

{ Find a vertex on parallel line of support.}
while $\gamma < \alpha + \pi$ do begin
$k := k + 1$; $\gamma := \gamma$ + ANGLE(k);
end;

{ Find a vertex on second perpendicular line of support.}
while $\delta < \alpha + \frac{3\pi}{2}$ do begin
$m := m + 1$; $\delta := \delta$ + ANGLE(m);
end;

{Find distances between parallel and perpendicular lines of support}
if $x_{i+1} = x_i$ then begin
$d_1 := |x_k - x_i|$
$d_2 := |y_m - y_j|$
end
else if $y_{i+1} = y_i$ then begin
$d_1 := |y_k - y_i|$
$d_2 := |x_m - x_j|$
end
else begin
$slope := \frac{(y_{i+1} - y_i)}{(x_{i+1} - x_i)}$; d_1 := PERPDIST($x_i, y_i, x_k, y_k, slope$);
d_2 := PERPDIST($x_j, y_j, x_m, y_m, \frac{-1}{slope}$);
end;

{Compute area of encasing rectangle anchored on current edge.}
$A_i := d_1 * d_2$;
if $i = 1$ or $A_i < A$ then begin
$A := A_i$; $edge := i$;
end;

end.

THEOREM 3.1. *Algorithm MINIMUM_AREA_RECTANGLE finds the area of a minimum area enclosing rectangle of an n-vertex convex polygon, and an edge at which such a rectangle can be anchored, in time* $O(n)$.

Proof. As i increases from 1 to n, α increases from 0 to a value less than 2π. It follows that the values of β, γ, and δ never exceed 4π. Hence the value of any of these three variables is altered at most $2n$ times. Hence the total time for all n executions of the loop is $O(2n) = O(n)$. □

Acknowledgement.

We thank Jon Bentley for pointing us to Shamos' work.

References

1. M. Adamowicz and A. Albano, "A two-stage solution of the cutting-stock problem," in *Information processing 71 (Proc. 1971 IFIP Congress)*, North-Holland; Amsterdam(1972).
2. H. Freeman and R. Shapira, "Determining the minimum-area encasing rectangle for an arbitrary closed curve," *Comm. ACM* **18** pp. 409-413 (1975).
3. R.L. Graham and F. Yao, "Finding the convex hull of a simple polygon," *J. Algorithms*, ((to appear)).
4. M. Haims and H. Freeman, "A multistage solution of the template-layout problem," *IEEE Trans. Sys. Sci. Cyb.* **SSC-6** pp. 145-151 (1970).
5. D.T. Lee, "Finding the convex hull of a simple polygon," pp. 305-310 in *Proc. 1982 Johns Hopkins Conf. System Sci.*, (1982).
6. M. Shamos, *Computational geometry*, Yale University(1978). (Ph.D. Dissertation)
7. I.M. Yaglom and V.G. Boltyanskii, *Convex Figures*, Holt, Rinehart, and Winston(1961).

4. Appendix. For implementation of the algorithm, we have developed improvements which appear to decrease actual running time. Since a sine or cosine calculation takes a significantly greater time than an addition or multiplication, we replace the ANGLE function by vector dot and cross products. The DOT and CROSS functions below essentially return arccos and arcsin (respectively) of the angle between their vector arguments. j can be computed by noting that arccos(0) < 0.0 for $\frac{\pi}{2} < 0 < \frac{3\pi}{2}$ radians. k can be computed by using arcsin(0) < 0.0 for $\pi < 0 < 2\pi$ radians, and m can be computed by using arccos(0) > 0.0 for $0 > \frac{3\pi}{2}$ radians.

The revised algorithm follows.

ALGORITHM MINIMUM_AREA_RECTANGLE

Input: a convex polygon P.

Outputs: The area A of a rectangle of minimum area enclosing P, and
an edge of P at which such a rectangle can be anchored.

{ Beginning of comments.

We assume that P is given in *standard form* ([6]): (the x and y coordinates of) its vertices $\{v_i\}$ $(1 \le i \le n)$ are listed in counterclockwise order, all vertices are distinct, no three consecutive vertices are collinear, and the vertex with smallest y-coordinate is listed first (if there is a tie, choose the one with least x-coordinate).

indices i, j, k, m below are mod n.

We use the following (constant-time) functions:

PERPDIST(x_1, y_1, x_2, y_2, s)
{returns the perpendicular distance from the line through (x_1, y_1) with slope s to the line through (x_2, y_2) with slope s.}

DOTPR(*cur*, *nxt*)
{returns the dot product of the vectors from vertices *cur* to *cur* +1 and *nxt* to *nxt* +1.}

CROSSPR(*cur*, *nxt*)
{returns the cross product of the vectors from vertices *cur* to *cur* +1 and *nxt* to *nxt* +1.}

end of comments. }

```
{Initialization}
j := 1;

{Find area of encasing rectangle anchored on each edge.}
for i := 1 to n do begin

        { Find a vertex on first perpendicular line of support }
        while DOTPR(i, j) > 0.0 do begin
            j := j + 1 ;
        end ;

        { Find a vertex on parallel line of support }
        if i = 1 then k := j ;
        while CROSSPR(i, k) > 0.0 do begin
            k := k + 1 ;
        end ;

        { Find a vertex on second perpendicular line of support }
        if i = 1 then m := k ;
        while DOTPR(i, m) < 0.0 do begin
            m := m + 1 ;
        end ;

        {Find distances between parallel and perpendicular lines of support}
        if x_{i+1} = x_i then begin
                d_1 := |x_k - x_i|
                d_2 := |y_m - y_j|
        end
        else if y_{i+1} = y_i then begin
                d_1 := |y_k - y_i|
                d_2 := |x_m - x_j|
        end
        else begin
                slope := (y_{i+1} - y_i) / (x_{i+1} - x_i) ;  d_1 := PERPDIST(x_i, y_i, x_k, y_k, slope);
                d_2 := PERPDIST(x_j, y_j, x_m, y_m, -1 / slope );
        end;

        {Compute area of encasing rectangle anchored on current edge.}
        A_i := d_1 * d_2;
        if i = 1 or A_i < A then begin
                A := A_i; edge := i;
        end;

end.
```