

# Parallel Algorithms for Separation of Two Sets of Points and Recognition of Digital Convex Polygons

Dilip Sarkar<sup>1</sup> and Ivan Stojmenović<sup>2,3</sup>

Received February 1992; Revised November 1992

---

Given two finite sets of points in a plane, the polygon separation problem is to construct a separating convex  $k$ -gon with smallest  $k$ . In this paper, we present a parallel algorithm for the polygon separation problem. The algorithm runs in  $O(\log n)$  time on a CREW PRAM with  $n$  processors, where  $n$  is the number of points in the two given sets. The algorithm is cost-optimal, since  $\Omega(n \log n)$  is a lower-bound for the time needed by any sequential algorithm. We apply this algorithm to the problem of finding a convex polygon, with the minimal number of edges, for which a given convex region is its digital image. The algorithm in this paper constructs one such polygon with possibly two more edges than the minimal one.

---

**KEY WORDS:** Algorithms; computational geometry; PRAMs; parallel algorithms; digital polygons.

## 1. INTRODUCTION

A convex  $k$ -gon is referred to as a  $k$ -separator of two sets of points in a plane if its interior contains one and avoids the other. Given two finite sets  $S_1$  and  $S_2$  of  $n_1$  and  $n_2$  points in a plane, the polygon separation problem is to construct a separating convex  $k$ -gon with smallest (integer)  $k$ . Edelsbrunner and Preparata defined this problem and presented an  $O((n_1 + n_2) \log(n_1 + n_2))$  time sequential algorithm.<sup>(1)</sup> When  $k =$

---

<sup>1</sup> Department of Mathematics and Computer Science, University of Miami, P. O. Box 249085, Coral Gables, Florida 33124.

<sup>2</sup> The research is sponsored by NSERC Operating Grant OGPIN 007.

<sup>3</sup> Computer Science Department, University of Ottawa, Ottawa, Ontario K1N 9B4, Canada.

$\Omega(\log(n_1 + n_2))$ , the algorithm is optimal since the lower-bound for the problem is  $O(n \log n)$ ,<sup>(1)</sup> where  $n = n_1 + n_2$ . They presented another algorithm, with time complexity  $O(kn)$ , which is more efficient than the previous one when  $k = O(\log n)$ .<sup>(1)</sup>

In this paper, we present an optimal parallel algorithm for the polygon separation problem. Our computation model is a Concurrent Read Exclusive Write (CREW) Parallel Random Access Machine (PRAM), where a memory cell can be read by two or more processors simultaneously, but only one processor can write into a memory cell at a time. Our parallel algorithm runs in  $O(\log n)$  time on a CREW PRAM with  $n$  processors. Thus, we show that the polygon separation problem is in the class  $NC$ . The reader is referred to Cook,<sup>(2)</sup> for more details on classification of parallel algorithms on the basis of their time complexities. A parallel algorithm is called cost-optimal if the product of the time complexity of the algorithm and the number of processors used by the algorithm is in the same order as that of the best-known sequential algorithm. A good discussion on cost-optimality can be found in Bar-On and Vishkin.<sup>(3)</sup> By defined measure, our parallel algorithm for the polygon separation problem is cost-optimal.

To design our parallel algorithm, we use the characterization of the polygon separation problem shown by Edelsbrunner and Preparata.<sup>(1)</sup> In the next section, following Edelsbrunner and Preparata,<sup>(1)</sup> we define terms and notations used in the rest of the paper. Section 3 presents characterization of the problem<sup>(1)</sup> and a high-level description of the algorithm. Section 4 presents our parallel algorithm for the polygon separation problem. In Sections 5 and 6, we apply this algorithm to the problem of finding a strict convex separator of two point sets and the problem of recognizing digital polygons with number of edges as little as possible.

## 2. DEFINITIONS

In this section we give some definitions and properties from Edelsbrunner and Preparata.<sup>(1)</sup> Without loss of generality we can assume that the set  $S_1$  is the internal set. Let  $C_1$  be the convex hull of  $S_1$ .  $C_1$  can be constructed in  $O(n_1 \log n_1)$  time (cf. Preparata and Shamos<sup>(4)</sup>). Only the vertices of  $C_1$  are relevant to the construction of the separator. Let  $l$  be any line not intersecting  $C_1$ ; the open half-plane  $h_+(l)$  containing  $C_1$  is called positive and the open half-plane  $h_-(l)$  not containing  $C_1$  is called negative. The left supporting line  $l(p)$  of a point  $p \in S_2$  is defined as the line through  $p$  and tangent to  $C_1$  directed from  $p$  to the contact point on the boundary of  $C_1$  such that  $C_1$  lies to the right of  $l(p)$ . Similarly, the right supporting

line  $r(p)$  of the point  $p$  can be defined. The angle of a directed line is defined with respect to the positive  $x$ -axis.

If we trace from  $p \in S_2$  the left supporting line  $l(p)$  and the right supporting line  $r(p)$  to  $C_1$ , each defines two half-planes. The intersection  $h_-(l(p)) \cap h_-(r(p))$  is called the remote wedge of  $p$ , denoted as  $W(p)$ . The union of all the remote wedges defined by all the points in  $S_2$  defines the region  $F$  of the plane whose interior must have void intersection with any convex separator of  $S_1$  and  $S_2$ . The region can be defined formally as,

$$F = \bigcup_{p \in S_2} W(p)$$

We will refer to  $F$  as the forbidden region of  $S_1$  and  $S_2$ . The complement of  $F$  is a star-shaped polygon called  $C_2$  whose reflex vertices are points of  $S_2$  and each of whose convex vertices is the intersection of two adjacent remote wedges. It is obvious that the kernel of  $C_2$  contains  $C_1$ . An edge of  $C_2$  is either bounded or unbounded. One extreme of every edge of  $C_2$  is a reflex vertex. The other extreme is called a *niche*. It is either a convex vertex of  $C_2$  or is at infinity.

Each edge  $e$  of  $C_2$  is assigned the direction from its niche to its reflex vertex and called an arc. An arc is called clockwise oriented if a ray, rotating clockwise in the plane around a point internal to the kernel of  $C_2$ , scans the reflex vertex point of the arc last. The set of clockwise oriented arcs will be denoted by  $A_-$ . Similarly, the set  $A_+$  of counterclockwise oriented arcs is defined.

Each arc  $e \in A^+$  is extended towards its terminus either up to the furthest intersection with an edge of  $C_2$ , if it exists, or to infinity otherwise. If an intersection exists, it is always with another member of  $A_+$ . An arc constructed in this way is called an *extended arc* and assigned the same direction as that of the arc that defined it. Figure 1 shows all counterclockwise arcs of  $C_2$ .

An arc  $e_j$  is the successor of an arc  $e_i$ , denoted as  $SUCC(e_i) = e_j$ , in either of these mutually exclusive cases: (i) if  $e_i$  has a finite terminus which lies on  $e_j$ ; (ii) if  $e_i$  has no finite terminus, then  $e_j$  has its niche at infinity, and letting  $l_k$  be the line containing  $e_k$ , the region  $h_+(l_i) \cap h_+(l_j)$  does not contain a connected component of  $F$ . For example, in Fig. 1 the successor of  $e_3$  is  $e_6$  (which is Case (i) of the definition) and the successor of  $e_4$  is  $e_1$  (which is Case (ii) of the definition).

### 3. SEQUENTIAL ALGORITHM

In this section we give a high-level description of the sequential algorithm.<sup>(1)</sup> Given two sets  $S_1$  and  $S_2$ , one can determine in  $O(n \log n)$  time

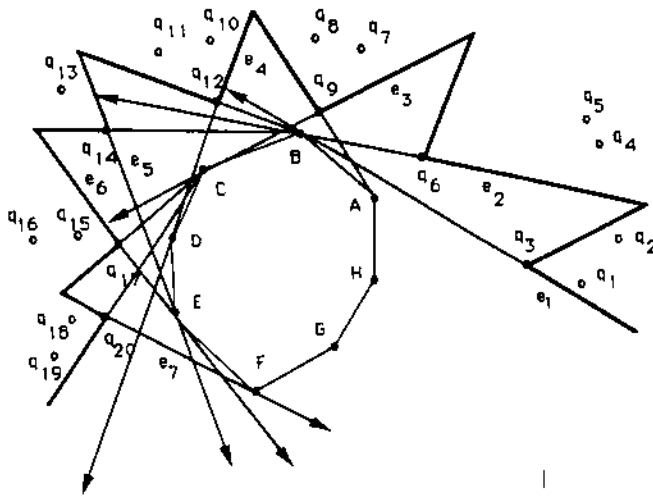


Fig. 1. Two separable sets of points.

whether they are polygon separable or not. If they are polygon separable, we can find the external and internal sets in  $O(n \log n)$  time.

Then the algorithm performs the following three tasks in order.

1. Construct  $F$ , the forbidden region
2. Construct a greedy separator (as defined next)
3. Find an optimal separator

A detailed description of the sequential algorithm is not essential for our algorithm. Thus, we present only a brief description of the three tasks stated previously.

1. Arrange the vertices of  $C_1$  in an array and use it to construct left and right supporting lines of each point  $p \in S_2$  in  $O(\log n)$  time.<sup>(4)</sup> These supporting lines define the forbidden region. Order the points of  $S_2$  in order of increasing angle of left supporting line and store in a list  $L$ . The vertices of  $F$  can be obtained by scanning the points of  $L$  starting with any point. Let us start with the first point  $q_1$  in  $L$  and assume that the points that currently define  $F$  are stored in a sequential list  $F_1$  and let  $p$  be the current point. At a generic step, we consider the remote wedge  $W(p)$  of  $p$  and scan  $F_1$  backwards until a point is found that lies outside the closure of  $W(p)$ , and eliminate all points scanned before. This generic step is performed for each point of  $L$  in turn. In the final step, we perform a generic step for the first point in the list  $F_1$ . The final points in

$F_1$  define  $F$ , the forbidden region. This step can be implemented in  $O(n \log n)$  time.

2. From the points in  $F_1$  construct the sequence of arcs in  $A_+$  and arrange them in a linear list  $A_1$ . Now construct the extended arc for each arc in the list  $A_1$  and find the successor of each arc  $e \in A_1$ . Select an arbitrary  $e \in A_1$  and let  $e_0 = e$ . Construct a sequence  $e_0, e_1, \dots, e_s$  such that  $SUCC(e_i) = e_{i+1}$ , for  $i = 0, 1, \dots, s-1$ , and  $e_{s-1} < e_0 \leq e_s$  in the cyclic order. Then, the polygon whose vertices are the intersection between consecutive extended arcs is a greedy separator of size  $s+1$ .

**Lemma 1.** (Ref. 1) There is (an integer)  $k$  such that each greedy separator has either  $k$  or  $(k+1)$  edges.

3. Now try to find a separator of size  $s$ . If one exists, it can be found in  $O(n_2)$  time. Thus, the sequential algorithm is  $O(n \log n)$  time.

#### 4. PARALLEL ALGORITHM

The parallel algorithm proceeds in the same way as the sequential algorithm. Initial steps, such as, determining the polygon separability, finding the internal set, and finding the convex hull of the internal set take only  $O(\log n)$  time on a CREW PRAM with  $n$  processors using one or more algorithms presented by Atallah and Goodrich,<sup>(5)</sup> Cole,<sup>(6)</sup> Goodrich,<sup>(7)</sup> and Ronse.<sup>(8)</sup> Thus, we will start with the construction of the forbidden region.

Let  $L_i = (q_{i_1}, q_{i_2}, \dots, q_{i_k})$  be a sublist of  $l_i$  points of  $L$  that satisfy all of the following three conditions: (i) if  $k > 1$  then for all  $i_k \neq i_j$ ,  $q_{i_k}$  is not included in the closure of the remote wedge  $W(q_{i_j})$ ; (ii) all points in  $L$  that are between  $q_{i_1}$  and  $q_{i_k}$  are in the closure of the remote wedges of the points in  $L_i$ ; and (iii) all the points in  $L_i$  preserve the ordering of the points in  $L$ . Any sublist  $L_i$  that satisfies these properties will be called *minimal sublist* of  $L$ . A minimal sublist  $L_i$  is said to be to the left of another minimal sublist  $L_j$  if all points in  $L_i$  are to the left of the left-most point in  $L_j$ . Two minimal sublists  $L_i$  and  $L_j$  are called *consecutive* if (i)  $L_i$  is to the left of  $L_j$  and (ii) the closure of the remote wedge of  $q_{j_1}$  includes all points between  $q_{i_1}$  and  $q_{i_k}$ , and possibly some points in  $L_i$  to the left of  $q_{i_1}$ .

**Lemma 2.** If  $L_i = (q_{i_1}, q_{i_2}, \dots, q_{i_k})$  and  $L_j = (q_{j_1}, q_{j_2}, \dots, q_{j_l})$  are two consecutive minimal sublists such that  $q_{i_k}$  is outside the closure of the remote wedge  $W(q_{j_1})$  but  $q_{i_{k+1}}$  is inside, then the combined list  $L_{i+j} = (q_{i_1}, q_{i_2}, \dots, q_{i_k}, q_{j_1}, \dots, q_{j_l})$  is also a minimal sublist.

*Proof.* Follows from the definitions given earlier and the Lemma 4.1 by Edelsbrunner and Preparata.<sup>(1)</sup> Observe that if  $p$  is in  $W(q)$  then  $W(p)$  is a subset of  $W(q)$ .  $\square$

The idea in this lemma will be used to obtain a minimal list  $F_1$  such that the closure of the remote wedges of the points in  $F_1$  include all points of  $L$ . Finally, we eliminate from the right end of the minimal list  $F_1$  those points that are included in the closure of the remote wedge of the first point in the list. The points in this constructed list are the reflex vertices of  $C_2$ . The construction of the minimal list is inherently parallel since, to start with, each point in  $S_2$  is a minimal sublist and we can recursively merge two adjacent minimal sublists to get a possibly bigger one until we have only a single minimal list. Thus, only  $\lceil \log n_2 \rceil$  parallel merging steps are required. If the time for merging two sublists is bounded by a constant, then the construction of a minimal sublist from the list  $L$  takes  $O(\log n_2)$  time. We will use  $n_2$  processors, one for each point in  $L$ , to construct the minimal list.

Suppose, at a merging step,  $L_i$  and  $L_j$  are two consecutive sublists and  $L_i$  is to the left of  $L_j$ ; a processor corresponding to every point in  $L_i$  will check whether its point is included in the closure of the remote wedge of the left-most point of the list  $L_j$ . If the point is inside, then it sets a flag  $INSIDE[i]$  true. We assume that  $i$ th processor is denoted as  $P_i$ , for  $0 \leq i \leq (n_2 - 1)$ . Next all the processors except  $P_0$  test if its point  $q[i]$  is in the closure of the remote wedge of the point in  $q[0]$  of  $P_0$ ; if so the flag  $INSIDE[i]$  is set true. Then the list of reflex vertices of  $C_2$  is constructed from the points in  $q[i]$  with  $INSIDE[i]$  false. We use an  $O(\log n_2)$  time prefix computation algorithm to compute the position of each point in the list.<sup>(3)</sup> Initially, a processor  $P_i$  has a value  $VAL[i] = 1$  if its flag  $INSIDE[i]$  is false, otherwise  $VAL[i] = 0$ . After computing the prefix, a processor receives  $S[i] = \sum_{j=0}^i VAL[j]$ . A processor  $P_i$  writes  $q[i]$  in location  $A_1[S[i]]$  if its flag is false (array  $A_1$  is defined in the previous section). Thus, we have completed the construction of the polygon  $C_2$ .

**Lemma 3.** Given the convex hull  $C_1$  of the internal set  $S_1$  and the points of the external set  $S_2$ , construction of the polygon  $C_2$  takes  $O(\log n)$  time if  $n_2$  processors are used.

We have completed Step 1 of the sequential algorithm and now Step 2 is dealt with. In order to determine  $SUCC(e_i)$  for each  $e_i \in A_1$ , we assign one processor to do a binary search on  $A_1$ . Thus, successors of all the arcs in  $A_1$  are found in  $O(\log n)$  time by  $n_2$  processors.

It is clear that an arc always has a successor. The successor of an arc is itself if  $S_1$  and  $S_2$  have a 1-gon separator. Once the successor of each arc has been found, it takes only constant time to find whether a 1-gon

separator exists. Thus, from now we assume that  $S_1$  and  $S_2$  have a  $k$ -gon separator for  $k > 1$ .

Next, we map our problem to the minimum circle cover problem, defined as follows. Given a set of circular arcs on a circle, a subset of it whose union equals the circle is called a cover. A cover with the smallest number of arcs in it is called a minimum cover. We map each arc  $e_i$  from  $A_1$  to a point on a circle such that the arcs preserve their mutual order (consecutive arcs are mapped onto consecutive points). Each pair  $(e_i, \text{SUCC}(e_i))$  is mapped to a circular arc on the circle. The minimum circle cover can be found in  $O(\log n)$  time with  $O(n)$  processors on a CREW PRAM.<sup>(9)</sup> The minimum cover defines also an optimal polygon separator, by inverse mapping.

**Theorem 1.** Given two point sets  $S_1$  and  $S_2$  of size  $O(n)$ , the preceding procedure finds an optimal polygon separator in  $O(\log n)$  time using  $n$  processors.

**Corollary 1.** The polygon separation problem is in the class  $NC$ . Moreover, the  $NC$  algorithm presented here is cost-optimal.

## 5. STRICT SEPARATION OF TWO POINT-SETS

The parallel algorithm presented in the previous section finds a minimum *nonstrict separator* (NSS) of  $S_1$  and  $S_2$ , because points of both  $S_1$  and  $S_2$  can be on the edges of the separability polygon. In this section we concentrate on *strict separators* for two point sets. A separating polygon is called a *strict separator* (SS) if points of  $S_2$  cannot be on the edges of separability polygon.

Suppose nonstrict separation of two point sets requires  $k$  edges. Let  $m$  be the number of edges required for strict separation. Obviously,  $k \leq m$ . Is  $k = m$ ? The answer is no. In Fig. 2 the points of  $S_2$  are represented by  $\times$  and points of  $S_1$  are represented by  $\bullet$ . It is obvious that in this example  $k = 4$  with the separating rectangle drawn by dotted lines, but  $m = 10$  with the separating octagon drawn by solid lines. In a NSS polygon if no edge contains any points of  $S_2$ , then trivially,  $k = m$ . But what happens if some of the edges of a NSS polygon contain points from  $S_2$ ? Two cases to be considered.

**Case 1.** Some edges of the NSS polygon have points from both  $S_1$  and  $S_2$  such that the points from  $S_1$  are not contiguous, that is, these points are separated by some points in  $S_2$ . In this case, no convex SS polygon exists.

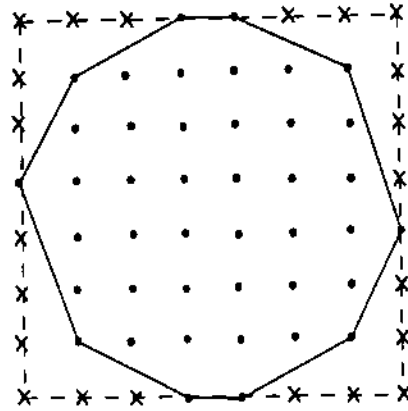


Fig. 2. An example to illustrate  $m > 2k$ .

**Case 2.** Some edges of the NSS polygon have points from  $S_2$ , but all points from  $S_1$  form a contiguous interval on the corresponding edge. In this case, if an edge contains only one point from  $S_1$ , we can replace it by two edges (see the left and right edges in Fig. 2); otherwise an edge contains two or more points from  $S_1$ , and we can replace it with three edges (see the upper and lower edges in Fig. 2). In both cases, the two new edges are tangents from the end points of the interval of points from  $S_1$  on the given edge to the rest of the set  $S_1$  (see Fig. 2). This proves the following theorem.

**Theorem 2.** If  $k$  and  $m$  are the numbers of vertices of the NSS and SS polygons of two point sets  $S_1$  and  $S_2$ , then  $k \leq m \leq 3k$ .

Therefore, the SS polygon obtained by the construction method described in this paper may not have a minimum number of edges. Thus, the problem of finding an optimal parallel algorithm for constructing a SS polygon with a minimum number of edges requires further investigation.

## 6. RECOGNITION OF DIGITAL CONVEX POLYGONS

A digital region is defined by a finite set  $R$  of digital points with integer coordinates (also called pixels). The region  $R$  is a convex digital polygon if there exists a convex polygon  $P$  such that the set of pixels that are inside or on the boundary of  $P$  is exactly  $R$ . We also say that  $R$  is the image of  $P$  and  $P$  is the origin of  $R$ . Obviously, a digital convex polygon may have many different origins. Two digital points  $(a, b)$  and  $(c, d)$  are called neighbors if  $\max(|a - c|, |b - d|) = 1$ . Let  $R''$  be the set of digital points that do not belong to the region  $R$ , and let  $R'$  be the set of digital

points from  $R''$  that have at least one neighbor from  $R$ . It is easy to see that the origin of  $R$  is a strict separator of  $R$  and  $R'$ , and a strict separator of  $R$  from  $R''$ . For example, in Fig. 2  $R$  is the set of all points marked as  $\bullet$  while  $R'$  and  $R''$  contain points marked as  $\times$ . One of the origins of  $R$  is a 10-gon that separates two kinds of points. An origin of  $R$  is called minimal if no origin exists which has fewer edges. One can easily verify that the problem of finding a minimal origin is equivalent to that of finding a strict separator of digitized regions. This problem was addressed by Kim<sup>(10)</sup> and a linear time sequential algorithm that finds a minimum origin of a digitized region was presented. However, the algorithm does not appear to be parallelizable. Here we present a parallel algorithm that is based on our parallel NSS algorithm. This algorithm finds an origin of  $R$  that has at most two more edges than the minimal one.

We will show that the SS problem for digital regions can be transformed to an instance of the NSS problem such that a solution to the NSS problem (with minor modification) gives a fairly good approximate solution of the original SS problem. In fact, it is a very simple transformation. The interior set  $S_1$  for the NSS problem consists of all the boundary points of  $R$  (a point is a boundary point if it has at least one neighbor from  $R'$ ) and the exterior set  $S_2$  is constructed from the digital points from  $R'$  in the following way. If  $(x, y)$  is such a digital point from  $R'$  then points  $(x + \delta, y)$ ,  $(x - \delta, y)$ ,  $(x, y + \delta)$ ,  $(x, y - \delta)$  are also included in the set  $S_2$ , in addition to point  $(x, y)$ . We will show that, for an appropriate choice of  $\delta$  and after a minor modification no point  $(x, y)$  from  $R'$  will belong to an edge of a separating polygon. Thus, the constructed separation is a SS separation for  $R$ . We show how to find an appropriate choice for the number  $\delta$ . First we prove the following lemma. Let  $\Delta x$  and  $\Delta y$  be maximal distances between any two points from  $R$  along  $x$ - and  $y$ -coordinates, respectively.

**Lemma 4.** Let  $e$  be the straight line passing through any two digital points from  $R$ . If a digital point  $T$  does not belong to  $e$  then the distance between  $T$  and  $e$  is greater than or equal to  $1/(\sqrt{\Delta x^2 + \Delta y^2}) = d$ .

*Proof.* Let  $(x_1, y_1)$  and  $(x_2, y_2)$  be any two points from  $R$ . The equation of a line  $e$  passing through them can be written as  $y(x_2 - x_1) - x(y_2 - y_1) - y_1(x_2 - x_1) + x_1(y_2 - y_1) = 0$ . The distance of a point  $T = (x', y')$  to  $e$  is  $|y'(x_2 - x_1) - x'(y_2 - y_1) - y_1(x_2 - x_1) + x_1(y_2 - y_1)| / \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$ . If the distance is not equal to zero then, since all numbers in this expression are integers, it is greater than  $1/\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \geq 1/\sqrt{(\Delta x^2 + \Delta y^2)}$ , as required.  $\square$

**Lemma 5.** Let  $P$  be a convex polygon that is an origin of a given digital region  $R$ . Every edge  $e$  of  $P$  can be replaced by another edge  $e'$  such that the obtained polygon  $P'$  is the origin of  $R$  and the distance from any point of  $R'$  to any edge of  $P'$  is  $\geq d$ .

*Proof.* Edge  $e$  separates some points of  $R$  from their neighbors from  $R'$ . All these points are at a distance  $\leq 1$  from  $e$ . Let  $CH(e)$  and  $CH'(e)$  be convex hulls of the points from  $R$  and  $R'$ , respectively (see Fig. 3). It is clear that  $e$  separates two disjoint polygons  $CH(e)$  and  $CH'(e)$ . Let  $U_1V_1$  and  $U_2V_2$  be critical support lines of  $CH(e)$  and  $CH'(e)$ , where  $V_1$  and  $V_2$  belong to  $CH(e)$ , while  $U_1$  and  $U_2$  belong to  $CH'(e)$ . [Note: a critical support line of two convex polygons  $P$  and  $Q$  is a line that is tangent to both  $P$  and  $Q$  such that  $P$  and  $Q$  lie on opposite sides of the line.] There are two cases:

- (1)  $V_1 \neq V_2$ . In this case, it is easy to show that the line  $e'$  extending any edge between  $V_1$  and  $V_2$  (for example,  $V_2V_1$ , where  $V_1$  is immediate neighbor of  $V_2$  lying between  $V_2$  and  $V_1$ ) also separates  $CH(e)$  and  $CH'(e)$ . Therefore,  $e'$  can replace  $e$  in  $P$  and passes through two points from  $R$ . By Lemma 4, any point from  $R'$  is at distance  $\geq d$ .
- (2)  $V_1 = V_2$  (that is, critical support lines intersect at a vertex of  $CH(e)$ ). Let  $e'$  be the line passing through  $V_1$  and parallel to  $U_1U_2$  (see Fig. 4). From Lemma 4, the distance from  $V_1$  to  $U_1U_2$  is  $\geq d$ . Hence, the distance from  $V_1$  or  $U_2$ , or any other pixel from  $CH'(e)$  to  $e'$  is  $\geq d$ , as required by the lemma.  $\square$

It follows from Lemma 5 that the SS problem for  $S_1$  and  $R'$  and the SS problem for  $S_1$  and  $S_2$  give polygon separators with the same minimal

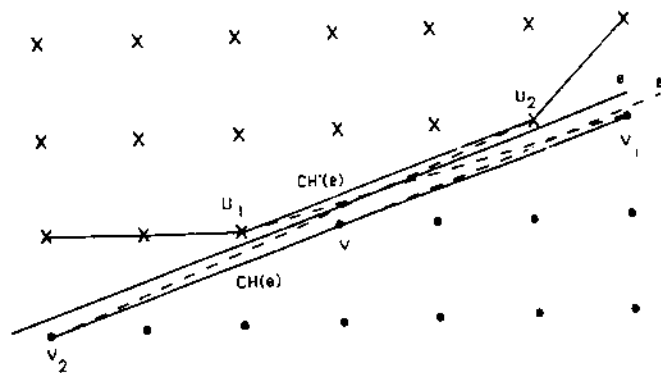


Fig. 3. Replacing  $e$  by  $e'$ .

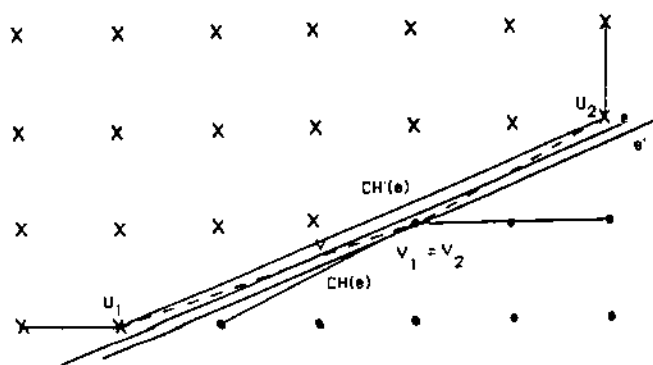


Fig. 4. Replace  $e$  by  $e'$ .

number of edges, if  $\delta$  is chosen such that  $\delta < d$ . Thus, it suffices to construct an SS polygon for  $S_1$  and  $S_2$ . Let  $k$  be the minimal number of edges of a NSS of  $S_1$  and  $S_2$  constructed by applying the algorithm of Section 4. Obviously,  $m \geq k$ , where  $m$  is the minimal number of edges of a SS of  $S_1$  and  $S_2$ . We only need to show that a so constructed NSS polygon  $P$  for  $S_1$  and  $S_2$  can be modified to give a SS polygon of  $R$  and  $R'$ .

Suppose that there exists a point  $T$  from  $R'$  which appears on the boundary of the NSS polygon of  $S_1$  and  $S_2$ . Then the angle  $\beta$  of the separating polygon containing  $T$  must be acute (see Fig. 5). But, any convex polygon may have at most 3 acute angles. Let  $O$  be the point where two lines meet to make the angle  $\beta$  (Fig. 5). Clearly,  $O$  is the vertex of  $P$  with either minimal or maximal  $x$ -coordinate; similarly for  $y$ -coordinate.

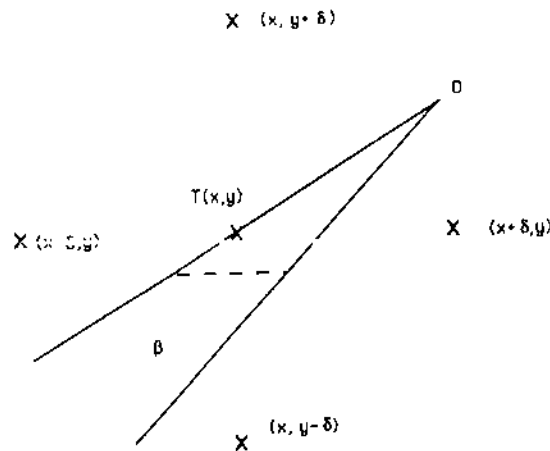


Fig. 5. Acute angle  $\beta$ .

Thus,  $P$  may have at most two such acute angles  $\beta$ . Each of two possible digital points  $T$  can be taken out of  $P$  by cutting  $P$  with a line (say, horizontally close to  $T$ , as indicated in Fig. 5 with dashed line).

In our construction we separate  $R$  from  $R'$  by separating from a larger set  $S_2$ . However, this is not necessarily the separation of  $R$  from  $R''$  which is required by the definition of digital polygons. We give an illustration in Fig. 6. The point  $S$  belongs to  $R''$  but not to either  $R$  or  $R'$ , and may be situated inside the separating polygon of  $S_1$  and  $S_2$ , which is undesirable. However, it is easy to show that the angle  $\beta$  of the separating polygon that contains the point  $S$  is acute; moreover it is less than  $\pi/4$ . ( $S$  is at distance at least 1 from two neighboring points from  $R'$  between which the angle must be contained.) It means that there are at most two such acute angles in the separating polygon. And each of them can be "cut" by a (horizontal or vertical) segment passing through the point from  $R$  that is nearest to  $S$  (see point  $V$  and corresponding dashed segment in Fig. 6). The new polygon separates  $R$  from  $R''$ . However, it may have two more edges than the minimal separating polygon.

Clearly, the same angles  $\beta$  may occur in above two cases, and the same kind of cut can be applied to resolve both problems. Therefore no more than two new edges are introduced. This completes our solution and confirms the following result.

**Theorem 3.** Let  $m$  be the number of edges in a minimal origin of a given digital region  $R$ . Then in  $O(\log n)$  time on a CREW PRAM with  $O(n)$  processors (where  $n$  is the number of boundary points from  $R$ ) one can construct an origin of  $R$  that has at most  $(m + 2)$  edges.

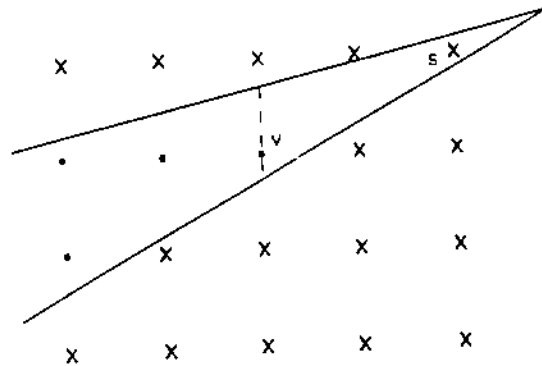


Fig. 6. Adding a new edge.

## 7. OPEN PROBLEMS

Several related problems need further investigation. The problem of constructing an origin of a digital region having minimal number of edges remains open, since our method may construct an origin that has two more edges than the one with minimal number of edges. Also, the strict separation problem needs further investigation, since our algorithm may produce a separator with three times more edges in the worst case.

Investigation is also necessary for both problems when separating polygons are nonconvex, for example, when the polygon is simple. The later problem is open for sequential algorithms as is the problem of finding combinatorial bounds on the number of edges.

## ACKNOWLEDGMENTS

The authors are grateful to four referees for their careful reading and remarks that have greatly improved the readability of the paper. Comments received by Anil Maheshwari are also appreciated.

## REFERENCES

1. H. Edelsbrunner and F. P. Preparata, Minimum Polygon Separation, *Information and Computation*, 77:218-232 (1988).
2. S. A. Cook, A Taxonomy of Problems with Fast Parallel Algorithms, *Information and Control*, 64:2-23 (1985).
3. I. Bar-On and U. Vishkin, Optimal Parallel Generation of Computation Tree Form, *ACM Trans. Prog. Lang. and Syst.*, 7:348-357 (1985).
4. F. P. Preparata and M. I. Shamos, *Computational Geometry*, Springer-Verlag, New York (1985).
5. M. J. Atallah and M. T. Goodrich, Efficient Parallel Solutions to Some Geometric Problems, *J. Parallel and Distributed Computing*, 3:492-507 (1986).
6. R. Cole, Parallel Merge Sort, *Proc. IEEE Symp. Found. Comput. Science*, pp. 511-516 (1986).
7. M. T. Goodrich, Finding the Convex Hull of a Sorted Point Set in Parallel, *Info. Proc. Lett.*, 26:173-179 (1987).
8. C. Ronse, A Bibliography on Digital and Computational Convexity (1961-1988), *IEEE Trans. PAMI*, 11:181-190 (1989).
9. D. Sarkar and I. Stojmenovic, An Optimal Parallel Circle-Cover Algorithm, *Infor. Proc. Lett.* 32:3-6 (1989).
10. C. E. Kim, Digital Convexity, Straightness, and Convex Polygons, *IEEE Trans. PAMI*,