

COMPUTING EXTERNAL WATCHMAN ROUTES ON PRAM, BSR, AND INTERCONNECTION NETWORK MODELS OF PARALLEL COMPUTATION

LAXMI P. GEWALI¹

*Department of Computer Science
University of Nevada, Las Vegas, NV 89154, USA*

and

IVAN STOJMENOVIC²

*Computer Science Department
University of Ottawa, Ottawa, CANADA*

Received 8 November 1993

Revised 31 January 1994

Accepted by A. Liestman

ABSTRACT

An external watchman route in the presence of a polygonal obstacle is a closed path such that each point in the exterior of the polygon is visible to some point along the route. We adapt the merging slopes technique of parallel computational geometry to develop a parallel algorithm for computing a shortest external watchman route in the presence of a convex polygon of n sides. The algorithm runs in $O(\log n)$ time using $O(\frac{n}{\log n})$ processors in the CREW-PRAM computational model; this is optimal within a constant factor. The algorithm can be easily adapted to compute a shortest watchman route in $O(\log n)$ time on a hypercube with $O(n)$ processors. We also discuss the computation of a shortest external watchman route on star and pancake networks. Finally, a constant time algorithm for solving the merging slopes problem on a BSR with n processors is described. This leads to algorithms with the same time and processor count for solving the external watchman route problem, for computing Minkowski sum, critical support lines, and separability of two convex polygons, for finding the maximum distance between two convex polygons, and for computing the smallest enclosing box, diameter, and width of a convex polygon.

Keywords: Computational geometry, visibility, watchman problem.

1. Introduction

Computing collision-free paths having visibility properties is an important problem in computational geometry having applications in robotics, computer graphics, and CAD/CAM [1,2]. The watchman route problem, first introduced in [3], asks

¹This research was supported in part by grants from CRAY Research Inc and Information Science Research Institute, UNLV

²Research supported in part by NSERC (grant OGPIN 007)

for a route (i.e. closed path) such that each point in a given space is visible to some point along the route. Note that two points in the exterior of a polygon are **visible** if the straight line segment connecting them does not intersect the interior of the polygon; similarly, two points inside a polygon are visible if the straight line segment connecting them does not intersect the exterior of the polygon. Algorithms dealing with watchman route problems can be found in [3-7]. An $O(n)$ time algorithm for computing the shortest watchman route inside a triangulated simple rectilinear polygon is given in [3] and an $O(n^4)$ time algorithm for simple triangulated polygon is reported in [4]. A linear time algorithm for computing the shortest watchman route in the exterior of a convex polygon can be found in [6].

Developing parallel algorithms for solving convex hull, shortest paths, and visibility problems has attracted the interest of many researchers in recent years [8-15]. In this paper we develop a parallel algorithm for computing the shortest watchman route in the exterior of a convex polygon of n sides. The algorithm runs in $O(\log n)$ time using $O(n)$ operations in the CREW-PRAM computational model; this number of operations is optimal within a constant factor. (Note that the time complexity of a sequential algorithm solving problem X is **optimal** if it can be proved that no sequential algorithm can solve X faster, asymptotically; and a parallel algorithm solving X is optimal if the total number of operations it uses is asymptotically the same as the time complexity of an optimal sequential algorithm solving X .) We show how our approach can be used directly to compute shortest watchman routes on BSR, hypercube, star, and pancake networks. A constant time algorithm for solving the merging slopes problem on a BSR with n processors is described. This leads to algorithms with the same time and processor count for solving the external watchman route problem, for computing Minkowski sum, critical support lines, and separability of two convex polygons, for finding the maximum distance between two convex polygons, and for computing the smallest enclosing box, diameter, and width of a convex polygon. We show that the shortest external watchman route can be computed in $O(\log n)$ time on a hypercube using $O(n)$ processors. For star and pancake networks we prove that the shortest external watchman route for a convex polygon with $m!$ vertices can be computed in $O(m^3 \log m)$ time.

2. Algorithm on PRAM

Consider a convex polygon P whose vertices in counterclockwise order are p_0, p_1, \dots, p_{n-1} . We denote the edge connecting p_i to p_{i+1} by e_i (the additions on indices are done modulo n). A route such that each point in the exterior of P is visible from some point along the route is an **external watchman route** for P . We are interested in developing a parallel algorithm for computing a shortest external watchman route for P . We use the term **wrapping routes** to indicate the external watchman routes that completely encircle the polygon. All other external watchman routes are called **2-leg routes**. Figure 1 shows examples of both types of routes.

Lemma 1. [6] A shortest external watchman route for a convex polygon P either follows the boundary of P or it is a 2-leg route with respect to some adjacent edges of P .

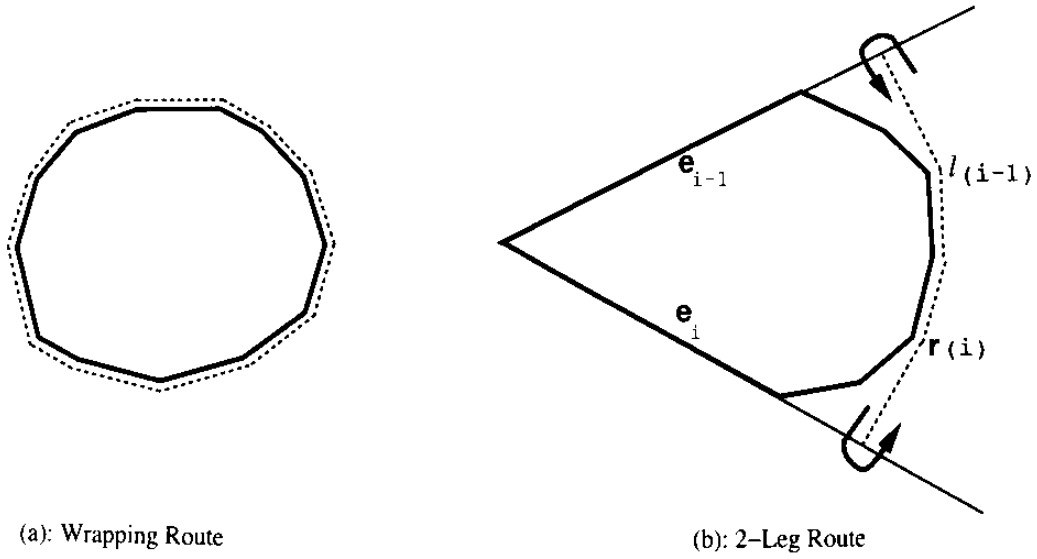


Fig. 1. Two kinds of watchman routes.

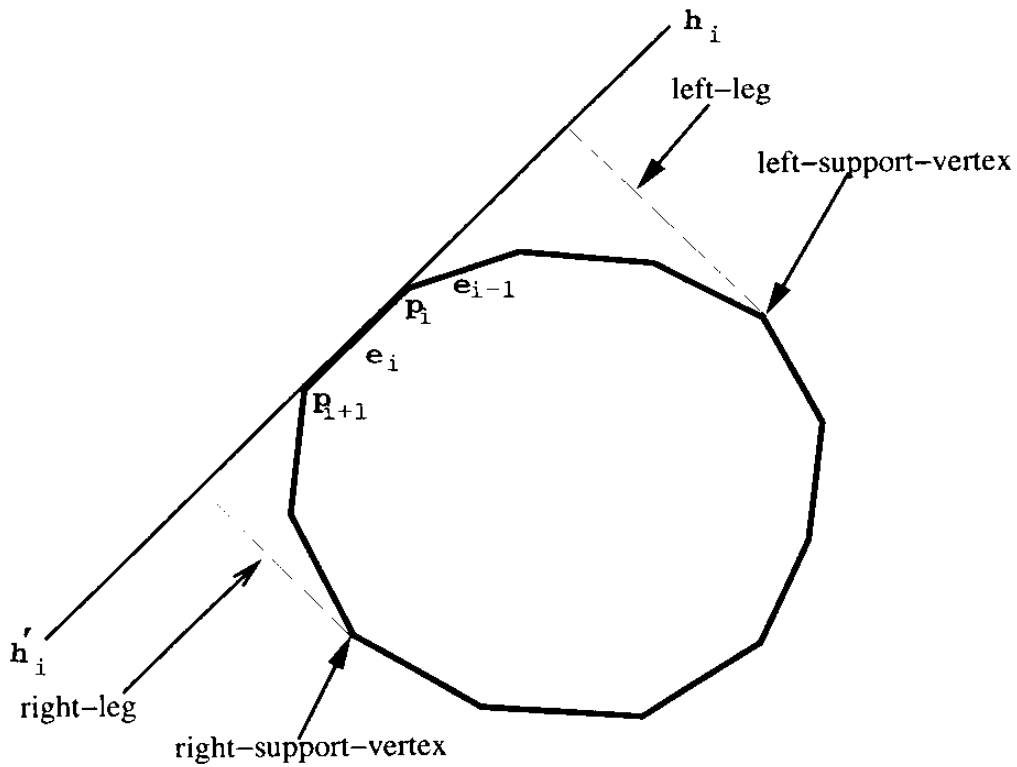


Fig. 2. Illustrating legs and support vertices.

It is straightforward to see that the shortest wrapping route is given by the boundary of the polygon. Computing the shortest 2-leg route efficiently is more involved. Let h_i (respectively, h'_i) denote the half line obtained by extending the edge $e_i = (p_i, p_{i+1})$ from p_{i+1} towards p_i (respectively, from p_i towards p_{i+1}). The shortest line segment connecting P with h_i which is perpendicular to h_i and tangent to P is called the **left leg** of e_i . The vertex of P that belongs to the left leg of e_i , denoted by $l(i)$, is called the **left-support-vertex** of e_i . Similarly, the **right leg** and **right-support-vertex** of e_i are defined by considering the shortest line segment tangent to P and perpendicular to h'_i (Figure 2). We denote the right support vertex of e_i by $r(i)$. The **2-leg route with respect to adjacent edges** e_{i-1} and e_i , denoted by $w(i)$, is the route that follows the left leg of e_{i-1} , the boundary of P from $l(i-1)$ to $r(i)$ in the clockwise direction, and the right leg of e_i as shown in Figure 1b.

The sequential algorithm given in [6] makes use of Lemma 1 and computes the shortest external watchman route as follows. It computes 2-leg routes with respect to each pair of adjacent edges by walking around the polygon and selects the one with the minimum length. It then compares the shortest 2-leg route with the boundary route and outputs the one with the shorter length. To compute a shortest 2-leg route efficiently in parallel, we need to use the **merging slopes technique** introduced in [14,15, 21] (see also [12]), which we describe next.

We start with a few definitions. The distance of a point p to an oriented edge e is the distance from p to the line containing e . If p lies to the left of e , then the distance is defined to be positive. Similarly, the distance is negative if p lies to the right of e . The α -distance of p to e is its distance to the edge e' obtained by rotating e by angle α in the counterclockwise direction around a point (the results of search queries discussed below do not depend on the choice of the rotation point; hence any point can be taken as rotation point). Given two convex polygons P and Q , and an angle α , the **extremal search problem** denoted by $ES(P, Q, \alpha)$ asks to find for each edge $e_j \in P$, a vertex $q_l \in Q$ with the smallest α -distance to e_j among vertices from Q . The vertex q_l is called the **associated vertex** of e_j in direction α . Extreme vertices of polygon Q with respect to an edge of P can be captured in term of α -distance. For example, the solution obtained by invoking $ES(P, Q, \frac{\pi}{2})$ gives the easternmost vertices of Q for each edge of P . Similarly, the westernmost, the southernmost, and the northernmost vertices of Q for each edge of P are obtained by invoking $ES(P, Q, \alpha)$, with α equal to $\frac{3\pi}{2}$, 0 , and π , respectively. In Figure 3, the associated vertices of e_j for α equal to 0 , π , $\frac{\pi}{2}$, and $\frac{3\pi}{2}$ are q_l (southernmost), q_j (northernmost), q_i (easternmost), and q_k (westernmost), respectively.

The use of the merging slope technique to identify the associated vertices of the edges of P can be briefly described as follows. (Details and other applications of merging slopes can be found in [12,14,15,21].)

Observation 1. The slopes of the of edges of a convex polygon are sorted in the order of the occurrence of these edges along the boundary. (Strictly speaking, the list is the concatenation of two sorted list if the first element is not a maximum or

minimum; but it can be converted to a sorted list by shifting in a straightforward way. Hence we can assume that the list is sorted.) Consider rotating the edges $e_i (0 \leq i \leq n - 1)$ of P by angle α in the counterclockwise direction (the slopes of the rotated edges are expressed in the interval $[0, 2\pi)$). Let s_i denote the slope of edge e_i of P rotated by angle α , and let s'_i denote the slope of the i -th edge (q_i, q_{i+1}) of Q . Let q_r be the associated vertex (in direction α) of edge $e_i \in P$. The crucial observation here is that the slope s'_i is the minimum slope greater than the slope s_i , among the slopes of the edges of Q . In other words, the slope s_i lies between the slopes of consecutive edges of Q . (In Figure 3, slope of e_j plus α ($\alpha = 0$) lies between the slopes of edges (q_{l-1}, q_l) and (q_l, q_{l+1}) ; the slope of e_j plus α ($\alpha = \frac{\pi}{2}$) lies between the slopes of edges (q_{i-1}, q_i) and (q_i, q_{i+1}) .) Hence by merging the slopes $s_i (0 \leq i \leq n - 1)$ and the slopes $s'_i (0 \leq i \leq m - 1)$, the associated vertices of each edges of P can be identified. This merging can be done in $O(\log n)$ time using $O(\frac{n}{\log n})$ processors in the CREW PRAM model of computation [16,17].

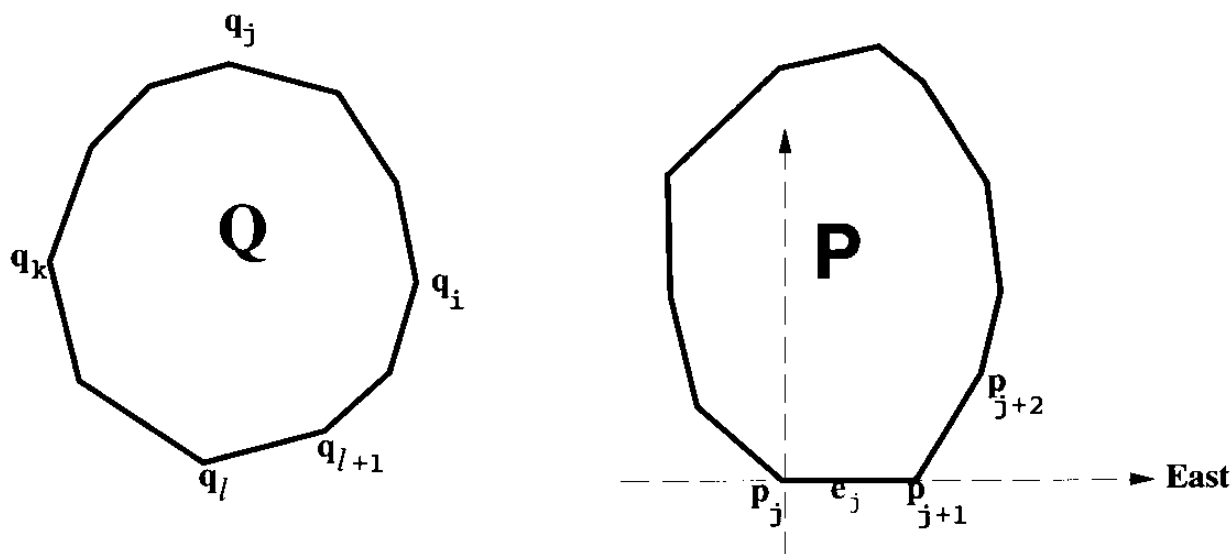


Fig. 3. Illustrating extreme vertices of Q for an edge of P .

Lemma 2. The right-support-vertices and the left-support-vertices of all edges of P can be computed in $O(\log n)$ time using $O(\frac{n}{\log n})$ processors in the CREW PRAM model of computation .

Proof. We apply the merging slopes technique to identify left-support and right-support vertices of all edges of P as follows. Let A denote the ordered list of the slopes of the edges of P . Let A_α denote the list obtained by adding α (modulo 2π) to the elements of A . In the merged list $A \cup A_\alpha$, an edge e_i of P appears between the edges of Q incident on the associated vertex of e_i . Hence the right support vertices can be identified by merging A and $A_{\frac{\pi}{2}}$ (invoking $ES(P, P, \frac{\pi}{2})$). Similarly, left-support-vertices can be identified by merging A and $A_{\frac{3\pi}{2}}$ (invoking $ES(P, P, \frac{3\pi}{2})$). Since parallel merging can be done in $O(\log n)$ time using $O(\frac{n}{\log n})$

processors in the CREW PRAM model, all left-support and right-support vertices can be computed within the stated time and number of processors. \square

After determining right-support-vertices and left-support-vertices of all edges the lengths of all 2-leg routes are computed in parallel by using parallel-prefix techniques and the shortest 2-leg route is determined. The length of the shortest external watchman route is the smallest of the shortest 2-leg route and the convex hull route. A formal description of the algorithm is given below.

Algorithm Shortest Watchman Route

Step 1: Compute $S_i = |e_0| + |e_1| + |e_2| + \dots + |e_i|$ for $i = 0, 1, \dots, n - 1$. The length of the perimeter of P is $C = S_{n-1}$.

Step 2: Compute left-support-vertices $l(i)$ and right-support-vertices $r(i)$ for all edges using the merging slopes technique and compute the lengths $L(i)$ and $R(i)$ of the left and the right legs of all edges.

Step 3: { Computation of shortest 2-leg route }

- a. Let $B(i)$ denote the length of the 2-leg route with respect to adjacent edges e_{i-1} and e_i . Let $l(i) = p_k$ and $r(i-1) = p_m$;
- b. Compute $B(i)$'s as follows:
 If $k < m$ then $B(i) = 2 * (L(i) + R(i-1) + S(m) - S(k))$
 Else $B(i) = 2 * (L(i) + R(i-1) + S(m) - S(k) + C)$
- c. Find the shortest 2-leg route D by examining $B(i)$'s.

Step 4: Report the smallest of C and D as the shortest watchman route.

Theorem 1. The shortest external watchman route for a convex polygon can be computed in $O(\log n)$ time using $O(\frac{n}{\log n})$ processors in the CREW-PRAM computational model.

Proof. Step 1 can be done in $O(\log n)$ time by using $O(\frac{n}{\log n})$ processors using the parallel prefix technique [8]. Step 2 can be done in $O(\log n)$ time with $O(\frac{n}{\log n})$ processor by using the parallel merging slopes technique (Lemma 2). Within the same parallel time and the same number of processors Step 3 can be done by using parallel minima finding methods [8,13]. Step 4 takes $O(1)$ time, and hence the total time and the total number of processors add up to $O(\log n)$ and $O(\frac{n}{\log n})$, respectively. \square

3. Algorithm on the BSR and Other Models

Recently Akl et al [18,19] introduced a new model of parallel computation, called the BSR (broadcasting with selective reduction) and showed that it is more powerful than any CRCW PRAM and yet requires no more resources for implementation than even EREW PRAM. The model allows constant time solutions to sorting,

parallel prefix and several other problems. Traditional PRAM (parallel random access machine) model assumes that during the execution of an algorithm several processors may read from, or write to, the same memory location simultaneously, such that each processor gains access to at most one location. An additional type of memory access is permitted in BSR, namely a broadcast instruction, by means of which all processors may gain access to all memory locations at the same time for the purpose of writing. At each memory location, a subset of the incoming broadcast data is selected and reduced to one value (using an appropriate reduction operator, taken from the set $\{\Sigma, \Pi, \wedge, \vee, \oplus, \cap, \cup\}$ denoting sum, product, AND, OR, XOR, max, and min, respectively); this value is finally stored in the memory location.

The selection process is carried out as follows. Along with each datum d_i a tag t_i is also broadcast, while each memory location x_j is associated with a limit l_j . A selection rule σ (taken from the following set $\{<, \leq, =, \geq, >, \neq\}$) is used to test the condition $t_i \sigma l_j$: if the latter is true then d_i is selected for reduction, otherwise, d_i is rejected. This is done simultaneously for all broadcast data $d_i, 1 \leq i \leq n$, and all memory locations $x_j, 1 \leq j \leq m$.

The BSR broadcast instruction is denoted by:

$$x_j := \mathcal{R}d_i | t_i \sigma l_j.$$

The above notation can be interpreted as follows. If $t_i \sigma l_j$ is satisfied then d_i is "accepted" by location x_j . The set of all data accepted by x_j is reduced to a single value by means of the binary associative operation \mathcal{R} , and stored in x_j . If no data are accepted by a given memory location x_j then x_j receives the value of the neutral element for operation \mathcal{R} . If only one datum is accepted, then x_j is assigned the value of that datum. The operation is performed for all $j, 1 \leq j \leq n$, in parallel.

For example, the prefix sum problem is to compute $p_j := x_1 + x_2 + \dots + x_j$ for $j = 1, 2, \dots, n$. It can be done by the following broadcast operation [18]:

$$p_j := \Sigma x_i | i \leq j$$

Another example is concurrent read. Processor j reads datum d_{p_j} from processor p_j as follows [20].

$$x_j := \Sigma d_i | i = p_j$$

We now describe an algorithm to solve the extremal search problem on BSR. As noted earlier in Observation 1, the associated vertex of e_j of P is the vertex q_k of Q such that the slope s'_k is the minimum possible slope which is greater than s_j . If no such slope is found then the result is the vertex q_0 . The algorithm can thus be coded elegantly as follows.

Algorithm Extremal_Search_On_BSR

for each processor $j, 1 \leq j \leq m$ do in parallel

$$s_j := \text{slope}(e_j) + \alpha \bmod 2\pi$$

$$w_j := \cup s'_i | s'_i \geq s_j$$

$$w_j := \cap q_i | s'_i = w_j$$

if all slopes are rejected then $w_j := q_0$

After the execution of the algorithm, the associated vertex of edge e_j will be

one of w_j, w_{j+1} ; a constant time additional check will select the proper associated vertex. Clearly, the broadcasting operation can be repeated to obtain, one by one, all necessary data about the associated vertex (the index, x and y coordinates, etc.). The algorithm runs in constant time on a BSR with n processors.

We have thus shown that all procedures needed to solve the shortest external watchman route problem (parallel prefix, concurrent read, and extremal search) can be computed in constant time on BSR and hence we have the following theorem.

Theorem 2. The shortest external watchman route for a convex polygon can be computed in constant time on BSR.

Our approach can be used to develop efficient parallel algorithms for computing the shortest external watchman routes on other models of parallel computers as well. In recent years, there has been a growth of research interest in developing parallel algorithms for solving computational geometry problems on hypercube, star, and pancake networks [12,14]. By using the data communication techniques and merging slopes techniques reported in [14], our algorithm can be modified in a straightforward way to compute the shortest external watchman route for a convex polygon on hypercube, star, and pancake networks. This is stated in the following theorem.

Theorem 3. The shortest external watchman route for a convex polygon can be computed in $O(\log n)$ time on a hypercube with $O(n)$ processors. Similarly, the shortest external watchman route for a convex polygon of $m!$ vertices can be computed in $O(m^3 \log m)$ time on a star or pancake network with $m!$ processors.

Proof. All techniques needed to solve the problem on given interconnection networks have the running times as indicated in the theorem. Step 1 requires parallel prefix, which is done in [21] for hypercube (generalized ranking algorithm from [22]) and in [14] for star and pancake models. Step 2 requires the merging slopes technique, solved in [21] (hypercube) and [14] (star and pancake). Step 3 is a concurrent read (to learn $S(m)$ and $S(k)$), followed by straightforward computations inside processors. Note that all processors which are supposed to read the same data $S(m)$ or $S(k)$ form a continuous interval (it follows as a consequence of applying the merging slopes technique). This kind of concurrent read is implemented as follows. The first processor in any such interval is recognized (it reads from a different processor than its neighbor with a smaller index), then reads $S(m)$ or $S(k)$ using exclusive read. This exclusive read can be implemented by two calls to distribution operation, in which the source address increases whenever the destination address increases. The distribution operation is described in [22] for hypercubes and in [14] for stars and pancakes. After the first processor in every interval learned $S(m)$ or $S(k)$, interval broadcasting (spreading data from each leader to its followers) is applied. For star and pancake networks, distribution and interval broadcasting are described in [14] while for hypercubes it is given in [22]. Each of these operations run in $O(n)$ time on a n -dimensional hypercube, and in $O(m^3)$ time on a star or a

pancake network with $m!$ processors. \square

4. Discussion

The merging slopes technique can be applied to solve some other problems, as discussed in [15] for CREW PRAM and mesh-connected computers, in [21] for hypercubes, and in [14] for the star and pancake models. Without giving details or definitions, we note here that the BSR solution to the merging slopes technique leads to constant time solutions with a number of processors equal to the input size for the following problems: for computing Minkowski (or vector) sum, critical support lines, and separability of two convex polygons, for finding the maximum distance between two convex polygons, and for computing the smallest enclosing box, diameter, and width of a convex polygon. Akl and Guenther [18] presents a constant time convex hull algorithm for n points in plane which uses n^2 processors on a BSR. The same result applies to some other applications of the merging slopes technique, namely diameter, width and smallest enclosing box of a set of points. It is an outstanding open problem to solve the convex hull problem with fewer processors.

We established that the shortest watchman route in the exterior of a convex polygon can be computed in $O(\log n)$ time using $O(n)$ operations, this number of operations is optimal within a constant factor. We also presented efficient algorithm to compute a shorest watchman route on other model of parallel computer, namely, the hypercube, the star, the pancake, and the BSR. The next step in this direction would be to develop an optimal parallel algorithm to compute the shortest watchman route inside a simple orthogonal polygon. A variation of the approach used in [11], together with the unfolding techniques used in [3], might be helpful in this direction. Also, it would be interesting to develop parallel algorithms for computing the shortest watchman route inside a simple polygon (not necessarily orthogonal). Since the best sequential algorithm to solve this problem known to date takes $O(n^4)$ time [4], it may be necessary to develop a faster sequential version before attempting to obtain an efficient parallel algorithm.

Acknowledgement. We thank referees for many helpful suggestions.

References

- [1] J. O'Rourke, *Art Gallery Theorems and Algorithms* (Oxford University Press, 1987).
- [2] G. T. Toussaint, ed., *Computational Morphology* (North Holland, 1988).
- [3] W. P. Chin and S. Ntafos, Optimum watchman routes, *Information Processing Letters* **28** (1988) 39–44.
- [4] W. P. Chin and S. Ntafos, Watchman routes in simple polygons, *Discrete and Computational Geometry* **6**, 1 (1991) 9–31.

- [5] S. Carlsson, B. J. Nilsson, and S. Ntafos, Optimum guard covers and m-watchman routes for restricted polygons, *Lecture Notes in Computer Science* (1991) 519.
- [6] S. Ntafos and L. P. Gewali, External watchman routes, to appear in *The Visual Computer*.
- [7] J. Czyzowicz, P. Egyed, H. Everett, D. Rappaport, T. Shermer, G. Toussaint, and J. Urrutia, The Aquarium Keeper's Problem, *The Second SIAM Symposium on Data Structures and Algorithms*, 1991.
- [8] S. G. Akl, *The Design and Analysis of Parallel Algorithms* (Prentice Hall, 1989).
- [9] A. Aggarwal, B. Chazelle, L. Guibas, C. O'Dunlaing, and C. Yap, Parallel computational geometry, *Algorithmica* 3 (1988) 293–327.
- [10] M. J. Atallah and D. Z. Chen, An optimal parallel algorithm for the visibility of a simple polygon from a point, *Proceedings of the Fifth ACM Symposium on Computational Geometry*, 1989.
- [11] H. ElGindy and M. Goodrich, Parallel algorithms for shortest path problems in polygons, *The Visual Computer* 3 (1988) 371–378.
- [12] S. G. Akl and K. A. Lyons, *Parallel Computational Geometry* (Prentice Hall, 1993).
- [13] J. JaJa, *Introduction to Parallel Algorithms* (Addison Wesley, 1992).
- [14] S. Akl, K. Qiu, and I. Stojmenovic, Data communication and computational geometry on the star and the pancake interconnection networks, *Proceedings of the 3rd IEEE Symposium on Parallel and Distributed Processing*, 1991.
- [15] I. Stojmenovic, *Parallel computational geometry*, Washington State Univ., Pullman, CS-87-176, 1987.
- [16] Y. Shiloach and U. Vishkin, Finding the maximum, merging and sorting in a parallel computation model, *Journal of Algorithms* 2 (1981) 88–102.
- [17] R. H. Barlow, D. J. Evans, and J. Shanehchi, A parallel merging algorithm, *Information Processing Letters* 13, 3 (1981) 103–106.
- [18] S. G. Akl and G. R. Guenther, Broadcasting with selective reduction, *Proceedings of 11th IFIP Congress*, San Francisco, Aug. 1989, 515–520.
- [19] S. G. Akl, L. Fava Lindon, and G. R. Guenther, Broadcasting with selective reduction on an optimal PRAM circuit, *Technique et Science Informatiques* 4 (1991) 261–268.

- [20] L. Fava Lindon and S. G. Akl, An optimal implementation of broadcasting with selective reduction, *IEEE Transactions on Parallel and Distributed Systems* **4**, 3 (1993) 256–269.
- [21] I. Stojmenovic, Computational geometry on a hypercube, *Proceedings of the International Conference on Parallel Processing*, Vol. III, 1988, 100–103.
- [22] D. Nassimi and S. Sahni, Data broadcasting in SIMD computers, *IEEE Transactions on Computers* **C-30**, 2 (1981) 101–106.