Convexity Problems on Meshes with Multiple Broadcasting

D. Bhagavathi, S. Olariu[†‡], *J. L. Schwing*[†], *W. Shen, L. Wilson,* and *J. Zhang*[†]

Department of Computer Science
Old Dominion University
Norfolk, VA 23529-0162
U.S.A.

Abstract The purpose of this work is to present simple time-optimal algorithms for a number of convexity-related problems on meshes with multiple broadcasting, that is, mesh-connected computers enhanced by the addition of row and column buses. More specifically, we show that with an n-vertex convex polygon P as input, the tasks of computing the diameter, the smallest-area enclosing rectangle, and the largest-area inscribed triangle sharing an edge with P, can be accomplished in $\Theta(\log n)$ on a mesh with multiple broadcasting of size $n \times n$.

Index Terms: convex polygons, meshes with multiple broadcasting, diameter, enclosing rectangle, inscribed triangle, time-optimal algorithms, image processing, computer vision.

1. Introduction

The notion of convexity is fundamental in image processing, computer graphics, pattern recognition, and computational geometry. In image processing, for example, convexity is a simple and important shape descriptor for objects in the image space [2,20,22]. In pattern recognition, convexity appears in clustering, and computing similarities between sets [7]. In computational geometry, convexity is often a valuable tool in devising efficient algorithms for a number of seemingly unrelated problems [19].

In this work we devise simple time-optimal algorithms for performing a number of tasks involving planar convex polygons. Given such a convex polygon P, we address the problems of computing the diameter of P, of computing a smallest-area enclosing rectangle, and of computing the largest-area inscribed triangle sharing an edge with P. These tasks are motivated by, and find applications to, problems in image processing, computer vision, and VLSI design. Specifically, the diameter of a convex polygon is of import in clustering [2,7,20,22], computer graphics [16], path planning [12,21], and in a number of facility location problems [19]. The smallest area enclosing rectangle arises in image processing [20] as well as in the compaction process in VLSI [18]. As it turns out, our algorithms are time-optimal in the model of computation that we describe next.

An $M \times N$ mesh-connected computer consists of MN identical processors positioned on a rectangular array. The processor located in row i ($1 \le i \le M$) and column j ($1 \le j \le N$) is referred to as $P(i,j)$. Every processor $P(i,j)$ is connected to its four neighbors $P(i-1,j)$, $P(i+1,j)$, $P(i,j-1)$ and $P(i,j+1)$, provided they exist. The mesh-connected computer has emerged as one of the more natural choices for solving a large number of computational tasks in image processing, computational geometry, and computer vision [4,10,13]. This is due, in part, to its simple interconnection topology and to the fact that many problems feature data that maps easily onto the mesh structure. In addition, meshes are particularly well suited for VLSI implementation [1,18].

However, due to their large communication diameter, meshes tend to be slow when handling data transfer operations over long distances. To overcome this problem, mesh-connected computers have recently been augmented by the addition of various types of bus systems [9,14,17]. One of the more promising such augmented systems, referred to as *mesh with multiple broadcasting* [9,17], involves augmenting the basic mesh-connected computer by the addition of row and column buses (see Figure 1). In

† This work was supported by NASA under grant NCC1-99
‡ Additional support by the National Science Foundation under grant CCR-8909996 is gratefully acknowledged

a mesh with multiple broadcasting, a processor can communicate with any of its four neighbors using local communication links, or can broadcast information along any of its row and column buses. As usual, only one processor is allowed to broadcast on a bus at one time. This architecture has proven to be feasible to implement in VLSI and is used in the DAP family of parallel machines [1].

Throughout this paper we assume a SIMD model: in each time unit, the same instruction is broadcast to all processors, which execute it and wait for the next instruction. Each instruction can consist of performing an arithmetic or boolean operation, communicating with one of its neighbors using a local link, broadcasting a value on a bus, or receiving a value from a specified bus. In $M \times N$ mesh with multiple broadcasting, each processor is assumed to have a constant number of registers of size $O(\log MN)$; furthermore, every operation involves handling at most $O(\log MN)$ bits of information. In accordance with other authors, we assume that communications along buses take $O(1)$ time, independent of the length of the bus [9,14,17].

Recently, efficient algorithms to solve a number of computational problems on meshes with multiple broadcasting have been proposed in the literature [3,9,11,15,17]. In particular, Olariu *et al.* [15] report time-optimal convex hull algorithms for both meshes with multiple broadcasting and reconfigurable meshes, while Kumar and Reisis [9] solve geometric problems on images.

The remainder of this paper is organized as follows: Section 2 discusses the time lower bounds for these three problems on meshes with multiple broadcasting; Section 3 proposes time-optimal algorithms to solve these three problems; finally, Section 4 summarizes the results and discusses some open problems.

2. Lower Bounds

Specifying an n-vertex polygon P in the plane amounts to enumerating its vertices as $p_1, p_2, \ldots, p_n$ ($n \geq 3$), here $p_i p_{i+1}$ ($1 \leq i \leq n-1$) and $p_n p_1$ define the edges of P. We note that the vertex representation of a polygon can be easily converted into an *edge* representation with P represented by a sequence $e_1, e_2, \ldots, e_n$ of edges, with e_i ($1 \leq i \leq n-1$) having p_i and p_{i+1} as its endpoints, and e_n having p_n and p_1 as its endpoints. Throughout this paper we assume that an N-vertex polygon is stored in an N-element array.

A polygon P is termed *simple* if no two of its non-consecutive edges intersect. Jordan's Curve Theorem guarantees that a simple polygon partitions the plane into two disjoint regions, the *interior* (bounded) and the *exterior* (unbounded), that are separated by the polygon. A simple polygon is *convex* if its interior is a convex set [19].

In this section, we derive lower bounds for the following problems.

- DIAMETER: given an n-vertex convex polygon, compute its diameter.
- ENCLOSING RECTANGLE: given an n-vertex convex polygon, determine an enclosing rectangle of minimum area.
- INSCRIBED TRIANGLE: given an n-vertex convex polygon, determine an inscribed triangle of maximum area sharing an edge with the given polygon.

Our optimality arguments will be stated first in the Parallel Random Access Machine model (PRAM, for short). This approach is motivated by a recent result of Lin *et al.* [11] that allows us to extend many lower bound results from the PRAM to meshes with multiple broadcasting. A PRAM consists of autonomous processors, each having access to a common memory. At each step, every processor performs the same instruction, with a number of processors masked out. In a Concurrent Read Exclusive Write PRAM (CREW-PRAM) model, a memory location can be simultaneously accessed by more than one processor in reading, but not in writing. The interested reader is referred to [23] for a thorough discussion on the PRAM family.

Additionally, we shall rely on a fundamental result of Cook *et al.* [5], asserting that the time lower bound for the OR problem on the CREW-PRAM is $\Omega(\log n)$ regardless of the number of processors used. For the sake of completeness, we define the problem and state the relevant result from [5].

- OR: given n bits, compute their logical OR.

Proposition 2.1. [5] OR has a time lower bound of $\Omega(\log n)$ on CREW-PRAM, independent of the number of processors and memory cells used.

We now show the time lower bounds for DIAMETER, ENCLOSING RECTANGLE, and INSCRIBED TRIANGLE problems to be $\Omega(\log n)$ on the CREW-PRAM, by reducing the OR problem to each of these problems. In all the derivations we use polar coordinates for convenience; as pointed

out in [19] this is not necessary.

Lemma 2.2. DIAMETER has a time lower bound of $\Omega(\log n)$ on the CREW-PRAM, independent of the number of processors and memory cells used.

Proof. We shall reduce OR to DIAMETER. For this purpose, assume the input to OR is n bits $b_1, b_2, \ldots, b_n$. Let ε be a positive real number satisfying $\cos\frac{\pi}{n} < \frac{1}{1+\varepsilon}$. With every bit b_i associate points p_i and p_{n+i} defined as follows: $p_i = (1+b_i\varepsilon, \frac{i\pi}{n})$ and $p_{n+i} = (1+b_i\varepsilon, \frac{(n+i)\pi}{n})$.

All points corresponding to 0-bits lie on the unit circle, all the others lie on the circle of radius $1+\varepsilon$. Note that ε has been chosen in such a way that the polygon P determined by the points $p_1, \ldots, p_{2n}$ is always convex. Further, note that the diameter of P is exactly 2 if and only if the OR of the input bits is 0. Since the construction of P takes O(1) time using n processors on the CREW-PRAM, the conclusion follows from Proposition 2.1. □

Lemma 2.3. ENCLOSING RECTANGLE has a time lower bound of $\Omega(\log n)$ on the CREW-PRAM, independent of the number of processors and memory cells used.

Proof. We shall reduce OR to ENCLOSING RECTANGLE. For this purpose, assume the input to OR is n bits $b_1, b_2, \ldots, b_n$. Let ε be a positive real number satisfying $\cos\frac{\pi}{2n} < \frac{1}{1+\varepsilon}$. With every bit b_i associate four points p_{2i-1}, p_{2i}, $p_{2n+2i-1}$ and p_{2n+2i} defined as follows: $p_{2i-1} = (1+\varepsilon-b_i\varepsilon, \frac{(2i-1)\pi}{2n})$, $p_{2i} = (1+\varepsilon-b_i\varepsilon, \frac{2i\pi}{2n})$, $p_{2n+2i-1} = (1+\varepsilon-b_i\varepsilon, \frac{(2n+2i-1)\pi}{2n})$, and $p_{2n+2i} = (1+\varepsilon-b_i\varepsilon, \frac{(2n+2i)\pi}{2n})$.

All points corresponding to 1-bits lie on the unit circle, all the others lie on the circle of radius $1+\varepsilon$. Note that ε has been chosen in such a way that the polygon P determined by the points $p_1, \ldots, p_{4n}$ is always convex. Further, note that the smallest enclosing rectangle has area $4\cos^2\frac{\pi}{4n}$ if and only if the OR of the input bits is 0. Since the construction of P takes O(1) time using n processors on the CREW-PRAM, the conclusion follows from Proposition 2.1. □

Lemma 2.4. INSCRIBED TRIANGLE has a time lower bound of $\Omega(\log n)$ on the CREW-PRAM, independent of the number of processors and memory cells used.

Proof. We shall reduce OR to INSCRIBED TRIANGLE. For this purpose, assume the input to OR is n bits $b_1, b_2, \ldots, b_n$. Let ε be a positive real number satisfying $\cos\frac{\pi}{2n} < \frac{1}{1+\varepsilon}$. With every bit b_i associate four points p_{2i-1}, p_{2i}, $p_{2n+2i-1}$ and p_{2n+2i} defined as follows: $p_{2i-1} = (1+b_i\varepsilon, \frac{(2i-1)\pi}{2n})$, $p_{2i} = (1+b_i\varepsilon, \frac{2i\pi}{2n})$, $p_{2n+2i-1} = (1+b_i\varepsilon, \frac{(2n+2i-1)\pi}{2n})$, and $p_{2n+2i} = (1+b_i\varepsilon, \frac{(2n+2i)\pi}{2n})$.

All points corresponding to 0-bits lie on the unit circle, all the others lie on the circle of radius $1+\varepsilon$. Note that ε has been chosen in such a way that the polygon P determined by the points $p_1, \ldots, p_{4n}$ is always convex. Further, note that the largest inscribed triangle has area $\sin\frac{\pi}{2n}$ if and only if the OR of the input bits is 0. Since the construction of P takes O(1) time using n processors on the CREW-PRAM, the conclusion follows from Proposition 2.1. □

To derive time lower bounds for DIAMETER, ENCLOSING RECTANGLE, and INSCRIBED TRIANGLE on meshes with multiple broadcasting, we use the following result recently developed by Lin *et al.* [11].

Proposition 2.5. [11] With $\alpha(n)$ preprocessing time, any subsequent unit-time computational step on an n-processor mesh with multiple broadcasting can be performed in one time unit on an n-processor CREW-PRAM with $O(n)$ extra memory, where $\alpha(n)$ stands for the inverse Ackermann function. □

Let $T_M(n)$ be the execution time of an algorithm for solving a given problem on an n-processor mesh with multiple broadcasting, then there exists a CREW-PRAM algorithm to solve the same problem in $T_P(n) = \alpha(n)+T_M(n)$ time using n processors and $O(n)$ extra memory by Proposition 2.5. Combining this observation with Lemmas 2.2, 2.3, and 2.4, we have the following result.

Theorem 2.6. DIAMETER, ENCLOSING RECTANGLE, and INSCRIBED TRIANGLE have a time lower bound of $\Omega(\log n)$ on meshes with multiple broadcasting of size $n \times n$. □

3. The Algorithms

The purpose of this section is to show that the time lower bounds derived in Theorem 2.6 are tight. We propose algorithms for DIAMETER, ENCLOSING RECTANGLE, and INSCRIBED TRIANGLE problems running in O(log n) time on meshes with multiple broadcasting of size $n \times n$. The input to each of these algorithms is a convex polygon $P=p_1, p_2, \ldots, p_n$, with p_j stored by processor $P(1,j)$ $(1 \leq j \leq n)$.

The following result is frequently used in our algorithms.

Proposition 3.1. [9] The prefix sums (also maxima or minima) of a sequence of n real numbers can be computed in O(log n) time on a mesh with multiple broadcasting of size $n \times n$. Furthermore, this is time-optimal. □

Our solution to the DIAMETER problem relies on the the notion of *antipodal pairs* [19]. Vertices p_i and p_j of a convex polygon P are an antipodal pair if P has parallel supporting lines through p_i and p_j.

Proposition 3.2. [19] The diameter of a convex polygon is the greatest distance between antipodal pairs. □

Our DIAMETER algorithm begins by mandating every processor $P(1,j)$ $(1 \leq j \leq n)$ broadcast the coordinates of the point it stores on the corresponding vertical bus. Next, every processor $P(i,i)$, $(1 \leq i \leq n)$, broadcasts the equation of its incident edges through row i. Every processor $P(i,j)$ with $i<j$ can now detect whether the point p_i and p_j are antipodal. If they are, $P(i,j)$ marks itself. As a result, the marked processor stores an antipodal pair.

It is easy to confirm (see also [19] page 180) that in every row of the mesh marked processors form an interval. Now detecting the leftmost marked processor in each row can be done in O(1) time. At the same time, the rank of each marked processor in the corresponding row, which we call row rank, can be decided in O(1) time. Similarly, the rank of each marked processor in the corresponding column, called column rank, can be decided. It is easy to see that either the row rank or column rank of a marked processor must be smaller than or equal to three. Now marked processors can be divided into two groups. The first group consists of the marked processors with row rank smaller than column rank. The second group consists of the others, that is, the marked processors with column rank smaller than or equal to row rank. The first group is further divided into three subgroups by their row ranks, and the second group is divided into three subgroups by their column ranks. Next, we send the antipodal pairs subgroup by subgroup to the first row, and identify the antipodal pair with the largest distance in each group. By Proposition 3.1, it takes O(log n) time. Finally, the diameter can be decided from these at most six antipodal pairs, and the whole process takes O(log n) time.

To summarize our findings, we state the following result.

Theorem 3.3. The DIAMETER problem can be solved in O(log n) time on a mesh with multiple broadcasting of size $n \times n$. Furthermore, this is time-optimal. □

We also note that a recent result in [15] computes the convex hull of a set of n points in the plane in O(log n) time on a mesh with multiple broadcasting of size $n \times n$. Therefore, we have the following result.

Corollary 3.4. The diameter of a set of n points in the plane can be computed in O(log n) time on a mesh with multiple broadcasting of size $n \times n$. Furthermore, this is time-optimal. □

Now we show how to solve the ENCLOSING RECTANGLE problem. Our solution to the ENCLOSING RECTANGLE problem relies on the following technical result.

Proposition 3.5. [8] The minimum area rectangle enclosing a convex polygon has one side collinear with one of the edges of the polygon. □

Our algorithm begins by having every processor $P(1,j)$ $(1 \leq j \leq n)$ broadcast the coordinates of the point it stores on the corresponding vertical bus. Next, every processor $P(i,i)$, $(1 \leq i \leq n)$, broadcasts the equation of the line determined by p_i and p_{i+1} to all the processors in row i. In each row, at most two adjacent processors detect that the points they hold are farthest away from the line $p_i p_{i+1}$. We retain the leftmost such processor in each row. Similarly lines, in each row, we can identify two points which admit two different supporting line perpendicular to $p_i p_{i+1}$. It is easy to confirm that in O(1) time processor $P(i,i)$ can compute the area of the enclosing rectangle having one edge collinear with $p_i p_{i+1}$.

Finally, what remains to be done is to compute the minimum of all the areas stored by processors $P(i,i)$ $(1 \leq i \leq n)$. This can be done as follows. First, every processor $P(i,i)$ $(1 \leq i \leq n)$ broadcasts the value of the area it stores vertically to processor $P(1,i)$. Once this information is available in the first row of the mesh, by Proposition 3.1, the minimum of these values can be computed in O(log n) time.

To summarize our findings we state the following result.

Theorem 3.6. The ENCLOSING RECTANGLE problem can be solved in O(log n) time on a mesh with multiple broadcasting of size $n \times n$. Furthermore, this is time-optimal. □

Theorem 3.6 together with the convex hull algorithm in [15] imply the following result.

Corollary 3.7. The enclosing rectangle of a set of n points in the plane can be computed in O(log n) time on a mesh with multiple broadcasting of size $n \times n$. Furthermore, this is time-optimal. □

Now we handle the INSCRIBED TRIANGLE problem. Our solution to the INSCRIBED TRIANGLE problem relies on the following technical result.

Proposition 3.8. [6] If $p_i p_k$ is a chord of a convex polygon, the function which calculates the area of triangle $p_i p_j p_k$ is unimodal as p_j assumes values of vertices of the polygon between p_i and p_k. □

We begin by having every processor $P(1,j)$ $(1 \le j \le n)$ broadcast the coordinates of the point it stores on the corresponding vertical bus. Next, every processor $P(i,i)$, $(1 \le i \le n)$, broadcasts the equation of the line $p_i p_{i+1}$ to all the processors in row i. At most two adjacent processors detect that the points they hold are farthest away from the line $p_i p_{i+1}$. We retain the leftmost such processor in each row and mandate it to send the coordinates of the point it holds to $P(i,i)$ along the bus in row i. It is easy to see that in O(1) time processor $P(i,i)$ can compute the area of the inscribed triangle sharing the edge $p_i p_{i+1}$ with the original polygon. Finally, every processor $P(i,i)$ $(1 \le i \le n)$ broadcasts the value of the area it stores vertically to processor $P(1,i)$. Once this information is available in the first row of the mesh, by Proposition 3.1, computing the maximum of these values can be performed in O(log n) time. The correctness of our algorithm is guaranteed by Proposition 3.8. Hence we have the following result.

Theorem 3.9. The INSCRIBED TRIANGLE problem can be solved in O(log n) time on a mesh with multiple broadcasting of size $n \times n$. Furthermore, this is time-optimal. □

4. Conclusions and Open Problems

To reduce communication diameter, mesh-connected computers have recently been augmented by the addition of various types of bus systems. One of the more promising such augmented systems, referred to as mesh with multiple broadcasting, involves augmenting the basic mesh-connected computer by the addition of row and column buses.

In this paper, we have established $\Omega(\log n)$ time lower bounds for the following problems with an n-vertex convex polygon P as input:

- computing the diameter;
- computing the smallest enclosing rectangle;
- computing a maximum-area inscribed triangle sharing an edge with P.

Furthermore, we showed that the bounds are tight by providing O(log n) algorithms to accomplish these three tasks on meshes with multiple broadcasting of size $n \times n$.

Other problems seem to be harder. First, we don't know how to determine the largest inscribed triangle in a given convex polygon. In [6] an elegant $O(n)$ time sequential algorithm is presented but it does not seem to be parallelizable to run in O(log n) time. Next, it would be nice to solve the symmetric problems of computing the smallest-area enclosing triangle as well as the largest inscribed circle and rectangle.

References

1. G. S. Almasi and A. Gottlieb, Highly parallel computing, Benjamin/Cummings, Redwood City, California, 1990.
2. D. H. Ballard and C. M. Brown, Computer Vision, Prentice-Hall, 1982.
3. D. Bhagavathi, P. Looges, S. Olariu, J. L. Schwing, and J. Zhang, Fast selection algorithms on meshes with multiple broadcasting, *Proc. of the Internat. Conference on Parallel Processing*, St. Charles, 1992.
4. W.-E. Blanz, D. Petkovic, and J. L. C. Sanz, Algorithms and Architectures for Machine Vision, in C. H. Chen, ed., *Signal Processing Handbook*, M. Dekker, New York, 1989.
5. S. A. Cook, C. Dwork, and R. Reischuk, Upper and lower time bounds for parallel random access machines without simultaneous writes, *SIAM Journal on Computing*, 15, (1986) 87-97.
6. D. P. Dobkin and L. Snyder, On a general method for maximizing and minimizing among certain

geometric problems, *Proc. 20th Annual Symp. on Foundations of Computer Science*, 1979, 9-17.
7. R. O. Duda and P. E. Hart, Pattern Classification and Scene Analysis, Wiley and Sons, New York, 1973.
8. H. Freeman and R. Shapira, Determining the minimum area encasing rectangle for an arbitrary closed curve, *Communications of the ACM*, 18, (1975), 409-413.
9. V. K. P. Kumar and D. I. Reisis, Image Computations on Meshes with Multiple Broadcast, *IEEE Trans. on Pattern Analysis and machine Intelligence*, 11, (1989), 1194-1201.
10. F. Thomson Leighton, Introduction to Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes, Morgan Kaufmann Publishers, San Mateo, California, 1992.
11. R. Lin, S. Olariu, J. L. Schwing, and J. Zhang, Simulating Enhanced Meshes, with Applications, Technical Report TR-91-041, Department of Computer Science, Old Dominion University, Norfolk, Virginia.
12. T. Lozano-Perez, Spatial Planning: A Configurational Space Approach, *IEEE Trans. on Computers*, C-32, 1983, 108-119.
13. M. Lu, Constructing the Voronoi diagram on a mesh-connected computer, *Proceedings of the International Conference on Parallel Processing*, (1985) 806-811.
14. M. Maresca and H. Li, Connection autonomy and SIMD computers: a VLSI implementation, *Journal of Parallel and Distributed Computing*, 7, (1989) 302-320.
15. S. Olariu, J. L. Schwing, and J. Zhang, Time-Optimal Sorting and Applications on $n \times n$ Enhanced Meshes, *Proc. Sixth Annual European Computer Conference*, The Hague, May 1992.
16. T. Pavlidis, Computer Graphics, Computer Science Press, Potomac, MD, 1978.
17. V. K. Prasanna and C. S. Raghavendra, Array Processor with Multiple Broadcasting, *Journal of Parallel and Distributed Processing*, 4 (1987) 173-190.
18. B. Preas and M. Lorenzetti, Eds., Physical design and automation of VLSI systems, Benjamin/Cummings,, Menlo Park, 1988.
19. F. P. Preparata and M. I. Shamos, Computational Geometry, an Introduction, Springer-Verlag, 1985.
20. A. Rosenfeld and A. Kak, Digital Picture Processing, Academic Press, vol. 1-2, 1982.
21. G. T. Toussaint, Movable Separability of Sets, in G.T. Toussaint ed., Computational Geometry, Elsevier Science Publishers, North-Holland, 1985.
22. D. Vernon, Machine vision, automated visual inspection and robot vision, Prentice-Hall, 1991.
23. U. Vishkin, Structural parallel algorithmics, Technical Report UMIACS-TR-91-53, April, 1991.

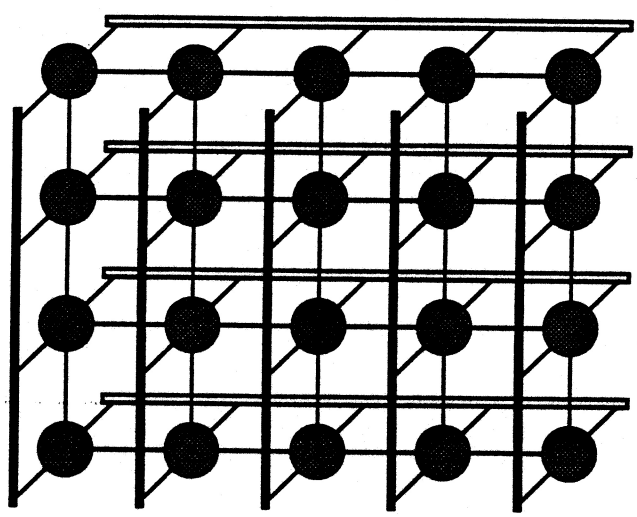

Figure 1: A 4x5 mesh with multiple broadcasting