Time-Optimal Proximity Graph Computations on Enhanced Meshes
(Extended Abstract)

Stephan Olariu* Ivan Stojmenović† Albert Y. Zomaya‡

Abstract

Proximity graphs play a fundamental role in pattern recognition, VLSI design, computer graphics, morphology, and image processing, among others. Our contribution is to propose time-optimal algorithms for constructing the Euclidian Minimum Spanning Tree and the Relative Neighborhood Graph of an n-vertex unimodal polygon as well as the Symmetric Furthest Neighbor Graph of n points in the plane. All our algorithm run on meshes enhanced with row and column buses. We begin by establishing a $\Omega(\log n)$ time lower bound for the tasks of computing the Euclidian Minimum Spanning Tree of an n-vertex unimodal polygon, as well as for the task of computing the Symmetric Furthest Neighbor Graph of a set of n points in the plane. This lower bound holds for both the CREW-PRAM and for meshes with multiple broadcasting. We then show that this time lower bound is tight by exhibiting algorithms for these tasks running in $O(\log n)$ time on a mesh with multiple broadcasting of size $n \times n$. We further show that the All Nearest Neighbors and the Relative Neighborhood Graph of an n-vertex unimodal polygon can be computed in $O(1)$ time on meshes with multiple broadcasting.

1 Introduction

Recently, an elegant and powerful architecture has been obtained by adding one bus to every row and to every column in the mesh, as illustrated in Figure 1. In [7] an abstraction of such a system is referred to as *mesh with multiple broadcasting.* The mesh with multiple broadcasting has been implemented in VLSI and is commercially available in the DAP family of multicomputers [7, 12]. In turn, due to its commercial availability, the mesh with multiple broadcasting has attracted a great deal of attention. Applications ranging from image processing [7, 12], to computer graphics and robotics [3, 11], to computational geometry and pattern recognition [3, 7, 11], to other fundamental problems [2, 4] have found efficient solutions on this platform and some of its variants. A mesh with

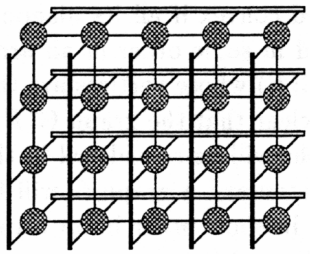

Figure 1: *A mesh with multiple broadcasting of size* 4×5

multiple broadcasting of size $n \times n$ consists of n identical SIMD processors positioned on a square array. We assume that processors know their coordinates within the mesh and have a *constant* number of registers of size $O(\log n)$. In unit time, each processor performs an arithmetic or boolean operation, communicates with one of its neighbors using a local connection, broadcasts a value on a bus or reads a value from a specified bus. We assume that communication along buses takes constant time. Each of these operations involves handling at most $O(\log n)$ bits of information. For

*Department of Computer Science, Old Dominion University, Norfolk, VA 23529-0162 USA

†Department of Computer Science, University of Ottawa, Ottawa, Ontario, K1N 9B4, Canada

‡Parallel Computing Research Lab, Dept. of Electrical and Electronic Eng, Univ. of Western Australia, Perth, WA 6907, Australia

practical reasons, only one processor is allowed to broadcast on a given bus at any one time. By contrast, all the processors on the bus can simultaneously read the value being broadcast.

A PRAM consists of SIMD processors, with unit-time access to a shared memory. In the CREW-PRAM, a memory location can be simultaneously accessed in reading but not in writing. A mesh with multiple broadcasting can be perceived as a restricted version of the CREW-PRAM: the buses are nothing more than *oblivious* concurrent read, exclusive write registers with the access restricted to certain sets of processors. Indeed, a square mesh with multiple broadcasting using p processors can be viewed as a CREW-PRAM with p processors where groups of $\sqrt{p}$ of these have concurrent read access to a register whose value is available for one time unit, after which it is lost. Given that the mesh with multiple broadcasting is, in this sense, *weaker* than the CREW-PRAM, it is very often quite a challenge to design algorithms in this model that match the performance of their CREW-PRAM counterparts. Typically, for the same running time, the mesh with multiple broadcasting uses more processors. This phenomenon will appear in our algorithms.

A classic problem in pattern recognition and classification is to compute for every point in a given set S, a point that is closest to it: this problem is known as the All-Nearest Neighbor problem (ANN, for short) and has been well studied in both sequential and parallel [6, 11, 13, 15]. Very recently, Olariu and Stojmenović [11] provided time-optimal algorithms for solving the ANN problem for points in the plane on the mesh with multiple broadcasting. A class of related problems involves associating a certain graph with the set S. This graph is, of course, application-specific. For example, in pattern recognition one is interested in the Euclidian Minimum Spanning Tree (EMST) of S, the Relative Neighborhood Graph (RNG) of S, the Symmetric Furthest Neighbor Graph (SFN) of S, the Gabriel Graph (GG) of S, and the Delaunay Graph (DG) of S, to name a few [13, 14, 15, 16]. In this context, our contribution is to provide time-optimal algorithms to construct the ANN, EMST and RNG, of an n-vertex unimodal polygon as well as the SFN of a set of n points in the plane.

We begin by establishing a $\Omega(\log n)$ time lower bound for both the task of computing the EMST of an n-vertex unimodal polygon and for the task of computing the SFN graph of a set of n points in the plane. This time lower bound holds for both the CREW-PRAM and for the mesh with multiple broadcasting. Next, we show that the bound is tight by exhibiting algorithms solving these problems in $O(\log n)$ time on a mesh with multiple broadcasting of size $n \times n$.

2 Background and Terminology

A polygon $P = p_1, p_2, \ldots, p_n$ is termed convex if all its diagonals lie entirely within P. Typically, P is represented by a linked list stored in some order in an array $P[1..n]$. With the polygon P we associate in a natural way the graph $G(P)$ whose vertices are the vertices of P and whose edges are precisely the edges of P. It should be clear that the graph $G(P)$ can be constructed from P in $O(1)$ time. Since a number of different graphs will be considered in this paper, it is appropriate to clarify how these graphs are specified. As customary, a graph G will be specified by a linked list of edges i.e. unordered pairs of vertices stored in some order in an array G. This representation facilitates the assignment of processors to the edges of G.

Call a real function $f : \{0, 1, \ldots, n-1\} \rightarrow R$ *unimodal* if there exists an integer i $(0 \leq i \leq n-1)$ such that f is strictly increasing in the interval $[0,i]$ and strictly decreasing in the interval $[i, n-1]$. A vertex p_i of a polygon P is said to be *unimodal* if the Euclidian distance function $d(p_i, p_j)$ is unimodal. The polygon P itself is termed unimodal if all its vertices are unimodal. Traditionally, convexity has played a central role in analyzing relevant features of the shape of a set of points [6, 13]. Recently, Toussaint [15] pointed out that the notions of convexity and unimodality are quite different: convex polygons need not be unimodal, and unimodal polygons need not be convex. Furthermore, in [15] it is argued that the key factor for obtaining very efficient algorithms for a large number of problems in computational geometry is not convexity, but rather unimodality. It is not surprising, therefore, that unimodality and multimodality have received considerable attention in the literature [1, 9, 14, 15, 16].

The *Euclidian Minimum Spanning Tree* (EMST) of a set S of points in the plane is a minimum spanning tree of the complete graph whose vertices are the points in S and whose edges are weighted by the Euclidian distance between the corresponding points. The *Relative Neighborhood Graph* (RNG) of a set S of points in the plane has been introduced by Toussaint [15] in an effort to capture many perceptually relevant features of the set S. Specifically, given a set S of points in the plane, RNG(S) has for vertices the points of S together with an edge between p and q if $d(p,q) \leq \max_{s \in S}\{d(p,s), d(q,s)\}$. An equivalent definition states that two vertices p, q are joined by an edge in RNG(S) if no other points of S lie inside LUNE(p,q), the "lune of influence" of p, q defined as the intersection of two open disks each of radius $d(p,q)$ centered at p, q respectively [14].

Given a set S a points in the plane, the points p and q of S are a *symmetric furthest pair* if both p and q are furthest from each other among the points of S, i.e. all other points lie inside $LUNE(p,q)$. Put differently, $d(p,q) = \max_{s \in S}\{d(p,s)\}$ and $d(q,p) = \max_{s \in S}\{d(q,s)\}$.

The *Symmetric Furthest Neighbor* Graph (SFN) associated with S has the points of S as vertices; vertices p and q are connected by an edge if and only if p and q are a symmetric furthest pair [16].

The *prefix computation* problem has turned out to one of the basic techniques in parallel processing, being a key ingredient in many algorithms. Recently, Olariu *et al.* [9] have shown that

Proposition 2.1. The prefix sums (also maxima or minima) of a sequence of n real numbers stored in one row of a mesh with multiple broadcasting of size $n \times n$ can be computed in $O(\log n)$ time. Furthermore, this is time-optimal. □

The convex hull of a set of points in the plane is defined as the smallest convex set that contains the original set [13]. Our arguments rely, in part, on the following result proved in [10].

Proposition 2.2. [10] The convex hull of planar set of n points stored in the first row of a mesh with multiple broadcasting of size $n \times n$ can be computed in $O(\log n)$ time. Furthermore, this is time-optimal. □

Vertices p_i and p_j of a convex polygon P are termed *antipodal* if P admits parallel supporting lines through p_i and p_j. A classic theorem in combinatorial geometry [13] asserts that the diameter of a convex polygon is the largest distance between antipodal pairs. Recently, the following result was proved in [3].

Proposition 2.3. [3] Let P be an n-vertex convex polygon stored one vertex per processor in one row of a mesh with multiple broadcasting of size $n \times n$. Any semigroup computation involving the antipodal pairs of P can be performed in $O(\log n)$ time. Furthermore, this is time-optimal. □

3 Lower Bounds

The purpose of this section is to derive lower bounds for the following problems.
EMST: given an n-vertex unimodal polygon compute its Euclidian Minimum Spanning Tree;
SFN: given a set S of points in the plane, compute its Symmetric Furthest Neighbor graph.

Our optimality arguments will be stated first in the CREW-PRAM. This approach is motivated by a recent result of Lin *et al.* [8] that allows us to extend many lower bound results from the CREW-PRAM to meshes with multiple broadcasting.

For further reference we now state 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. We define the problem and state the relevant result from [5].
OR: given n bits $b_1, b_2, \ldots, b_n$, compute their logical OR.

Proposition 3.1. The time lower bound for computing the OR of n bits on the CREW-PRAM is $\Omega(\log n)$, independent of the number of processors and memory cells used. □

In addition, we shall rely on the following recent result of Lin *et al.* [8].

Proposition 3.2. Any computation that takes $O(t(n))$ computational steps on an n-processor mesh with multiple broadcasting can be performed in $O(t(n))$ computational steps on an n-processor CREW-PRAM with $O(n)$ extra memory. □

It is important to note that Proposition 3.2 guarantees that if $T_M(n)$ is 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 $T_P(n) = T_M(n)$ time using n processors and $O(n)$ extra memory. In other words, "too fast" an algorithm on the mesh with multiple broadcasting implies "too fast" an algorithm for the CREW-PRAM. This observation is exploited in [8] to transfer known computational lower bounds for the PRAM to the mesh with multiple broadcasting.

We show that the time lower bound for EMST, RNG, and SFNG is $\Omega(\log n)$ on the CREW-PRAM, by reducing the OR problem to each of these problems.

We first present the lower bound argument for the EMST problem. For this purpose, let $b_1, b_2, \ldots, b_n$ be an arbitrary input to the OR problem. Consider a circle C and let $p_1, p_2, \ldots, p_{2n}$ be equally spaced points on the boundary of C. It is easy to construct C such that the points are 1 unit apart. Associate with every bit b_i, $(1 \leq i \leq n)$, the pair of points p_i and p_{n+i} on C. If $b_i = 1$ then perturb p_i and p_{n+i} by a small amount ϵ clockwise. Let P be the resulting polygon with its vertices denoted in clockwise order by $p_1, p_2, \ldots, p_{2n}$. It is easy to see that P is unimodal.

If $b_1 = 1$ then the answer to OR is 1 and we are done. Otherwise, consider any algorithm that correctly returns the EMST of the polygon P. Clearly, EMST has $2n - 1$ edges. In $O(1)$ time n processors can identify the *unique* edge of P that does not belong to the EMST. Now the answer to OR is 1 if and only if the length of this edge is larger than 1. Since the construction of P takes $O(1)$ time using n processors on the CREW-PRAM, Proposition 3.1 implies the following result.

Theorem 3.3. The task of computing the Euclidian Minimum Spanning Tree of an n-vertex unimodal polygon has a time lower bound of $\Omega(\log n)$ on the CREW-PRAM, regardless of the number of processors and memory cells available. □

It is important to note that the polygon P used in the reduction above is always convex, in addition to being unimodal. Therefore, we have the following result.

Theorem 3.4. The task of computing the Euclidian Minimum Spanning Tree of an n-vertex polygon polygon has a time lower bound of $\Omega(\log n)$ on the CREW-PRAM, independent of the number of processors and memory cells used. □

Next, we discuss the time lower bound for the SFN problem for a set S of points in the plane. Let $b_1, b_2, \ldots, b_n$ be an arbitrary input to the OR problem. With every bit b_i, $1 \leq i \leq n$, associate the point $p_i = (\frac{i}{n}, 0)$ if $b_i = 0$ and $p_i = (\frac{i}{n}, 2)$ otherwise. Now, the set S consists of all the points p_i defined above, along with the point $p_0 = (0, 0)$. It is easy to confirm that our construction guarantees that p_0 and p_n are symmetric furthest neighbors at distance 1 if and only if the OR of $b_1, b_2, \ldots, b_n$ is 0. Since the construction of P takes $O(1)$ time using n processors on the CREW-PRAM, Proposition 3.1 implies the following result.

Theorem 3.5. The task of computing the Symmetric Furthest Neighbor graph of a set of n points in the plane has a time lower bound of $\Omega(\log n)$ on the CREW-PRAM, regardless of the number of processors and memory cells available. □

By Proposition 3.2 and Theorems 3.3, 3.4, and 3.5 combined we have the following result.

Theorem 3.6. The task of constructing the EMST of an n-vertex unimodal polygon as well as the task of constructing the SFN of a set of n points in the plane has a time lower bound of $\Omega(\log n)$ on a mesh with multiple broadcasting of size $n \times n$. □

4 The Algorithms

Our goal is to design simple time-optimal algorithms to compute the ANN, EMST and RNG of an n-vertex unimodal polygon as well as for computing the SFN of a set of n points in the plane.

Recently, the authors [11] have exhibited a rather difficult algorithm for computing the ANN of an arbitrary set of points in the plane. To solve the ANN problem for unimodal polygons we rely on the following simple fact whose proof follows directly from definition.

Observation 4.0. In a unimodal polygon every edge is adjacent to its nearest neighbor. □

Observation 4.0 suggests an $O(1)$ time algorithm for the ANN problem. We omit the details.

Recently, Olariu [9] has developed simple linear-time sequential algorithms to compute the EMST and RNG of a unimodal polygon. We now recall a number of relevant properties of unimodal

polygons established in [9]. Let $P = p_0, p_1, \ldots, p_{n-1}$ be a unimodal polygon.

Proposition 4.1. [9] If a diagonal of P is an edge of RNG(P) then its endpoints belong to the two longest edges of P. □

Call an edge e of RNG(P) *P-critical* if $e \notin P$. It is worth noting that Proposition 4.1 asserts that if p_ip_j is P-critical, then p_i and p_j must be endpoints of the two longest edges in P. A natural question to ask is: "How many P-critical edges can RNG(P) contain?" The answer to this question is given by the following result.

Proposition 4.2. [9] The RNG of a unimodal polygon P contains at most one P-critical edge. □

An important consequence of Proposition 4.2 is the following.

Corollary 4.3. The EMST of an n-vertex unimodal polygon P shares at least $n-2$ edges with P. □

The following result makes Corollary 4.3 more precise, by characterizing the diagonals of P which can belong to MST(P).

Proposition 4.4. [9] A diagonal of P is an edge in EMST(P) if and only if it is a P-critical edge in RNG(P). □

We begin by showing that given an n vertex unimodal polygon $P = p_1, p_2, \ldots, p_n$, stored one vertex per processor in the first row of a mesh with multiple broadcasting of size $n \times n$, the task of detecting the edges of $P(G)$ that do not belong to $RNG(P)$ can be performed in O(1) time. For definiteness, assume that for every i, $1 \leq i \leq n$, processor $P(1, i)$ stores the vertex p_i. Using the vertical buses we replicate the contents of the first row in all the rows of the platform. Next, for every i, $1 \leq i \leq n-1$, processor $P(i, i)$ broadcasts the edge p_ip_{i+1} to all the processors in row i. Similarly, processor $P(n, n)$ broadcasts the edge p_np_1 throughout row n. Every processor in row i that detects that the point it stores in inside $LUNE(p_i, p_{i+1})$ sets a local register to 1. All other processors set the register to 0. The following result is key for our O(1) time algorithm for computing the $RNG(P)$.

Lemma 4.5. For every row i, the set of processors that store a 1 form an interval. □

By virtue of Lemma 4.5, the *leftmost* processor in row i that stores a vertex that lies inside $LUNE(p_i, p_{i+1})$ can be detected in O(1) time. In turn, this processor informs $P(i, i)$ that the edge p_i, p_{i+1} cannot belong to $RNG(P)$. To summarize our findings we state the following result.

Theorem 4.6. The task of computing the RNG(P) of an n-vertex unimodal polygon can be performed in O(1) time on a mesh with multiple broadcasting of size $n \times n$. □

We are now in a position to present an algorithm to compute the EMST(P) of an n-vertex unimodal polygon $P = p_1, p_2, \ldots, p_n$. We begin by computing the two longest edges of P and denote them by p_ip_{i+1} and p_jp_{j+1}, respectively. Let p_rp_s be any critical edge of the polygon determined by p_i, p_{i+1}, p_j, and p_{j+1}. If such a critical edge exists then EMST(P) and RNG(P) coincide and are obtained from G(P) by removing the edges p_ip_{i+1} and p_jp_{j+1} and adding the critical edge p_rp_s.

On the other hand, if such a critical edge does not exist, then EMST(P) is obtained from G(P) by removing the edge p_ip_{i+1}. The correctness of this simple and surprising algorithm is guaranteed by the following result in [9].

Proposition 4.7. Given a unimodal polygon P, the above procedure correctly computes the EMST(P). □

By Proposition 2.1, computing the longest and the second longest edges of P as well as testing whether or not LUNE(p_i, p_{i+1}) is empty takes O($\log n$) time. Thus, Proposition 2.1, Theorems 3.6, and 4.7 combined imply the following result.

Theorem 4.8. Let P be an n-vertex unimodal polygon well stored in a mesh with multiple broadcasting of size $n \times n$. The Euclidian Minimum Spanning Tree of P can be computed in O($\log n$) time on a mesh with multiple broadcasting of size $n \times n$. Furthermore, this is time-optimal on this architecture. □

To compute the SFN of a set S of points in the plane, we need the following result [16].

Proposition 4.9. Every symmetric furthest pair of points in S is an antipodal pair in the convex hull $CH(S)$. □

Proposition 4.9 motives us to proceed in the following sequence of steps.

Step 1. Compute the convex hull $CH(S)$ of S. We note that using the time-optimal algorithm of [10] the task specific to this step can be performed in $O(\log n)$ time;

Step 2. Generate all the antipodal pairs of $CH(S)$. It is well-known that the number of these antipodal pairs is bounded by $O(n)$. It is also worth noting that the algorithm of [3] generates these pairs in $O(\log n)$ time and stores them in the first row of the mesh;

Step 3. For each such pair determine if it is a symmetric furthest pair. This can be done in $O(\log n)$ time using the first steps of the diameter-finding algorithm of [3].

Thus, this simple algorithm runs in $O(\log n)$ time. To summarize, we state the following result.

Theorem 4.10. Let S be an arbitrary set of n points in the plane, stored one point per processor in the first row of a mesh with multiple broadcasting of size $n \times n$. The Symmetric Furthest Neighbor graph of S can be computed time-optimally in $\Theta(\log n)$ time on this architecture. □

References

[1] A. Aggarwal and R. C. Melville, Fast computation of the modality of polygons, *Journal of Algorithms*, **7**, (1986), 369–381.

[2] A. Bar-Noy and D. Peleg, Square meshes are not always optimal, *IEEE Transactions on Computers*, **C-40**, (1991), 196–204.

[3] D. Bhagavathi, S. Olariu, J. L. Schwing, W. Shen, L. Wilson, and J. Zhang, Convexity problems on meshes with multiple broadcasting, *Journal of Parallel and Distributed Computing*, **27**, (1995), 142–156.

[4] Y. C. Chen, W. T. Chen, G. H. Chen and J. P. Sheu, Designing efficient parallel algorithms on mesh connected computers with multiple broadcasting, *IEEE Transactions on Parallel and Distributed Systems*, **1**, (1990), 241–246.

[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] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*, Wiley and Sons, New York, 1973.

[7] V. P. Kumar and C. S. Raghavendra, Array processor with multiple broadcasting, *Journal of Parallel and Distributed Computing*, **2**, (1987), 173–190.

[8] R. Lin, S. Olariu, J. L. Schwing, and J. Zhang, Simulating enhanced meshes, with applications, *Parallel Processing Letters*, **3**, (1993), 59–70.

[9] S. Olariu, A simple linear-time algorithm for computing the RNG and MST of unimodal polygons, *Information Processing Letters*, **7**, (1989), 243–247.

[10] S. Olariu, J. L. Schwing, and J. Zhang, Optimal convex hull algorithms on enhanced meshes, *BIT*, **33**, (1993), 396–410.

[11] S. Olariu and I. Stojmenović, Time-optimal proximity problems on meshes with multiple broadcasting, *Journal of Parallel and Distributed Computing*, to appear.

[12] D. Parkinson, D. J. Hunt, and K. S. MacQueen, The AMT DAP 500, 33^{rd} *IEEE Comp. Soc. International Conf.*, 1988, 196–199.

[13] F. P. Preparata and M. I. Shamos, *Computational Geometry — An Introduction*, Springer-Verlag, Berlin, 1988.

[14] G. T. Toussaint, The relative neighborhood graph of a finite planar set, *Pattern Recognition*, **12**, (1980), 261–268.

[15] G. T. Toussaint, Complexity, convexity and unimodality, *International Journal of Computer and Information Sciences*, **13**, (1984), 197–217.

[16] G. T. Toussaint, The symmetric all-furthest neighbor problem, *Computers and Mathematics with Applications*, **9**, (1983), 747–754.