Constant Time Computational Geometry On Reconfigurable Meshes With Buses*

Madhusudan Nigam and Sartaj Sahni

University of Florida

Gainesville, FL 32611

Technical Report 92-35

ABSTRACT

We develop O(1) time algorithms for the following computational geometry problems: convex hull, smallest enclosing box, ECDF search, and triangulation. Our algorithms are for the reconfigurable mesh with buses architecture and run on the RMESH, PARBUS, and MRN models.

Keywords And Phrases
Convex hull, enclosing box, ECDF search, triangulation, reconfigurable mesh with buses.

* This research was supported, in part, by the National Science Foundation under grant MIP-9103379.
1. Introduction

Several different mesh like architectures with reconfigurable buses have been proposed in the literature. These include the content addressable array processor (CAAP) of Weems et al. [WEEM87,89], the polymorphic torus of Li and Maresca [LI89ab, MARE89], the reconfigurable mesh with buses (RMESH) of Miller et al. [MILL88abc], the processor array with a reconfigurable bus system (PARBUS) of Wang and Chen [WANG90ab], and the reconfigurable network (RN) of Ben-Asher et al. [BENA91].

The CAAP [WEEM87,89] and RMESH [MILL88abc] architectures appear to be quite similar. So, we shall describe the RMESH only. In this, we have a bus grid with an \( n \times n \) arrangement of processors at the grid points (see Figure 1 for a 4x4 RMESH ). Each grid segment has a switch on it which enables one to break the bus, if desired, at that point. When all switches are closed, all \( n^2 \) processors are connected by the grid bus. The switches around a processor can be set by using local information. If all processors disconnect the switch on their north, then we obtain row buses (Figure 2). Column buses are obtained by having each processor disconnect the switch on its east (Figure 3). In the exclusive write model two processors that are on the same bus cannot simultaneously write to that bus. In the concurrent write model several processors may simultaneously write to the same bus. Rules are provided to determine which of the several writers actually succeeds (e.g., arbitrary, maximum, exclusive or, etc.). Notice that in the RMESH model it is not possible to simultaneously have \( n \) disjoint row buses and \( n \) disjoint column buses that, respectively, span the width and height of the RMESH. It is assumed that processors on the same bus can communicate in \( O(1) \) time. RMESH algorithms for fundamental data movement operations and image processing problems can be found in [MILL88abc, MILL91ab, JENQ91abc].

An \( n \times n \) PARBUS (Figure 4) [WANG90ab] is an \( n \times n \) mesh in which the interprocessor links are bus segments and each processor has the ability to connect together arbitrary subsets of the four bus segments that connect to it. Bus segments that get so connected behave like a single bus. The bus segment interconnections at a processor are done by an internal four port switch. If the up to four bus segments at a processor are labeled N (North), E (East), W (West), and S (South), then this switch is able to realize any set, \( A = \{ A_1, A_2 \} \), of connections where \( A_i \subseteq \{ N, E, W, S \} \), \( 1 \leq i \leq 2 \) and the \( A_i \)'s are disjoint. For example \( A = \{ \{ N, S \}, \{ E, W \} \} \) results in connecting the North and South segments together and the East and West segments together. If this is done in each processor, then we get, simultaneously, disjoint row and column buses (Figure 5 and 6). If A
= \{\{N,S,E,W\},\phi\}, then all four bus segments are connected. PARBUS algorithms for a variety of applications can be found in [MILL91ab, WANG90ab, LIN92, JANG92abcde]. Observe that in an RMESH the realizable connections are of the form \(A = \{A_1\}, A_1 \subseteq \{N,E,W,S\}\).

The polymorphic torus architecture [LI89ab, MARE89] is identical to the PARBUS except that the rows and columns of the underlying mesh wrap around (Figure 7). In a reconfigurable network (RN) [BENA91] no restriction is placed on the bus segments that connect pairs of processors or on the relative placement of the processors. I.e., processors may not lie at grid points and a bus segment may join an arbitrary pair of processors. Like the PARBUS and polymorphic torus, each processor has an internal switch that is able to connect together arbitrary subsets of the bus segments that connect to the processor. Ben-asher et al. [BENA91] also define a mesh restriction (MRN) of their reconfigurable network. In this, the processor and bus segment arrangement is exactly as for the PARBUS (Figure 4). However the switches internal to processors are able to
obtain only the 10 bus configurations given in Figure 8. Thus an MRN is a restricted PARBUS.

While we have defined the above reconfigurable bus architectures as square two dimensional meshes, it is easy to see how these may be extended to obtain non square architectures and architectures with more dimensions than two.

RMB algorithms for computational geometry problems has been explored in [MILL87, JANG92e, REIS92]. In [MILL87] RMESH algorithms for several geometric
problems on digitized images were developed. These include closest figure, extreme points of every figure, diameter, smallest enclosing box, and smallest enclosing circle.

An $O(1)$ time convex hull algorithm for a set of $N$ planar points is given in [REIS92]. The algorithm uses an $N \times N$ configuration and is based on the grid convex hull algorithm of [MILL88]. Unfortunately, the algorithm of [REIS92] is flawed. In section 2, we give a corrected constant time RMESH convex hull algorithm. This may also be run on an $N \times N$ PARBUS and MRN. Jang and Prasanna [JANG92e] develop constant time PARBUS algorithms for the all pairs nearest neighbor problem on a set of $N$ planar points, 2-set dominance counting for $N$ planar points, and the 3-dimensional maxima problem. All these algorithms are for $N \times N$ PARBUS configuration. Their algorithm for the nearest neighbor problem is easily run on an $N \times N$ RMESH and MRN in constant time. Jang and Prasanna [JANG92e] state that their 2-set dominance counting algorithm can be simulated by an RMESH using the switch simulation of [JANG92d]. However the simulation of [JANG92e] requires 16 RMESH processors for each PARBUS processor simulated. Hence a $4N \times 4N$ RMESH is needed for the simulation. In section 4, we consider the ECDF search problem which is closely related to the 2-set dominance counting problem and develop a constant time algorithm that requires only an
The third geometry problem considered in [JANG92e] is the 3-dimensional maxima problem. In this, we are given a set $S$ of three tuples (points in three dimensional space) and are required to find all points $p \in S$ such that there is no $q \in S$ with the property that each coordinate of $q$ is greater than the corresponding coordinate of $p$ (i.e., no $q$ in $S$ dominates $p$). The algorithm suggested in [JANG92e] is a modification of their 2-set dominance counting algorithm and is unnecessarily complicated. We observe that the problem is trivially solved in constant time using the algorithm of Figure 9. The input $N$ points are in row 1 of the $N \times N$ PARBUS/RMESH/MRN. The algorithm of Figure 9 can be used to solve, in constant time, the $k$-dimensional maxima problem for any $k$.

The remaining computational geometry problems we consider in this paper are the smallest enclosing rectangle problem (section 3) and the problem of triangulating a set of $N$ planar points (section 5). While these problems have not been considered before for the RMB architecture, parallel algorithms for other architectures have been developed. For example, Miller and Stout [MILL89] show how to find the smallest enclosing
Figure 6  Column buses in a PARBUS

Figure 7  4 x 4 Polymorphic torus
rectangle of a set of $N$ planar points in $O(N^{1/2})$ on an $N \times N$ mesh and Wang and Tsin [WANG87] develop an $O(\log N)$ time CREW PRAM algorithm to triangulate a set of $N$ planar points. Miller and Stout [MILL89] also develop optimal mesh algorithms for the convex hull of $N$ planar points and for several other computational geometry problems. Mesh algorithms for some other computational geometry problem are considered in [JEON90, LU85,86].

2. Convex Hull

In this section, we consider the problem of determining the Convex Hull of a set of $N$ planar points on $N \times N$ Reconfigurable Meshes with Buses (RMBs). While the RMESH algorithms we develop can be run without modification on PARBUSs and MRNs, improved performance results from the use of algorithms for subtasks such as ranking and sorting that are tailored to the PARBUS/MRN architecture.
Step 1: Broadcast the $N$ points on row 1 to all rows using column buses.

Step 2: (Use the $i'$th row to determine if the $i'$th point of $S$ is a non dominated point)
- Processor $[i,i]$ broadcasts its point to all processors in its row using a row bus, $1 \leq i \leq N$
- Processor $[i,j]$, $1 \leq i, j \leq N$ compares the coordinates of the point it received in step 1 to those of the points it received in step 2. If each coordinate of the step 1 point is larger than the corresponding coordinate of its step 2 point, then PE$[i,j]$ sets its $T$ variable to 0. Otherwise, $T$ is set to 1.
- Using row bus splitting, PE$[i,1]$ determines if there is a 0 in row $i$, $1 \leq i \leq N$. If so, it sets its $U$ variable to 0 (the $i'$th point is dominated). Otherwise, $U$ is set to 1 (the $i'$th point is not dominated).

Step 3: (Route the results back to row 1)
- Using row buses PE$[i,1]$ sends its $U$ value to PE$[i,i]$. Using column buses, PE$[i,i]$ sends the $U$ value just received to PE$[1,i]$, $1 \leq i \leq N$.

**Figure 9** Simple Constant time algorithm for 3-dimensional maxima

Let $S$ be a set of $N$ distinct points in a plane. Let $E(S)$ be the set of extreme points in $S$ and let $CH(S)$ denote an ordering of $E$ such that $CH(S)$ defines a convex polygon. This convex polygon is the convex hull of $S$. The extreme points problem is that of finding $E(S)$ while in the convex hull problem, we are to find the ordered set $CH(S)$. Our algorithms to find $E(S)$ and $CH(S)$ make use of the following known results:

**Lemma 1**: [Theorem 3.8, [PREP85], p104] The line segment $l$ defined by two points is an edge of the convex hull iff all other points lie on $l$ or to one side of it. □

**Lemma 2**: [[PREP85], p105] Let $p_1$ and $p_2$, respectively, be the lexicographically lowest and the highest points of $S$. Let $CH(S)$ be $(p_1, l_1, l_2, \ldots, l_k, p_2, u_1, \ldots, u_j)$. The ordering is a counter clockwise ordering of the extreme points and is such that the interior of the defined convex polygon is on the left as one traverses the extreme points in this order. The points $l_1, \ldots, l_k, p_2$ have the property that the polar angle made by each of these points, the positive $x$-axis, and the preceding point of $CH(S)$ is least. The points
$u_1, u_2, ..., u_j$ have the property that the polar angle made by each point, the negative $x$-axis, and the preceding point of $CH(S)$ is least (see Figure 10). \qed

---

**Figure 10** Convex hull of a set of points

### 2.1. $N^3$ Processor Algorithm

Before developing the $O(1)$ $N \times N$ processor algorithm, we develop a simple $O(1)$ $N^2 \times N$ processor algorithm. An $O(1)$ $N^2 \times N$ processor algorithm for $E(S)$ or $CH(S)$ is easily obtained by using either Lemma 1 or 2. We elaborate on the algorithm based on Lemma 1. The strategy employed by this algorithm is given in Figure 11. It assumes that the $N$ points of $S$ are initially in the top row of the $N^2 \times N$ RMESH/PARBUS/MRN. The $i$'th row, $1 \leq i \leq N^2$ of the RMESH/PARBUS/MRN is used to determine if points $j$ and $k$ where $i = (j-1)N + k$ of $S$ form an edge of the convex hull. In case they do, then $j$ and $k$ are in $E(S)$ and $j$ immediately precedes or immediately follows $k$ in $CH(S)$.

The correctness of Figure 2 is a direct consequence of Lemma 1 and the fact [PREP85, p100] that an ordering of the points of $E$ by polar angle about an internal point yields the convex hull. The complexity of the algorithm is readily seen to be $O(1)$. Note that if only $E(S)$ is to be computed then steps 9-11 may be omitted.
Step 1: Use column buses to broadcast the points of $S$ from row 1 to all other rows.

Step 2: In row $i = (j - 1)N + k$, $1 \leq i \leq N^2$, points $j$ and $k$ are broadcast to all processors using a row bus.

Step 3: Using the points $j$ and $k$ received in step 2, each processor computes $a$, $b$, and $c$ such that $ax + by + c = 0$ defines the straight line through points $j$ and $k$.

Step 4: Let $(u, v)$ be the coordinates of the point of $S$ assigned to processor $[e, f]$ in step 1. Processor $[e, f]$ sets its flag to 1 if $au + bv + c > 0$, -1 if $au + bv + c < 0$, 0 if $au + bv + c = 0$ and $(u,v)$ is on the line segment between points $j$ and $k$ of step 3 (this includes the cases when $(u,v)$ is point $j$ or $k$), 2 otherwise.

Step 5: Using row buses and row bus splitting, PE$[i, 1]$, $i = (j - 1)N + k$, $1 \leq i \leq N^2$, determines if there is a flag $= 2$ on row $i$. If so, $(j, k)$ is not an edge of the convex hull. If not, it determines if there is a 1 and later if there is a -1. If both a 1 and -1 are present, $(j,k)$ is not an edge of the convex hull. Otherwise it is.

---

**Figure 11** $O(1) N^2 \times N$ processor algorithm for $E(S)$ and $CH(S)$

(continued on next page)

### 2.2. $N^2$ Processor Algorithm

The convex hull of $S$ can be computed in $O(1)$ time on an $N \times N$ RMESH using the algorithm of Figure 12. The algorithm for an $N \times N$ PARBUS/MRN is similar. Step R0 can be done using the $O(1)$ sorting algorithm for $N$ data items on an $N \times N$ RMESH [NIGA92]. Step R1 is done using the $O(1) N^3$ processor algorithm of the preceding section on each $N \times N^{1/2}$ sub RMESH. For R2, we use the $i$’th $N \times N^{1/2}$ sub RMESH to determine which points of $E_i$ are also in $E$. This is done using the algorithm of Figure 13.
Step 6: If $\text{PE}[i,1], i = (j-1)N + k, i \leq N^2, j > k$ detects that $(j,k)$ is an edge of the convex hull, then using a row bus it broadcasts a 1 to PEs $[i,j]$ and $[i,k]$.

Step 7: PEs that receive a 1 in step 6 broadcast this to the PEs in row 1 of the same column (note 0 or 4 PEs in each column receive a 1 in step 6; using column bus splitting, all but one of these may be eliminated to avoid concurrent writes to a column bus).

Step 8: Row 1 PEs now mark the points of $S$ they contain as being in or out of $E(S)$ (note: a point is in $E(S)$ if and only if it receives a 1 value in step 7).

Step 9: Using row bus splitting in row 1, any three extreme points are accumulated in $\text{PE}[1,1]$. In case $|E(S)| = 2$, the remainder of this step is omitted. The centroid of these three points is computed. Since no three points of $E$ can be colinear, the centroid is an interior point of the convex hull.

Step 10: The centroid computed in step 9 is broadcast to all points of $E(S)$ using a row bus. Each of these points computes the polar angle made* by the point using the centroid as the origin.

Step 11: The points of $E(S)$ are sorted by polar angle using the $O(1)$ RMESH/PARBUS?MRN sort algorithm of [NIGA92].

---

**Figure 11** $O(1) N^2 \times N$ processor algorithm for $E(S)$ and $CH(S)$

(continued from previous page)

**Theorem 1:** The extreme points algorithm of Figure 13 is correct.

**Proof:** To establish the correctness of Figure 13, we need to show that exactly those points of $\bigcup_{k=1}^{N^{1/2}} E_k$, that are not in $E(S)$ are eliminated in steps T5 and T6. It is easy to see

* An alternative to using the polar angle is discussed on p100 of [PREP85].
R0: Sort $S$ by $x$-coordinate and within $x$ by $y$-coordinate.

R1: Partition $S$ into $N^{1/2}$ subsets, $S_i$, $1 \leq i \leq N^{1/2}$ using the ordering just obtained. Each subset is of size $N^{1/2}$. Determine the convex hull $CH_i$ and extreme points $E_i$ of $S_i$ using an $N \times N^{1/2}$ sub RMESH for each $i, 1 \leq i \leq N^{1/2}$.

R2: Combine the extreme points $E_i$, $1 \leq i \leq N^{1/2}$, to obtain the extreme points $E$ of $S$.

R3: Obtain the convex hull of $S$ from $E$.

---

**Figure 12** $N \times N$ RMESH algorithm for convex hull

that no point eliminated in T5 or T6 can be in $E$ as in T5 the eliminated points are not in the convex hull of $E_i \cup E_j$ and in T6 they are not in the convex hull of $\cup E_k$. To see that the points not eliminated are all in $E(S)$, suppose that there is a non eliminated point $p \in E_i$ that is not in $E(S)$. Let $Q$ be the set of non eliminated points. Clearly, $E(S)$ = extreme points of $Q$ and $p$ is in the interior of the convex hull of $Q$ (see Figure 16), $CH(Q)$. Without loss of generality, assume that $|E(S)| > 2$. Since $p$ is in the interior of $CH(Q)$, there must be a $q \in Q$ such that $q \notin S_i$ and the line from $p$ to $q$ does not go through the interior of $CH(S_i)$ and does not touch the boundary of $CH(S_i)$ except at $p$. Without loss of generality, assume that $q \in S_j$ and $q$ is to the right of $p$ or vertically above $p$. If $p$ lies between the two tangents from $q$ to $S_i$, then $p$ lies between the tangents of $S_i$ and $S_j$ and so is eliminated in step T5 (see Figure 17). Since $p \in Q$, $p$ is not eliminated in T5 and hence must be on a tangent from $q$ to $S_i$ (Figure 18). Furthermore, if $p$ is not a tangent point with respect to $S_i$ and $S_j$, then $p$ lies between the two tangents of $S_i$ and $S_j$ and is eliminated in T5. So, we may assume $p$ is one of the $3N^{1/2} - 2$ points considered in step T6.

First suppose that $p$ is a point of the leftmost partition $S_1$. Since $p$ is in the interior of $CH(Q)$, part of the boundary of $CH(Q)$ passes above $p$ and part below $p$. Let $a$ be the leftmost point in $E_1$, $b$ the first point in $CH(Q)$ that the lower boundary reaches outside of $S_1$, and $c$ the first point in $CH(Q)$ that the upper boundary reaches outside of $S_1$ (see
T1: Let \( R_{ji} \) denote the \((j,i)\)'th \( N^{1/2} \times N^{1/2} \) sub RMESH of the entire RMESH. Move \( E_i \) and \( E_j \) into \( R_{ji} \) such that each row of \( R_{ji} \) has \( E_i \) (one point to a processor) and each column has \( E_j \) (one point to a processor). This can be accomplished by broadcasting row 1 along column buses to all rows and then broadcasting the diagonal (of the \( N \times N \) RMESH) processor data along row buses to all columns.

T2: Each PE in the \( k' \)th column of \( R_{ji} \) computes the slope of the line that joins the \( k' \)th point of \( E_i \) and the point of \( E_j \) it contains. These slopes form a tritonic sequence (they may decrease, increase, decrease or may increase, decrease, increase) as the points of \( E_j \) are in convex hull order.

T3: The minimum and maximum of the slopes in each column of \( R_{ji} \) are found using bus splitting. These are stored in row 1 of \( R_{ji} \).

T4: The maximum of the at most \( N^{1/2} \) minimums found in step T3 is determined by forming the cross product of the minimums in the \( N \) processors of \( R_{ji} \). The minimum of the maximums of step T3 is similarly found.

Figure 13 Substeps for step R2 (continued on next page)

Figure 19). If \( b = c \), then \( p \) is not on the tangent from \( c \) to \( CH(S_1) \) and so must have been eliminated in step T5. So, \( b \neq c \). Let \( d \) and \( e \), respectively, be the points in \( S_1 \) to which \( c \) and \( b \) are connected in \( CH(S_1) \). Note that it is possible that \( d = a \) or \( e = a \) (or both). If \( c \in S_j \), then \( d \) and \( c \) must be tangent points with respect to \( S_1 \) and \( S_j \) (as otherwise, some points of \( S_1 \) and/or \( S_j \) lie above the upper boundary of \( CH(S) \)). Similarly, \( e \) and \( b \) are tangent points with respect to \( S_1 \) and \( S_k \) where \( b \in S_k \). It is clear that \( p \) is not an extreme point of \{\( a, b, c, d, e, p \)\}. Since \{\( a, b, c, d, e, p \)\} is a subset of the set used in step T6, \( p \) must have been eliminated in this step.

Next, suppose that \( p \) is a point of \( S_{N^{1/2}} \). The proof for this case is similar to that for \( p \in S_1 \). Finally, suppose \( p \in S_j, i \neq \{1, N^{1/2} \} \). Let \( a, b, c, d \) be the first points of the portions of the boundary of \( CH(Q) \) that are above and below \( p \) that are not in \( S_i \) (Figure
T5: The minimum and maximum of step T4 define the two tangents of $E_i$ and $E_j$. These are used to eliminate points of $E_i$ that are now known to not be in $E$ (see Figure 14).

T6: The tangents of T5 define up to 4 tangent points. Over all $R_{ji}$, $1 \leq i \leq N^{1/2}$, there are $3 \ N^{1/2} - 2$ non eliminated tangent points plus non eliminated points of $S_i$. The extreme points of this set of points is computed. Tangent points of $E_i$ that are not in the set of extreme points just computed are eliminated (see Figure 15).

T7: The points of $\bigcup E_i$ not eliminated in steps T5 and T6 define $E$.

---

Figure 13 Substeps for step R2 (continued from previous page)

---

Figure 14 Examples of extreme points eliminated in step T5
\[ i = 2 \]
\[ * = \text{tangent points} \]
\[ X = \text{extreme points of E2 eliminated in step T5} \]
\[ + = \text{extreme points of E2 eliminated in step T6} \]

**Figure 15** Example of extreme point elimination in step T6

---

**Figure 16** Example for correctness proof

20). Note that it is possible that \( a = b \) and \( c = d \). Also, note that \( a, b, c, \) and \( d \) must exist as \( |S_j| = N^{1/2} \) for each \( j \) (i.e., as no \( S_j \) is empty).

Suppose that \( a \in S_j \). There must be a tangent point with respect to \( S_j \) and \( S_i \) that is in the triangle defined by \( a, a' \) and \( a'' \) (see Figure 20). This is so as \( p \) is a tangent point with respect to \( S_j \) and the other end of the tangent must lie in the said triangle. Let this
tangents from \( q \) to \( S_i \)

---

tangents between \( S_i \) and \( S_j \)

---

**Figure 17** Tangents

---

**Figure 18** \( p \) is on tangent from \( q \)

point be \( A \). Similarly, there are tangent points \( B, C, D \) in the corresponding triangles associated with \( b, c, \) and \( d \). From the triangles in which \( A, B, C, D \) are located, we see that \( p \) is contained in the convex hull of \( \{A, B, C, D, p\} \) and so must be eliminated in step T6. Note that when \( a = b \) and \( c = d \), there must be a tangent point, \( e \), of \( E_i \) that is on
the upper and lower part of the boundary of \( CH(S) \). Now, \( p \) is not an extreme point of \( \{ A, B, C, D, e, p \} \) (we need point \( e \) as it is possible that \( A = B \), and \( C = D \)).

One may verify that each of the steps T1 through T7 can be done in \( O(1) \) time. For step T6, a modified version of the \( N^3 \) processor algorithm of the previous section is run in each \( N \times N^{1/2} \) sub RMESH. Since the number of edges to consider is less than \( (3N^{1/2})^2 = 9N \), the algorithm is run at most nine times. In each run of the algorithm an edge needs to compare with \( 3N^{1/2} - 2 \) points using \( N^{1/2} \) processors. This is done in three passes.

For step R3, the points of \( E \) may be sorted by polar angle about an interior point as in steps 10 and 11 of Figure 2.

---

**Figure 19** \( p \in S_1 \)
3. Smallest Enclosing Rectangle

It is well known [FREE75] that the smallest enclosing rectangle of a set of \( N \) planar points has at least one side that is an extension of an edge of the convex hull. Hence, the smallest enclosing rectangle may be found by first computing the convex hull; then determining for each convex hull edge, the smallest enclosing rectangle that has one side which is an extension of this edge; and finally determining the smallest of these rectangles. The algorithm is given in Figure 21. Its correctness is immediate and one readily sees that its complexity is \( \mathcal{O}(1) \) on an \( N \times N \) RMESH, PARBUS and MRN.

4. ECDF Search On An \( N \times N \) RMESH

Let \( S \) be the set of \( N \) distinct planar points. Point \( p \in S \) dominates point \( q \in S \) if and only if each coordinate of \( p \) is greater than the corresponding coordinate of \( q \). In the ECDF search problem [STOJ86], we are to compute for each \( p \in S \), the number, \( D(p,S) \), of points of \( S \) that are dominated by \( p \). Stojmenovic [STOJ86] provides an \( N \ (N = |S|) \) processor, \( \mathcal{O}(\log^2 N) \) time hypercube algorithm for this problem.

On an \( N \times N \) RMB, \( D(p,S) \) for each point \( p \in S \), may be computed using a strategy similar to that used in [JANG92e] to solve the 2-set dominance counting problem on a PARBUS in \( \mathcal{O}(1) \) time. A high level description of the strategy is given in Figure 22. For simplicity in presentation, we assume that no two points of \( S \) have the same \( x \)- or the
Step 1: Find the convex hull of the \( N \) points.

Step 2: [Construct convex hull edges]
(a) The convex hull points are broadcast to all rows using column buses. These are saved in variable \( R \) of each processor.

(b) \( \text{PE}[i,i] \) broadcasts its \( R \) value using a row bus to the \( S \) variable of all processors on row \( i \), \( 1 \leq i \leq p \), \( p = \) number of convex hull points.

(c) \( \text{PE}[i,i+1] \) broadcasts its \( R \) value using a row bus to the \( T \) variable of all processors on row \( i \), \( 1 \leq i \leq p - 1 \). For \( i = p \), \( \text{PE}[p,1] \) does this. (Note: Now each PE in row \( i \) contains the same edge of the convex hull in its \( S \) and \( T \) variables)

Step 3: [Determine area of minimum rectangle for each edge]
(a) Using its \( R \), \( S \), and \( T \) values, each PE computes the perpendicular distance \( D \) between point \( R \) and the straight line defined by the points \( S \) and \( T \). Since, the \( R \)'s are in convex hull order, the \( D \) values in a row form a tritonic sequence whose minimum, \( h \), can be found in \( O(1) \) time using row bus splitting. \( h \) is the minimum height of the rectangle for the row edge.

---

**Figure 21** Minimum enclosing rectangle (continued on next page)

same \( y \)-coordinates.

Step 1 can be done by sorting the \( N \) points in \( O(1) \) time by \( x \)-coordinate [NIGA92]. The point in \( \text{PE}[1, j] \) belongs to partition \( X_u \) where \( u = \lfloor \frac{j-1}{N^{1/2}} \rfloor + 1 \). Step 2 is similarly accomplished by another sort (this time by \( y \)-coordinate). Following this each point knows which \( X \) and which \( Y \) partition it belongs to. The computation of \( DX(p) \) is done using an \( N \times N^{1/2} \) submesh for the points in each \( X \) partition. The \( i \)'th such submesh is used for \( X_i \). For this, each \( p \in X_i \) uses an \( N^{1/2} \times N^{1/2} \) partition of the \( N \times N^{1/2} \) submesh. In processor \( k \) of row 1 of this \( N^{1/2} \times N^{1/2} \) partition, a variable \( T \) is set to 1 iff \( p \)
Step 3: (b) Let $P$ be the perpendicular through the middle of the edge defined by points $S$ and $T$. Each processor computes the perpendicular distance of its point $R$ from the infinite line $P$ (use negative distances for points on one side of $P$ and positive distances for points on the other side). These distances form a quadratonic sequence and its maximum, $d_{\text{max}}$, and minimum, $d_{\text{min}}$, can be found in $O(1)$ time using row bus splitting.

(c) The minimum area, $A$, of the rectangle for row $i$ is $h^*(d_{\text{max}} - d_{\text{min}})$. Let this be stored in the $A$ value of $\text{PE}[i,1]$.

Step 4  [Determine overall minimum rectangle]
Compute the minimum of the $A$’s of PEs $[i,1]$, $1 \leq i \leq p$. This is done by forming the cross product of the $A$’s in a $p \times p$ sub array and comparing the two $A$’s in each PE.

\[\text{Figure 21 Minimum enclosing rectangle (continued from previous page)}\]

dominates the $k'$th point of $X_i$. $T$ is set to zero otherwise. The $T$’s in each $N^{1/2} \times N^{1/2}$ partition may be summed in $O(1)$ time using the RMESH ranking algorithm of [JENQ91a]. The result is $DX(p)$. The computation of $DX(p)$ is best done after Step 1 of Figure 22 as at this time the points of $S$ are in the needed order. $DY(p)$ may be computed in a similar way following Step 2. However when setting the value of $T$, we note that $T$ is to be set to 1 iff $p$ dominates both the $k'$th point of $Y_j$ and the $k'$th point is not in the same $X$ partition as $p$.

Let $M_{ij} = \sum_{u < i} \sum_{v < j} |S_{uv}|$. To compute $M_{ij}$, we first sort the points of $S$ by their $Y$ partition index and within each $Y$ partition, they are sorted by the $X$ partition index. Following this, each processor in row 1 determines if the $X$ partition number of the point in the processor to its left is less than that of its own point. If so, then using its own column index and the $Y$ partition number of its point, it can compute $R_{ij} = \sum_{u < i} S_{uj}$, where $i$ and $j$ are, respectively, the $X$ and $Y$ partition numbers of its point ($R_{ij} = q - (j - 1)N^{1/2}$ where $q$ is the processor’s column index). Finally, $M_{ij} = \sum_{v < j} R_{iv}$. Before describing
Step 1: Partition $S$ into $N^{1/2}$ sets $X_i$, $1 \leq i \leq N^{1/2}$ such that $|X_i| = N^{1/2}$, $1 \leq i \leq N^{1/2}$, and no point in $X_i$ has a larger $x$-coordinate than any of the points in $X_{i+1}$, $1 \leq i \leq N^{1/2}$.

Step 2: Partition $S$ into $N^{1/2}$ sets $Y_j$, $1 \leq j \leq N^{1/2}$ such that $|Y_j| = N^{1/2}$, $1 \leq j \leq N^{1/2}$, and no point in $Y_j$ has a larger $y$-coordinate than any of the points in $Y_{j+1}$, $1 \leq j \leq N^{1/2}$.

Step 3: Let $S_{ij} = X_i \cap Y_j$ (see Figure 23). For each $p \in S_{ij}$, it is the case that
\[
D(p, S) = \# \text{ of points dominated by } p \text{ in } (Y_j - S_{ij}) + \# \text{ of points dominated by } p \text{ in } X_i + \sum_{u < i} \sum_{v < j} |S_{uv}| = DY(p) + DX(p) + \sum_{u < i} \sum_{v < j} |S_{uv}|.
\]
Compute $D(p, S)$ using this equation.

**Figure 22** Strategy to compute $D(p, S)$

**Figure 23** Partitioning of the points of $S$

To do this final summation, we provide, in Figure 24, the steps in the overall ECDF algorithm.
Step 1: Each point determines its X partition. This is done by sorting the points by x-coordinate.

Step 2: Compute $DX(p)$ for each point. This is done using an $N^{1/2} \times N^{1/2}$ block for each $p \in S$.

Step 3: Each point determines its Y partition. For this, sort all points by y-coordinate.

Step 4: Compute $DY(p)$ as in step 2.

Step 5: Sort $S$ by Y partition and within Y partition by X partition. Each $p \in S$ determines its $R_{ij}$ value where $i$ and $j$ are such that $p \in X_i$ and $p \in Y_j$.

Step 6: Each point $p$ determines $M_{ij} = \sum_{v \times j} R_{iv}$.

Step 7: Each point $p$ computes $D(p,S) = DY(p) + DX(p) + M_{ij}(p)$.

**Figure 24** RMESH algorithm to compute $D(p,S)$

Now, let us consider the computation of $M_{ij} = \sum_{v \times j} R_{iv}$. For each $i$, the $N^{1/2}$ $M_{ij}$ values ($1 \leq j \leq N^{1/2}$) are computed using the $i$’th $N \times N^{1/2}$ submesh. Initially, we have $R_{iv}$ stored in the $R$ variable of the PEs in the $v$’th $N^{1/2} \times N^{1/2}$ submesh of the $i$’th $N \times N^{1/2}$ submesh (Figure 25(a)). The steps required to compute $M_{ij}$ are given in Figure 26. Its correctness is easily verified and its complexity is $O(1)$.

5. Triangulation

In this section, we develop an $O(1)$ time algorithm to triangulate a set $S$ of $N$ planar points. The algorithm is for an $N \times N$ RMESH, an $N \times N$ PARBUS and an $N \times N$ MRN. For simplicity, we assume that the $N$ points have distinct x-coordinates and that no four are cocircular. Our algorithm is easily modified to work when these assumption do
not hold. The overall strategy is given in Figure 27. We begin, in step 1, by partitioning the N points into $N^{2/3}$ sets, $S_i$, each of size $N^{1/3}$. For this, we assume $N$ is a power of 3. The partitioning is done by $x$-coordinate. Each $S_i$ contains points that are to the left of those in $S_{i+1}$, $1 \leq i < N^{2/3}$. This partitioning is possible because of our assumption that the $x$-coordinates are distinct. To accomplish the partitioning, the $N$ points are sorted by $x$-coordinate using the $O(1)$ sorting algorithm of [NIGA92].

In step 2, the Voronoi diagram of each $S_i$ is computed. For this, each point $P \in S_i$ computes its Voronoi polygon which defines all points in the plane that are closer to $p$ than to any of the other points in $S_i$. As noted in [PREP85], for any two points $p$ and $q$, the set of points that are closer to $p$ than to $q$ is the half plane containing $p$ that is defined by the perpendicular bisector of the line joining $p$ and $q$. The Voronoi polygon, $V(p)$, (i.e., the polygon whose interior is the set of points closer to $p$ than to any other point in $S_i$) is defined by the intersection of the $N^{1/3} - 1$ half planes obtained by considering all $q \in S_i - \{p\}$. $V(p)$ is comprised of portions of at most $N^{1/3} - 1$ perpendicular bisectors. Consider the five points $\{a, b, c, d, e\}$ of Figure 28 and suppose we are determining which portion of the perpendicular bisector $\alpha\beta$ of $ab$ is a boundary of $V(a)$. The perpendicular bisector of $ac$ intersects $\alpha\beta$ at $f$. Since points in the half plane above the perpendicular bisector of $ac$ are closer to $c$ than to $a$, only the portion of $\alpha\beta$ with $x$-coordinate $\geq f.x$ ($f.x$ is the $x$-coordinate of $f$) may contribute to $V(a)$. Similarly, since the
Step 1: [Decompose $R$’s using the radix $N^{1/4}$]

PE[1, 1] of each of the $N^{1/2} \times N^{1/2}$ submeshes computes $a$ and $b$ such that $R = a \star N^{1/4} + b$ and $0 \leq b < N^{1/4}$. Since $|Y_v| = N^{1/2}$, each $R_{iv} \leq N^{1/2}$. Hence, $0 \leq a \leq N^{1/4}$.

Step 2: [Prefix sum the $a$’s]

(a) The $i$’th $N \times N^{1/2}$ submesh is partitioned into $N^{1/4} \times N^{1/2}$ submeshes. In each of these, the $N^{1/4}$ $a$ values contained are routed to the row 1 processors such that the $k$’th group of $N^{1/4}$ such processors contain the $k$’th $a$ value (Figure 15(b)).

(b) In each $N^{3/4} \times N^{1/2}$ submesh, the unary representation of the $a$’s is computed. For this, the $r$’th processor in an $N^{1/4}$ sets its $D$ value to 1 if $r$ is less than or equal to its $a$ value. Otherwise, it sets $D$ to zero.

---

**Figure 26** Computation of $M_{ij}$ (continued on next page)

perpendicular bisector of $ad$ intersects $\alpha\beta$ at $g$, only the portion of $\alpha\beta$ with $x$-coordinate $\geq g.x$ may contribute to $V(a)$. Finally, since the perpendicular bisector of $ae$ intersects $\alpha\beta$ at $h$ and since points below (or to the right) of this bisector are closer to $e$ than to $a$, only the portion of $\alpha\beta$ with $x$-coordinate $\leq h.x$ can contribute to $V(a)$. So, by looking at the three other perpendicular bisectors, we see that only the portion of $\alpha\beta$ that has $x$-coordinate $\geq mx = \max \{f.x, g.x\}$ and $\leq mn = h.x$ is a boundary of $V(a)$. Note that if $mx \leq mn$, then $\alpha\beta$ does not contribute to $V(a)$.

This strategy for step 2 of Figure 17 may be implemented on an RMESH or PARBUS. First, the points of $S_i$ are broadcast to all rows of the $N \times N^{1/3}$ submesh using column buses. Next, we compute $V(p)$ for each point $p \in S_i$ using $N^{2/3} \times N^{1/3}$ partitions of the $N \times N^{1/3}$ partition. The $j$’th such partition (from the top) is used to compute $V(p)$ for the $j$’th point $p \in S_i$. For this, the $N^{2/3} \times N^{1/3}$ partition is further partitioned into $N^{1/3} \times N^{1/3} \times N^{1/3}$ partitions as in Figure 29. The $k$’th $N^{1/3} \times N^{1/3}, A_k$, is used to determine the portion of the perpendicular bisector of $pp_k$ (where $p_k$ is the $k$’th point of $S_i$) that contributes to $V(p), 1 \leq k \leq N^{1/3}, p_k \neq p$. For this, the points $p$ and $p_k$
Step 2: (c) The one’s in row 1 of the $N^{3/4} \times N^{1/2}$ submesh are prefix summed using the RMESH ranking algorithm of [JENQ91a]. Let the overall sum for each group of $N^{1/4}$ a’s be stored in variable $A$ of the [1,1] PE of the $N^{3/4} \times N^{1/4}$ submesh.

(d) [Prefix sum the $A$’s]
We now have $N^{1/4}$ $A$ values to be prefix summed. Each is in the range $[0, N^{1/2}]$. The $A$’s are decomposed using radix $N^{1/4}$ such that $A = c \times N^{1/4} + d$ where $0 \leq d < N^{1/4}$. The $N^{1/4}$ $c$’s may be prefix summed using an $N^{1/2} \times N^{1/2}$ submesh by routing them to row 1 of any $N^{1/2} \times N^{1/2}$ submesh, computing their unary representation (as in (b) above), and ranking (as in (c) above). The $d$’s are prefix summed similarly. By adding together corresponding pairs of prefix sums of $c$’s and $d$’s, the prefix sums of the $A$’s are obtained.

(e) [Obtain prefix sum of $a$’s] The prefix sum of the $a$’s is obtained by adding together the prefix sum of the $a$’s in each $N^{1/4}$ group (as computed in (c)) and appropriate prefix sum of $A$ (as computed in (d)).

Step 3: [Prefix sum the $b$’s] This is similar to Step 2.

Step 4: [Obtain $M_{ij}$] $M_{ij}$ is the sum of the $a$ and $b$ prefix sums computed in step 3 and step 4 for the $j$’th (from the bottom) $N^{1/2} \times N^{1/2}$ submesh of the $i$’th $N \times N^{1/2}$ submesh.

---

**Figure 26** Computation of $M_{ij}$ (continued from previous page)

are broadcast to all processors $A_k$. Each processor then computes the perpendicular bisector $\alpha\beta$ of $pp_k$ as well as that of $pq$ where $q$ is the point initially in the processor. The processors initially containing $p$ or $p_k$ are excluded. The intersection between $\alpha\beta$ and the bisector of $pq$ is obtained. Let the $x$-coordinate of this intersection be $\gamma$. The processor now sets itself to $\leq \gamma$ or $\geq \gamma$ depending on which portion of $\alpha\beta$ has not been eliminated. In case $\alpha\beta$ is parallel to the bisector of $pq$ and on the same side of $p$, then the processor
Step 1: Partition $S$ into $N^{2/3}$ sets, $S_i$, each containing $N^{1/3}$ points. The partitioning is done such that all points in $S_i$ are to the left of those in $S_{i+1}$, $1 \leq i < N^{2/3}$.

Step 2: Using an $N \times N^{1/3}$ submesh for each $S_i$, compute the Voronoi diagram for $S_i$, $1 \leq i \leq N^{2/3}$.

Step 3: Compute the straight-line dual of each of the $N^{2/3}$ Voronoi diagrams of step 2.

Step 4: Partition the region not in the union of the convex hulls of the $S_i$’s into basic polygons and triangulate these.

---

**Figure 27** Triangulating $N$ planar points

---

**Figure 28** Determining portion of $V(a)$ contributed by $\alpha \beta$
sets itself to $\geq \infty$, if $\alpha \beta$ is farther from $p$ than the bisector of $pq$ and to $\leq \infty$ otherwise. Following this, the maximum $mx$ of the $\geq$ values and the minimum $mn$ of the $\leq$ values is found using all processors of $A_k$. If $mn < mx$ then $\alpha \beta$ contributes to $V(p)$ and the contributing endpoints together with the associated vertices $p$ and $p_k$ are saved in processor $[1,1]$ of $A_k$.

The purpose of computing the straight line dual in step 3 is to obtain the triangulation of the regions defined by the convex hull of each of the $S_i$’s. Theorem 5.11 of [PREP85] states that the straight line dual of the Voronoi diagram of $S$ is a triangulation of $S$ (Strictly speaking, this is true only if no four points of $S$ are cocircular. However, when four or more points are cocircular, the completion of the triangulation is easy once the dual has been obtained). The straight line dual of the Voronoi diagram of $S_i$ is easy to obtain. Suppose that $(u,v)$ is an edge of this diagram. $(u,v)$ is a portion of some perpendicular bisector. Let the points of $S_i$ associated with this bisector be $a$ and $b$. Then, $(a,b)$ is an edge of the straight line dual. Since, in the computation of the Voronoi diagram (step 3 of Figure 17), we associate with each bisector the two points of $S_i$ that generate the perpendicular bisector, the edges of the dual are easily generated.

When implementing steps 2 and 3, it is best to combine them so that when we detect $mn < mx$ (in step 2), the edge $(p, p_k)$ is generated provided $p.x > p_k.x$ (this avoids the generation of two copies of each edge of the dual). Since the number of edges in the triangulation of $S_i$ is at most $3 \frac{N^{1/3}}{} - 6$, these edges may be routed to the $N^{1/3}$ row 1

Figure 29 $N^{1/3} \times N^{1/3}$ partitions
processors in $O(1)$ time. Each processor in row 1 gets at most 3 edges.

Following step 3, the regions enclosed by the convex hulls of the $S_i$’s, has been triangulated (Figure 30) and we need to triangulate the region outside. For this, we use the technique of Wang and Tsin [WANG87]. The exterior region is partitioned into ‘simple polygons’ (Figure 30) which are then triangulated independently. The algorithm of [WANG87] to partition the external region is given in Figure 31. This has been modified to account for the fact that we have $N^{2/3}$ $S_i$’s while Wang and Tsin have only $N^{1/2}$.

Step 1 is easily done in $O(1)$ time using an $N \times N^{1/3}$ submesh for each $S_i$ using the $N^3$ processor algorithm of Figure 2. Step 2 can be done in $O(1)$ time using an $N^{1/3} \times N^{1/3}$ submesh for each $S_j, j < i$. The least slope $U_{ij}, j < i$ can be found by first finding the least slope $U_{ij}$ in each group of $N^{1/3} U_{ij}$’s. This leaves us with at most $N^{1/3}$ candidates from which the least slope one can be found. The computation is confined to one $N \times N^{1/3}$ submesh for each $S_i$. The leftmost and rightmost points of $S_i$ are easily identified in $O(1)$ time. This may be done by using an $N^{1/3} \times N^{1/3}$ submesh for each $S_i$ and comparing each point of $S_i$ against each other point or by using the convex hull of $S_i$ and comparing adjacent points in the hull. Hence, step 4 is also easy to do in $O(1)$ time.

Wang and Tsin [WANG87] have shown that the algorithm of Figure 32 correctly identifies the exterior polygon to which each point of the convex hull of the $S_i$’s belongs. In this algorithm, $l_i$ and $r_i$, respectively, denote the leftmost and rightmost points of $CH(i)$; $a_i$ and $b_i$ denote the end points of $U_i$ with $b_i$ being a point of $CH(S_i)$ and $a_i$ being a point of $CH(j)$ for some $j < i$; and $B(u)$ denotes the exterior polygon to which point $u$ belongs. The portion of the upper convex hull of $S_i$ that is comprised of the hull points from $b_i$ to $l_i$ (counterclockwise) defines a unique exterior polygon (see Figure 30).
Step 1: Compute the convex hull $CH(i)$ of each $S_i$, $1 \leq i \leq N^{2/3}$.

Step 2: For each $S_i$, determine the upper, $U_{ij}$, and lower, $L_{ij}$, tangents of $S_i$ with respect to $S_j$, $1 \leq j < i \leq N^{2/3}$.

Step 3: For each $S_i$, determine the upper tangent $U_i$ that has the least slope among all $U_{ij}$, $j < i$ and the lower tangent $L_i$ that has maximum slope among all $L_{ij}$, $j < i$.

Step 4: For each $S_i$, $1 < i \leq N^{2/3}$ determine the line $M_i$ that joins the rightmost point of $CH(i-1)$ and the leftmost point of $CH(i)$.

Step 5: Identify the exterior polygons formed by the $U_i$'s, $L_i$'s, $M_i$'s, and the boundaries of the convex hull of the $S_i$'s. There are exactly $2N^{2/3} - 2$ such polygons.

**Figure 31** WANG’s Algorithm [WANG87] to partition the exterior of the $CH(i)$’s into polygons

This polygon is called $P_i$. While it is possible for a point $u$ to belong to more than one $P_i$, $B(u)$ is defined to be the largest $j$ such that $u$ is a point of $P_j$. Note that points on the lower hull of the $S_i$’s are on polygons bounded from above by the $M_i$’s. These are labeled $Q$’s in Figure 30. $B(u)$ may be similarly defined for these points.

To implement the algorithm of Figure 32, it is best to identify the $a_i$, $b_i$, $r_i$, $l_i$ vertices associated with each $S_i$ during the computation of $u_i$ and $l_i$ in step 3 of Figure 31 and during the computation of $M_i$ in step 4 of Figure 31. Step 1 of Figure 32 is easily performed in $O(1)$ time using an $N \times N^{1/3}$ submesh for each $S_i$. For step 2, we note that there is exactly one $b_i$ associate with each $S_i$, $i > 1$. The corresponding $a_i$ can be stored along with it when $U_i$ is computed in Figure 31. This results in a pair $(a_i, b_i)$ stored in a single processor. To compute $B(a_i)$, we need to accumulate in the $N \times N^{1/3}$ submesh associated with $S_i$, all $(a_j, b_j)$ pairs for $j > i$. This can be done by having the PE that
{Find $B(u)$ for points in the upper hull of the $S_i$’s}

Step 1: For all points $u$ that are between their $b_i$ and $l_i$ when moving along $CH(i)$ counterclockwise from $b_i$ ($b_i$ excluded, $l_i$ included), set $B(u) = i$.

Step 2: If $u$ is an $a_i$, then $B(u) = \text{least } j \text{ such that } a_j.x < a_i.x < b_j.x$.

Step 3: If $u$ is a $b_i$, then $B(u)$ is computed similar to the step 2 computation of $B(a_i)$.

Step 4: For $u = r_i$, set $B(u) = i + 1$.

Step 5: Let $b_i \rightarrow r_i$ denote the segment of the upper hull of $CH(i)$, $1 \leq i \leq N^{2/3}$ with endpoints $b_i$ and $r_i$. Sort $b_i$, $r_i$, and all $a_j$’s in $b_i \rightarrow r_i$ into ascending order by $x$-coordinate.

Step 6: Let $b_i = a_{i_1}, a_{i_2}, ..., a_{i_t} = r_i$ be the sorted sequence of $a_j$’s in $b_i \rightarrow r_i$. For each $u \neq a_{i_j}$, $1 \leq j \leq t$ in $b_i \rightarrow r_i$, set $B(u) = B(a_{i_j})$ iff $a_{i_{j-1}}.x < u.x < a_{i_j}.x$.

{Find $B(u)$ for points in the lower hull of the $S_i$’s}
[Similar to steps 1-6]

---

**Figure 32** Algorithm to determine $B(u)$

contains ($a_i, b_i$) route the pair to the row $i$ processor on its column (using a column bus) and then broadcasting these pairs along row buses to the $N \times N^{1/3}$ submeshes that need them. The least $j$ that satisfies the inequality of step 2 is now easily found in $O(1)$ time (in each $N \times N^{1/3}$ submesh). Step 3 is similar to step 2 and step 4 is trivially done in $O(1)$ time.

The segment $b_i \rightarrow r_i$ of the upper hull of $CH(i)$ is easily identified from $CH(i)$ in $O(1)$ time for each $S_i$. All points on this segment that are $a_j$’s have been identified during
step 3 of Figure 31. The sort of step 5 can therefore be done in $O(1)$ time using an $N^{2/3} \times N^{1/3}$ submesh of the $N \times N^{1/3}$ submesh that corresponds to each $S_i$. Following the sort, each point of $b_i \rightarrow r_i$ that is not an $a_j$ can determine its $u$ value in $O(1)$ time using a row of $N^{1/3}$ processors. For this, the sorted $a_i$'s are broadcast down column buses of each $N \times N^{1/3}$ submesh and the unique $j$ satisfying the inequality of step 6 is found using row bus splitting. Once the $B(u)$'s have been computed, the convex hull points of all the $S_i$'s are sorted by their $B(u)$ values. This brings points in the same polygon next to each other. The sort is accomplished in $O(1)$ time using the algorithm of [NIGA92]. To triangulate each polygon, it is desirable to have the polygon points ordered by the $S_i$'s from which they came and within $S_i$, ordered in convex hull order. This can be incorporated into the sort by $B(u)$. Each polygon can now be triangulated using an $N \times d_i$ (where $d_i$ is the size of the polygon) submesh. This triangulation is fairly straightforward as each polygon is comprised of two monotone segments. The steps are given in [WANG87] for a PRAM and are easily performed on an RMESH or PARBUS of size $N \times d_i (N \geq d_i)$ in $O(1)$ time.

6. Conclusions

We have developed constant time RMB algorithms for the convex hull, the smallest enclosing rectangle, ECDF searching, and triangulation problems. All of our algorithms are for the case of $N$ planar points and all work on $N \times N$ RMBs. The algorithms apply to all the three models (RMESH/PARBUS/MRN).

7. References


