Constant Time Convex Hull And Enclosing Box On Reconfigurable Meshes With Buses*

Madhusudan Nigam and Sartaj Sahni
University of Florida
Gainesville, FL 32611

ABSTRACT

We develop O(1) time algorithms to compute the convex hull and the smallest enclosing box of a set of planar points. 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, reconfigurable mesh with buses.

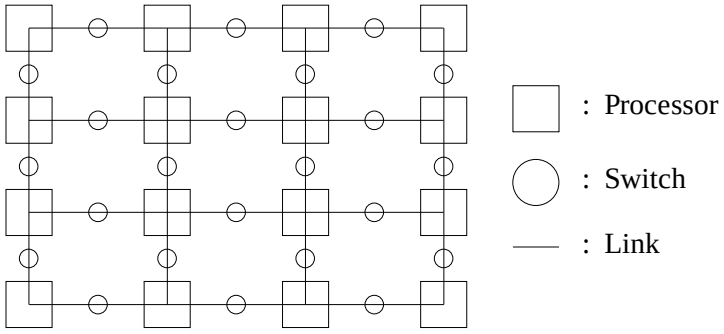
1 Introduction

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 ?F{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. Column buses are obtained by having each processor disconnect the switch on its east. 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.,

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

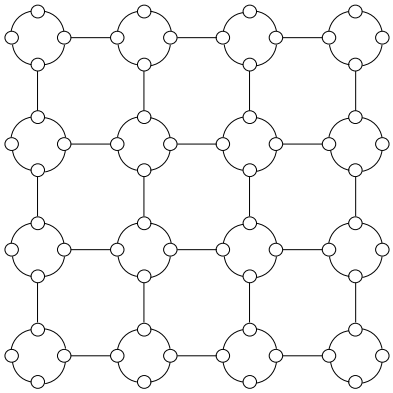
arbitrary, maximum, exclusive or, etc.). Notice that in the RMESH model it is not possible to simultaneously have n disjoint row buses and n disjoint column buses that, respectively, span the width and height of the RMESH. It is assumed that processors on the same bus can communicate in O(1) time. RMESH algorithms for fundamental data movement operations and image processing problems can be found in [MILL88abc, MILL91ab, JENQ91abc].



?F{1} 4×4 RMESH

An $n \times n$ PARBUS (?F{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 proccessor are done by an internal four port switch. If the upto 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. If A = {{N,S,E,W},ϕ}, 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 (?F{7}). In a reconfigurable network (RN) [BENA91] no restriction is placed on the bus segments that connect pairs of



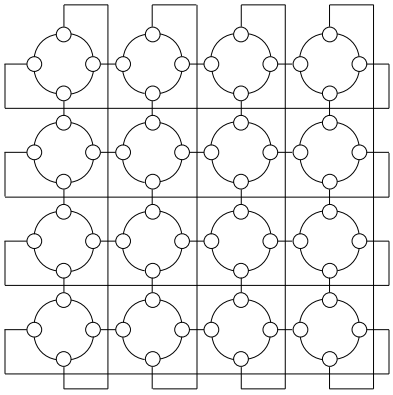
?F{4} 4 x 4 PARBUS

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 ?F{8}. Thus an MRN is a restricted PARBUS.

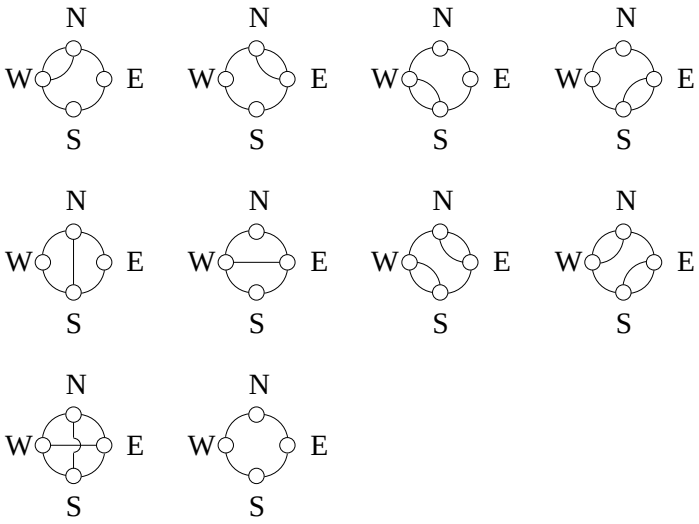
While we have defined the above reconfigurable bus architectures as square two dimensional meshes, it is easy to see how these may be extended to obtain non square architectures and architectures with more dimensions than two.

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]. Unfortunatley, 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



?F{7} 4 x 4 Polymorphic torus



?F{8} Local Configurations allowed for a switch in MRN

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.

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 ?F{9}. The input N points are in row 1 of the $N \times N$ PARBUS/RMESH/MRN. The algorithm of ?F{9} can be used to solve, in constant time, the k-dimensional maxima problem for any k.

In section 3, we consider the smallest enclosing rectangle problem. While this problems has 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 rectangle of a set of N planar points in $O(N^{1/2})$ on an $N \times N$ mesh. 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 problems are considered in [JEON90, LU85,86 and WANG87].

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 taht are tailored to the PARBUS/MRN architecture.

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

---

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}

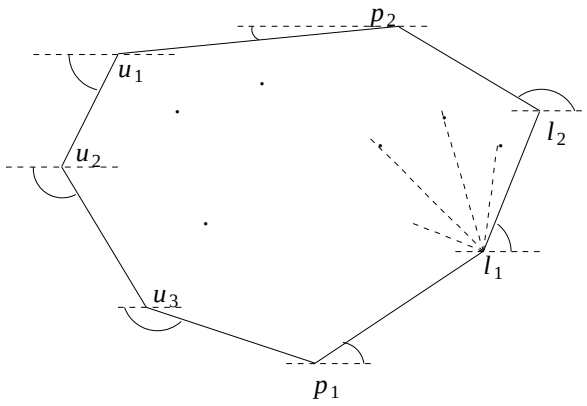
Using row buses PE$[i,1]$ sends its U value to PE$[i,i]$. Using column buses, PE$[i,i]$ sends the U value just received to PE$[1,i]$, $1 \leq i \leq N$.

?F{9} Simple Constant time algorithm for 3-dimensional maxima

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 $u_1, u_2, \ldots, u_j$ 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 ?F{10}). □



?F{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 ?F{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, $1 \le i \le N^2$ 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 $CH(S)$.

Step 1: Use column buses to broadcast the points of S from row 1 to all other rows.

Step 2: In row $i = (j-1)N + k$, $1 \le i \le N^2$, 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], $i = (j-1)N + k$, $1 \le i \le N^2$, 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.

Step 6: If PE[i,1], $i = (j-1)N + k$, $i \le \le N^2$, $j > k$ 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 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].

?F{11} O(1) $N^2 \times N$ processor algorithm for $E(S)$ and $CH(S)$

The correctness of ?F{11} 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.

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 ?F{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 ?F{13}.

R0: Sort S by x-coordinate and within x by y-coordinate.

R1: Partition S into $N^{1/2}$ subsets, S_i, $1 \le i \le N^{1/2}$ 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,$1 \le i \le N^{1/2}$.

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

R3: Obtain the convex hull of S from E.

?F{12} $N \times N$ RMESH algorithm for convex hull

Theorem 1: The extreme points algorithm of ?F{13} is correct.

Proof: To establish the correctness of ?F{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 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 ?F{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

An alternative to using the polar angle is discussed on p100 of [PREP85].

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.

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 ?F{14}).

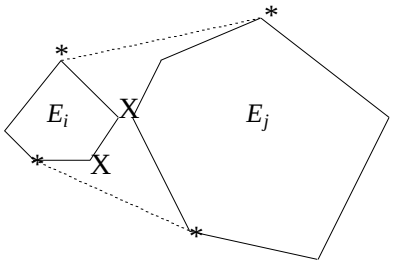
T6: The tangents of T5 define up to 4 tangent points. Over all R_{ji}, $1 \le i \le N^{1/2}$, there are $3\,N^{1/2} - 2$ non eliminated tangent points plus non eliminated points of S_i. The extreme points of this set of points is computed. Tangent points of E_i that are not in the set of extreme points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

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

?F{13} Substeps for step R2

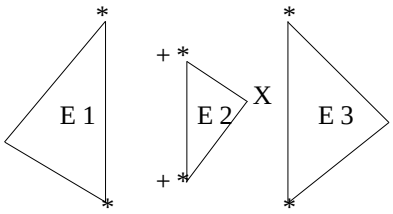
tangents from q to S_i, then p lies between the tangents of S_i and S_j and so is eliminated in step T5 (see ?F{17}). Since $p \in Q$, p is not eliminated in T5 and hence must be on a tangent from q to S_i (?F{18}). Furthermore, if p is not a tangent point with respect to S_i and S_j, then p lies between 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,



Tangents = Broken lines
* = tangent points of E_i and E_j
X = points of E_i eliminated

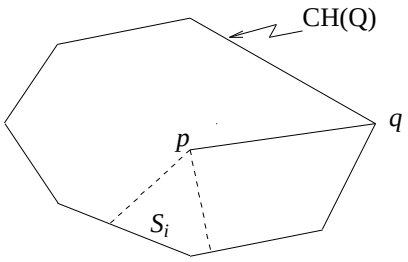
?F{14} Examples of extreme points eliminated in step T5



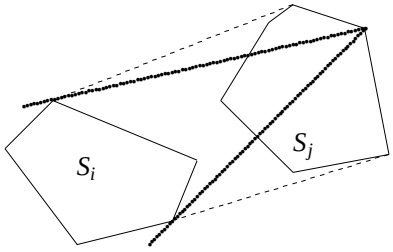
i = 2
*= tangent points
X = extreme points of E2 eliminated in step T5
\+ = extreme points of E2 eliminated in step T6

?F{15} Example of extreme point elimination in step T6

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 ?F{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, 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



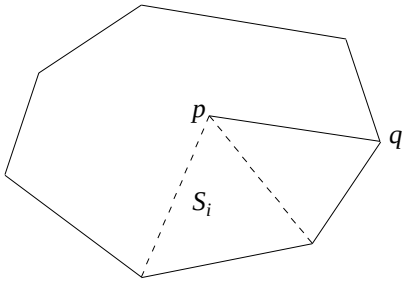
?F{16} Example for correctness proof



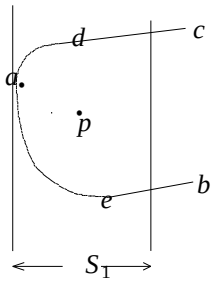
?F{17} Tangents

(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 $p \in S_i$, $i \notin \{1, N^{1/2}\}$. 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 (?F{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



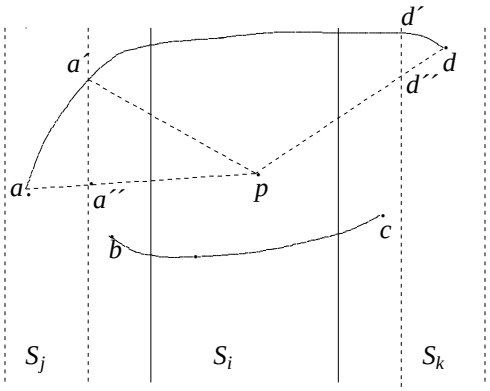
?F{18} p is on tangent from q



?F{19} $p \in S_1$

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 ?F{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 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 $C = D$). □



?F{20} $p \in S_i, i \notin \{1, N^{1/2}\}$

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 ?F{11}.

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 ?F{21}. Its correctness is immediate and one readily sees that its complexity is O(1) on an $N \times N$ RMESH, PARBUS and MRN.

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) PE[i,i] broadcasts its R value using a row bus to the S variable of all processors on row i, $1 \le i \le p$, p = number of convex hull points.

(c) PE[i,$i+1$] broadcasts its R value using a row bus to the T variable of all procesors on row i, $1 \le i \le p-1$. For $i = p$, 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.

(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, $d_{\max}$, and minimum, $d_{\min}$, can be found in O(1) time using row bus splitting.

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

Step 4 [Determine overall minimum rectangle]

Compute the minimum of the A´s of PEs [i,1], $1 \le i \le p$. 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.

?F{21} Minimum enclosing rectangle

4 Conclusions

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

5 References

[BENA91] Y. Ben-Asher, D. Peleg, R. Ramaswami, and A. Schuster, "The power of reconfiguration," *Journal of Parallel and Distributed Computing*, 13, 139-153, 1991.

[FREE75] H. Freeman and R. Shapira, "Determining the minimal-area encasing rectangle foran arbitrary closed curve," *Communications of ACM*, 18, 409-413, 1975.

[JANG92a] J. Jang and V. Prasanna, "An optimal sorting algorithm on reconfigurable meshes," *Proceddings 6th International Parallel Processing Symposium*, IEEE Computer Society, Los Alamitos, CA, 130-137, March 1992.

[JANG92b] J. Jang, and V. K. Prasanna, "A fast sorting algorithm on higher dimension reconfigurable mesh," *Proceedings 26th Conference on Information Sciences and Systems*, 1992.

[JANG92c] J. Jang, H. Park, and V. K. Prasanna, "A fast algorithm for computing histogram on reconfigurable mesh," Department of EE-Systems, Technical Report IRIS-290, University of Southern California, LosAngeles, Feb 1992.

[JANG92d] J. Jang, H. Park, and V. K. Prasanna, "An optimal multiplication algorithm on reconfigurable mesh," Department of EE-Systems, Technical Report IRIS-291,University of Southern California, LosAngeles, March 1992.

[JANG92e] J. Jang and V. K. Prasanna, "Efficient Parallel Algorithms for Some Geometric Problems on Reconfigurable Mesh," *Proceedings of 1992 International Conference on Parallel Processing*, CRC Press, Boca Raton, FL, 127-130, Aug 1992.

[JENQ91a] J. Jenq and S. Sahni, "Reconfigurable mesh algorithms for image shrinking, expanding, clustering, and template matching," *Proceedings 5th International Parallel Processing Symposium*, IEEE Computer Society Press, Los Alamitos, CA, 208-215, 1991.

[JENQ91b] J. Jenq and S. Sahni, "Reconfigurable mesh algorithms for the Hough transform," *Proc. 1991 International Conference on Parallel Processing*, CRC Press, Boca Raton, FL, 34-41, 1991.

[JENQ91c] J. Jenq and S. Sahni, "Reconfigurable mesh algorithms for the area and perimeter of image components," *Proc. 1991 International Conference on Parallel Processing*, CRC Press, Boca Raton, FL, 280-281, 1991.

[JEON90] C.S. Jeong, and D.T. Lee, "Parallel Geometric Algorithms on Mesh-connected Computers," *Algorithmica*, 5, 155-177, 1990.

[LI89a] H. Li and M. Maresca, "Polymorphic-torus architecture for computer vision,"

IEEE Trans. on Pattern & Machine Intelligence, 11, 3, 133-143, 1989.

[LI89b] H. Li and M. Maresca, "Polymorphic-torus network," *IEEE Trans. on Computers*, C-38, 9, 1345-1351, 1989.

[LU85] M. Lu and P. Varman, "Solving geometric proximity problems on mesh-connected computers," *Proceedings 1985 Workshop on Computer Architecture, Pattern Analysis, Image Database Management*, 248-255, 1985.

[LU86] M. Lu, "Constructing the Voronoi diagram on a Mesh-Connected Computer," *Proceedings 1986 International Conference on Parallel Processing*, The Pennsylvania State University Press, University Park, PA, 806-809, 1986.

[MILL87b] R. Miller, V. K. Prasanna Kumar, D. Reisis and Q. Stout, "Parallel Computation on Reconfigurable meshes," Department of EE-systems, Technical Report IRIS#229, University of Southern California, LosAngeles, 1987.

[MILL88] R. Miller, and Q. F. Stout, "Efficient Parallel Convex Hull Algorithms," *IEEE Transactions on Computers, c-37, 12, 1605-1618, Dec 1988.*

[MILL89] R. Miller, and Q.F. Stout, "Mesh Computer Algorithms for Computational Geometry," *IEEE Transaction on Computers*, 38, 3, 321-340, 1989.

[MILL91a] R. Miller, V. K. Prasanna Kumar, D. Reisis and Q. Stout, "Efficient parallel algorithms for intermediate level vision analysis on the reconfigurable mesh", *Parallel Architectures and Algorithms for Image Understanding*, Viktor K. Prasanna ed., 185-207, Academic Press, NewYork, 1991

[MILL91b] R. Miller, V. K. Prasanna Kumar, D. Reisis and Q. Stout, "Image processing on reconfigurable meshes", *From Pixels to Features II*, H. Burkhardt ed., Elsevier Science Publishing, Amsterdam, 1991.

[NIGA92] M. Nigam, and S. Sahni, "Sorting n Numbers on $n \times n$ Reconfigurable Meshes with Buses, CIS, University of Florida, Technical Report TR-92-04, 1992

[PREP85] F.P. Preparata and M.I. Shamos, *Computational Geometry: An Introduction*, Springer-Verlag, New York, 1985.

[REIS92] D.I. Reisis, "An Efficient Convex Hull Computation on the Reconfigurable Mesh," *Proceedings 1992 International Parallel Processing Symposium*, IEEE Computer Society Press, Los Alamitos, CA, 142-145, 1992.

[WANG87] C.A. Wang, and Y.H. Tsin, "An O($\log n$) time parallel algorithm for triangulating a set of points in the plane," *Information Processing Letters*, 25, 1, 15-47, 1988.

[WANG90a] B. Wang and G. Chen, "Constant time algorithms for the transitive closure and some related graph problems on processor arrays with reconfigurable bus systems," *IEEE Trans. on Parallel and Distributed Systems*, 1, 4, 500-507, 1990.

[WANG90b] B. Wang, G. Chen, and F. Lin, "Constant time sorting on a processor array with a reconfigurable bus system," *Information Processing Letters*, 34, 4, 187-190, 1990.

[WEEM87] C. C. Weems, S. P. Levitan, A. R. Hanson, E. M. Riseman, J. G. Nash, and D. B. Sheu, "The image understanding architecture," COINS Technical Report 87-76, University of Massachusetts at Amherst, 1987.

[WEEM89] C. C. Weems, S. P. Levitan, A. R. Hanson, E. M. Riseman, J. G. Nash, and D. B. Sheu, "The image understanding architecture, " *International Journal of Computer Vision*, 2, 251-282, 1989.