Constant Time ECDF Search And Triangulation 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.

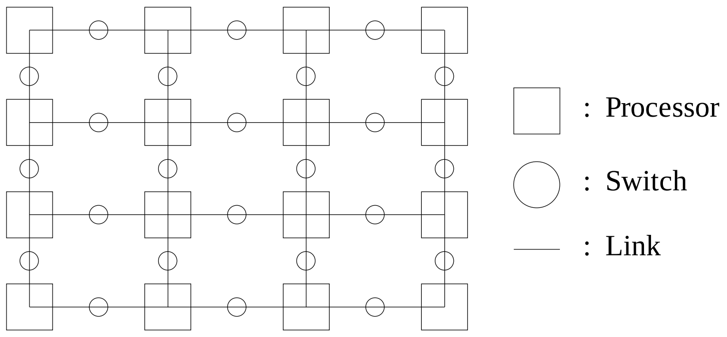
1 Introduction

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.

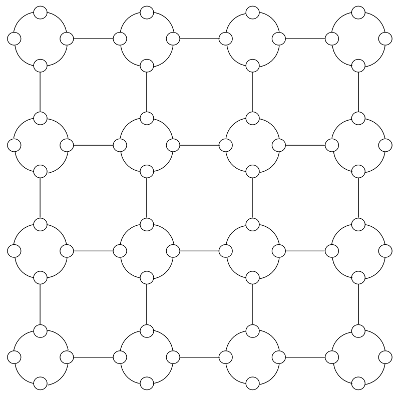
arbitrary, maximum, exclusive or, etc.). Notice that in the RMESH model it is not possible to simultaneously have *n* disjoint row buses and *n* disjoint column buses that, respectively, span the width and height of the RMESH. It is assumed that processors on the same bus can communicate in O(1) time. 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}, $1 \le i \le 2$ 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



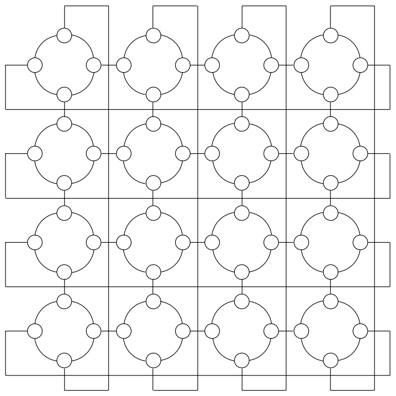
?F{4} 4 x 4 PARBUS

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.

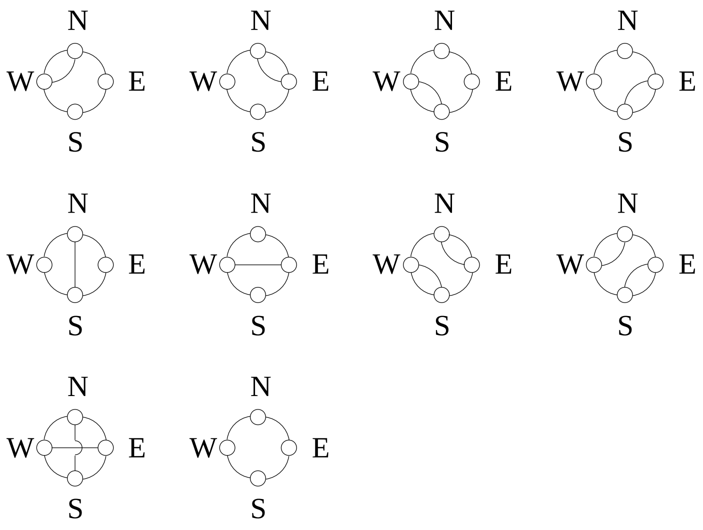
While we have defined the above reconfigurable bus architectures as square two dimensional meshes, it is easy to see how these may be extended to obtain non square architectures and architectures with more dimensions than two.

RMB algorithms for computational geometry problems has been explored in [MILL87, JANG92e, REIS92, and NIGA93]. 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 [NIGA93]. The algorithm uses an $N \times N$ configuration. 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



?F{7} 4 x 4 Polymorphic torus



?F{8} Local Configurations allowed for a switch in MRN 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 2, 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 $N \times N$ RMESH.

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. A simple constant time algorithm for this problem appears in [NIGA93]. This paper also contains a constant time algorithm for the smallest enclosing rectangle problem.

In section 3, we consider the problem of triangulating a set of N planar points. While this problem has not been considered before for the RMB architecture, parallel algorithms for other architectures have been developed. For example, Wang and Tsin [WANG87] develop an O(log$N$) time *CREW* PRAM algorithm to triangulate a set of N planar points.

2 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, 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 O(1) time. A high level description of the strategy is given in ?F{22}. For simplicity in presentation, we assume that no two points of S have the same x- or the same y-coordinates.

Step 1 can be done by sorting the N points in O(1) time by x-coordinate [NIGA92]. The point in 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

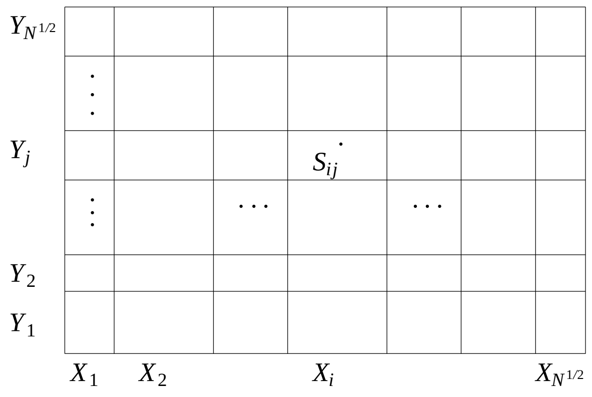
Step 1: Partition S into $N^{1/2}$ sets X_i, $1 \le i \le N^{1/2}$ such that $|X_i| = N^{1/2}$, $1 \le i \le N^{1/2}$, and no point in X_i has a larger x-coordinate than any of the points in X_{i+1}, $1 \le i \le N^{1/2}$.

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

Step 3: Let $S_{ij} = X_i \cap Y_j$ (see ?F{23}). For each $p \in S_{ij}$, it is the case that $D(p,S) =$ # of points dominated by p in $(Y_j - S_{ij})$ + # of points dominated by p 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.

?F{22} Strategy to compute $D(p,S)$



?F{23} Partitioning of the points of S

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 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 ?F{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 how to do this final summation, we provide, in ?F{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.

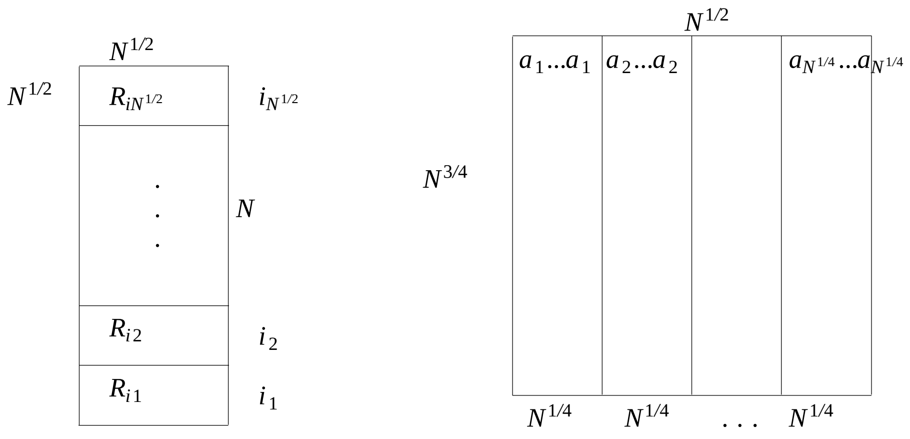
Step 5: Sort S by Y partition and within Y partition by X partition. Each $p \in S$ determines its R_{ij} value where i and j are such that $p \in X_i$ and $p \in Y_j$.

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

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

?F{24} RMESH algorithm to compute $D(p,S)$

Now, let us consider the computation of $M_{ij} = \sum_{v<j} R_{iv}$. For each i, the $N^{1/2}$ M_{ij} values $(1 \le j \le N^{1/2})$ 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 (?F{25}(a)). The steps required to compute M_{ij} are given in ?F{26}. Its correctness is easily verified and its complexity is O(1).



(a) i´th $N \times N^{1/2}$ submesh (b) $N^{3/4} \times N^{1/2}$ submesh

?F{25}

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 $R = a*N^{1/4} + b$ and $0 \le b < N^{1/4}$. Since $|Y_v| = N^{1/2}$, each $R_{iv} \le N^{1/2}$. Hence, $0 \le a \le N^{1/4}$.

Step 2: [Prefix sum the a´s]

(a) The i´th $N \times N^{1/2}$ submesh is partitioned into $N^{1/4}$ $N^{3/4} \times N^{1/2}$ 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.

(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.

?F{26} Computation of M_{ij} (continued on next page)

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 $A = c*N^{1/4} + d$ where $0 \le d < N^{1/4}$. 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.

?F{26} Computation of M_{ij} (continued from previous page)

3 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 ?F{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}, $1 \le i < N^{2/3}$. 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].

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}, $1 \le i < N^{2/3}$.

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

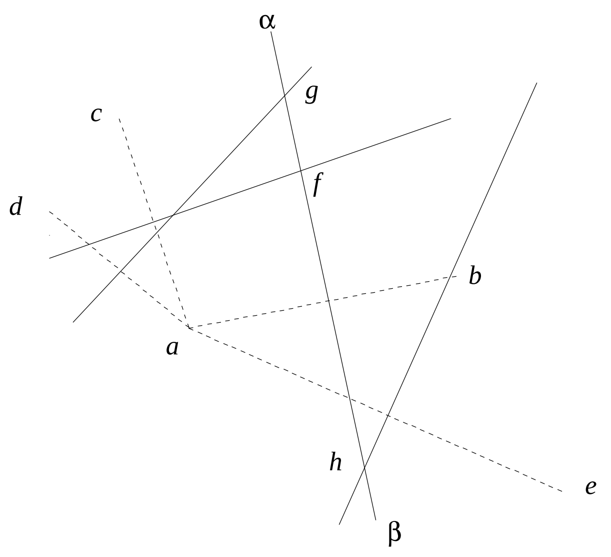
Step 3: Compute the straight-line dual of each of the $N^{2/3}$ Voronoi diagrams of step 2.

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

?F{27} Triangulating N planar points

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 ?F{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 $\ge f.x$ ($f.x$ is the x-coordinate of f) may contribute to $V(a)$. Similarly, since the perpendicular bisector of ad intersects $\alpha\beta$ at g, only the portion of $\alpha\beta$ with x-coordinate $\ge g.x$ 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 $\le h.x$ 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 $\ge mx = \max\ \{f.x,\ g.x\}$ and $\le m.n = h.x$ is a boundary of $V(a)$. Note that if $mx \le mn$, then $\alpha\beta$ does not contribute to $V(a)$.

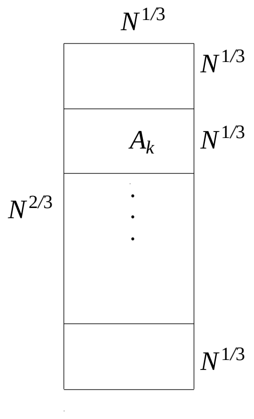
This strategy for step 2 of 17 may be implemented on an RMESH or PARBUS. First, the points of S_i are broadcast to all rows of the $N \times N^{1/3}$ submesh using column buses. Next, we compute $V(p)$ for each point $p \in S_i$ using $N^{2/3} \times N^{1/3}$ partitions of the $N \times N^{1/3}$ partition. The j´th such



?F{28} Determining portion of $V(a)$ contributed by $\alpha\beta$

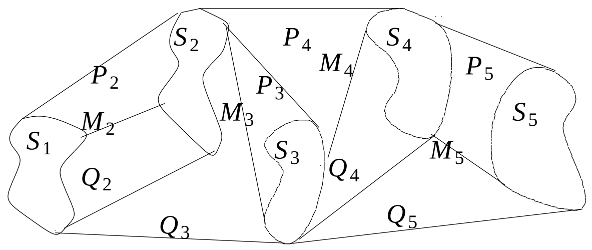
partition (from the top) is used to compute $V(p)$ for the j´th point $p \in S_i$. For this, the $N^{2/3} \times N^{1/3}$ partition is further partitioned into $N^{1/3}$ $N^{1/3} \times N^{1/3}$ partitions as in ?F{29}. The k´th $N^{1/3} \times N^{1/3}$, A_k, is used to determine the portion of the perpendicular bisector of pp_k (where p_k is the k´th point of S_i) that contributes to $V(p)$, $1 \le k \le N^{1/3}$, $p_k \ne p$. For this, the points p and p_k 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 γ. The processor now sets itself to $\le \gamma$ or $\ge \gamma$ 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 sets itself to $\ge \infty$, if $\alpha\beta$ is farther from p than the bisector of pq and to $\le \infty$ otherwise. Following this, the maximum mx of the $\ge$ values and the minimum mn of the $\le$ values is found using all processors of A_k. If $mn < mx$ then $\alpha\beta$ contibutes to $V(p)$ and the contributing end-points 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



?F{29} $N^{1/3} \times N^{1/3}$ partitions

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 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.



?F{30} Partitioning region exterior to convex hulls

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 $3N^{1/3} - 6$, these edges may be routed to the $N^{1/3}$ row 1 processors in O(1) time. Each processor in row 1 gets atmost 3 edges.

Following step 3, the regions enclosed by the convex hulls of the S_i´s, has been triangualted (?F{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' (?F{30}) which are then triangulated independently. The algorithm of [WANG87] to partition the external region is given in ?F{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: Compute the convex hull $CH(i)$ of each S_i, $1 \le i \le N^{2/3}$.

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

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, $1 < i \le N^{2/3}$ 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.

?F{31} WANG's Algorithm [WANG87] to partition the exterior of the $CH(i)$´s into polygons

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 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 ?F{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 ?F{30}). 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 ?F{30}. $B(u)$ may be similarly defined for these points.

{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 $B(u) =$ least j 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 $CH(i)$, $1 \le i \le N^{2/3}$ 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 $u \ne a_{i_j}$, $1 \le j \le t$ in $b_i \rightarrow r_i$, set $B(u) = B(a_{ij})$ iff $a_{ij-1}.x < u.x < a_{ij}.x$.

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

?F{32} Algorithm to determine $B(u)$

To implement the algorithm of ?F{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 ?F{31} and during the computation of M_i in step 4 of ?F{31}. Step 1 of ?F{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 ?F{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 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 satisties 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 ?F{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.

4 Conclusions

We have developed constant time RMB algorithms for the ECDF searching, and triangulation problems. Our algorithms are for all three 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.

[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.

[MILL87] 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.*

[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.