Theory Comput. Systems **30**, 547–558 (1997)

Theory of Computing Systems

A Randomized Parallel Three-Dimensional Convex Hull Algorithm for Coarse-Grained Multicomputers*

F. Dehne,[1] X. Deng,[2] P. Dymond,[2] A. Fabri,[3] and A. A. Khokhar[4]

[1]School of Computer Science, Carleton University,
Ottawa, Ontario, Canada K1S 5B6
dehne@scs.carleton.ca

[2] Department of Computer Science, York University,
North York, Ontario, Canada M3J 1P3
{deng,dymond}@cs.yorku.ca

[3]Department of Computer Science, Utrecht University,
3508 TB Utrecht, The Netherlands
andreas@cs.ruu.nl

[4]School of EE and Department of Computer Science, Purdue University,
West Lafayette, IN 47907, USA
ashfaq@cs.purdue.edu

Abstract. We present a randomized parallel algorithm for constructing the three-dimensional convex hull on a generic p-processor coarse-grained multicomputer with arbitrary interconnection network and n/p local memory per processor, where $n/p \geq p^{2+\varepsilon}$ (for some arbitrarily small $\varepsilon > 0$). For any given set of n points in 3-space, the algorithm computes the three-dimensional convex hull, with high probability, in $O((n \log n)/p)$ local computation time and $O(1)$ communication phases with at most $O(n/p)$ data sent/received by each processor. That is, with high probability, the algorithm computes the three-dimensional convex hull of an arbitrary point set in time $O((n \log n)/p + \Gamma_{n,p})$, where $\Gamma_{n,p}$ denotes the time complexity of one communication phase. The assumption $n/p \geq p^{2+\varepsilon}$ implies a coarse-grained, limited parallelism, model which is applicable to most commercially available multiprocessors.

In the terminology of the BSP model, our algorithm requires, with high probability, $O(1)$ supersteps, synchronization period $L = \Theta((n \log n)/p)$, computation cost $O((n \log n)/p)$, and communication cost $O((n/p)g)$.

* This work was partially supported by the Natural Sciences and Engineering Research Council of Canada and the ESPRIT Basic Research Actions No. 7141 (ALCOM II).

1. Introduction

1.1. *The Model*

Speedup results for theoretical PRAM algorithms do not necessarily match the speedups observed on real machines [6], [35]. Given sufficient slackness in the number of processors, Valiant's BSP approach [36] simulates PRAM algorithms optimally on distributed memory parallel systems. Valiant points out, however, that one may want to design algorithms that utilize local computations and minimize global operations. The BSP approach requires that g (= local computation speed/router bandwidth) is low, or fixed, even for an increasing number of processors. Gerbessiotis and Valiant [22] describe circumstances where PRAM simulations cannot be performed efficiently, among others if the factor g is high. Unfortunately, this is true for most currently available multiprocessors. The algorithm presented here considers this case for the three-dimensional convex hull problem. Furthermore, as pointed out in [36], the cost of a message also contains a constant overhead cost s. The value of s can be fairly large and the total message overhead cost can have a considerable impact on the speedup observed (see, e.g., [16]).

We therefore use a slightly enhanced version of the BSP model, referred to as a *coarse-grained multicomputer* model. It is comprised of a set of p processors $P_1, \ldots, P_p$ with $O(n/p)$ local memory per processor and an arbitrary communication network (or shared memory). Algorithms consist of alternating local computation and global communication rounds. Each communication round consists of routing a single h-relation with $h = \tilde{O}(n/p)$,[1] i.e., each processor sends $\tilde{O}(n/p)$ data and receives $\tilde{O}(n/p)$ data. We require that all information sent from a given processor to another processor in one communication round is packed into one message. In the BSP model, a communication round is equivalent to a superstep with communication cost $\tilde{O}((n/p)g)$.

Finding an optimal algorithm in the coarse-grained multicomputer model is equivalent to minimizing the number of communication rounds as well as the total local computation time. This considers all parameters discusssed above that affect the final observed speedup, and it requires no assumption on g. Furthermore, it has been shown that minimizing the number of supersteps also leads to improved portability across different parallel architectures [36], [18]. The above model has been used (explicitly or implictly) in parallel algorithm design for various problems [11], [15]–[17], [19], [27] and has shown very good practical timing results.

1.2. *The Three-Dimensional Convex Hull Problem*

For the three-dimensional convex hull problem studied in this paper, Amato and Preparata [5] give an almost work optimal deterministic NC^1 algorithm for the CREW PRAM. Chow [13] presented an $O(\log^3 n)$-time algorithm for the EREW PRAM as well as a polylogarithmic-time algorithm for the cube connected cycles interconnection network with distributed memory. Reif and Sen [34] were the first to give a time and work optimal randomized algorithm for the CREW PRAM. Goodrich *et al*. [25] adapted this algorithm

[1] $X = \tilde{O}(f(n))$ denotes $X = O(f(n))$ with high probability. More precisely, $X = \tilde{O}(f(n))$ if and only if $(\forall c > c_0 > 1)$ $\mathrm{prob}\{X \geq cf(n)\} \leq 1/n^{g(c)}$ where c_0 is a fixed constant and $g(c)$ is a polynomial in c with $g(c) \to \infty$ for $c \to \infty$ [31].

for an external memory model with an array of disks and multiple processors. For higher-dimensional convex hulls Amato *et al.* [4] presented $O(\log n)$-time algorithms with $O(n \log n + n^{\lfloor d/2 \rfloor})$ work. Dehne *et al.* [15] proposed an algorithm for the coarse-grained multicomputer model. The time complexity is $\tilde{O}((n \log n)/p + \Gamma_{n,p})$, assuming $n/p \geq p^{\varepsilon}$ ($\varepsilon > 0$) and uniform point distribution. Under this assumption, the algorithm performs only a constant number of communication phases. $\Gamma_{n,p}$ denotes the time complexity of one communication phase with at most $\tilde{O}(n/p)$ data sent/received by each processor.

1.3. *The Results*

In this paper we improve considerably on previous results in [15] and [17]. We present a randomized parallel algorithm for p-processor coarse-grained multicomputers with local memories of size $n/p \geq p^{2+\varepsilon}$ ($\varepsilon > 0$ arbitrarily small) and arbitrary interconnection network, which computes the convex hull of n arbitrary points in 3-space in time $\tilde{O}((n \log_2 n)/p + \Gamma_{n,p})$.

Every processor spends a total local computation time of $\tilde{O}((n \log n)/p)$. The algorithm uses only $\tilde{O}(1)$ global communication phases with at most $\tilde{O}(n/p)$ data sent/received by each processor.

With respect to [15], the algorithm presented here allows for an arbitrary input distribution. In particular, it allows for inputs created by mapping a two-dimensional Voronoi diagram problem to a three-dimensional convex hull problem [12] (which could not be handled in [15]). The techniques used in this paper are very different from the ones presented in [15] and [17]. The randomization methods presented are very different from the ones previously reported, e.g., in [34] and related papers.

Using Valiant's terminology for the BSP model [36], our algorithm requires, with high probability, only $O(1)$ supersteps, synchronization period $L = \Theta((n \log n)/p)$, computation cost $O((n \log n)/p)$, and communication cost $O((n/p)g)$. It is not obvious how to achieve $O(1)$ supersteps by techniques described in [25] and [34] or divide-and-conquer methods.

2. Outline of the Algorithm

Let S be a set of n points in 3-space (in general position).

Input: Each point of S is stored with exactly one processor. Processor p_i stores n/p points of S.
Output: Processor p_i stores a set E_i of $O(n/p)$ edges of $\mathrm{CH}(S)$, $1 \leq i \leq p$. Each edge of $\mathrm{CH}(S)$ is stored with exactly one processor.

1. Computing a sample convex hull
   All processors p_i compute globally a random sample $R \subset S$ of size $O(n/p)$.
   R is broadcast to all processors.
   Each processor computes the convex hull $\mathrm{CH}(R)$. We can assume without loss of generality that the origin lies in the interior of $\mathrm{CH}(R)$ and that each face is a triangle. The points inside $\mathrm{CH}(R)$ are removed.

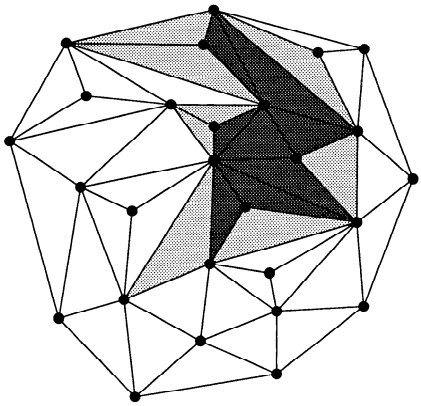

Fig. 1. A set R_i of faces (dark shading), and the set Q_i of faces which are edge-adjacent to faces of R_i (light shading).

2. Computing the generating sets
   With each face t of $\mathrm{CH}(R)$ associate a cone C_t and a closed half-space H_t. The cone C_t is defined by the origin as apex and rays starting at the origin and passing through the three vertices of t. The closed half-space H_t is defined by the plane through t and does not contain the origin. Let K_t and T_t denote the subsets of points of S that are not inside $\mathrm{CH}(R)$ but contained in C_t and H_t, respectively. (Note that K_t and T_t contain the vertices of t.)
   Partition the set of faces of $\mathrm{CH}(R)$ among the p processors. Let R_i be the subset of faces assigned to processor p_i, $1 \leq i \leq p$. Details on how the set of faces of $\mathrm{CH}(R)$ is partitioned is discussed in Section 3.2. Let Q_i be the subset of faces of $\mathrm{CH}(R)$ which are not in R_i but edge adgacent to at least one face of R_i, $1 \leq i \leq p$. See Figure 1 for an illustration. Define $S_i = (\bigcup_{t \in R_i} K_t) \cup (\bigcup_{t \in Q_i} T_t)$, $1 \leq i \leq p$.
   Processor p_i computes S_i and its convex hull $\mathrm{CH}(S_i)$, $1 \leq i \leq p$. The points inside $\mathrm{CH}(S_i)$ are removed.
3. Disconnecting internal edges
   Each processor p_i creates two copies of each edge (v, w) of $\mathrm{CH}(S_i)$, one with key v and one with key w. All these edges are sorted globally with respect to their endpoints.
   For each endpoint v and incident edge (v, w) consider the induced ray starting at v and passing through w. The convex hull $\mathrm{CH}(\mathcal{R}_v)$ of the set $\mathcal{R}_v$ of rays induced by the incident edges of v is computed. The incident edges inside $\mathrm{CH}(\mathcal{R}_v)$ are removed, and v is removed if v is inside $\mathrm{CH}(\mathcal{R}_v)$.
   Each remaining edge e is sent back to processor p_i if e was originally part of $\mathrm{CH}(S_i)$.
4. Selecting the global convex hull edges in each generating set
   (a) Each processor p_i computes for each triangle t of R_i the point of S_i with maximum distance to the plane defined by t. We refer to such a point as a *furthest point* in S_i.

(b) On each processor p_i let G_i be the subgraph of $\mathrm{CH}(S_i)$ induced by those edges that remain after Step 3. Each processor p_i computes the set E_i of edges of G_i reachable in G_i from a furthest point in S_i.

3. Algorithm Details and Analysis

We now discuss in more detail the steps outlined in the above algorithm and give a probablistic analysis. Let $k \in \mathbb{N}$ be a parameter to be determined later.

3.1. *Computing a Sample Convex Hull*

We need the following:

Lemma 1 [31]. *Consider a random variable X with binomial distribution. Let n be the number of trials, each of which is successful with probability q. The expectation of X is $E(X) = nq$.*

(a) $\mathrm{prob}\{X > cnq\} \leq e^{-(1/2)(c-1)^2 nq}$, *for* $c > 1$.
(b) *Let* $\beta \geq 4$ (*not necessarily fixed*). *If* $\beta^2 nq \geq \alpha \ \ln(n)$ *for some constant* $\alpha > 0$, *then* $X = \tilde{O}(\beta nq)$.

We now outline how to select a random sample $R \subset S$ of size $\tilde{O}(n/p)$. Each processor p_i performs a biased random coin flip for each point of S stored at p_i, such that each point is selected with probability pk/n. From the above lemma it follows that $|R| = \tilde{O}(pk)$ and that if $k = \Omega(\ln(n))$, then the number of points selected by a processor p_i is $\tilde{O}(k)$.

Since we store R at each processor, it is necessary that $|R| \leq \tilde{O}(n/p)$. Hence, we obtain:

Requirement 1. $k \leq n/p^2$.

In order to compute the convex hull $\mathrm{CH}(R)$ of R we apply any optimal sequential ($O((n/p)\log_2(n/p))$ time) three-dimensional convex hull algorithm [14], [33].

3.2. *Computing the Generating Sets*

In Step 2 of the algorithm we compute p subsets S_i of S, such that each edge of the convex hull $\mathrm{CH}(S)$ is generated in the computation of a convex hull $\mathrm{CH}(S_i)$. Before we give an algorithm for computing the generating sets S_i, we show that in Step 2 a superset of the set of convex hull edges is generated.

Lemma 2. *Every edge of* $\mathrm{CH}(S)$ *is an edge of at least one* $\mathrm{CH}(S_i)$, $1 \leq i \leq p$.

Proof. Let (v, w) be an edge in $\mathrm{CH}(S)$. Consider the line $l(v, w)$ parallel to (v, w) and contained in the plane defined by v, w and the origin such that $l(v, w)$ is tangent to the sample convex hull $\mathrm{CH}(R)$. Recall that the origin lies inside $\mathrm{CH}(R)$. The line

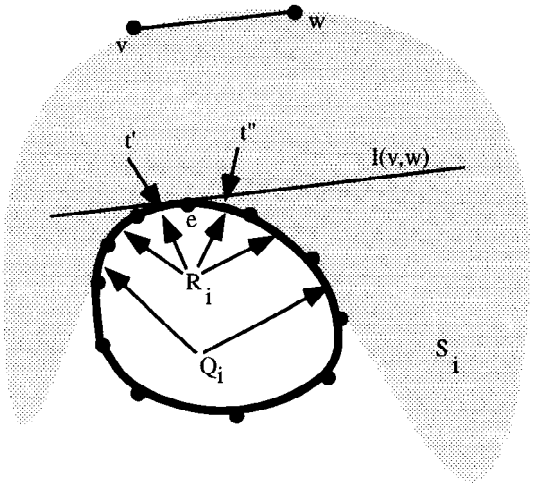

Fig. 2. Two-dimensional illustration of the proof of Lemma 2.

$l(v, w)$ intersects an edge e of CH(R) (if it intersects at a vertex we consider any edge incident to this vertex). See Figure 2 for a two-dimensional illustration (two-dimensional projection). Let t' and t'' be the faces of CH(R) adjacent to e. By construction there exists at least one R_i such that $\{t', t''\} \cap R_i \neq \emptyset$. It follows from the convexity of CH(R) that $(T_{t'} \cup T_{t''}) \subset S_i = (\bigcup_{t \in R_i} K_t) \cup (\bigcup_{t \in Q_i} T_t)$. Thus, $\{v, w\} \subset S_i$ and, hence, (v, w) is an edge of CH(S_i). □

We next discuss how we can construct p generating sets S_i. It is important to establish that no S_i contains more than $\tilde{O}(n/p)$ points of S.

For each face t of CH(R) we can easily compute $|K_t|$, the number of points in a cone C_t, by constructing a subdivision hierarchy [20] for the convex polyhedron CH(R). On this subdivision hierarchy we perform a ray-shooting query for each point $s \in S$, using a ray that starts at the origin and passes through s. Each query can be answered in time $O(\log|\text{CH}(R)|)$. The search structure is of size $O(|\text{CH}(R)|)$ and can be computed sequentially in time $O(|\text{CH}(R)|)$. Each point is contained in exactly one cone. Each processor performs the ray-shooting queries for its point set, and the p results for each of the n/p faces are added after a global sort of the faces.

Lemma 3.

(a) *For any fixed $c \geq 5$, with probability at least $1 - 1/n^{c-4}$ there exists no triangle t such that $|T_t| \geq c((n \ln(n))/pk)$.*
(b) *For any fixed $c \geq 5$, with probability at least $1 - 1/n^{c-4}$ there exists no triangle t such that $|K_t| \geq c((n \ln(n))/pk)$.*

Proof. (a) Consider a fixed $c \geq 5$. For some fixed triangle t, $\text{prob}\{|T_i| = j\} \leq (1 - pk/n)^j$ and

$$\begin{aligned}
\text{prob}\left\{|T_t| \geq c\frac{n\ln(n)}{pk}\right\} &\leq \sum_{j=c((n\ln(n))/pk)}^{n} \text{prob}\{|T_t| = j\} \\
&\leq \left(n - c\frac{n\ln(n)}{pk}\right)\left(1 - \frac{pk}{n}\right)^{c((n\ln(n))/pk)} \\
&= \left(n - c\frac{n\ln(n)}{pk}\right)\left(\left(1 - \frac{1}{n/pk}\right)^{n/pk}\right)^{c\ln(n)} \\
&\leq \left(n - c\frac{n\ln(n)}{pk}\right)e^{-c\ln(n)}.
\end{aligned}$$

As there are less than n^3 possible triangles,

$$\begin{aligned}
&\text{prob}\{\text{there exists a } t \text{ with } |T_t| \geq c((n\ln(n))/pk)\} \\
&\quad \leq n^3(n - c((n\ln(n))/pk))e^{-c\ln(n)}.
\end{aligned}$$

The claim follows from

$$\begin{aligned}
& n^3\left(n - c\frac{n\ln(n)}{pk}\right)e^{-c\ln(n)} \leq \frac{1}{n^{c-4}} \\
\Leftrightarrow \quad & n^{c-4}n^3\left(n - c\frac{n\ln(n)}{pk}\right) \leq e^{c\ln(n)} \\
\Leftrightarrow \quad & (c-4)\ln(n) + 3\ln(n) + \ln\left(n - c\frac{n\ln(n)}{pk}\right) \leq c\ \ln(n) \\
\Leftrightarrow \quad & c - 4 \leq \frac{1}{\ln(n)}(c\ln(n) - 4\ln(n)) = c - 4.
\end{aligned}$$

(b) Follows immediately, since $K_t \subset T_t$ for each face t of $\text{CH}(R)$. □

It is easy to partition the faces of $\text{CH}(R)$ into p subsets R_i such that, for each R_i, $\sum_{t\in R_i} |K_t| = O(n/p)$. However, a set R_i from such a partitioning can have $O(pk)$ edge-adjacent faces not in R_i and can induce a set S_i containing $O((n/p)\log_2 n)$ points. Our construction of the p sets R_i is based on the following separator theorem:

Lemma 4 [28]. *Let G be any m-vertex planar graph. Then the vertices of G can be partitioned in three sets V_1, U, and V_2, such that no edge joins a vertex in V_1 with a vertex in V_2, neither V_1 nor V_2 have total size exceeding $2m/3$, and separator U contains no more than $2\sqrt{2}\sqrt{m}/(1 - \sqrt{\frac{2}{3}})$ vertices. Furthermore V_1, U, and V_2 can be determined sequentially in time $O(m)$.*

We apply the separator theorem to the dual graph G of $\text{CH}(R)$. Define the cost of a node of G corresponding to a face t of $\text{CH}(R)$ as $|K_t|$. In order to partition the vertices of G into p subsets $W_1, \ldots, W_p$, we recursively apply the separator theorem

until the total node cost of every subgraph is at most n/p. This yields less than $2p$ subproblems and we pair them arbitrarily. Since $|K_t| \leq (cn\ln(n))/pk$, there are at most $l = \log_2 p + \log_2 \ln(n) + \log_2 c$ levels of recursion to reduce the total node cost of each subgraph to n/p. At the ith level, the total number of nodes in (up to 2^{i-1}) separators is bounded by $2^{i-1}d\sqrt{m/1.5^{i-1}}$, where $d = 2\sqrt{2}/(1-\sqrt{\frac{2}{3}})$ and $m = \theta(pk)$. The total number of nodes in all separators obtained in this procedure is bounded by $\sum_{i=1}^{l} 2^{i-1}d\sqrt{m/1.5^{i-1}} = O(p\sqrt{k\ln(n)})$.

Let B be the multiset of points contained in sets T_t for faces t corresponding to some separator, counting the multiple presence of points. The total number $|B|$ of such points is bounded by $O(p\sqrt{k\ln(n)})$ times $O((n\log n)/pk)$, which is $O((n\ln(n)\sqrt{\ln(n)})/\sqrt{k})$.

Requirement 2. $k \geq (l\ln(n)\sqrt{\ln(n)})^2$.

Choosing $k \geq (l\ln(n)\sqrt{\ln(n)})^2$, we obtain $|B| = O(n/l)$. A point is in B if and only if it is bounded away from the origin by one of T_t for some t in the separators. The complement of B is the set of points in the convex hull defined by all the half-spaces containing the origin and bounded by a hyperplane T_t, for all t in one of the separators. This convex hull can be constructed sequentially within time $O((n\log n)/p)$. In time $O(\log(|R|))$ we determine whether each point does not belong to B. Then we determine those points of B which correspond to the first level separator. All these points are assigned to S_i, $1 \leq 1 \leq p$. After removing these points from B, the remaining points in B can be partitioned into two parts: Those in sets T_t where t corresponds to a vertex in V_1 and those in sets T_t where t corresponds to a vertex in V_2. This partition can be obtained using a Kirkpatrick subdivision hierarchy for the convex hull $CONV(R)$ in $O(\log(|R|))$ time for each point in B. Therefore, for each point in B, the number of operations on each level is $O(\log n)$ and the total number of operations on l levels is $O(l|B|\log n)$, which is $O(n\log n)$. Since each point of B requires the same number of operations, they can be distributed over the p processors so that each processor needs $O((n\log n)/p)$ operations.

As the level of recursion increases from 1 to l, the number of adjacent faces decreases geometrically. The number of faces which are edge-adjacent to a face of $\mathrm{CH}(R)$ in R_i is bounded by

$$\sum_{j=1}^{l}\sqrt{(\tfrac{1}{2})^j|\mathrm{CH}(R)|} = \sqrt{|\mathrm{CH}(R)|}\sum_{j=0}^{l}\sqrt{(\tfrac{1}{2})^j} = O(\sqrt{|\mathrm{CH}(R)|}) = \tilde{O}(\sqrt{pk}).$$

Hence, it follows from Lemma 3 that

$$|S_i| = \tilde{O}\left(\frac{n}{p} + \sqrt{pk}\,\frac{n\ln(n)}{pk}\right).$$

Since it is necessary that S_i is of size $\tilde{O}(n/p)$ we obtain:

Requirement 3. $k \geq p\ln^2(n)$.

Requirements 1–3 hold if $n/p \geq p^{2+\varepsilon}$ for some fixed $\varepsilon > 0$.

3.3. *Disconnecting Internal Edges*

In Step 3 of the algorithm we start removing from generating sets S_i those edges that are not edges of $\mathrm{CH}(S)$.

Amato and Preparata [5] observed that, given a set of rays $\mathcal{R}_v$ in $\mathbb{R}^3$ which all start at vertex v, the problem of computing their convex hull $\mathrm{CH}(\mathcal{R}_v)$ can be reduced to a two-dimensional convex hull computation. The algorithm first determines a plane H_v that does not pass through v and intersects all rays of $\mathcal{R}_v$. If there is no such plane H_v, then v is inside the convex hull $\mathrm{CH}(\mathcal{R}_v) = \mathrm{R}^3$. Hence, v is inside $\mathrm{CH}(S)$. If such a plane H_v exists, we can determine the convex hull $\mathrm{CH}(\mathcal{R}_v)$ by computing the two-dimensional convex hull of the intersection points of the rays with H_v.

We execute a sequential algorithm if $|\mathcal{R}_v| \leq O(n/p)$. Otherwise, we reduce the problem to solving, in parallel, the following linear programming problem:

Let the plane H_v be defined by equation $ax + by + cz = d$. It passes through v and all vertices w_i of edges (v, w_i) lie on one side of H_v. Let (x_i, y_i, z_i) be the coordinates of point w_i. This defines the constraints $ax_i + by_i + cz_i \geq d$ for the vertices w_i and the function $f(a, b, c) = ax_v + by_v + cz_v - d$, with (x_v, y_v, z_v) as coordinates of vertex v. This problem can be solved in parallel time $\tilde{O}((n \log_2 n)/p + T_s(n, p))$ with $O(1)$ communication rounds [17].

Lemma 5. *After execution of Step* 3 *the following hold*:

(a) *No edge of* $\mathrm{CH}(S)$ *is removed.*
(b) *The edges that are contained in* $\mathrm{CH}(S)$ *form a single connected component which does not contain any vertex or edge that is not in* $\mathrm{CH}(S)$.

Proof. (a) Step 2 removes only edges and vertices which lie inside the convex hull $\mathrm{CH}(\mathcal{R}_v)$ of rays incident to a vertex v. Clearly, an edge inside $\mathrm{CH}(\mathcal{R}_v)$ also lies inside $\mathrm{CH}(S)$.

(b) Assume that a vertex not in $\mathrm{CH}(S)$ is connected to a vertex v in $\mathrm{CH}(S)$ by some path of edges remaining after Step 3. Consider the edge of this path incident to v. This edge is deleted in Step 3 as it lies inside $\mathrm{CH}(\mathcal{R}_v)$; a contradiction. □

3.4. *Selecting the Global Convex Hull Edges in Each Generating Set*

Step 4(a) computes for each triangle $t \in R_i$ the point of S_i that has maximal distance to the plane through t. Such a furthest point is clearly in $\mathrm{CH}(S)$.

To determine the furthest points for R_i we use a subdivision hierarchy [20] for the convex polyhedron $\mathrm{CH}(S_i)$. We perform for each plane through a triangle in R_i a search to determine its tangents to $\mathrm{CH}(S_i)$. The size of the search structure is $O(|\mathrm{CH}(S_i)|)$ and it can be constructed sequentially in time $O(|\mathrm{CH}(S_i)|)$. The sequential search needs logarithmic time for each triangle. Hence, the time complexity of Step 4(a) is $O(|R_i| \log_2 |S_i|)$.

In Step 4(b) each processor p_i computes the union of those connected components of G_i that contain a furthest point in S_i. The sequential time complexity is $O(|G_i|) = O(|S_i|)$.

Lemma 6.

(a) *Every edge of* $\mathrm{CH}(S)$ *is contained in at least one set* E_i, $1 \leq i \leq p$.
(b) *No set* E_i, $1 \leq i \leq p$, *contains an edge that is not in* $\mathrm{CH}(S)$.

Proof. (a) Consider an edge (v, w) of $\mathrm{CH}(S)$. By Lemma 2, there exists a set S_i such that (v, w) is in $\mathrm{CH}(S_i)$. From Lemma 5 it follows that (v, w) is not deleted in Step 3. Due to the definition of R_i, v is contained in H_t for some $t \in R_i$, and $\mathrm{CH}(S) \cap H_t \subset S_i$. By Lemma 5 and the fact that H_t is a half-space it follows that the edges incident to $\mathrm{CH}(S) \cap H_t$ form a connected component. Furthermore, a point in S_i with maximum distance to the plane defined by t is in $\mathrm{CH}(S) \cap H_t$. Hence, $(v, w) \in E_i$.

(b) Follows immediately from Lemma 5. □

4. Summary

Lemmas 2, 5, and 6 imply the following:

Theorem 1. *The convex hull of n points in* 3*-space can be computed on a p-processor coarse-grained multicomputer with* $n/p \geq p^{2+\varepsilon}$, $\varepsilon > 0$, *in time* $\tilde{O}((n \log_2 n)/p + \Gamma_{n,p})$.

By standard transformation of two-dimensional Voronoi diagram construction to three-dimensional convex hull computation [12] we obtain:

Corollary 1. *The Voronoi diagram of a set of n points in the Euclidean plane can be computed on a p-processor coarse-grained multicomputer with* $n/p \geq p^{2+\varepsilon}$, $\varepsilon > 0$, *in time* $\tilde{O}((n \log_2 n)/p + \Gamma_{n,p})$.

Note that in order to compute the Voronoi diagram, the algorithm can be simplified. We do not need Step 4 of the algorithm, as the Voronoi diagram problem is equivalent to a convex hull problem where all points are vertices of the convex hull. All internal edges which are produced in Step 2 of the algorithm are removed in Step 3.

References

[1] A. Aggarwal, A.K. Chandra, and M. Snir. On Communication Latency in PRAM Computations. *Proc. SPAA*89, pages 11–21.
[2] A. Aggarwal, B. Chazelle, L. Guibas, C. Ó'Dúnlaing, and C. Yap. Parallel Computational Geometry. *Algorithmica*, Vol. 3, pages 293–327, 1988.
[3] N. Alon and N. Meggido. Parallel Linear Programming in Fixed Dimension Almost Surely in Constant Time. *Proc. FOCS*90, pages 574–582.
[4] N.M. Amato, M. Goodrich, and E. Ramos. Parallel Algorithms for High-Dimensional Convex Hulls. *Proc.* 35*th Annual IEEE Symposium on Foundations in Computer Science*, pages 683–694, 1994.
[5] N.M. Amato and F.P. Preparata. An NC^1 Parallel 3D Convex Hull Algorithm. *Proc.* 9*th Annual ACM Symposium on Computational Geometry*, pages 289–297, 1993.
[6] R.J. Anderson and L. Snyder. A Comparison of Shared and Nonshared Memory Models of Computation. *Proceedings of IEEE*, Vol. 79, No. 4, pages 480–487, 1995.

[7] M.J. Atallah and M.T. Goodrich. Efficient Parallel Solutions to Some Geometric Problems. *Journal of Parallel and Distributed Computing*, Vol. 3, pages 492–507, 1986.
[8] M.J. Atallah and M.T. Goodrich. Parallel Algorithms for Some Functions of Two Convex Polygons. *Algorithmica*, Vol. 3, pages 535–548, 1988.
[9] M.J. Atallah and J.-J. Tsay. On the Parallel-Decomposability of Geometric Problems. *Proc*. 5*th Annual ACM Symposium on Computational Geometry*, pages 104–113, 1989.
[10] K.E. Batcher. Sorting Networks and Their Applications. *Proc*. *AFIPS Spring Joint Computer Conference*, pages 307–314, 1968.
[11] G.E. Blelloch, C.E. Leiserson, B.M. Maggs, C.G. Plaxton, S.J. Smith, and M. Zagha. A Comparison of Sorting Algorithms for the Connection Machine CM-2. *Proc*. *SPAA*91, pages 3–16.
[12] K.Q. Brown. Voronoi Diagrams from Convex Hulls. *Information Processing Letters*, Vol. 9, pages 223–228, 1979.
[13] A. Chow. Parallel Algorithms for Geometric Problems. Ph.D. thesis, University of Illinois at Urbana-Champaign, 1980.
[14] K.L. Clarkson, K. Mehlhorn, and R. Seidel. Four Results on Randomized Incremental Constructions. *Computational Geometry Theory and Application*, Vol. 3, No. 4, pages 185–212, 1993.
[15] F. Dehne, A. Fabri, and C. Kenyon. Scalable and Architecture Independent Parallel Geometric Algorithms with High Probability Optimal Time. *Proc*. 6*th IEEE Symposium on Parallel and Distributed Processing*, pages 586–593, 1994.
[16] F. Dehne, A. Fabri, and A. Rau-Chaplin. Scalable Parallel Geometric Algorithms for Coarse Grained Multicomputers. *Proc*. 9*th Annual ACM Symposium on Computational Geometry*, pages 298–307, 1993.
[17] X. Deng. A Convex Hull Algorithm for Coarse Grained Multiprocessors. *Proc*. 5*th International Symposium on Algorithms and Computation*, pages 162–173, 1994.
[18] X. Deng and P. Dymond. Efficient Routing and Message Bounds for Optimal Parallel Algorithms. *Proc*. *IPPS* 1995, Santa Barbara, pages 556–562, April 1995.
[19] X. Deng and N. Gu. Good Programming Style on Multiprocessors. *Proc*. *IEEE Symposium on Parallel and Distributed Processing*, Dallas, pages 538–543, October 1994.
[20] D.P. Dobkin and D.G Kirkpatrick. Fast Detection of Polyhedral Intersection. *Theoretical Computer Science*, Vol. 27, pages 241–253, 1983.
[21] H. Edelsbrunner. *Algorithms in Combinatorial Geometry*. EATCS Monographs on Theoretical Computer Science, Vol. 10. Springer-Verlag, Berlin, 1987.
[22] A.V. Gerbessiotis and L.G. Valiant. Direct Bulk-Synchronous Parallel Algorithms. *Proc*. 3*rd Scandinavian Workshop on Algorithm Theory*, pages 1–18. Lecture Notes in Computer Science, Vol. 621. Springer-Verlag, Berlin, 1992.
[23] P. Gibbons. A More Practical PRAM Model. *Proc*. 1989 *ACM Symposium on Parallel Algorithms and Architectures*, pages 158–168, 1989.
[24] M.T. Goodrich. Planar Separators and Parallel Polygon Triangulation. To appear in *Journal of Computer and System Science*.
[25] M.T. Goodrich, J.J. Tsay, D.E. Vengroff, and J.S. Vitter. External-Memory Computational Geometry. *Proc*. 34*th IEEE Symposium on Foundations of Computer Science*, pages 714–723, 1993.
[26] F.T. Leighton. *Introduction to Parallel Algorithms and Architectures*: *Arrays*, *Trees*, *Hypercubes*. Morgan Kaufmann, San Mateo, CA, 1992.
[27] Hui Li and K.C. Sevcik. Parallel Sorting by Overpartitioning. *Proc*. *SPAA*94, pages 46–56, 1994.
[28] R.J. Lipton and R.E. Tarjan. A Separator Theorem for Planar Graphs. *SIAM Journal on Applied Mathematics*, Vol. 36, No. 2, pages 177–189, 1979.
[29] K. Mehlhorn. *Graph Algorithms and NP-Completeness*. Springer-Verlag, New York, 1984.
[30] R. Miller and Q.F. Stout. Efficient Parallel Convex Hull Algorithms. *IEEE Transactions on Computers*, Vol. 37, No. 12, pages 1605–1618, 1988.
[31] K. Mulmuley. *Computational Geometry*: *an Introduction Through Randomized Algorithms*. Prentice-Hall, New York, 1993.
[32] C.H. Papadimitriou and M. Yannakakis. Towards an Architecture Independent Analysis of Parallel Algorithms. *Proc*. 20*th Symposium on Theory of Computing*, pages 510–513, 1988.
[33] F.P. Preparata and M.I. Shamos. *Computational Geometry*: *an Introduction*. Springer-Verlag, New York, 1985.
[34] J.H. Reif and S. Sen. Optimal Parallel Algorithms for 3D Convex Hulls and Related Problems. *SIAM*

Journal of Computing, Vol. 21, pages 446–485, 1992, pages 394–404, 1989, with corrigendum in *SIAM Journal of Computing*, Vol. 23, pages 447–448, 1994.

[35] L. Snyder. Type Architectures, Shared Memory and the Corollary of Modest Potential. *Annual Review of Computer Science*, Vol. 1, pages 289–317, 1986.

[36] L.G. Valiant. A Bridging Model for Parallel Computation. *Communications of the ACM*, Vol. 33, pages 103–111, 1990.

[37] L.G. Valiant. General Purpose Parallel Architectures. *Handbook of Theoretical Computer Science*, edited by J. van Leeuwen, pages 943–972. MIT Press/Elsevier, Cambridge, MA/Amsterdam, 1990.

Received October 30, 1995, *and in revised form April* 15, 1996, *and in final form September* 17, 1996.