Scalable And Architecture Independent Parallel Geometric Algorithms With High Probability Optimal Time*

Frank Dehne[1], Claire Kenyon[2], and Andreas Fabri[3]

[1] School of Computer Science, Carleton University, Ottawa, Canada K1S 5B6, dehne@scs.carleton.ca
[2] LIP, URA CNRS 1398, Ecole Norm. Supérieure de Lyon, 69364 Lyon Cedex 07, France, kenyon@lip.ens-lyon.fr
[3] INRIA, BP 93, 06 902 Sophia Antipolis, France, fabri@cassiopee.inria.fr

Abstract*

We present parallel computational geometry algorithms that are scalable, architecture independent, easy to implement, and have, with high probability, an optimal time complexity for uniformly distributed random input data. Our methods apply to multicomputers with arbitrary interconnection network or bus system.

The following problems are studied in this paper: (1) lower envelope of line segments, (2) visibility of parallelepipeds, (3) convex hull, (4) maximal elements, (5) Voronoi diagram, (6) all-nearest neighbors, (7) largest empty circle, and (8) largest empty hyperrectangle. Problems 2-8 are studied for d-dimensional space, $d=O(1)$. We implemented and tested the lower envelope algorithm and convex hull algorithm (for $d=3$ and $d=4$) on a CM5. The results indicate that our methods are of considerable practical relevance.

1. Introduction

In this paper we continue our work I in [5] to design parallel computational geometry algorithms that are scalable, architecture independent (i.e. portable), easy to implement, and have an optimal or nearly optimal time complexity.

The Model Of Computation

To ensure that the complexity analysis matches the timings observed in actual implementations, it is important that the model of computation is based only on assumptions that are true for contemporary multicomputers (and can be expected to hold for future ones as well). Our algorithms are designed for a *generic multicomputer* $M_\alpha(p,n)$ with only the following small set of assumptions:

(A1) $M_\alpha(p,n)$ has p processors $P_1, ..., P_p$ connected by some arbitrary interconnection network or bus system.

(A2) Every processor P_i has $\Theta(n/p)$ local memory.

(A3) $n/p \geq p^\alpha$ for some arbitrarily small but fixed constant $\alpha > 0$.

(A4) The total data size, n, is a large number (but the number of processors, p, is not necessarily large).

These assumptions are evidently true for contemporary multicomputers. Assumption A3 is based on the well known fact that memory is inexpensive and each processor has, in practice, more than just a few local registers of memory. Note that $\alpha > 0$ can be arbitrarily small, but for all practical purposes one can safely assume that $\alpha > 1/10$ (i.e. each processor has a local memory of at least $p^{1/10}$). It seems reasonable to expect that the above assumptions will also hold in the future.

* Research partially supported by the Natural Sciences and Engineering Research Council of Canada and the ESPRIT Basic Research Actions Nr. 3075 (ALCOM) and Nr. 7141 (ALCOM II). This work was done while the first author was visiting the Lab. de l'Inform. du Parallélisme at the Ecole Norm. Supérieure de Lyon.

The Problems

We present extremely simple parallel algorithms for solving the following problems on any multi-computer $M_\alpha(p,n)$:

Problem 1. Given a random set S of n non-intersecting line segments in $[0,1]^2$, compute its *lower envelope*, $LE(S)$.

Problem 2. Given a random set S of n parallelepipeds in $[0,1]^d$, all perpendicular to the x-axis, $3 \le d = O(1)$, compute their *visible portion*, $VIS(S)$, with respect to the view along the x-axis. For d=3, Problem 2 is the well known *visibility problem* for parallel rectangles in $[0,1]^3$.

Problem 3. Given a random set S of n points in $\mathbb{R}^d$, $d=O(1)$, compute its *convex hull*, $CH(S)$.

Problem 4. Given a random set S of n points in $\mathbb{R}^d$, $d=O(1)$, compute the subset, $MAX(S)$, of *maximal element*s of S.

Problem 5. Given a random set S of n points in $[0,1]^d$, compute the *Voronoi diagram* of S.

Problem 6. Given a random set S of n points in $[0,1]^d$, $d=O(1)$, compute for each $s \in S$ its *nearest neighbor* in S.

Problem 7. Given a random set S of n points in $[0,1]^d$, $d=O(1)$, compute the *largest empty circle* inside the convex hull of S.

Problem 8. Given a random set S of n points in $[0,1]^d$, $d=O(1)$, compute the *largest empty hyperrectangle* inside the smallest hyperrectangle containing S.

The Results

For standard uniform data distribution, our algorithms for Problems 1-4 require, with high probability, a communication time and local computation time of at most

$$(k+1)\ (T_{pSum}(p) + T_{compr}(p,n)) \quad \text{and} \quad (k+2)\ \left(\frac{T_1(n)}{p} + O\left(\frac{n}{p}\right)\right),$$

respectively, where $k = \left\lceil \frac{1}{2\alpha} + \frac{1}{2} \right\rceil$ is a fixed constant and $T_1(n)$ denotes the (high probability) sequential time complexity of the respective problem. Note that, for all practical purposes one can safely assume that $k \le 6$. For machines like the CM5, Intel iPSC or Paragon, k=1. Even for a CM2, $k \le 3$.

For Problems 5-8 our algorithms require for standard uniform data distribution, with high probability, a communication time and local computation time of at most

$$(2^{3d+d})\ T_{sort}(p,n) \quad \text{and} \quad 2^{3d}\ \left(\frac{T_1(n)}{p} + O\left(\frac{n}{p}\right)\right), \text{ respectively, } d=O(1).$$

$T_{pSum}(p)$, $T_{compr}(p,n)$, and $T_{sort}(p,n)$ denote the parallel time complexity on an $M_\alpha(p,n)$ to compute the partial sums of p integers (one stored at each processor), compress a subset of data of size $n' \le n$ into $p' \le p$ processors (where all data items in a processor are moved to the same destination processor), and sort n data items, respectively.

It is easy to see that a partial sum and compression operation can be implemented by a constant number of sorts, and that all of the above problems reduce to sorting. Hence, from a theoretical point of view, our algorithms have, with high probability, an asymptotically optimal time complexity.

An important feature of our methods is that they require, with high probability, only a fixed small number of communication rounds, independent of the problem size. Each communication round is either a partial sum or compress operation (Problems 1-4) or a sort operation (Problems 5-8). All other computation is local.

Comparison With Previous Work

For many architectures, our results are a considerable improvement over existing methods. This applies in particular to previous fine-grained ($n/p=O(1)$) algorithms, even if they are (fine-grained) optimal. For example, it is impossible for fine-grained mesh algorithms, even optimal ones, to yield optimal speedups for ratios $n/p \ne O(1)$ by applying the usual simulation method (also called "virtual

processors" in many multicomputer operating systems). For $n/p \neq O(1)$, our methods are considerably faster. In fact, for meshes, our algorithms are, with high probability, optimal for the entire range $n/p \geq p^{\alpha}$, $\alpha>0$. For hypercubic networks, no previous algorithms exist that can match our speedups for $n/p \neq O(1)$.

As indicated above, our methods require, with high probability, only a fixed small number of communication rounds, independent of the problem size. This is the main reason why they perform well. In contrast, previous fine grained algorithms for the above problems (see e.g. [1] for an overview), require a number of communication rounds that increases with the problem size (typically $\log(n)$, $\log\log(n)$, $\log^*(n)$, etc. rounds).

The results and techniques presented in this paper are different from [5. Our previous work presented *deterministic* algorithms. Those results were also based on the idea of using only $O(1)$ communication rounds, but we used deterministic methods to reach that goal, which are very different from the high probability methods presented now. In [5], we could solve Problem 1, and Problem 4 for $d \leq 3$. For Problems 2,3,5-8 there existed so far no efficient scalable solution (except for simulating fine-grained algorithms). Furthermore, our previous deterministic methods applied only to the case $n/p \geq p$, i.e. $\alpha \geq 1$. Our new methods presented in this paper solve Problems 2-8, with high probability, for any $\alpha > 0$. In addition, the new algorithms solve Problems 2-8 for arbitrary fixed dimension, and with an improved time complexity (compared to those cases solved in [5]). For Problem 1 we obtain, with high probability, an improved time complexity. The obvious drawback is, of course, that our new time complexity analysis holds only for uniform data distribution. It is an interesting open problem to extend our methods to the non-uniform case. We will indicate heuristics for which we conjecture that they solve at least some cases of non-uniform distributions.

Practical Significance, Experimental Results

Our algorithms are simple and easy to implement. The constants in the time complexity analysis are small. Except for a small (fixed) number of communication rounds, all other computation is sequential and consists essentially of solving on each processor a small (fixed) number of subproblems of size n/p. Hence, it allows to use existing sequential code for the respective problem. For the communication rounds, we can use well studied existing code for data compression, partial sum, and sorting.

For coarse grained machines, our methods imply communication through few large messages rather than having many small messages. This is important for machines like the Intel iPSC where each message creates a considerable overhead. Note, however, that our analysis accounts for the length of messages.

We have implemented and tested our algorithms for the lower envelope and convex hull (d=3 and d=4) on a CM5 and obtained very fast running times which match nicely the theoretical analysis. Section 4 presents and discusses these expiremental results. They indicate that our methods are of considerable practical relevance.

2. Hull Problems

We study the following *Generic Hull Problem*, $H(S)$, which includes Problems 1-4 listed above: *"For a random set S of n objects from some universe U, compute* $H(S) \subseteq U$*"* where $H(S)$ has the properties

(P1) $H(A_1 \cup ... \cup A_j) = H(H(A_1) \cup ... \cup H(A_j))$ for any A_1, ..., $A_i \subseteq S$, and

(P2) there exists a function $h(n)$ such that

(a) for any random subset $A \subseteq S$, $E(|H(A)|) \leq h(|A|)^{\dagger}$, and

(b) $h(n) \leq n^{\delta}$ for some $0 < \delta < \min\{\varepsilon, 1/8\}$.

---

† For some random variable X let $E(X)$ and $\Pr\{X=y\}$ denote the expected value of X and the probability that X takes a certain value y, respectively.

For Problems 1-4 listed above, S corresponds to a random set of line segments, parallelepipeds in $[0,1]^d$ or points in $\mathbb{R}^d$, and $H(S)$ corresponds to $LE(S)$, $VIS(S)$, $CH(S)$ and $MAX(S)$, respectively. We will show later that these four problems have Properties P1 and P2.

We observe that Assumption A3 in Section 1 is equivalent to

$$1 \le p \le n^{1-\varepsilon}, \text{ where } \varepsilon = \frac{\alpha}{1+\alpha} \quad .$$

In the following Sections 2.1 and 2.2 we will first show how to compute $H(S)$ if $p \le \sqrt{n}$ and then how to generalize our method for $1 \le p \le n^{1-\varepsilon}$.

2.1. The Generic Hull Problem, $H(S)$, For $p \le \sqrt{n}$

We study the following extremely simple algorithm for computing $H(S)$.

Algorithm 1 *Architecture*: A multicomputer $M_\alpha(p,n)$ with $p \le \sqrt{n}$. *Input:* Each processor P_i stores a random subset S_i of n/p objects of S. (The subsets S_i are disjoint.) *Output:* $H(S)$.

(1) Each processor P_i computes sequentially $H(S_i)$. Let $S' = H(S_1) \cup ... \cup H(S_p)$, $n' = |S'|$.
(2) IF $n' \le n/p$ THEN S' is compressed into processor P_1 which computes sequentially $H(S)=H(S')$ STOP.
(3) The set S' is compressed into $p' \le 2\, n'\, p\, /\, n$ processors as follows: The sequence $H(S_1), \ldots, H(S_p)$ is split into p' maximal subsequences of consecutive $H(S_j)$, such that the total size of each subsequence (i.e. the total number of objects) is at most n/p. Let S_i' be the set of objects in the i-th subsequence, $1 \le i \le p'$. Set S_i' is stored at processor P_i.
(4) Each processor P_i $(1 \le i \le p')$ computes sequentially $H(S_i')$. Let $S'' = H(S_1') \cup ... \cup H(S_{p'}')$, $n'' = |S''|$.
(5) IF $n'' \le n/p$
THEN S'' is compressed into processor P_1 which computes $H(S)=H(S'')$ sequentially
ELSE continue with some deterministic algorithm.

The correctness of Algorithm 1 follows immediately from Property P1. Note that, for the "else" case in Step 5, any deterministic parallel algorithm may be applied. As we will show in the remainder, it is so highly unlikely that it will ever be used, that it doesn't seem worthwhile to invest much programming efforts here. A trivial solution could be, e.g., to use a sequential algorithm, running on P_1 and with memory accesses to other processors' memories implemented by message passing. For some special cases, simple and efficient deterministic scalable parallel solutions have been presented in [5].

In the remainder of this section we will prove

Lemma 1 *With high probability, either $n' \le n/p$ or $n'' \le n/p$.*

This implies immediately

Theorem 1 *Given a multicomputer $M_\alpha(p,n)$ with $p \le \sqrt{n}$ then, with high probability, Algorithm 1 computes H(S), $|S| = n$, with a communication time of at most 2 ($T_{pSum}(p) + T_{compr}(p,n)$) and a local computation time of at most 3 $T_H(n)/p + O(n/p)$ where $T_H(n)$ denotes the sequential time complexity for solving H(S).*

We now start proving Lemma 1. Consider some $0 < \delta < 1/4$. It follows from Property P2 that $h(n) \le n^\delta$. In order to prove Lemma 1, we will consider three cases.

2.1.1. Case 1: $p \le n^{\frac{1-\delta}{3}}$

From *Chebyshev's Inequality* (see e.g. [6] *p*.233) and the above assumption it follows that

$\Pr\{\text{there exist some } 1 \le i \le p \text{ such that } |H(S_i)| > n^{\frac{2}{3}(\frac{1}{2}-\delta)} h(n)\} \le p \Pr\{|H(S_i)| > n^{\frac{2}{3}(\frac{1}{2}-\delta)} h(n)\} \le \frac{p}{n^{\frac{4}{3}(\frac{1}{2}-\delta)}} \le \frac{n^{\frac{1-\delta}{3}}}{n^{\frac{4}{3}(\frac{1}{2}-\delta)}} \le \frac{1}{n^{\frac{1}{3}-\delta}} \to 0$ for $n \to \infty$, because $\delta \le \frac{1}{4}$. Hence, it follows with probability at least $1 - \frac{1}{n^{\frac{1}{3}-\delta}} \to 1$ (for $n \to \infty$) that after Step 1 of Algorithm 1 we obtain a set S' of size

$$n' \le p\ n^{\frac{2}{3}(\frac{1}{2}-\delta)} h(n) \le n^{\frac{1-\delta}{3}}\ n^{\frac{2}{3}(\frac{1}{2}-\delta)} n^{\delta} = n^{\frac{2}{3}}.$$

From $p \le n^{\frac{1-\delta}{3}}$ it follows that $\frac{n}{p} \ge n^{\frac{2}{3}}$. Hence, $n' \le \frac{n}{p}$ with high probability, and Lemma 1 follows for Case 1.

2.1.2. Case 2: $n^{\frac{1-\delta}{3}} \le p \le \frac{1}{\sqrt{2}} n^{\frac{1-\delta}{2}}$

Since $n^{\frac{1}{3}(1-\delta)} \le p$, it follows that $p \to \infty$ for $n \to \infty$. Thus, it follows from the *Law of Large Numbers* (see e.g. [6] *p*.243) that $\Pr\{n' \le 2\ p\ h(n)\} \to 1$ for $n \to \infty$. We observe that $2\ p\ h(n) \le \frac{n}{p}$ if $p \le \frac{\sqrt{n}}{\sqrt{2h(n)}}$. The latter holds if $p \le \frac{\sqrt{n}}{\sqrt{2n^{\delta}}}$, which is true if $p \le \frac{1}{\sqrt{2}} n^{\frac{1-\delta}{2}}$. Hence, $n' \le \frac{n}{p}$ with high probability, and Lemma 1 follows for Case 2.

2.1.3. Case 3: $\frac{1}{\sqrt{2}} n^{\frac{1-\delta}{2}} \le p \le \sqrt{n}$

Since $\frac{1}{\sqrt{2}} n^{\frac{1-\delta}{2}} \le p$, it follows that $p \to \infty$ for $n \to \infty$. Thus, it follows from the *Law of Large Numbers* that, with high probability, $1/2\ p\ h(n) \le n' \le 2\ p\ h(n) \le 2\ \sqrt{n}\ n^{\delta}$. Since $p' \le \frac{2n'}{n/p}$ and $p \le \sqrt{n}$. it follows that, with high probability, $p' \le \frac{4ph(n)}{n/p} \le 4\ h(n) \le 4\ n^{\delta}$.

We now study Steps 3 and 4 of Algorithm 1. For each S_i', $1 \le j \le p'$, there exist sets S_{t_i}, S_{t_i+1}, …, S_{u_i} such that $S_i' = H(S_{t_i}) \cup H(S_{t_i+1}) \cup \ldots \cup H(S_{u_i})$. Hence, by Property P1, $H(S_i') = H(H(S_{t_i}) \cup H(S_{t_i+1}) \cup \ldots \cup H(S_{u_i})) = H(S_{t_i} \cup S_{t_i+1} \cup \ldots \cup S_{u_i})$. Define $\text{orig}(S_i') = S_{t_i} \cup S_{t_i+1} \cup \ldots \cup S_{u_i}$, then $H(S_i') = H(\text{orig}(S_i'))$.

Lemma 2 *For each $1 \le i \le p'$, orig(S_i') is a random subset of S and, hence, $E(|H(S_i')|) = E(|H(\text{orig}(S_i'))|) \le h(n)$.*

Proof Sketch. orig(S_i') = $S_{t_i} \cup S_{t_i+1} \cup \ldots \cup S_{u_i}$. Clearly, S_{t_i}, is a random set of n/p objects. For each $l \in [t_i, u_i]$, $S_{t_i} \cup S_{t_i+1} \cup \ldots \cup S_l \cup \ldots \cup S_{u_i}$ is in bijection with $S_l \cup S_{t_i} \cup S_{t_i+1} \cup \ldots \cup S_{l-1} \cup S_{l+1} \cup \ldots \cup S_{u_i}$, in that order. Hence, it follows that S_l has the same distribution as S_{t_i}. Thus, orig(S_i') is the union of random subsets of S, and Lemma 2 follows. □

Hence, it follows from *Chebyshev's Inequality* that

Pr{there exist some $1 \le i \le p'$ such that $|H(S_i')| > n^{\frac{1}{3}-2\delta} h(n)$} $\le p'$ Pr{$|H(S_i)| > n^{\frac{1}{3}-2\delta} h(n)$} $\le \frac{p'}{n^{\frac{2}{3}-4\delta}} \le \frac{4n^{\delta}}{n^{\frac{2}{3}-4\delta}} \to 0$ because $\delta < \frac{1}{8}$. Thus, after Step 4 of Algorithm 1 we obtain, with high probability, a set S'' of size $n'' \le p' \, n^{\frac{1}{3}-2\delta} h(n) \le 4 \, n^{\delta} n^{\frac{1}{3}-2\delta} n^{\delta} \le n^{\frac{1}{2}}$. From $p \le \sqrt{n}$, it follows that $\frac{n}{p} \ge \sqrt{n}$. Hence, $n'' \le \frac{n}{p}$ with high probability, and Lemma 1 follows for Case 3. This concludes the proof of Lemma 1.

2.2. The Generic Hull Problem, $H(S)$, For $1 \le p \le n^{1-\varepsilon}$

We will now generalize Algorithm 1 to solve $H(S)$ on a multicomputer $M_\alpha(p,n)$. Let $k \ge \frac{1}{2(\varepsilon-\delta)} - \frac{1}{6} = \frac{1+\alpha}{2\alpha - 2\delta(1+\alpha)} - \frac{1}{6}$ be a fixed positive integer constant.

Algorithm 2 *Architecture*: A multicomputer $M_\alpha(p,n)$. *Input:* Each processor P_i stores a random subset S_i of n/p objects of S. (The subsets S_i are disjoint.) *Output:* $H(S)$. Let $n^{(0)} = n$, $p^{(0)} = p$, and $S^{(0)} = S$.

(1) For $j = 0 \ldots k$ do

(a) Each processor P_i, $1 \le i \le p^{(j)}$, computes sequentially $H(S_i^{(j)})$. Let $S^{(j+1)} = H(S_1^{(j)}) \cup \ldots \cup H(S_{p^{(j)}}^{(j)})$, $n^{(j+1)} = |S^{(j+1)}|$.

(b) IF $n^{(j+1)} \le n/p$ THEN $S^{(j+1)}$ is compressed into processor P_1 which computes sequentially $H(S) = H(S^{(j+1)})$. STOP.

(c) The set $S^{(j+1)}$ is compressed into $p^{(j+1)} \le 2 \, n^{(j+1)} \, p / n$ processors as follows: The sequence $H(S_1^{(j)}), \ldots, H(S_{p^{(j)}}^{(j)})$ is split into $p^{(j+1)}$ maximal subsequences of consecutive $H(S_i^{(j)})$, such that the total size of each subsequence (i.e. the total number of objects) is at most n/p. Let $S_i^{(j+1)}$ be the set of objects in the i-th subsequence, $1 \le i \le p^{(j+1)}$. Set $S_i^{(j+1)}$ is stored at processor P_i.

(2) IF $n^{(k+1)} \le n/p$
THEN in the previous Step 1c, $S^{(k+1)}$ was compressed into processor P_1 which can now compute $H(S) = H(S^{(k+1)})$ sequentially
ELSE continue with some determinstic algorithm.

The correctness of Algorithm 1 follows, again, immediately from Property P1. In the remainder of this section we will prove

Lemma 3 *With high probability, $n^{(j)} \le n/p$ for some $j \le k+1$.*

This implies immediately

Theorem 2 *Given a multicomputer $M_\alpha(p,n)$ then, with high probability, Algorithm 2 computes H(S), |S| = n, with a communication time of at most (k+1) ($T_{pSum}(p) + T_{compr}(p,n)$) and a local computation time of at most (k+2) ($T_H(n)/p + O(n/p)$), where $T_H(n)$ denotes the sequential time complexity for solving H(S) and $k \geq 1 / (2(\varepsilon-\delta)) - 1/6 = (1+\alpha) / (2\alpha - 2\delta(1+\alpha)) - 1/6$ is a fixed positive integer constant.*

We now prove Lemma 3. If $p \leq \sqrt{n}$ then we apply the analysis of Algorithm 1 given in Section 2.1. For the remainder assume that $\sqrt{n} \leq p \leq n^{1-\varepsilon}$. The basic idea is to iterate Lemma 2 and the analysis for Case 3 in Section 2.1. We obtain that, with high probability,

$$p^{(j+1)} \leq \frac{4p^{(j)}h(n)}{\frac{n}{p}} \leq p^{(j)} \frac{4n^{\delta}}{n^{\varepsilon}} \text{ for all } j \geq 0.$$

Hence, with high probability, $p^{(j)} \leq p \left(\frac{4n^{\delta}}{n^{\varepsilon}}\right)^{j}$ for all $j \geq 0$. We observe that $p \left(\frac{4n^{\delta}}{n^{\varepsilon}}\right)^{j} \leq n^{\frac{\varepsilon-\delta}{3}}$ if $i \geq \frac{1}{2(\varepsilon-\delta)} - \frac{1}{6}$. Unless the for loop in Algorithm 2 is stopped for $j < k$, it follows that Pr{there exist some $1 \leq i \leq p$ such that $H(S_i^{(k)}) > n^{\frac{2}{3}(\varepsilon-\delta)} h(n)$} $\leq \frac{p^{(k)}}{n^{\frac{4}{3}(\varepsilon-\delta)}} \leq \frac{n^{\frac{\varepsilon-\delta}{3}}}{n^{\frac{4}{3}(\varepsilon-\delta)}} \leq \frac{1}{n^{\varepsilon-\delta}} \to 0$ for $n \to \infty$, because $\delta < \varepsilon$. Thus, with high probability $n^{(k+1)} \leq p^{(k)} n^{\delta} n^{\frac{2}{3}(\varepsilon-\delta)} \leq n^{\frac{\varepsilon-\delta}{3}} n^{\delta} n^{\frac{2}{3}(\varepsilon-\delta)} \leq n^{\varepsilon} \leq \frac{n}{p}$. This concludes the proof of Lemma 3.

2.3. Applications

We now study how to apply Algorithm 2 for solving Problems 1-4 listed in Section 1. Clearly, all four problems have Property P1 indicated above. We now study conditions under which Problems 1-4 have Property P2.

Lemma 4 *For a uniform random set S of n non-intersecting line segments in $[0,1]^2$, $E(|LE(S)|) \leq 2 \ln(n)$.*

An outline of the proof (incl. more details about the underlying distribution) will be given in the complete version of this paper. See [8] for details.

Lemma 5 *For a uniform random set S of n parallelepipeds in $[0,1]^d$,*

$$E(|VIS(S)|) \leq \frac{2^{d-1}}{(d-1)!} \ln^{d-1}(n).$$

An outline of the proof (incl. more details about the underlying distribution) will be given in the complete version of this paper. See [8] for details. With regards to Problem 3, it has been shown in [3, 10-12 that

$$E(|CH(S)|) \leq O(\ln(n)^{f(d)}), f(d)=O(1)$$

for a random set S of n points in $\mathbb{R}^d$, $d=O(1)$. If the points are chosen independently from a d-dimensional normal distribution, then $f(d) = (d-1)/2$ [10]. If they have their components chosen independently from any set of continuos distributions (possibly different for each component), then $f(d) = d-1$ [3]. It has been shown in [3] that

$$E(|MAX(S)|) \leq \log(n)^{(d-1)}$$

for a random set S of n points in $\mathbb{R}^d$, $d=O(1)$, where the coordinates of each point are independent and chose from an identical, continuous distribution. Hence, for the data distributions indicated above, Problems 1-4 have Property P2 for any $\delta > 0$. We apply Algorithm 2 and Theorem 2 with $k = \lceil 1/(2a) + 1/2 \rceil$ and obtain

Theorem 3 *For any multicomputer $M_\alpha(p,n)$, Algorithm 2 solves Problems 1-4 (for the data distributions indicated above), with high probability, with a communication time of at most $(k+1)(T_{pSum}(p) + T_{compr}(p,n))$ and a local computation time of at most $(k+2)(T_1(n)/p + O(n/p))$, where $k = \lceil 1/(2a) + 1/2 \rceil$ is a fixed constant and $T_1(n)$ denotes the (high probability) sequential time complexity of the respective problem.*

3. Proximity Problems

Due to space limitations, we will now present only an outline of our solutions for Problems 5-8. We will also restrict our presentation to the case d=2. The complete proofs and the generalization to $d=O(1)$ will be presented in the final version of this paper.

Our approach is to use a grid method similar to [4]. The non-trivial difference is that the results in [4] are *expected* sequential time complexities, whereas our parallel time complexity results are *with high probability*. Consider the following *rectangular partitioning* of S into p subsets R_1, ..., R_p: Partition $[0,1]^2$ by $\sqrt{p}$ -1 vertical lines into $\sqrt{p}$ vertical slabs K_1, ..., $K_{\sqrt{p}}$ such that each slab contains exactly $n/\sqrt{p}$ points of S. Partition each slab K_j, $1 \le j \le \sqrt{p}$, by $\sqrt{p}$ -1 horizontal lines into $\sqrt{n}$ rectangles $r_{(j-1)\sqrt{p}+1}$, ..., $r_{(j-1)\sqrt{p}+\sqrt{p}}$ such that each rectangle contains exactly n/p points of S. Let R_i be the set of points contained in rectangle r_i, $1 \le i \le p$. Denote by w_i and h_i the width and height of rectangle r_i, respectively. Two rectangles r_i and $r_{i'}$ are called adjacent if they are within distance $1/(2\sqrt{p})$. For each $i \in \{1, ..., \sqrt{p}\}$ define neighbors(i) = $\{i' \mid 1 \le i' \le \sqrt{p}, i' \ne i, r_i$ and $r_{i'}$ are adjacent$\}$ and N_i as the union of all $R_{i'}$ such that $i' \in$ neighbors(i).

We study the following algorithm for solving Problems 5-8. For $S' \subseteq S$, let $\lambda(s,S)$ refer to the Voronoi polygon of s with respect to S', nearest neighbor of s in S', or the largest empty circle or rectangle with respect to S' that has s on its border, respectively

Algorithm 3 (outline) *Architecture*: A multicomputer $M_\alpha(p,n)$. *Input:* Each processor P_i stores a random subset of n/p unique points of S. *Output:* Each processor P_i stores $\lambda(s,S)$ for n/p points $s \in S$.

(1) With two global sort procedures, a rectangular partitioning R_1, ..., R_p of S is created such that each processor P_i stores subset R_i and the coordinates of r_i.

(2) IF for every processor P_i: $1/(2\sqrt{p}) \le h_i \le 2/\sqrt{p}$ and $1/(2\sqrt{p}) \le w_i \le 2/\sqrt{p}$ THEN

(2a) Every processor P_i receives a copy of all $R_{i'}$, $i' \in$ neighbors(i), and computes $\lambda(s, R_i \cup N_i)$ for each $s \in R_i$.

(2b) If Algorithm 3 is used to compute the Voronoi diagram, check if the rays of the open Voronoi polygons have strictly increasing angles. If this is not the case, report that the result is incorrect and continue with Step 2c.

ELSE

(2c) use some deterministic algorithm.

Lemma 6 *Algorithm 3 correctly computes $\lambda(s,S)$ for all $s \in S$.*

Proof Sketch. The "if" clause in Step 2 ensures that there exists no empty circle with radius $1/(2\sqrt{p})$ and that, for each $s \in R_i$, all points within distance $1/(2\sqrt{p})$ are contained in $R_i \cup N_i$. It is easy to see that in such a case, $\lambda(s,R_i \cup N_i) = \lambda(s,S)$ for Problems 6-8. For Problem 5, we also need to check the correctness of the Voronoi edges outside $[0,1]^2$. This is done by the test in Step 2b. We show that if the rays of open Voronoi polygons have strictly increasing angles then the Voronoi edges outside $[0,1]^2$ were computed correctly. Note that the test can be performed with every processor on the border of $[0,1]^2$ having only the data of its two direct neighbors. The complete proof will be presented in the final version of this paper. □

The following two lemmas show that the "if" clause in Step 2 and the test in Step 2b are successful with high probability.

Lemma 7 *With high probability, for all* $1 \le i \le p$, $1/(2\sqrt{p}) \le h_i \le 2/\sqrt{p}$ *and* $1/(2\sqrt{p}) \le w_i \le 2/\sqrt{p}$.

The proof will be presented in the final version of this paper. For a uniform point distribution we can show via *Chernoff Bounds* (see e.g. [6] *p*. 193) that, for growing *n*, the probability that there exists any h_i or w_i outside the above bounds converges very fast to 0.

Lemma 8 *If Algorithm 3 is used to compute the Voronoi diagram then, with high probability, after Step 2a all Voronoi edges outside* $[0,1]^2$ *have been correctly computed.*

The basic idea for the proof is to show via *Chernoff Bounds* that with high probability all Voronoi edges outside $[0,1]^2$ are bisectors of points in adjacent rectangles of our rectangular partitioning. The complete proof will be presented in the final version of this paper.

The above three lemmas imply

Theorem 4 *For any multicomputer* $M_\alpha(p,n)$*, Algorithm 2 solves Problems 5-8, with high probability, with a communication time of at most* $(2^{3d}+d)\ T_{sort}(p,n)$ *and a local computation time of at most* $2^{3d}\ (T_1(n)/p + O(n/p))$*, where* $T_1(n)$ *denotes the (high probability) sequential time complexity of the respective problem.*

Note that for Problem 5 and the case $d>2$, the Voronoi diagram to be reported can be of more than linear size. With high probability, each processor P_i can compute the Voronoi polygons for its point set R_i based on the point set $R_i \cup N_i$ which is of size $O(n/p)$. There is no problem as long as, during this sequential computation of $\lambda(s,R_i \cup N_i)$ for each $s \in R_i$, the reported results need not be stored in the local memory of P_i or if there is extra memory available for storing the output. Otherwise, it is impossible (for any algorithm) to solve the problem on the given architecture. The expected output size per processor is $O(n/p)$ [4], and we conjecture that this bound also holds with high probability. For Problems 6-8, the output size per processor is always $O(n/p)$, for any dimension $d=O(1)$.

4. Experimental Results

We implemented Algorithm 2 for Problem 1 and Problem 3 (d=3 and d=4) on a CM5. Our CM5 partition had 32 processors with 40 MB memory per processor. All timing results are obtained in multi-user mode. The sequential code for the lower envelope problem was a plane sweep algorithm which we implemented ourselves, and for the convex hull we used the "Quickhull Algorithm" [2]. The observed running times are shown in Figures 1 and 2, and in Table 1.

Curves (a) and (b) in Figure 1 show the local computation times and communication times, respectively, for computing the lower envelope of a random set of non-intersecting line segments (we will discuss the data generation in detail in the full version of the paper). Table 1 shows the number of communication rounds observed, and the exact numerical values of the observed times. For each data

point, we made 20 experiments. The running time indicated is the average of those 20 experiments, together with the double-sided confidence interval with significance level 99.9%. For all data points, the number of communication rounds observed was always identical for all 20 experiments. Our CM5 configuration allowed to process n = 8 Meg (= 8388608) line segments, i.e. n/p = 256 K (= 262144) line segments can be stored in the local memory of one processor. The program required 0.114±0.66% seconds communication time and 13.7±0.06% seconds local computation time. We then simulated a smaller local memory size, n/p, on our given machine. (Note that, for n/p < 256 K we allowed in our simulation only n/p, instead of the possible 256 K, line segments to be compressed into one processor in each iteration of Step 1c in Algorithm 2.) For $n/p \geq 512$ line segments per processor, our algorithm terminated always after one communication round. We observe that the communication time is extremely small, for all cases. Essentially, all time is spent on local computation, were each processor first computes the lower envelope for n/p line segments, and then processor P_1 solves another such problem.

In order to study better the behavior of Algorithm 2, we changed it to iterate Step 1 until the remaining data size is at most n/p, or until a time-out occurs. For n/p between 64 and 256 we observed two communication rounds, for n/p = 32 we needed 4 communication rounds, and for smaller n/p our algorithm was not successful. The latter cases are due to the fact that, for our experiments, p=32 is a small number and, hence, for small ratios n/p we violate Assumption A4 which requires that n is a large number. Clearly, it makes even no sense to solve the lower envelope problem for n < 1 K line segments on a CM5.

Curves (c) and (d) in Figure 1, and the respective rows in Table 1, show the local computation times and communication times, respectively, for computing the lower envelope of a random set of (possibly) *intersecting* line segments. The probabilistic analysis of this case is still an open problem. The obtained empirical results suggest, however, that Algorithm 2 has a very similar behavior for this case.

Figure 2 and the respective rows in Table 1 show the results for 3D and 4D convex hull. The experimental setup was the same as for the lower envelope problem. For all data points, the number of communication rounds observed was always identical for all 20 experiments. For n/p between 256 and 2K, or 1 K and 8 K, we observed for 3D or 4D convex hull, respectively, that Algorithm 2 requires between 2 and 3 communication rounds. For smaller n/p, we have again a violation of Assumption A4 which requires that n is a large number. Since h(n) is a larger function for Problem 3 with d = 3 and d = 4 than for Problem 1, we require larger values of n to obtain high probability for a constant number of communication rounds. For $n/p \geq 4$ K (d = 3) or $n/p \geq 16$ K (d = 4), respectively, our algorithm terminated always after one communication round. Again, we observe that the communication time is extremely small. Hence, we observe a speedup very close to $p/2$.

5. Open Problems

An important problem, which we plan to study in the future, is to extend our methods to non-uniform data distributions. For example, for Algorithm 2 it seems helpful to randomly permute the remaining data at the end of each iteration and simply compress them as much as possible. We conjecture that this will improve the performance and solve at least some cases which are currently not covered. For Algorithm 3, one can adapt the rectangular partitioning scheme to the given distribution and, if necessary, iterate Step 2a. It is an interesting open problem to give a probabilistic analysis of such algorithmic extensions. [7, 9, 13]

6. References

[1] S. G. Akl and K. A. Lyons, *Parallel Computational Geometry*, Prentice Hall, 1993.

[2] C. B. Barber, D. P. Dobkin, and H. Huhdanpaa, "The quickhull algorithm for convex hull," Tech. Report No. GCG53, The Geometry Center, University of Minnesota, Minneapolis, MN 55454, USA, 1993.

[3] J. L. Bentley, H. T. Kung, M. Schkolnick, and C. D. Thompson, "On the average number of maxima in a set of vectors and applications," *J. ACM*, Vol. 25, 1978, pp. 536-543.

[4] J. L. Bentley, Weide, and A. Yao, "Optimal expected time algorithms for closest point problems," *ACM Transactions on Mathematical Software*, Vol. 6, No. 4, 1980, pp. 563-580.
[5] F. Dehne, A. Fabri, and A. Rau-Chaplin, "Scalable parallel computational geometry for coarse grained multicomputers," in Proc. *ACM Symposium on Computational Geometry*, 1993, pp. 298-307.
[6] W. Feller, *An introduction to probability theory and its applications*, John Wiley, 1950.
[7] T. Hagerup and J. Katajainen, "Improved parallel bucketing algorithms for proximity problems," in Proc. *26th Hawaii International Conference on System Sciences*, 1993, pp. 318-327.
[8] C. Mathieu-Kenyon, "The complexity of an axial view of a parallelepipedic scene: an average-cas analysis" unpublished manuscript, also presented at the Journées de Géométrie Algorithmique, INRIA, Sophia-Antipolis, 1990.
[9] P. D. MacKenzie and Q. F. Stout, "Ultra-fast expected time parallel algorithms," in Proc. *2nd ACM-SIAM Symposium on Discrete Algorithms*, 1991, pp. 414-423.
[10] H. Raynaud, "Sur l'envelope convexe des nauges des poits aléatoires dans R^n," *J. Appl. Prob.*, Vol. 7, 1972, pp. 35-48.
[11] A. Rényi and R. Sulanke, "Über die konvexe Hülle von n zufällig gerwählten Punkten," *Z. Wahrsch. Verw. Gebiete*, Vol. 2, 1963, pp. 75-84.
[12] A. Rényi and R. Sulanke, "Zufällige konvexe Polygone in einem Ringgebiet," *Z. Wahrsch. Verw. Gebiete*, Vol. 9, 1968, pp. 146-157.
[13] Q. Stout, "Constant-time geometry on PRAMs," in Proc. *International Conference on Parallel Processing*, 1988, pp. 104-107.

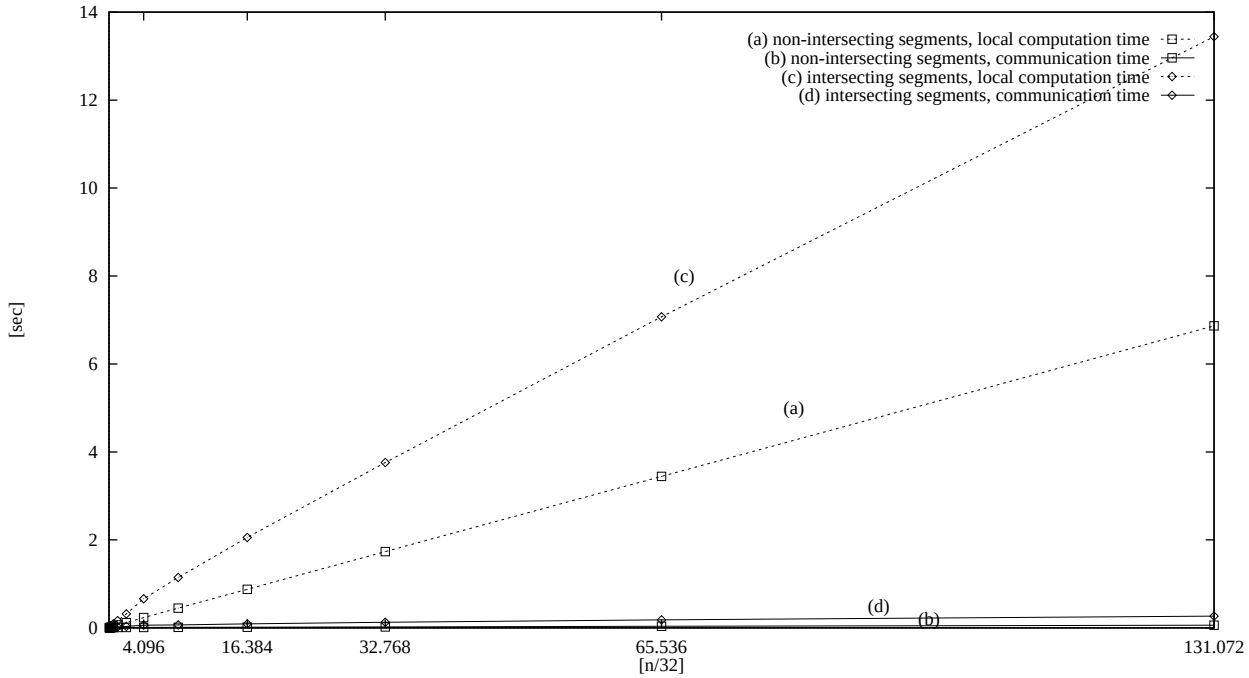

Figure 1: Timing Results For The Lower Envelope Problem.

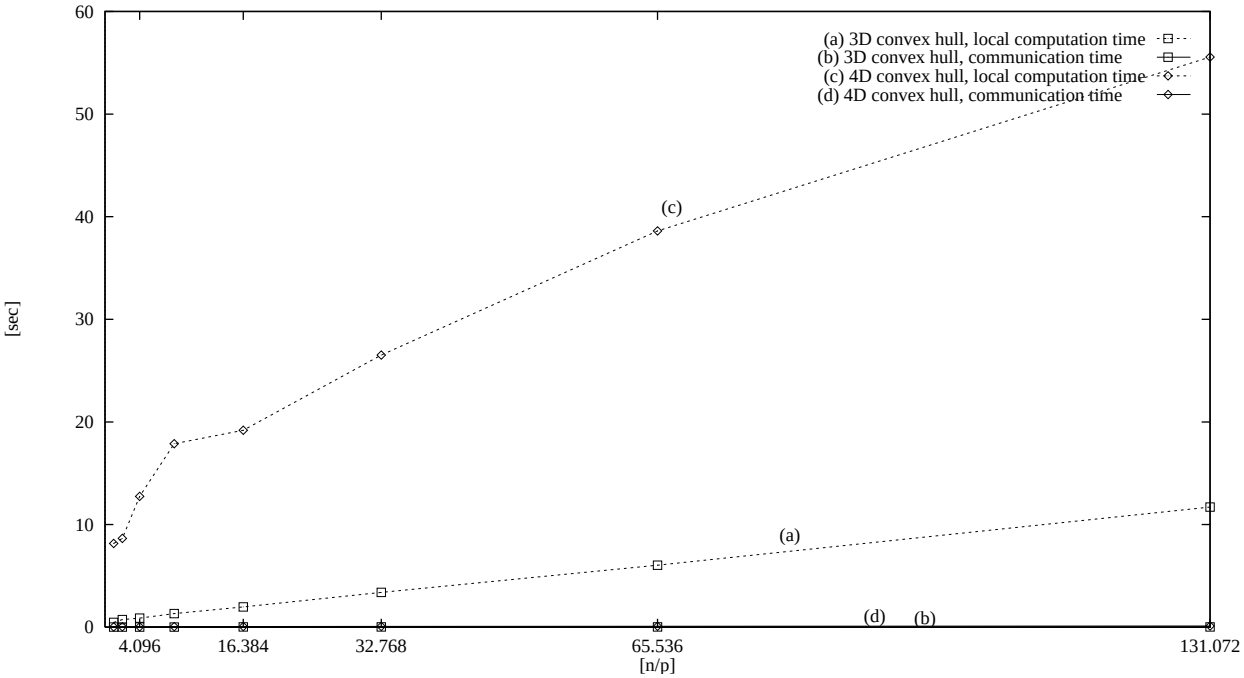

Figure 2: Timing Results For The 3-Dim. and 4- Dim. Convex Hull Problem.

n	1 K	2 K	4 K	8 K	16 K	32 K	64 K	128 K	256 K	512 K	1 Meg	2 Meg	4 Meg	8 Meg
n/p	32	64	128	256	512	1 K	2 K	4 K	8 K	16 K	32 K	64 K	128 K	256 K
Figure 1a&b: Problem 1 (lower envelope)														
rounds	4	2	2	2	1	1	1	1	1	1	1	1	1	1
T_{local} [sec]	0.003 ±11%	0.005 ±4.8%	0.010 ±5.7%	0.020 ±2.4%	0.039 ±3.0%	0.065 ±2.1%	0.119 ±1.2%	0.233 ±1.1%	0.447 ±0.6%	0.876 ±0.4%	1.734 ±0.3%	3.442 ±0.2%	6.866 ±0.1%	13.7± 0.06%
T_{comm} [sec]	0.004 ±7.2%	0.004 ±5.9%	0.004 ±3.1%	0.006 ±9.4%	0.007 ±2.4%	0.008 ±7.7%	0.009 ±7.7%	0.010 ±6.7%	0.012 ±5.2%	0.015 ±3.7%	0.022 ±3.0%	0.035 ±2.1%	0.060 ±1.2%	0.114 ±0.7%
Figure 1c&d: Problem 1 (lower envelope) for intersecting line segments														
rounds	*	5	5	4	2	2	2	2	1	1	1	1	1	1
T_{local} [sec]		0.010 ±3.9%	0.022 ±4.5%	0.050 ±4.8%	0.090 ±5.3%	0.163 ±3.1%	0.318 ±2.0%	0.659 ±1.6%	1.146 ±1.7%	2.054 ±1.1%	3.759 ±0.5%	7.072 ±1.2%	13.44 ±0.5%	25.72 ±0.3%
T_{comm} [sec]		0.006 ±2.5%	0.009 ±8.6%	0.014 ±11%	0.015 ±12%	0.020 ±4.3%	0.032 ±4.0%	0.059 ±33%	0.065 ±3.0%	0.089 ±2.4%	0.126 ±1.8%	0.182 ±4.7%	0.265 ±2.1%	0.372 ±2.0%
Figure 2a&b: Problem 3 (convex hull), dimension $d = 3$														
rounds	*	*	*	3	2	2	2	1	1	1	1	1	1	1
T_{local} [sec]				0.320 ±13%	0.352 ±9%	0.469 ±5.5%	0.740 ±4.4%	0.879 ±7.3%	1.325 ±8.4%	1.975 ±11%	3.388 ±13%	6.035 ±13%	11.70 ±10%	24.52 ±11%
T_{comm} [sec]				0.016 ±5.9%	0.013 ±11%	0.014 ±1.2%	0.016 ±0.7%	0.016 ±1.1%	0.018 ±0.9%	0.020 ±2.3%	0.022 ±1.2%	0.024 ±3.7%	0.026 ±1.0%	0.031 ±14%
Figure 2c&d: Problem 3 (convex hull), dimension $d = 4$														
rounds	*	*	*	*	*	3	2	2	2	1	1	1	1	1
T_{local} [sec]						8.162 ±4.4%	8.659 ±4.3%	12.75 0±3.1 %	17.87 6±4.6 %	19.19 4±5.6 %	26.52 3±3.6 %	38.61 4±6.7 %	55.55 2±5.8 %	91.65 ±6.9%
T_{comm} [sec]						0.025 ±1.6%	0.036 ±91%	0.037 ±15%	0.049 ±1.4%	0.066 ±3.9%	0.082 ±3.5%	0.095 ±0.9%	0.116 ±3.1%	0.137 ±2.1%

Table 1: Table Of Timing Results and Confidence Intervals.