Parallel Algorithms for Higher-Dimensional Convex Hulls

(Preliminary Version)

NANCY M. AMATO*
ICSI & Texas A&M Univ.
amato@cs.uiuc.edu

MICHAEL T. GOODRICH†
Johns Hopkins Univ.
goodrich&cs.jhu.edu

EDGAR A. RAMOS‡
Univ. of Illinois
ramos@cs.uiuc.edu

Abstract

We give fast randomized and deterministic parallel methods for constructing convex hulls in $\mathbb{R}^d$, for any fixed d. Our methods are for the weakest shared-memory model, the EREW PRAM, and have optimal work bounds (with high probability for the randomized methods). In particular, we show that the convex hull of n points in $\mathbb{R}^d$ can be constructed in $O(\log n)$ time using $O(n \log n + n^{\lfloor d/2 \rfloor})$ work, with high probability. We also show that it can be constructed deterministically in $O(\log^2 n)$ time using $O(n \log n)$ work for $d = 3$ and in $O(\log n)$ time using $O(n^{\lfloor d/2 \rfloor} \log^{c(\lceil d/2 \rceil - \lfloor d/2 \rfloor)} n)$ work, for $d \geq 4$, where $c > 0$ is a constant, which is optimal for even $d \geq 4$. We also show how to make our 3-dimensional methods output-sensitive with only a small increase in running time.

These methods can be applied to other problems as well. A variation of the convex hull algorithm for even dimensions deterministically constructs a $(1/r)$-cutting of n hyperplanes in $\mathbb{R}^d$ in $O(\log n)$ time using optimal $O(nr^{d-1})$ work; when $r = n$, we obtain their arrangement and a point location data structure for it. With appropriate modifications, our deterministic 3-dimensional convex hull algorithm can be used to compute, in the same resource bounds, the intersection of n balls of equal radius in $\mathbb{R}^3$. This leads to a sequential algorithm for computing the diameter of a point set in $\mathbb{R}^3$ with running time $O(n \log^3 n)$, which is arguably simpler than an algorithm with the same running time by Brönnimann et al.

1 Introduction

The convex hull is a well studied structure, and the convex hull construction problem is a fundamental problem in computational geometry. Besides being of interest in its own right, the convex hull construction problem has as a dual the important problem of computing the intersection of n halfspaces. Moreover, the construction of d-dimensional Delaunay triangulations and Voronoi diagrams can be reduced to the construction of $(d+1)$-dimensional convex hulls [23]. In $\mathbb{R}^d$, the size of the convex hull of n points is $\Theta(n^{\lfloor d/2 \rfloor})$ in the worst case, and its construction requires $\Omega(n \log n + n^{\lfloor d/2 \rfloor})$ work [23, 48].

*This research supported in part by an AT&T Bell Laboratories Graduate Fellowship and by NSF Grant CCR-9315696.

†This research supported by the NSF under Grants IRI-9116843 and CCR-9300079. A portion of this effort was done while this author was visiting the University of Illinois at Urbana-Champaign.

‡This research supported in part by NSF Grant CCR-9118874.

1.1 Related work

Optimal deterministic sequential algorithms have long been known for the cases $d = 2, 3$ [28, 47]. In higher dimensions, $d \geq 4$, Seidel proposed two deterministic algorithms. His first algorithm [54] ran in $O(n \log n + n^{\lceil d/2 \rceil})$ time[1], which is optimal for even d, and later he gave an $O(n^{\lfloor d/2 \rfloor} \log n)$ solution [55]. For some time, the only solutions optimal in higher dimensions were the randomized incremental algorithm of Clarkson and Shor [15], and the subsequent randomized method of Seidel [56]. Recently, Chazelle [10] gave the first deterministic algorithm that is optimal in higher dimensions, which was simplified by Brönnimann *et al.* [8]. The optimality of the above algorithms is measured with respect to the worst-case size complexity of the resulting convex hull. However, when the size of the output is considered, it may be possible to beat the worst-case lower bounds since the size of the convex hull may range from $O(1)$ to $O(n^{\lfloor d/2 \rfloor})$. Accounting for output size, the lower bound becomes $\Omega(h + n \log h)$, where h is the size of the convex hull. The first *output-sensitive* algorithm, due to Kirkpatrick and Seidel [33], computed the convex hull in $\mathbb{R}^2$ in $O(n \log h)$ time. Clarkson and Shor [15] gave an optimal randomized output-sensitive solution for 3-dimensional convex hulls, which was optimally derandomized by Chazelle and Matoušek [13]. In higher dimensions, the only deterministic output-sensitive method known, due to Seidel [55], runs in time $O(n^2 + h \log n)$, which can be slightly improved to $O(n^{2-(2/(\lfloor d/2 \rfloor + 1))+\epsilon} + h \log n)$, for any fixed $\epsilon > 0$, using a technique of Matoušek [42]. All of these methods for $d \geq 3$ seem inherently sequential.

The parallel construction of the convex hull has also received much attention. For exclusive-write PRAMs (the EREW and CREW models) it is known that $\Omega(\log n)$ time is required to compute the convex hull in $\mathbb{R}^d, d \geq 2$ [19]. Optimal deterministic 2-dimensional convex hull algorithms running in $O(\log n)$ time using $O(n \log n)$ work for the CREW PRAM were given by Atallah and Goodrich [6, 7] and Aggarwal *et al.* [2], and for the EREW PRAM by Miller and Stout [44]. For 3-dimensional convex hulls, using n processors on a CREW PRAM, $O(\log^3 n)$ time was achieved by Chow [14] and Aggarwal *et al.* [2], $O(\log^2 n \log^* n)$ time was obtained by Dadoun and Kirkpatrick [20], and $O(\log^2 n)$ time was achieved Amato and Preparata [3]. For some time, the only solution to the 3-dimensional convex hull problem optimal with respect to time or work was the $O(\log n)$ time and $O(n \log n)$ work randomized algorithm for the CREW PRAM of Reif and Sen [51]. Recently, by de-

[1] Throughout this paper we assume d is a fixed constant.

randomizing Reif and Sen's algorithm, Goodrich obtained an $O(\log^2 n)$ time work-optimal method for the EREW PRAM, and a time-optimal method using $O(n^{1+\epsilon})$ work was given by Amato and Preparata [4] for the CREW PRAM, where $\epsilon > 0$ is any fixed constant. There is also a parallel output-sensitive algorithm by Ghouse and Goodrich [25]. Using the CRCW PRAM model, they give an $O(\log n)$ time and $O(n \log h)$ work method for $\mathbb{R}^2$, and an $O(\log^2 n)$ time and $O(\min\{n \log^2 h, n \log n\})$ work method for $\mathbb{R}^3$. We know of no previous parallel algorithms for d-dimensional convex hull construction for $d > 3$.

1.2 Our results

We give fast randomized and deterministic parallel methods for constructing the convex hull of a set P of n points in $\mathbb{R}^d$, which is the smallest convex set containing the points in P. Our methods are actually for the dual problem of computing the intersection $H^\cap$ of a set H of n halfspaces in $\mathbb{R}^d$ containing some known point o. In particular, we give $O(\log n)$-time randomized parallel algorithms that use optimal $O(n \log n + n^{\lfloor d/2 \rfloor})$ work, with high probability. In addition, we give deterministic methods that run in $O(\log^2 n)$ time using $O(n \log n)$ work for $d = 3$ and in $O(\log n)$ time using $O(n^{\lfloor d/2 \rfloor} \log^{c(\lceil d/2 \rceil - \lfloor d/2 \rfloor)} n)$ work, for $d \geq 4$, where $c > 0$ is a constant, which is optimal for even d. For $d = 3$ we also show how to make our methods output-sensitive so as to have a work bound of $O(n \log h)$ while only slowing down the construction by an $O(\log n)$ factor. Our methods are for the EREW PRAM, which is the weakest of the synchronous shared-memory models. It is also the easiest to simulate on more-realistic parallel models, and, by a scheme due to Cole [16, 17], algorithms for this model can sometimes be used to derive faster sequential parametric searching algorithms (e.g., see [1, 11, 43]) than would be possible using concurrent-read parallel methods.

Our methods are all based upon a parallel divide-and-conquer scheme, where one subdivides the space into cells that should contain fewer halfspaces and then recurses on each cell in parallel. The difficulty in applying this approach, however, is that existing methods for producing efficient partitions would produce a constant-factor "blow up" in the total problem size with each recursive call. Eventually, this blow-up leads to the total problem size becoming too large to process optimally. We get around this problem using a number of new ideas. For example, in our randomized method for $d \geq 4$ we show how to use a parallel analogue to Matoušek's shallow-cutting lemma [39], together with a technique we call *biased sampling*, to "garbage collect" the size blow-up after a certain number of iterations. (Incidentally, this biased sampling technique was recently discovered independently in a slightly different form by Rajasekaran and Ramaswami [49].) In addition, so as to get an n-polynomial probability[2] bound, rather than just an expected-time bound, we use a duration-unknown scheduling lemma for randomized computations. In our optimal deterministic method for even dimensions we get around the size blow-up problem by using a shallow-cutting analogue to a partition sparsity concept introduced by Chazelle [9]. Finally, we get around the size blow-up problem for the 3-dimensional halfspace intersection by using a new "pruning" computation, which removes halfspaces that cannot ultimately contribute vertices to the final intersection. This type of technique was first introduced by Reif and Sen [51] in their randomized 3-dimensional halfspace intersection algorithm. Our pruning computation is quite different from theirs, however, and is considerably simpler. Incidentally, Reif and Sen pose as open problems whether one can achieve $O(\log n)$ time and optimal-work with high probability in the EREW PRAM model, and whether one can improve the confidence for randomized CREW PRAM $O(\log n)$-time optimal-work construction from n-polynomial to n-exponential. Our methods answer both of these questions in the affirmative.

The methods developed in our convex hull algorithms can be applied to other problems as well. We give a simple variation of our convex hull algorithm for even dimensions that deterministically constructs a $(1/r)$-cutting of n hyperplanes in $\mathbb{R}^d$ in $O(\log n)$ time using optimal $O(nr^{d-1})$ work. When $r = n$, this method yields the entire hyperplane arrangement $\mathcal{A}(H)$ and a point location data structure for it. We also show that with appropriate modifications, our deterministic 3-dimensional convex hull algorithm can be used to compute, in the same resource bounds, the intersection of n balls of equal radius in $\mathbb{R}^3$. Using the algorithm for ball intersection together with parametric search [43] as in previous works [11, 40], we obtain a sequential algorithm for computing the diameter of a point set in $\mathbb{R}^3$ with running time $O(n \log^3 n)$ that is arguably simpler than the algorithm with the same running time by Brönnimann *et al*. [8].

We present some important constructions for hyperplane set systems in the next section. In Section 3 we give our convex hull methods for $d \geq 4$, and we give some specialized methods for $d = 3$ in Section 4.

2 Hyperplane Set Systems

We begin by describing a general framework for set systems, which is most similar to a framework given by Clarkson and Shor [15]. Let $(X, \mathcal{F})$ be an *$\mathcal{F}$-generated set system*, i.e., let X be a set of n elements and let $\mathcal{F}$ be a function, called the generator function, that maps subsets of X to subsets of 2^X, whose elements are commonly referred to as *ranges*. For example, in what we will call the *hyperplane set system*, X would denote a set of hyperplanes and the set of ranges $\mathcal{F}(Y), Y \subseteq X$, could represent all the subsets of X defined by the intersection of the hyperplanes in X with the interiors of simplices whose vertices are also vertices in the arrangement $\mathcal{A}(Y)$ of the hyperplanes in Y. The

[2] We use "n-polynomial" to refer to a probability that is at least $1 - 1/n^c$, for some constant $c \geq 1$; we use "n-exponential" to refer to a probability that is at least $1 - 1/2^{n^\delta}$, for some constant $\delta > 0$.

VC-exponent [5][3] of such a set system is the infimum of all numbers k such that $|\mathcal{F}(Y)|$ is $O(|Y|^k)$. For set systems with finite VC-exponent, D, we will assume that $\mathcal{F}$ is defined so that each range in $R \in \mathcal{F}(Y)$ is determined by some subset of at most D members of Y, which are called the *triggers* for R. It is easy to see, for example, that the hyperplane set system has finite VC-exponent, since there are at most $O(\binom{|Y|}{d(d+1)})$ combinatorially distinct simplices defined by vertices in $\mathcal{A}(Y)$. That is, each range in $R \in \mathcal{F}(Y)$ is determined by at most $d(d+1)$ "trigger" hyperplanes. Let us therefore assume for the remainder of this section that we are dealing with set systems with finite VC-exponent.

Adapting a definition from [57], we say that a $(1/r)$-*approximation* [37] of X is a subset Y such that, for any $R \in \mathcal{F}(X)$,

$$\left| \frac{|Y \cap R|}{|Y|} - \frac{|R|}{|X|} \right| < \frac{1}{r}.$$

Matoušek [37] shows how to compute a $(1/r)$-approximation of size $O(r^2 \log r)$ in $O(nr^{O(1)})$ time. Specializing to hyperplanes, we also have the following:

Lemma 2.1 *One can construct a $(1/r)$-approximation of a set H of n hyperplanes in $\mathbb{R}^d$ with the following complexities in the EREW PRAM model:*

1. *size $O(n^\epsilon r^{2+\epsilon})$ in $O(\log n)$ time using $O(nr^c)$ work, for any fixed constant $\epsilon > 0$ and some constant $c > 1$,*
2. *size $O(n^\epsilon r^2)$ in $O(\log n)$ time using $O(n)$ work, for any fixed constant $\epsilon > 0$ and some constant $c > 1$, which is a $(1/r)$-approximation with n-exponential probability*[4]*,*
3. *size $O(r^d)$ in $O(\log n \log r)$ time using $O(n \log r)$ work, provided $r \leq n^\delta$, for some constant $\delta > 0$.*

Proof: Part (1) is a special case of a result of Goodrich [27] for set systems with finite VC-exponent. Part (2) follows by taking a random sample of size $O(n^\epsilon r^2)$, which by a Chernoff bound (e.g., see [29]), can be shown to form a $(1/r)$-approximation with n-exponential probability. Part (3) follows from a straightforward parallelization of a result of Matoušek [41]. ■

As it turns out, these size bounds are too large to be of direct use in our convex hull algorithms, however.

2.1 Semi-nets

Let Y be a subset of X, and let a parameter $r \in [1, n]$ be given. Further, let $N_Y(s, X)$ denote the number of ranges from X generated by $\mathcal{F}(Y)$ of size s such that $Y \cap R = \emptyset$. We say such ranges are *missed* by Y. Define $f_0(r)$ to be the expected number of missed ranges generated by an r-sized (fully independent) random sample S of X (with all such samples equally likely). Y is a $(1/r)$-*semi-net*[5] of order $\omega \geq 0$ if

$$\sum_{0 \leq t \leq r} N_Y(tn/r, X) \max\{t^\omega, 1\} \;=\; O(f_0(r)),$$

where the sum ranges over all values of t from 0 to r for which $N_Y(tn/r, X)$ is non-zero. The next lemma implies that a suitably-defined random sample is a $(1/r)$-semi-net, even if it is defined by random variables that are only k-wise independent. It is a k-wise independent version of a result of Chazelle and Friedman [12].

Lemma 2.2 *Let $(X, \mathcal{F})$ be an $\mathcal{F}$-generated set system with constant VC-exponent D. If Y is a subset of X defined by n k-wise independent indicator random variables, for $k \geq 3D + 2\omega + 4$, each of which is 1 with probability r/n, then, with probability at least 1/2, Y is a $(1/r)$-semi-net of order ω with size $O(r)$, provided that f_0 is non-decreasing.*

Proof: Goodrich [27] shows that if Y is defined as above, then with probability at least 3/4, $|Y| = r \pm \Theta(r^{1/2})$. Let $\mathcal{F}_s(X)$ denote the set of ranges in $\mathcal{F}(X)$ of size s. To establish the probability that Y is a $(1/r)$-semi-net, let us apply Markov's inequality to a bound on the expectation

$$E[\sum_{0 \leq t \leq r} N_Y(tn/r, X) \max\{t^\omega, 1\}]. \tag{1}$$

If we let p_1 denote the probability $\Pr(R \in \mathcal{F}(Y))$ and we let p_2 denote the probability $\Pr(Y \cap R = \emptyset \mid R \in \mathcal{F}(Y))$, then we can re-write (1) as

$$\sum_{0 \leq t \leq r} \sum_{R \in \mathcal{F}_{\frac{tn}{r}}(X)} p_1 p_2 \max\{t^\omega, 1\}.$$

Using a result from [27], we can bound p_1 by $\min\{b/t^{(k-D)/2}, 1\}$, for some constant $b > 1$. Moreover, this requires just $(k - D)$-wise independence [27]. Thus, since each range in $\mathcal{F}(Y)$ is determined by at most D triggers, we can bound p_2 by $(r/n)^D$ using the additional D-wise independence available in the random variables. Thus, we can bound (1) by

$$b \sum_{0 \leq t \leq r} |\mathcal{F}_{\frac{tn}{r}}(X)|(r/n)^D \cdot \min\{t^{-(D+\omega+2)}, 1\} \cdot \max\{t^\omega, 1\},$$

which can be re-written

$$b \sum_{0 \leq t < 1} |\mathcal{F}_{\frac{tn}{r}}(X)|(r/n)^D + b \sum_{1 \leq t \leq r} |\mathcal{F}_{\frac{tn}{r}}(X)|(r/n)^D t^{-(D+2)}.$$

Clarkson and Shor [15] show (using a slightly different notation) that

$$\sum_{0 \leq s \leq k} |\mathcal{F}_s(X)| \leq C(k)^D f_0(n/k),$$

[3] There is a related notion, known as the VC-dimension [30, 37], and this is subsumed in the above definition, since a set system with VC-dimension d has VC-exponent d as well [53, 57].

[4] Note that the probability is on the event that the construction yields a $(1/r)$-approximation, not on the running time.

[5] Our definition of a semi-net is motivated by the $(1/r)$-semi-cutting notion introduced by Chazelle [10], as well as proof techniques given in [12, 27, 38].

for some constant $C > 0$. Thus, the expectation (1) is at most

$$bCf_0(r) + b\sum_{1\le t\le r}(tn/r)^D f_0(r/t)(r/n)^D t^{-(D+2)},$$

which can be bounded by

$$bCf_0(r) + b\sum_{1\le t\le r} f_0(r/t)t^{-2} \le af_0(r),$$

for some constant $a > 1$, provided that f_0 is non-decreasing. ■

This, in turn, allows us to derive the following:

Lemma 2.3 *Let $(X, \mathcal{F})$ be an $\mathcal{F}$-generated set system with finite VC-exponent. Then one can construct a $(1/r)$-semi-net of constant order ω for $(X, \mathcal{F})$ of size $O(r)$ in $O(\log n)$ time using $O((nr)^c)$ work, for some constant $c > 1$, in the EREW PRAM model.*

Proof: (Sketch) Given Lemma 2.2, the proof is a straightforward application of the limited independence parallel derandomization technique (e.g., see Luby [34, 35] or Karloff and Mansour [32]). ■

The only possibly difficult step in the above proof is in the computation of $f_0(r)$ for a given set system. Note, however, that it is sufficient for us to use an upper bound for $f_0(r)$ in such a case, as that can only improve the probability of acceptance of some Y. In any case, the above construction is, of course, not work-efficient. To achieve work efficiency we employ a "composition" lemma like the following:

Lemma 2.4 *If A is a $(1/2r)$-approximation to a set system $(X, \mathcal{F})$, and Y is a $(1/2r)$-semi-net of order ω for A, then Y is a $(1/r)$-semi-net of order ω for X, provided $f(2r) \le 2^D f(r)$ for some constant $D > 0$.*

Proof: The sum

$$\sum_{0\le t\le r} N_Y(tn/r, X)\max\{t^\omega, 1\} \qquad (2)$$

can be viewed as a weighted sum over all ranges in $\mathcal{F}(Y)$ with each range R being given weight $\max\{(|R|r/n)^\omega, 1\}$. Since Y is a $(1/2r)$-semi-net for A,

$$\sum_{0\le t\le 2r} N_Y(t|A|/2r, A)\max\{t^\omega, 1\} = O(f_0(2r)), \qquad (3)$$

which can be viewed as a weighted sum over all ranges in $\mathcal{F}(Y)$ (restricted to A) with each range $A \cap R$, $R \in \mathcal{F}(Y)$, being given weight $\max\{(|A \cap R|2r/|A|)^\omega, 1\}$. Since A is a $(1/r)$-approximation, $|A \cap R| \ge |A|(|R|/n - 1/2r)$. Thus, for each range R given weight t^ω in (2) there is a corresponding range $A \cap R$ given weight at least $(2t-1)^\omega$ in (3). Therefore, since $(2t-1)^\omega \ge t^\omega$ for $t \in [1, r]$,

$$\sum_{0\le t\le r} N_Y(tn/r, X)\max\{t^\omega, 1\} = O(f_0(2r)),$$

which is $O(f_0(r))$ by assumption. ■

This immediately gives us the following:

Lemma 2.5 *Let $(X, \mathcal{F})$ be an $\mathcal{F}$-generated set system with finite VC-exponent, so that $f(2r) \le 2^D f(r)$ for all $1 \le r \le n$. Then one can construct a $(1/r)$-semi-net of constant order ω for $(X, \mathcal{F})$ of size $O(r)$ in the EREW PRAM model with the following complexities:*

1. *in $O(\log n)$ time using $O(nr^c)$ work, for some constant $c > 1$, or*
2. *$O(\log n)$ time using $O(n \log r)$ work, provided that $r \le n^\alpha$, for some constant $\alpha > 0$, yielding a $(1/r)$-semi-net with n-exponential probability, or*
3. *$O(\log n \log r)$ time using $O(n \log r)$ work, provided that $r \le n^\alpha$, for some constant $\alpha > 0$.*

Proof: Each of the bounds follows from applying Lemma 2.4 to a $(1/2r)$-semi-net constructed via Lemma 2.3 from a $(1/2r)$-approximation constructed via Lemma 2.1. ■

We next give an application of this lemma in the hyperplane set system.

2.2 Cuttings

Let H be a collection of n hyperplanes in $\mathbb{R}^d$. For any simplex s, let $H_{|s}$ denote the set of hyperplanes of H intersecting the interior of s. The set $H_{|s}$ is often referred to as the *conflict list* for s relative to H [15]. In hyperplane set systems the ranges are $H_{|s}$ sets. A $(1/r)$-*cutting* [38] of H is a partition $\mathcal{C}$ of (possibly unbounded) d-simplices that cover $\mathbb{R}^d$ and such that $|H_{|s}| \le n/r$ for each $s \in \mathcal{C}$. Chazelle and Friedman [12] show that there exists a $(1/r)$-cutting of size $O(r^d)$, where the size of a $(1/r)$-cutting $\mathcal{C}$ is the number of simplices in $\mathcal{C}$. Chazelle [9] shows that such a $(1/r)$-cutting can be constructed in $O(nr^{d-1})$ time, for any $r \in [1, n]$, and Matoušek [41] shows that such a cutting can be constructed in $O(n \log r)$ time for $r \le n^\alpha$, for some small constant $\alpha > 0$ that depends upon d. Goodrich [27] gives parallel analogues to these results, showing that $(1/r)$-cuttings of size $O(r^d)$ can be constructed in $O(\log n \log r)$ time in the EREW PRAM model with $O(nr^{d-1})$ work for any $r \in [1, n]$ (the $\log r$ factor is removed in subsection 3.5) and with $O(n \log r)$ work for $r \le n^\alpha$.

For the purposes of convex hull construction, however, these results are not quite what we need, for a generic $(1/r)$-cutting has too many simplices. We can produce more specialized $(1/r)$-cuttings, however, via semi-nets.

As we have already observed, in the case of hyperplane set systems, ranges in $\mathcal{F}(Y)$ are determined by simplices, whose vertices are taken from the vertices of $\mathcal{A}(Y)$. Define $\mathcal{F}_0(Y)$ to be the set of simplices for ranges generated by $\mathcal{F}(Y)$ that miss Y (i.e., a simplex s is in $\mathcal{F}_0(Y)$ if its corresponding range $H_{|s}$ is in $\mathcal{F}(Y)$ and $Y \cap R = \emptyset$). In addition, define $\mathcal{T}(Y)$ to be the set of simplices in a canonical triangulation [12] of the arrangement $\mathcal{A}(Y)$ of Y, restricted to those simplices in $\mathcal{F}_0(Y)$.

Lemma 2.6 *Let H be a collection of n hyperplanes in $\mathbb{R}^d$, and let Y be an $O(r)$-sized $(1/r)$-semi-net for H of order at least ω (in some hyperplane set system), for some constant $\omega > d+1$. Given $\mathcal{T}(Y)$, together with the conflict lists for its simplices, one can construct a $(1/r)$-cutting of size $O(|\mathcal{T}(Y)|)$ (complete with its conflict lists) in $O(\log n)$ time using $O(|\mathcal{T}(Y)|n/r)$ work in the EREW PRAM model.*

Proof: We use an adaptation of proof techniques used by Chazelle and Friedman [12] and Matoušek [38]. For each simplex s in $\mathcal{T}(Y)$, if $|H_{|s}| > n/r$, then we form a $(1/t)$-cutting $\mathcal{C}_s$ of size at most $O(t^{d+1})$ for $H_{|s}$, where $t = |H_{|s}|r/n$. This can be done deterministically in $O(\log n)$ time using $O((n/r)t^{\omega})$ work, for some constant $\omega > d+1$, by a method of Goodrich [27]. We can also easily construct the new conflict lists with this complexity. If Y has order ω, then the total size and work bounds are as claimed above. ■

On the surface this appears to be no better than the existing $(1/r)$-cutting constructions [9, 12, 27, 41], for in the standard hyperplane set system $|\mathcal{T}(Y)|$ is $O(r^d)$. We can do better than this, however, by restricting the way ranges are generated in our set system.

2.3 Shallow cuttings

The first such restriction we consider is to "shallow" cuttings [39]. Let o denote a fixed origin with respect to $\mathbb{R}^d$. Define the *level* of a point $p \in \mathbb{R}^d$ relative to H to be the number of hyperplanes in H that are crossed by the open segment $\overline{op}$. Given $l \leq n$, a collection $\mathcal{C}$ of simplices in $\mathbb{R}^d$ is an *l-shallow $(1/r)$-cutting* of H if the simplices in $\mathcal{C}$ cover all points of level at most l (and possibly more than this) and if $|H_{|s}| \leq n/r$ for each $s \in S$. Matoušek [39] shows that an l-shallow $(1/r)$-cutting of n hyperplanes in $\mathbb{R}^d$ of size $O(r^{\lfloor d/2 \rfloor}(l(r/n)+1)^{\lceil d/2 \rceil})$ can be constructed in polynomial time for any $r \in [1, n]$, and in $O(n \log r)$ time for $r \leq n^{\alpha}$. Our construction loosely follows his.

Let Y be a given subset of H. We define a set system, which we call the *l-shallow hyperplane system* so that $\mathcal{F}(Y)$ is the set of subsets of H that are defined by simplices generated by Y in the usual hyperplane set system restricted to simplices that contain some point on level at most l in $\mathcal{A}(H)$. This set system is a subset of the usual hyperplane set system; hence, it too has finite VC-exponent.

Fact 2.7 ([15]) *The number of vertices on level at most l in $\mathcal{A}(H)$ is $O(n^{\lfloor d/2 \rfloor}(l+1)^{\lceil d/2 \rceil})$.*

Lemma 2.8 *Let Y be an $O(r)$-sized $(1/r)$-semi-net of H of order $\omega \geq 0$ (with respect to the l-shallow hyperplane set system). Then $|\mathcal{T}(Y)|$ is $O(r^{\lfloor d/2 \rfloor}(l(r/n)+1)^{\lceil d/2 \rceil})$.*

Proof: Note that $|\mathcal{T}(Y)| \leq \sum_{0 \leq t \leq r} N_Y(tn/r, H)$ (with respect to the l-shallow hyperplane set system); hence, it is $O(f_0(r))$, where, in this case, $f_0(r)$ is proportional to the expected number of vertices in $\mathcal{T}(S)$, taken over all r-sized random samples $S \subseteq H$. The probability that any vertex on level at most l in $\mathcal{A}(H)$ becomes a vertex in such a $\mathcal{T}(S)$ is $(r/n)^d$. Thus, by Fact 2.7 and the linearity of expectation, $\mathbf{E}(|\mathcal{T}(S)|)$ is $O((r/n)^d n^{\lfloor d/2 \rfloor}(l+1)^{\lceil d/2 \rceil})$, which can also be expressed as $O(r^{\lfloor d/2 \rfloor}(l(r/n)+1)^{\lceil d/2 \rceil})$. ■

Thus, by Lemma 2.3, we can construct a semi-net Y as above in $O(\log n)$ time using polynomial work. We can also derive a composition lemma for such "shallow" semi-nets:

Lemma 2.9 *If A is a $(1/2r)$-approximation to H in the hyperplane set system and Y is a $(1/2r)$-semi-net of order ω for A in the l'-shallow set system, where $l' = |A|(l/n+1/r)$, then Y is a $(1/r)$-semi-net of order ω for H in the l-shallow set system.*

Proof: The proof follows that for Lemma 2.4, with the added observation that any point on level at most l in $\mathcal{A}(H)$ must be on level at most l' in $\mathcal{A}(A)$. ■

This allows us to derive the following:

Theorem 2.10 *Given a set H of n hyperplanes in $\mathbb{R}^d$, one can construct an l-shallow $(1/r)$-cutting for H, including conflict lists, with $k = O(r^{\lfloor d/2 \rfloor}(l(r/n)+1)^{\lceil d/2 \rceil})$ simplices, in the EREW PRAM model with the following complexities:*

1. *in $O(\log n)$ time using $O(nr^c + nk)$ work, for some constant $c > 1$,*
2. *$O(\log n)$ time using $O(n \log n + kn/r + k^c)$ work, for some constant $c > 1$, with n-polynomial probability (or allowing for concurrent reads), provided that $r \leq n^{\alpha}$, for some constant $\alpha > 0$,*
3. *$O(\log n \log r)$ time using $O(n \log r + kn/r + k^c)$ work, for some constant $c > 1$, provided that $r \leq n^{\alpha}$, for some constant $\alpha > 0$.*

Proof: Lemma 2.1 gives the complexities for constructing a $(1/2r)$-approximation A and Lemma 2.3 gives the complexity of constructing a $(1/2r)$-semi-net Y for A. In $O(\log n)$ time and $O(r^c)$ work it is a simple matter to then construct $\mathcal{T}(Y)$. In order to apply Lemma 2.6, then, to complete the construction, we need only construct the $H_{|s}$ lists for each $s \in \mathcal{T}(Y)$. We can easily do this in $O(\log n)$ time and $O(nk)$ work in the EREW PRAM model. Alternatively, we could build a data structure for $\mathcal{T}(Y)$ (actually, its dual) and use this structure to build each of the $H_{|s}$ lists. The method, which we describe in the full version, yields a running time of $O(\log n)$ time using $O(n \log n + k^c + nk/r)$ work, with n-polynomial probability (or using concurrent reads), or, alternatively in $O(\log n \log r)$ time with $O(n \log r + k^c + nk/r)$ work. ■

2.4 Sparse cuttings

Another notion that will prove important in some of our methods is that of *sparsity*. This concept was introduced by Chazelle [10], extended to the parallel domain by Goodrich

[27], and, more recently, used by Pellegrini [46] in the design of efficient sequential data structures.

Define $\mathcal{V}(H,s)$ to be the set of all vertices in an arrangement $\mathcal{A}(H)$ of a set of hyperplanes, H, that are in the interior of a simplex s. In addition, define a restriction of the l-shallow set systems to the hyperplanes in $H_{|s}$ and to simplices $s \cap s'$ for each simplex s' generated by the usual hyperplane set system. Call this the *s-restricted l-shallow* set system, and observe that it too has finite VC-exponent. In addition, define an *s-restricted l-shallow $(1/r)$-cutting* to be an l-shallow $(1/r)$-cutting for the hyperplanes in $H_{|s}$, restricted to completely lie in s.

Lemma 2.11 *Given a simplex s, let Y be an $O(r)$-sized $(1/r)$-semi-net of $H_{|s}$ of constant order $\omega \geq 0$ (with respect to the s-restricted l-shallow hyperplane set system). Then $|\mathcal{T}(Y)|$ is $O(r^{\lfloor d-1/2 \rfloor}(l(r/n)+1)^{\lceil d-1/2 \rceil} + (r/n)^d|\mathcal{V}(H,s)|)$.*

Proof: The proof follows that of Lemma 2.8 for the first term, which reflects the complexity of the simplices in $\mathcal{T}(Y)$ that intersect the boundary of s. For the second term simply observe that the probability that a vertex in $\mathcal{V}(H,s)$ becomes a vertex in $\mathcal{A}(Y) \cap s$ is $(r/n)^d$. ■

Thus, by Lemma 2.3, we can construct such a semi-net in $O(\log n)$ time using polynomial work. To do this more efficiently, however, we need yet another composition lemma.

Lemma 2.12 *Given a simplex $s \in \mathbb{R}^d$, let A be a $(1/r^d)$-approximation of $H_{|s}$ in the standard hyperplane set system, and let Y be a $(1/2r)$-semi-net of order ω for A in the s-restricted l'-shallow set system, where $l' = |A|(l/n + 1/r)$. Then Y is a $(1/r)$-semi-net of order ω for $H_{|s}$ in the s-restricted l-shallow set system with $|\mathcal{T}(Y)|$ being $O(r^{\lfloor (d-1)/2 \rfloor}(l(r/n)+1)^{\lceil (d-1)/2 \rceil} + (r/n)^d|\mathcal{V}(H,s)|)$.*

Proof: The proof follows that of Lemmas 2.4 and 2.9 to establish the semi-net and shallowness properties for Y. This also establishes the first term in the size bound for $|\mathcal{T}(Y)|$. Thus, we have only to establish the second term in that bound. Chazelle [9] shows that $|\mathcal{V}(A,s)|$ can be used to estimate $|\mathcal{V}(H,s)|$:

$$\left| \frac{|\mathcal{V}(H,s)|}{|H|^d} - \frac{|\mathcal{V}(A,s)|}{|A|^d} \right| < \frac{1}{r^d}.$$

Thus, $(r/|A|)^d|\mathcal{V}(A,s)|$ is $O((|A|/n)^d|\mathcal{V}(H,s)|+1)$, which establishes the lemma. ■

Finally, we can use this to prove the following:

Theorem 2.13 *Given a simplex s and a set $H_{|s}$ of n hyperplanes in $\mathbb{R}^d$, one can construct an s-restricted l-shallow $(1/r)$-cutting for $H_{|s}$, including conflict lists, with $k = O(r^{\lfloor d-1/2 \rfloor}(l(r/n)+1)^{\lceil d-1/2 \rceil} + (r/n)^d|\mathcal{V}(H,s)|)$ simplices, in the EREW PRAM model with the complexities being the same as in Theorem 2.10 (using this value of k).*

Proof: The proof follows that of Theorem 2.10. ■

This parallel result will prove useful in our deterministic method for even dimensions.

Having presented the important ingredients in our methods, let us now describe our algorithms.

3 d-Dimensional Convex Hulls

Given a set H of n halfspaces in $\mathbb{R}^d$ containing a point o, we show in this section how to optimally construct the intersection $H^\cap$ in parallel. Let us first give a simple sub-optimal method, however.

3.1 A simple sub-optimal method

We begin by observing that Theorem 2.10 can be used to design a fast, simple EREW PRAM algorithm that uses $O(n^{\lfloor d/2 \rfloor}\log^c n)$ work, for some constant $c > 0$. We begin by finding a 0-shallow $(1/r)$-cutting, T, of size $O(r^{\lfloor d/2 \rfloor})$, where $r = n^\alpha$, together with its conflict lists, using Theorem 2.10(1), for some constant $\alpha < 1/(2c+d)$. We then recurse on each non-empty $H_{|s}$, $s \in T$. At the bottom level, when the problem size is constant, we complete the construction using a "brute force" method. It is easy to show that there are $O(\log\log n)$ levels in this recursion and the work bound can be characterized $w(n) \leq Cn^\alpha w(n^{1-\alpha}) + O(n^{1+\alpha(\lfloor d/2 \rfloor + c)})$. Thus, we get the following:

Lemma 3.1 *Given a set H of n halfspaces (all containing o) in $\mathbb{R}^d$, $d \geq 4$, one can compute $H^\cap$ in $O(\log n)$ time using $O(n^{\lfloor d/2 \rfloor}\log^c n)$ work, for some constant $c > 0$, in the EREW PRAM model.*

3.2 Optimal expected work

The above procedure runs very fast, but it is not work efficient. As mentioned above, Chazelle [10] gives a sequential deterministic method for intersecting d-dimensional halfspaces that is optimal, but which is difficult to parallelize. This is because it involves a seemingly inherently-sequential application of the conditional probabilities derandomization method. Interestingly, the randomized method Chazelle gives is easy to parallelize to run in $O(\log^2 n)$ expected time using $O(n^{\lfloor d/2 \rfloor})$ expected work in the EREW PRAM model. It is not so clear how one could speed this running time up, however, nor is it clear how one could increase the success probability without making the work bound sub-optimal. To achieve these results we design a new method.

3.3 Optimal work with high probability

To perform this construction faster and with higher probability we begin by forming a random sample S of H of size $r = n/\log^{c_0} n$, for some constant $c_0 > 0$ to be determined in the analysis. By Lemma 2.2, S is a $(1/r)$-semi-net with

probability at least $1/2$. We then run the $O(\log n)$-time, work-inefficient method of Lemma 3.1 on H, except that we only consider hyperplanes from S when forming the shallow cuttings. This is the *bias* in our sampling. We terminate this procedure when we have formed a complete description of $S^{\cap}$, together with its triangulation T and a representation of the conflict list, with respect to H, for each edge in this triangulation. We then remove all edges from T that do not belong to a canonical triangulation of $S^{\cap}$, and form a canonical triangulation T' using the edges that remain.

We can choose c_0 large enough so that the total work for this step is $O(n^{\lfloor d/2 \rfloor}/\log n)$. This gives us all the preconditions needed to be able to apply Lemma 2.6 to construct a 0-shallow $(1/r)$-cutting for H, together with all the conflict lists for this cutting. If S is indeed a $(1/r)$-semi-net, then the size of this cutting is $O(r^{\lfloor d/2 \rfloor})$, and each conflict list has size at most n/r. Thus, we may complete the construction by applying the following Duration-Unknown Scheduling Lemma[6] to the collection of tasks defined by running the method of Section 3.2 on $H_{|s}$, for each simplex s in T':

Lemma 3.2 *Suppose one is given a set of n tasks, each of which runs (independently) in t steps with w work on an EREW PRAM, with probability at least $1/2$. Then one can perform all n tasks in $O(\log n + t \log\log n)$ time using $O(nw)$ work on an EREW PRAM, with n-exponential probability.*

Proof: (Sketch) The method is based upon be a combination of parallel divide-and-conquer and failure sweeping [25, 36]. If n is smaller than some suitably large constant, we solve the duration-unknown scheduling problem by replicating the n tasks a constant number of times and running all copies in parallel. Otherwise, we divide the n tasks into $n^{1/2}$ groups of size $n^{1/2}$ each and recursively solve the duration-unknown scheduling problem for each in parallel, except that we terminate any recursive calls that take more than the specified time with their $n^{1/2}w/t$ processors. We then perform a parallel prefix computation to compress all the unfinished tasks into an array of size $n^{3/4}$, if possible. Finally, we make $n^{1/6}$ copies of each unfinished task and run all these copies in parallel. In the full version we show, using Chernoff bounds [29], that this scheme runs in the claimed bounds with n-exponential probability. ■

Again, assuming that S is a $(1/r)$-semi-net, the total running time is therefore $O(\log n)$ using $O(n^{\lfloor d/2 \rfloor})$ work, with n-exponential probability. Of course, the assumption on S simply implies that this algorithm runs with these bounds with probability at least $1/2$. To turn this into an algorithm that has these bounds with high probability we replicate the first phase of our algorithm $O(\log n)$ times and run all these versions in parallel. With n-polynomial probability, one of these calls will terminate in the specified time and work bounds with our discovery that the originally-chosen S is indeed a semi-net. We then complete the algorithm as above using this S. This gives us the following:

[6] This is a probabilistic analogue of a result of Cole and Vishkin [18].

Theorem 3.3 *One can compute the intersection of n half-spaces in $\mathbb{R}^d$ in $O(\log n)$ time using $O(n^{\lfloor d/2 \rfloor})$ work in the EREW PRAM model, with n-polynomial probability, for $d \geq 4$.*

3.4 An optimal deterministic method for even dimensions

We can also derive an optimal-work deterministic convex hull method for even dimensions. Our method is the same as that in Section 3.1 except that in each level of the recursion we compute an s-restricted 0-shallow $(1/r)$-cutting (Theorem 2.13). Although at first glance this algorithm may appear to suffer from the "constant-factor" blow-up problem mentioned in Section 3.1, a careful analysis shows that it is actually work-optimal for even dimensions.

The basis cutting $\mathcal{C}_0$ consists of a constant number of simplices that cover all vertices of $\mathcal{A}(H)$, with $n_0 = n$ and $r_0 = 1$ (a $\mathcal{C}_0$ with d simplices can be constructed from a large simplex s_0 covering all vertices of $\mathcal{A}(H)$ by connecting o to each $(d-1)$-face of s_0). Inductively, we have a collection of simplices $\mathcal{C}_{i-1}$ with common apex o (the origin) that cover all vertices of level 0, and no vertices of level $> n_{i-1}$ in $\mathcal{A}(H)$, such that $|H_{|s}| \leq n_{i-1}$ for each $s \in \mathcal{C}_{i-1}$, where $n_{i-1} = n_{i-2}/r_{i-1}$ and $r_{i-1} = n_{i-2}^{\epsilon}$. $\mathcal{C}_i$ is obtained from $\mathcal{C}_{i-1}$ by refining $s \in \mathcal{C}_{i-1}$, if $|H_{|s}| > n_i$, using an s-restricted 0-shallow $(1/\rho_s)$-cutting of $H_{|s}$, where $\rho_s = r_i|H_{|s}|/n_{i-1} < r_i$ (so each $s' \in \mathcal{C}_i$ has $|H_{|s'}| \leq |H_{|s}|/\rho_s = n_{i-1}/r_i = n_i$). Thus, $n_i = n^{(1-\epsilon)^i}$ for $i \geq 0$, $r_i = n^{(1-\epsilon)^{i-1}\epsilon}$ for $i \geq 1$, and there will be $k = O(\log\log n)$ iterations before we obtain a set $\mathcal{C}_k$ that covers only the vertices of level 0 in $\mathcal{A}(H)$, i.e., $H^{\cap}$.

We first examine the size of $\mathcal{C}_i$. From Theorem 2.13 we get the recurrence

$$\begin{aligned}
|\mathcal{C}_i| &\leq \sum_{s \in \mathcal{C}_{i-1}} \left\{ \rho_s^{\lfloor \frac{d-1}{2} \rfloor} + \left(\frac{\rho_s}{|H_{|s}|} \right)^d |\mathcal{V}(H_{|s}, s)| \right\} \\
&\leq A r_i^{\lfloor \frac{d-1}{2} \rfloor} |\mathcal{C}_{i-1}| + B \left(\frac{r_i}{n_{i-1}} \right)^d n^{\lfloor \frac{d}{2} \rfloor} n_{i-1}^{\lceil \frac{d}{2} \rceil} \\
&= A r_i^{\lfloor \frac{d-1}{2} \rfloor} |\mathcal{C}_{i-1}| + B r_i^d \left(\frac{n}{n_{i-1}} \right)^{\lfloor \frac{d}{2} \rfloor}
\end{aligned}$$

for some constants $A, B > 1$. The bound on $\sum_s |\mathcal{V}(H_{|s}, s)|$ follows from Fact 2.7 since $\mathcal{C}_{i-1}$ covers no vertices of $\mathcal{A}(H)$ with level $> n_{i-1}$. This recurrence's solution is $|\mathcal{C}_i| \leq C r_i^d (n/n_{i-1})^{\lfloor d/2 \rfloor}$, for some constant C. Verifying this inductively we need

$$\begin{aligned}
C &\geq AC r_i^{\lfloor \frac{d-1}{2} \rfloor} \left(\frac{r_{i-1}}{r_i} \right)^d \left(\frac{n_{i-1}}{n_{i-2}} \right)^{\lfloor \frac{d}{2} \rfloor} + B \\
&= AC r_i^{\lfloor \frac{d-1}{2} \rfloor} \left(\frac{r_{i-1}}{r_i} \right)^d \left(\frac{1}{r_{i-1}} \right)^{\lfloor \frac{d}{2} \rfloor} + B \\
&= AC r_{i-i}^{\Delta_1} + B,
\end{aligned}$$

where $\Delta_1 = \lfloor (d-1)/2 \rfloor - \lfloor d/2 \rfloor + \epsilon(\lceil (d-1)/2 \rceil + 1)$. Note that $\Delta_1 > 0$ for d odd. However, if d is even, then ϵ can be chosen sufficiently small, e.g., $\epsilon < 1/d$, so that $\Delta_1 < 0$. Thus, the solution is valid for even d and C large enough, e.g., $C = A + B$. Also, since r_k and n_{k-1} are constant, the size of the final cutting is $|\mathcal{C}_k| = O(n^{\lfloor d/2 \rfloor})$.

We now show that the algorithm is work optimal for even $d \geq 4$. By Theorem 2.13, for each $s \in \mathcal{C}_i$ we perform $O(n_{i-1} r_i^F)$ work, for some constant $F > \lfloor d/2 \rfloor$.

$$\begin{aligned}
\sum_{i=0}^{k-1} n_i r_{i+1}^F |\mathcal{C}_i| &= \sum_{i=0}^{k-1} n_i r_{i+1}^F r_i^d n^{\lfloor \frac{d}{2} \rfloor} n_{i-1}^{-\lfloor \frac{d}{2} \rfloor} \\
&= n + n^{\lfloor \frac{d}{2} \rfloor} \sum_{i=1}^{k-1} n_i^{1-\lfloor \frac{d}{2} \rfloor} r_i^{\lceil \frac{d}{2} \rceil - (1-\epsilon)F} \\
&= n + n^{\lfloor \frac{d}{2} \rfloor} \sum_{i=1}^{k-1} n^{(1-\epsilon)^{i-1}\Delta_2}
\end{aligned}$$

where $\Delta_2 = 1 - \lfloor d/2 \rfloor + \epsilon(d - 1 - F + \epsilon F)$. Thus, for ϵ sufficiently small, $\Delta_2 < 0$, $\sum_{i=1}^{k} n^{(1-\epsilon)^{i-1}\Delta_2} = O(1)$ and the total work is $O(n^{\lfloor d/2 \rfloor})$.

This implies the following:

Theorem 3.4 *Given a set H of n halfspaces in $\mathbb{R}^d$, for even $d \geq 4$, one can (deterministically) compute $H^\cap$ in $O(\log n)$ time using $O(n^{\lfloor d/2 \rfloor})$ work in the EREW PRAM model.*

By well-known reductions, this theorem immediately implies deterministic optimal-work parallel methods for 3-dimensional Delaunay triangulations and Voronoi diagrams, which have a wide number of applications (see e.g., [24]).

3.5 Cuttings, arrangements, & point location

The same method used to construct convex hulls for even dimensions, when applied to the whole arrangement $\mathcal{A}(H)$ of hyperplanes H rather than to a single cell, can be used to construct $(1/r)$-cuttings for H with optimal work $O(nr^{d-1})$ (when the conflict list for each simplex is part of the output) in $O(\log n)$ time in the EREW PRAM model. From this, solutions follow for the problems of constructing hyperplane arrangements and point location data structures for hyperplane arrangements.

For a $(1/r)$-cutting, the basis $\mathcal{C}_0$ is a single simplex covering all the vertices of $\mathcal{A}(H)$ and the set of simplices $\mathcal{C}_{i-1}$ covers all of the vertices of $\mathcal{A}(H)$ (the simplices are not required to have a common apex). For refining a simplex $s \in \mathcal{C}_{i-1}$, if $|H_{|s}| > n_i$, we use an s-restricted $|H_{|s}|$-shallow $(1/\rho_s)$-cutting where $\rho_s = r_i |H_{|s}|/n_{i-1}$, with $r_i = (Kr)^{(1-\epsilon)^{i-1}\epsilon}$ for an appropriate K. With this choice, $n_i = n/(Kr)^{1-(1-\epsilon)^i}$, so after $k = O(\log \log n)$ iterations we have $n_k = n/r$.

Since $\sum_{s \in \mathcal{C}_{i-1}} |\mathcal{V}(H_{|s}, s)| = O(n^d)$, we obtain the recurrence $|\mathcal{C}_i| \leq A r_i^{d-1} |\mathcal{C}_{i-1}| + B(nr_i/n_{i-1})^d$, whose solution is $|\mathcal{C}_i| \leq C(nr_i/n_{i-1})^d$ for an appropriate C. Thus, $|\mathcal{C}_k| = O(r^d)$. For some constant $F > d$, and appropriate $\epsilon > 0$, the total work performed is $\sum_{i=0}^{k-1} n_i r_{i+1}^F |\mathcal{C}_i| = O(nr^{d-1})$.

When $r = n$, we obtain the complete hyperplane arrangement $\mathcal{A}(H)$ (actually, it is split into pieces, but it can be put together by going backwards in the construction). This provides an EREW PRAM analogue to a CREW PRAM result of Goodrich [26].

Theorem 3.5 *Given a set H of n hyperplanes in $\mathbb{R}^d$, a $(1/r)$-cutting for H of size $O(r^d)$ can be constructed in $O(\log n)$ time using $O(nr^{d-1})$ work in the EREW PRAM model. In particular, $\mathcal{A}(H)$ can be constructed in $O(\log n)$ time using $O(n^d)$ work in the EREW PRAM model.*

The hierarchical structure obtained can be used to perform point location, as indicated by Chazelle [9]. Since we are interested in the ability to answer simultaneous queries, we construct for each $s \in \mathcal{C}_i$ the point location data structure of Dobkin and Lipton [21]. This reduces the problem to point location in a collection of *slabs*, a collection of hyperplanes restricted to an infinite prism whose section is a simplex, so that they do not intersect inside the prism. We give the details in the full version and summarize here the result as follows.

Theorem 3.6 *Given a set H of n hyperplanes in $\mathbb{R}^d$, a point location data structure of size $O(n^d)$ can be constructed in $O(\log n)$ time and $O(n^d)$ work in the EREW PRAM model. $O(n)$ searches can be performed in $O(\log n \log \log n)$ time or in $O(\log n)$ time with n-polynomial probability in the EREW PRAM model, and in $O(\log n)$ time in the CREW PRAM model.*

The deterministic result for the EREW PRAM model uses an appropriate replication of the data structure, while the randomized result uses a technique of Reif and Sen [52].

4 3-Dimensional Convex Hulls

Let H be a set of n halfspaces in $\mathbb{R}^3$ containing a known point o. We wish to compute their intersection $H^\cap$.

4.1 A simple optimal expected work method

We begin by describing a simple randomized method that achieves an expected running time of $O(\log n)$ and performs $O(n \log n)$ expected work. The structure of the algorithm is similar to the sub-optimal method for $\mathbb{R}^d$ of Section 3.1. This general approach was first used for $\mathbb{R}^3$ by Clarkson and Shor [15] in their sequential randomized output-sensitive algorithm, and was also used in the randomized parallel algorithm of Reif and Sen [51] for the CREW PRAM model.

We select a random sample $R \subset H$, construct the convex polyhedron $R^\cap$, and triangulate its boundary. Then, we decompose $R^\cap$ into a set S_R of simplices, where a simplex is formed by a triangular face of $R^\cap$ and the point o. Next, for

each simplex $s \in S_R$, we find the conflict list $H_{|s} \subset H \backslash R$. This method is then applied recursively to each simplex $s \in S_R$, with input $H_{|s}$, to compute $H^{\cap}_{|s} \cap s$. Clarkson and Shor [15] show that, for appropriate constants c_0 and c_1, both of the following conditions hold with probability at least $1/2$: (i) $\sum_{s \in S_R} |H_{|s}| \leq c_0 n$, and (ii) $\max_{s \in S_R} |H_{|s}| \leq c_1 (n/r) \log r$.

We examine the resource bounds of one level in the recursion for a sample size $r = n^{\epsilon}, 0 < \epsilon < 1$. We construct (and triangulate) $R^{\cap}$ in $O(\log n)$ time using $O(n^{2\epsilon} \log n)$ work [4]. As noted by Reif and Sen [51], the set of simplices cut by a halfspace $h \in H \backslash R$ can be found by locating the point $\mathcal{D}(h)$ in the arrangement $\mathcal{A}(G)$, where $G = \{\mathcal{D}(p) | p \in V \cup o\}$, V is the vertex set of $R^{\cap}$, and $\mathcal{D}$ is the standard duality transform which in $\mathbb{R}^3$ maps points to planes and planes to points. By Theorem 3.6, the set of simplices cut by each halfspace in $H \backslash R$ can be found in $O(\log n)$ time using $O(n^{4\epsilon} + n \log n) = O(n \log n)$ expected work. Then, processor allocation (i.e., one processor for each plane/simplex pair) can be done by a prefix sums computation. Therefore, each level in the recursion takes $O(\log n)$ time using $O(n \log n)$ work, where n is the input size for the level.

Since the size of each subproblem is roughly $O(n^{1-\epsilon})$, the expected depth of the recursion is $O(\log \log n)$, and the total expected running time $O(\log n)$. In order for the entire algorithm to perform $O(n \log n)$ work the total size of the subproblems at any level in the recursion must not exceed $O(n)$. Unfortunately, since halfspaces can appear in multiple subproblems, this method has same the "constant-factor" blow-up problem as the method for $\mathbb{R}^d$ of Section 3.1, i.e., over the $O(\log \log n)$ recursive calls the total size of the subproblems can only be bounded by $O(n \log^{O(1)} n)$.

4.1.1 Pruning

In their CREW PRAM algorithm, Reif and Sen [51] overcome this size blow-up problem by first constructing $T^{\cap}$, where T is a subset of the halfspaces in H that are known to be bounding halfspaces of $H^{\cap}$. They then use $T^{\cap}$ to identify redundant halfspaces in $H \backslash (T \cup R)$.

We use a different approach to avoid this problem. Our basic strategy is to eliminate any known redundancy in the input to the subproblems. Specifically, before the recursive call on a simplex s, we remove those halfspaces from $H_{|s}$ that we know cannot contribute a vertex to $H^{\cap}$ in s. This gives a simple, efficient way to avoid the size blow-up problem.

Consider a simplex s. We show that a sufficient number of redundant halfspaces in $H_{|s}$ can be found using only the intersection of $H^{\cap}$ with $\mathrm{bd}(s)$, the boundary of s. This intersection consists of three two-dimensional convex polygon chains, called *contours*, one on each of the three faces of s that contain o. We can construct the contours in $O(\log n)$ time and $O(n \log n)$ work using the optimal EREW PRAM planar convex hull algorithm of Miller and Stout [44]. Note that all halfspaces contributing an edge to a contour contribute a face to $H^{\cap}$. It is convenient to separately consider the set of halfspaces $H^{c}_{|s} \subset H_{|s}$ that contribute an edge to one of the contours, and the set $H^{nc}_{|s} = H_{|s} \backslash H^{c}_{|s}$. Define $H^{nc} = \bigcup_{s \in S_R} H^{nc}_{|s}$, $H^{c} = \bigcup_{s \in S_R} H^{c}_{|s}$, and $H^{\cap}_{s} = H^{\cap} \cap s$.

Identifying redundant halfspaces in $H^{nc}_{|s}$. A halfspace $h \in H^{nc}_{|s}$ may or may not contribute a vertex to $H^{\cap}_{s}$. We say that a halfspace $h \in H^{nc}_{|s}$ and a contour[7] of s *generate* the ray r that originates at the closest point p on that contour to the bounding plane of h, "shoots" through the interior of s, and is contained in the bounding planes of the (at most two) halfspaces in $H^{c}_{|s}$ that contributed p to the contour.

Observation 4.1 *If a ray generated from $h \in H^{nc}_{|s}$ and a contour of s does not intersect the bounding plane of h, then h cannot contribute to $H^{\cap}_{s}$, i.e., h can be pruned from $H_{|s}$.*

We say that the halfspace $h \in H^{nc}_{|s}$ is *pinned* to the simplex s if its bounding plane is pierced (not necessarily in s) by all three rays generated from h and a contour of s. By Observation 4.1, only those halfspaces $h \in H^{nc}_{|s}$ pinned to a simplex need to be retained in $H_{|s}$. The closest point queries needed to construct the rays can be done by binary search on the contours in $O(\log n)$ time using $O(n)$ work on an EREW PRAM [45]. The following lemma shows that after pruning $|H^{nc}| = O(n)$.

Lemma 4.2 *A halfspace $h \in H^{nc}$ is pinned to at most one simplex $s \in S_R$.*

Proof: Suppose that some $h \in H^{nc}$ is pinned to both $H^{nc}_{|s}$ and $H^{nc}_{|s'}$, $s \neq s'$. In this case, the bounding plane of h must be pierced by all the rays generated from h and the six contours of s and s'. However, this violates the convexity of $H^{\cap}$, $H^{\cap}_{|s}$ and $H^{\cap}_{|s'}$. ■

Identifying redundant halfspaces in $H^{c}_{|s}$. Consider a halfspace $h \in H^{c}_{|s}$. If h does not contribute a vertex to $H^{\cap}_{s}$, then it must contribute a face to $H^{\cap}_{s}$ that is bounded by edges on at least two of the contours on $\mathrm{bd}(s)$. We call such a face *trivial*, and remove h from $H^{c}_{|s}$. The trivial faces can be identified by labeling each contour edge with the halfspaces that define it, and lexicographically sorting the labels; this takes $O(\log n)$ time using $O(n \log n)$ work on an EREW PRAM [17]. The fact that the contours can be used to identify the trivial faces was noted by Clarkson and Shor [15], and was also used in the parallel algorithm of Reif and Sen [51].

Lemma 4.3 *After removing trivial faces, $|H^{c}| = O(n)$.*

Proof: This follows from the facts that $|H^{\cap}| = O(n)$, a halfspace retained in $H^{c}_{|s}$ contributes a vertex to $H^{\cap}_{s}$, and vertices have degree three (assuming nondegeneracy). ■

In order to construct $H^{\cap}_{s}$, the trivial faces removed in the lth level of the recursion must be replaced before returning to the $(l-1)$st level. Fortunately, this is not difficult.

[7] In the full version we also deal with some boundary cases that are not explicitly covered by this definition.

Lemma 4.4 *Let P_s be a maximal subset of $H^c_{|s}$ such that every $h \in P_s$ contributes a trivial face to $H^{\cap}_s$ and these faces define at most one subchain on each contour of s. Then the portion of the convex polyhedron returned from the previous level that lies beyond $P^{\cap}_s$ (i.e., in $\overline{P^{\cap}_s}$) is defined by the (at most three) halfspaces responsible for the contour edges incident to the contour subchains associated with P_s.*

Proof: We give the proof in the full version. ■

Thus, we can achieve optimal expected time and work.

4.2 Achieving high probability

We can modify the algorithm of the previous subsection, however, to achieve the time and work bounds with high probability. Here n_0 denotes the size of the initial problem, and n the size of a subproblem in some stage of the recursion.

n_0-polynomial probability. Assuming that sufficient processors are available at each stage, time $O(\log n_0)$ with n_0-polynomial probability can be achieved by stopping the recursion when $n = O(\log^k n_0)$ for an appropriate k, and then finishing with a nonoptimal algorithm, for example the algorithm of Amato and Preparata [3]. The total expected work is $O(n_0 \log n_0)$ (and $O(n_0 \log^2 n_0)$ with n_0-polynomial probability). To obtain optimal work with n_0-polynomial probability, we use the polling technique of Reif and Sen [51]. This applies to the EREW PRAM model.

n_0-exponential probability. To achieve even higher probability we need two tools. The first one is to obtain a "good" sample at each stage with n-exponential probability using Theorem 2.10 to obtain a 0-shallow $(1/r)$-cutting in the CREW PRAM model. This is algorithm $A(n)$ to be used below. Its resource bounds hold with failure probability $O(\exp(-n^c))$ for some $0 < c < 1$ for a subproblem of size n. Since n decreases as the algorithm progresses, we need a second technique to boost the degrading exponential probability to n_0-exponential. This is achieved with the failure sweeping technique [25, 36]. Let algorithm $B(n_0)$ be as follows: Run $A(n_0)$ until subproblems of size $n_0^{\alpha}, 0 < \alpha < 1$ are obtained, for each middle stage where $n_0^{\alpha} \leq n \leq \log^k n_0$ do failure sweeping, and substitute the last stages by the deterministic algorithm of [4].

We summarize the results in the following theorem. Both solve open problems of Reif and Sen [51].

Theorem 4.5 *The convex hull problem in $\mathbb{R}^3$ of size n can be solved in the EREW PRAM model with work $O(n \log n)$ and time $O(\log n)$ with n-polynomial probability, and in the CREW PRAM model with the same bounds with n-exponential probability.*

4.3 Deterministic and output sensitive algorithms

Goodrich [27] has derandomized Reif and Sen's [51] three-dimensional convex hull algorithm to obtain a deterministic algorithm with optimal work and $O(\log^2 n)$ time on an EREW PRAM. His techniques [27] can be applied to our algorithm yielding a simpler deterministic algorithm with the same resource bounds.

We obtain a deterministic output-sensitive parallel algorithm using optimal $O(n \log h)$ work, where $h = |H^{\cap}|$, but increased running time $O(\log^3 n)$, by applying the technique used in the sequential randomized output-sensitive method of Clarkson and Shor [15], which was also used in its derandomized version by Chazelle and Matoušek [13].

Suppose that we know the value of $h = |H^{\cap}|$, and that $h < n^{\epsilon}$, for some $\epsilon > 0$ (otherwise the $O(n \log n)$ work method suffices). By Theorem 2.10(3), we obtain a 0-shallow $(1/h)$-cutting of size $O(h)$. Everything proceeds as before except the contours are computed using a parallel version of Kirkpatrick and Seidel's [33] output-sensitive planar convex hull algorithm, due to Ghouse and Goodrich [25], which can be implemented to run in $O(\log^2 n)$ time using $O(n \log h)$ work on an EREW PRAM. Note that since the contours are part of $H^{\cap}$, their size is bounded by h. We then recurse on the $s \in S_R$ with non-empty $H_{|s}$. We spend $O(\log^2 n_i)$ time and perform $O(n_i \log h)$ at the ith stage of the recursion, where n_i is the total input size for the ith stage, and $n_0 = n$. From the $(1/h)$-cutting, and since there are at most h simplices at any stage (pruning ensures that $H_{|s} = \emptyset$ if there is no vertex of $H^{\cap}$ in s), $n_i = O(n/h^{i-1})$, $i \geq 1$. Therefore the total work is $\sum_{i=0}^{\log_h n} O(n \log h/h^i) = O(n \log h)$, and the time is $\sum_{i=0}^{\log_h n} O(\log^2(n/h^i)) = O(\log^3 n/\log h)$.

Since h is not known in advance, we try a sequence of output sizes $h_1, h_2, h_3, \ldots$, where h_1 is some appropriate constant, and $h_{i+1} = h_i^2$. The output-sensitive method is run using the values h_i as estimates of h. If the ith execution performs more than $Cn \log h_i$ work, for some appropriate constant C, or, at any stage of the recursion, finds that there are more than h_i simplices containing vertices of $H^{\cap}$, then execution is stopped and the next value is tried. This process terminates when $h < h_i$. The total work is $\sum_{i=0}^{\log\log h} cn \log h_1^{2^i} = O(n \log h)$, and the time is $\sum_{i=0}^{\log\log h} O(\log^3 n/\log h_1^{2^i}) = O(\log^3 n)$.

Theorem 4.6 *The convex hull of n points in $\Re^3$ can be constructed in $O(\log^2 n)$ time and $O(n \log n)$ work, or in $O(\log^3 n)$ time and $O(n \log h)$ work, in the EREW PRAM model.*

4.4 Ball intersection and diameter in $\mathbb{R}^3$

Let X be a set of n points in $\mathbb{R}^3$. For $r > 0$ and $x \in X$ let $b = b(x, r)$ be the (closed) ball of radius r centered at x. Let $B = B(X, r) = \{b(x, r) : x \in X\}$. We are interested in computing the intersection $B^{\cap}$ of these balls, a convex body with linear boundary complexity [31]. The interest in this object originates in its relevance to the computation of the *diameter* of X, the largest distance between any pair of points in X. The relation was pointed out by Clarkson and Shor [15] who gave optimal $O(n \log n)$ time random-

ized algorithms for both problems. Deterministically, the current best algorithms have running times $O(n \log n)$ and $O(n \log^3 n)$ respectively [8]. The algorithms we describe here match those running times and are arguably simpler[8].

Ball intersection. With appropriate variations, the method used to compute the intersection of halfspaces can be used to compute $B^{\cap}$ in $O(\log n)$ time using $O(n \log n)$ work in the EREW PRAM model. We only describe the necessary variations, further details are given in the full version. Some of the other required ingredients have been described in [40].

(i) A point o in the interior can be determined using techniques similar to those for solving linear programming problems, in $O(\log^2 n)$ time with $O(n)$ work. (ii) For a sample $R \subset B$, $R^{\cap}$ can be computed by a brute force method. A canonical triangulation S_R can be obtained as follows [40]: the boundary is triangulated by drawing for each face and vertex the segment of great circle on the face through the vertex and the poles; then the trapezoids on the boundary are joined to the interior point o to form *bricks*. (iii) For brick $s \in S_R$, let $B_s = \{x \in X : s \not\subseteq b(x, r)\}$. Our set system consists of sets B_s. In [40], using a linearization technique, it is shown that a sample R of size $r = n^{\epsilon}$ with appropriate properties ($|B_s| \leq c_1(n/r) \log r$ for each s and $\sum_{s \in S_R} |B_s| \leq c_2 n$) can be computed in $O(n \log n)$ time. Using the techniques of [27], that sample can be obtained in $O(\log^2 n)$ time with $O(n \log n)$ work in the EREW PRAM model. (iv) The conflict lists B_s can be computed using point location in an arrangement of hyperplanes as in [40] by using the mentioned linearization technique. (Alternatively, in a sequential algorithm, the conflict lists can be obtained using a hierarchical decomposition [22] to determine a first intersection point between each bounding sphere and the boundary of $R^{\cap}$, and then walking to determine all the intersection.) (v) The contours are computed using a two-dimensional version of the same method (note that each portion of contour lies on a type of cylindrical surface; still the complexity of a contour is linear in the number of balls involved). (vi) Let $B^c_{|s}$ and $B^{nc}_{|s}$ be defined as for halfspaces. $B^{nc}_{|s}$ can be computed by first recursively computing $B^c_{|s} \cap$ and then using a hierarchical decomposition ($b \in B^{nc}_{|s}$ if $B^c_{|s} \cap \not\subseteq b$). Note that in the recursive computation always $B^{nc}_{|s}$ is empty. This was used in [51]. (vii) The detection and removal of trivial faces is similar to that for the case of halfspaces, we only point out that a single ball can contribute more than one piece of trivial face inside a brick s.

Diameter. Using the algorithm for ball intersection together with parametric search [43] as in previous works [11, 40], we obtain a sequential algorithm for the diameter problem with running time $O(n \log^3 n)$.

First, we need to make the ball intersection algorithm into an oracle that determines whether $D > r$, $D < r$ or $D = r$. For this each point of X is located inside a brick $s \in S_R$ in each stage of the algorithm. Using the linearization technique of [40], this becomes a problem of point location among hyperplanes. The total work is $O(n \log n)$. If at any moment, a point is not in any brick then $D > r$. Otherwise $D \leq r$ with $D = r$ if some points are exactly on the boundary of $B^{\cap}$.

Second, using parametric search, one obtains in a straightforward manner a diameter algorithm with running time $O(n \log^4 n)$ (and a parallel algorithm running in time $O(\log^5 n)$): a $O(\log^2 n)$ factor due to the parallel algorithm, a $O(\log n)$ factor due to the number of times the oracle must be run to resolve a batch of comparisons, and a factor $O(n \log n)$ due to the oracle. We need a closer look to save a $\log n$ factor. Specifically, we find that, except for the construction of an approximation of size n^{ϵ} with work $O(n \log n)$, all the algorithm can be implemented so that the comparisons involving the radius are presented in $O(\log n)$ parallel batches. The problem with the approximation construction is that it consists of $O(\log n)$ stages each requiring the construction of a $(1/c)$-approximation, for some constant c, using $O(n)$ work as described in [27], which in turn requires $\Theta(\log \log n)$ stages (so a running time $O(n \log^3 n \log \log n)$ can be achieved). One can verify that the game of Cole [16, 17] can be played on the computation graph of the algorithm for constructing a $(1/c)$-approximation, so that the $\log \log n$ factor is saved. The bound $O(n \log^3 n)$ on the running time follows.

[8] Previous work, using only elementary techniques (not using geometric sampling) could only achieve running times $O(n \log^2 n)$ and $O(n \log^5 n)$ respectively [50].

Acknowledgements

We would like to thank Ken Clarkson, Herbert Edelsbrunner, Yossi Matias, Jiří Matoušek, Franco Preparata, and Jack Snoeyink for helpful comments concerning the topics of this paper.

References

[1] P. K. Agarwal, M. Sharir, and S. Toledo. Applications of parametric searching in geometric optimization. In *Proc. 3rd ACM-SIAM Sympos. Discrete Algorithms*, pages 72–82, 1992.

[2] A. Aggarwal, B. Chazelle, L. Guibas, C. Ó'Dúnlaing, and C. Yap. Parallel computational geometry. *Algorithmica*, 3:293–327, 1988.

[3] N. M. Amato and F. P. Preparata. The parallel 3D convex hull problem revisited. *Internat. J. Comput. Geom. Appl.*, 2(2):163–173, 1992.

[4] N. M. Amato and F. P. Preparata. An NC[1] parallel 3D convex hull algorithm. In *Proc. 9th Annu. ACM Sympos. Comput. Geom.*, pages 289–297, 1993.

[5] P. Assouad. Densité et dimension. *Ann. Inst. Fourier, Grenoble*, 3:232–282, 1983.

[6] M. J. Atallah and M. T. Goodrich. Efficient parallel solutions to some geometric problems. *J. Parallel Distrib. Comput.*, 3:492–507, 1986.

[7] M. J. Atallah and M. T. Goodrich. Parallel algorithms for some functions of two convex polygons. *Algorithmica*, 3:535–548, 1988.

[8] Hervé Brönnimann, Bernard Chazelle, and Jiří Matoušek. Product range spaces, sensitive sampling, and derandomization. In *Proc. 34th Annu. IEEE Sympos. Found. Comput. Sci. (FOCS 93)*, pages 400–409, 1993.

[9] B. Chazelle. Cutting hyperplanes for divide-and-conquer. *Discrete Comput. Geom.*, 9(2):145–158, 1993.

[10] B. Chazelle. An optimal convex hull algorithm in any fixed dimension. *Discrete Comput. Geom.*, 10:377–409, 1993.

[11] B. Chazelle, H. Edelsbrunner, L. Guibas, and M. Sharir. Diameter, width, closest line pair and parametric searching. *Discrete Comput. Geom.*, 10:183–196, 1993.

[12] B. Chazelle and J. Friedman. A deterministic view of random sampling and its use in geometry. *Combinatorica*, 10(3):229–249, 1990.

[13] B. Chazelle and J. Matoušek. Derandomizing an output-sensitive convex hull algorithm in three dimensions. Technical report, Dept. Comput. Sci., Princeton Univ., 1992.

[14] A. L. Chow. *Parallel algorithms for geometric problems*. Ph.D. thesis, Dept. Comput. Sci., Univ. Illinois, Urbana, IL, 1980.

[15] K. L. Clarkson and P. W. Shor. Applications of random sampling in computational geometry, II. *Discrete Comput. Geom.*, 4:387–421, 1989.

[16] R. Cole. Slowing down sorting networks to obtain faster sorting algorithms. *J. ACM*, 34:200–208, 1987.

[17] R. Cole. Parallel merge sort. *SIAM J. Comput.*, 17(4):770–785, 1988.

[18] R. Cole and U. Vishkin. Approximate parallel scheduling, part i: the basic technique with applications to optimal parallel list ranking in logarithmic time. *SIAM Journal on Computing*, 17(1):128–142, 1988.

[19] S. A. Cook, C. Dwork, and R. Reischuk. Upper and lower time bounds for parallel random access machines without simultaneous writes. *SIAM J. Comput.*, 15:87–97, 1986.

[20] N. Dadoun and D. G. Kirkpatrick. Parallel construction of subdivision hierarchies. *J. Comput. Syst. Sci.*, 39:153–165, 1989.

[21] D. P. Dobkin and R. J. Lipton. Multidimensional searching problems. *SIAM J. Comput.*, 5:181–186, 1976.

[22] D.P. Dobkin and D.G. Kirkpatrick. Fast detection of polyhedral intersection. *Theoretical Computer Science*, 27:241–253, 1983.

[23] H. Edelsbrunner. *Algorithms in Combinatorial Geometry*, volume 10 of *EATCS Monographs on Theoretical Computer Science*. Springer-Verlag, Heidelberg, West Germany, 1987.

[24] H. Edelsbrunner and E. Mücke. Three-dimensional alpha shapes. *ACM Trans. on Graphics*, 1994. To appear.

[25] M. Ghouse and M. T. Goodrich. In-place techniques for parallel convex hull algorithms. In *Proc. 3rd ACM Sympos. Parallel Algorithms Architect.*, pages 192–203, 1991.

[26] M. T. Goodrich. Constructing arrangements optimally in parallel. *Discrete Comput. Geom.*, 9:371–385, 1993.

[27] M. T. Goodrich. Geometric partitioning made easier, even in parallel. In *Proc. 9th Annu. ACM Sympos. Comput. Geom.*, pages 73–82, 1993.

[28] R. L. Graham. An efficient algorithm for determining the convex hull of a finite planar set. *Inform. Process. Lett.*, 1:132–133, 1972.

[29] T. Hagerup and C. Rüb. A guided tour of Chernoff bounds. *Information Processing Letters*, 33(10):305–308, 1990.

[30] D. Haussler and E. Welzl. Epsilon-nets and simplex range queries. *Discrete Comput. Geom.*, 2:127–151, 1987.

[31] A. Heppes. Beweis einer vermutung von a. vázsonyi. *Acta Math. Acad. Sci. Hungar.*, 7:463–466, 1956.

[32] H. Karloff and Y. Mansour. On construction of k–wise independent random variables. In *Proc. ACM Sympos. Theory of Computing*, pages 564–573, 1994.

[33] D. G. Kirkpatrick and R. Seidel. The ultimate planar convex hull algorithm? *SIAM J. Comput.*, 15:287–299, 1986.

[34] M. Luby. A simple parallel algorithm for the maximal independent set problem. *SIAM J. Comput.*, 15(4):1036–1053, 1986.

[35] M. Luby. Removing randomness in parallel computation without a processor penalty. In *Proc. IEEE Sympos. Foundations of Computer Science*, pages 162–173, 1988.

[36] Y. Matias and U. Vishkin. Converting high probability into nearly-constant time—with applications to parallel hashing. In *23rd ACM Symp. on Theory of Computing*, pages 307–316, 1991.

[37] J. Matoušek. Approximations and optimal geometric divide-and-conquer. In *Proc. 23rd Annu. ACM Sympos. Theory Comput.*, pages 505–511, 1991. Also to appear in *J. Comput. Syst. Sci.*

[38] J. Matoušek. Cutting hyperplane arrangements. *Discrete Comput. Geom.*, 6:385–406, 1991.

[39] J. Matoušek. Reporting points in halfspaces. *Comput. Geom. Theory Appl.*, 2(3):169–186, 1992.

[40] J. Matoušek and O. Schwarzkopf. A deterministic algorithm for the three-dimensional diameter problem. In *Proc. 25th Annu. ACM Sympos. Theory Comput.*, pages 478–484, 1993.

[41] J. Matoušek. Efficient partition trees. *Discrete Comput. Geom.*, 8:315–334, 1992.

[42] J. Matoušek. Linear optimization queries. *J. Algorithms*, 14:432–448, 1993. The results combined with results of O. Schwarzkopf also appear in *Proc. 8th ACM Sympos. Comput. Geom.*, 1992, pages 16–25.

[43] N. Megiddo. Applying parallel computation algorithms in the design of serial algorithms. *J. ACM*, 30:852–865, 1983.

[44] R. Miller and Q. F. Stout. Efficient parallel convex hull algorithms. *IEEE Trans. Comput.*, C-37(12):1605–1618, 1988.

[45] W. Paul, U. Vishkin, and H. Wagener. Parallel dictionaries on 2-3 trees. In *Proceedings of the Tenth ICALP*, pages 597–609, 1983.

[46] M. Pellegrini. On point location and motion planning among simplicies. In *Proc. ACM Sympos. Theory of Computing*, pages 95–104, 1994.

[47] F. P. Preparata and S. J. Hong. Convex hulls of finite sets of points in two and three dimensions. *Commun. ACM*, 20:87–93, 1977.

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

[49] S. Rajasekaran and S. Ramaswami. Optimal parallel randomized algorithms for the Voronoi diagram of line segments in the plane and related problems. In *Proc. 10th Annu. ACM Sympos. Comput. Geom.*, 1994.

[50] E. A. Ramos. An algorithm for intersecting equal radius balls r^3. Technical Report UIUCDCS-R-94-1851, Dept. of Computer Science, University of Illinois at Urbana-Champaign, 1994.

[51] J.H. Reif and S. Sen. Optimal parallel randomized algorithms for three-dimensional convex hulls and related problems. *SIAM J. Comput.*, 21(3):466–485, 1992.

[52] J.H. Reif and S.Sen. Randomized algorithms for binary search and load balancing on fixed connection networks with geometric applications. In *Proc. 2nd ACM Sympos. Parallel Algorithms Architect.*, pages 327–337, 1990.

[53] N. Sauer. On the density of families of sets. *J. Combin. Theory Ser. A*, 13:145–147, 1972.

[54] R. Seidel. A convex hull algorithm optimal for point sets in even dimensions. M.Sc. thesis, Dept. Comput. Sci., Univ. British Columbia, Vancouver, BC, 1981. Report 81/14.

[55] R. Seidel. Constructing higher-dimensional convex hulls at logarithmic cost per face. In *Proc. 18th Annu. ACM Sympos. Theory Comput.*, pages 404–413, 1986.

[56] R. Seidel. Small-dimensional linear programming and convex hulls made easy. *Discrete Comput. Geom.*, 6:423–434, 1991.

[57] V. N. Vapnik and A. Y. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory Probab. Appl.*, 16:264–280, 1971.