Geometric Partitioning Made Easier, Even in Parallel

(Preliminary Version)

Michael T. Goodrich*

Dept. of Computer Science, The Johns Hopkins Univ., Baltimore, MD 21218
Email: goodrich@cs.jhu.edu

Abstract

We present a simple approach for constructing geometric partitions in a way that is easy to apply to new problems. We avoid the use of VC-dimension arguments, and, instead, base our arguments on a notion we call the *scaffold dimension*, which subsumes the VC-dimension and is simpler to apply. We show how to easily construct $(1/r)$-nets and $(1/r)$-approximations for range spaces with bounded scaffold dimension, which immediately implies simple algorithms for constructing $(1/r)$-cuttings (by straightforward recursive subdivision methods). More significant than simply being a conceptual simplification of previous approaches, however, is that our methods lead to asymptotically faster and more-efficient EREW PRAM parallel algorithms for a number of computational geometry problems, including the development of the first optimal-work NC algorithm for the well-known 3-dimensional convex hull problem, which solves an open problem of Amato and Preparata. Interestingly, our approach also yields a faster sequential algorithm for the distance selection problem, by the parametric searching paradigm, which solves an open problem posed by Agarwal, Aronov, Sharir, and Suri, and reiterated by Dickerson and Drysdale.

1 Introduction

One general type of geometric structure that has received a considerable amount of attention of late is the *geometric partition* [9, 10, 11, 14, 15, 34, 35, 36]. The general framework is that one is given a collection X of n geometric objects in $\mathbb{R}^d$, such as lines, and a parameter r, and one wishes to construct a partitioning of the space into $O(r^d)$ constant-sized cells so that each cell intersects as few objects as possible. As one might expect, such a structure is especially suited to the recursive design of efficient geometric data structures and to the design of fast geometric divide-and-conquer algorithms (e.g., see [1]).

This paper addresses the problem of constructing such partitions. We are primarily interested in the parallel complexity of these constructions, but one should keep in mind that any parallel algorithm will always have an associated sequential algorithm, which is derived from a simple simulation of the parallel method. Thus, the discovery of a new parallel method implies a new sequential method as well. Typically, this derived sequential algorithm is more complicated than the existing sequential methods, so it is often of little interest. Nevertheless, there are some occasions when the implied sequential method is conceptually simpler than the previous sequential methods (e.g., see [18, 45]), hence, is possibly of independent interest. Indeed, as we show in this paper, such is the case for geometric partitioning.

Before we describe our results, however, let us review some of the previous work for this important problem. As shown by Clarkson [14] and Haussler and Welzl [29], and extended by Clarkson and Shor [15], one can often apply random sampling to construct a geometric partitioning of space so that each cell intersects $O((n \log r)/r)$ objects on average. Moreover, Chazelle and Friedman [9] show that one can, in fact, construct such a partitioning deterministically in polynomial time, and Berger, Rompel, and Shor [7] and Motwani, Naor, and Naor [39] show that one can in certain situations achieve NC implementations[1] of Chazelle and Friedman's algorithm. Unfortunately, the running time of Chazelle and Friedman's algorithm is quite high, as are the time and processor bounds of the implied parallel algorithms (they run in $O(\log^4 n)$ time using a number of processors proportional to the time bound of Chazelle and Friedman's algorithm). In addition, all of these deterministic methods employ the complicated *conditional probabilities* technique (see Alon, Spencer, and Erdös [4]) to derandomize random sampling; hence, they do not lead to conceptually simple algorithms.

One can improve the running time of the Chazelle and Friedman algorithm for the case when the range space determined by X and $\mathcal{R}$, the set of combi-

*This research supported by the NSF and DARPA under Grant CCR-8908092, by the NSF under Grants CCR-9003299 and IRI-9116843.

9th Annual Comptutational Geometry,5/93/CA,USA
© 1993 ACM 0-89791-583-6/93/0005/0073...$1.50

[1] We use NC to denote the class of problems solvable in polylogarithmic time using a polynomial number of processors [25, 30, 31].

natorially distinct ways to intersect X by "cells," has bounded Vapnik-Chervonenkis (VC) dimension[2], as shown by a number of results due to Matoušek (e.g., [34, 35, 36]). Unfortunately, Matoušek's algorithms are even more complicated than the previous methods, as his methods also use the conditional probabilities technique, together with an intricate divide-and-conquer strategy, and even call Chazelle and Friedman's algorithm as a subroutine. Moreover, since they are formulated for range spaces with bounded VC-dimension, they are quite cumbersome to apply, since determining the VC-dimension of most range spaces is often quite difficult[3].

All of these methods are based upon related notions of $(1/r)$-approximations and $(1/r)$-nets, which are subsets of X used to construct good geometric partitions for X. For any range R, a subset $Y \subseteq X$ *$(1/r)$-approximates* R if

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

Given a set $\mathcal{R}$ of ranges of X, we say Y is a *$(1/r)$-approximation* [34, 35] of $(X, \mathcal{R})$ if Y $(1/r)$-approximates every range in $\mathcal{R}$. Relaxing this requirement a bit, we say that Y is a *$(1/r)$-net* [29] of $(X, \mathcal{R})$ if $Y \cap R \neq \emptyset$ for each $R \in \mathcal{R}$ such that $|R| \geq |X|/r$. This is clearly a simpler notion, since a $(1/r)$-approximation is automatically a $(1/r)$-net. Note that X is a $(1/n)$-approximation for $(X, \mathcal{R})$. Recent results of Matoušek, Welzl, and Wernisch [37] give a tighter analysis of the best achievable sizes for $(1/r)$-approximations and $(1/r)$-nets based upon discrepancy arguments. Although their analysis uses the VC-dimension notion, it can also be viewed as a first step towards the avoidance of a strict reliance on this notion, as their analysis is actually based upon a derived parameter known as the *shatter function* [34, 35, 36, 37]. Indeed, Chazelle and Matoušek [11] have very recently produced a method for deterministically constructing $(1/r)$-approximations and $(1/r)$-nets based upon these new approaches [37] using shatter functions and discrepancy bounds. Their methods are simpler than the previous algorithms, but they still use the conditional probability technique for derandomization and they still rely on VC-dimension arguments (albeit now only indirectly).

In this paper we give fast and efficient efficient parallel algorithms for constructing $(1/r)$-approximations, $(1/r)$-nets, and geometric partitions. Our methods run in $O(\log^2 n)$ time using an efficient number of processors in the EREW PRAM model[4]. For example, given n hyperplanes in $\mathbb{R}^d$, and a parameter r, we show that, in $O(\log^2 n)$ time using $O(nr^{d-1})$ work[5], one can construct a $O(r^d)$-sized $(1/r)$-cutting [8, 9, 29, 34, 35, 36] of $\mathbb{R}^d$, which is a decomposition of $\mathbb{R}^d$ into simplices such that each simplex intersects at most n/r hyperplanes.

Our methods avoid the inherently-sequential conditional probabilities technique and are designed in a framework that subsumes the cumbersome VC-dimension notion in a manner that is simple to analyze and apply. For these reasons, the sequential methods derived from our parallel procedures are actually simpler than the previous sequential methods. We achieve these simplifications (and our fast parallel running times) by using the simple *restricted independence* derandomization technique [4, 32, 33] to derandomize algorithms based on random sampling. The only unfortunate aspect of using this technique is that the necessary processor bounds are quite high. Still, we show that one can significantly reduce the number of processors needed to construct the partitions by using a simple divide-and-conquer strategy.

We show the utility of our methods by giving a number of applications to well-known computational geometry problems. For example, we show that one can construct the convex hull of n points in $\mathbb{R}^3$ in $O(\log^2 n)$ time using $O(n \log n)$ work on an EREW PRAM, which is the first optimal-work *NC* algorithm for solving this problem. This solves an open problem of Amato and Preparata [5]. In addition, by an application of the powerful parametric searching technique [3, 12, 16, 17, 19, 20], we show that our methods actually improve the sequential complexity for the problem of selecting the k-th smallest distance determined by n points in the plane [2, 24]. Our method runs in $O(n^{4/3} \log^{8/3} n)$ time in the worst case, which matches the expected time complexity of the previous randomized method due to Agarwal *et al.* [2]. This solves an open problem of Agarwal *et al.*, which was reiterated by Dickerson and Drysdale [24].

2 Our Framework

Our framework is to model the geometric partitioning problem in a fashion resembling the model introduced by Chazelle and Friedman [9], which is a synthesis of the model introduced by Clarkson and Shor [14, 15]

[2] The *VC-dimension* of a range space $(X, \mathcal{R})$ is usually defined as the maximum size of a subset A of X such that $\mathcal{R}|_A = 2^A$ (e.g., see [29, 34, 35, 36]). Thus, to show that some $(X, \mathcal{R})$ has VC-dimension d one must prove that there exists a subset A of size d such that $\mathcal{R}|_A = 2^A$, and, for *every* subset B of size $d+1$, $\mathcal{R}|_B \neq 2^B$.

[3] The interested reader is challenged to determine the VC-dimension of the space where X is a set of line segments in the plane and the ranges are the sets of segments intersecting arbitrary triangles.

[4] This is the synchronous parallel model where processor share a common main memory that does not allow for concurrent access (it is an exclusive-read, exclusive-write storage).

[5] The *work* performed by a parallel algorithm is the product of the running time and the number of processors needed. It corresponds to the running time of the derived sequential algorithm.

and the model introduced by Haussler and Welzl [29] and Matoušek [34, 35, 36]. Let X be a set of n (geometric) objects and let $(X, \mathcal{R})$ be a range space for X, i.e., $\mathcal{R}$ is a set of subsets of X, each of which is called a *range*. In addition to X and $\mathcal{R}$, we also assume we are given a function $\mathcal{F}$ that maps each subset $Y \subseteq X$ to a subset of 2^X so that $\mathcal{F}(X) = \mathcal{R}$. We call $\mathcal{F}$ the *generator* function for the range space $(X, \mathcal{R})$, and we let $|\mathcal{F}(Y)|$ denote the number of ranges it generates for Y. A range space such as $(X, \mathcal{R})$ has *scaffold dimension* d if d is the smallest integer such that $|\mathcal{F}(Y)|$ is $O(|Y|^d)$ for any $Y \subseteq X$.

As an example, consider the range space $(X, \mathcal{R})$ where X is a set of n points in general position in the plane and $\mathcal{R}$ is the set of all combinatorially distinct ways of intersecting the points of X with a disk. It is easy to show that each $R \in \mathcal{R}$ can be defined in terms of at most three points of X, which *generate* R. It is therefore easy to see that this range space has scaffold dimension 3. Determining the VC-dimension of this range space is more of a challenge, which we leave to the interested reader.

Given a range space with bounded scaffold dimension, our goal is to find small-sized $(1/r)$-approximations and $(1/r)$-nets of X for $1 < r < n$.

3 Probabilistic Preliminaries

Our approach to constructing small-sized $(1/r)$-approximations and $(1/r)$-nets of range spaces with bounded scaffold dimension is to derandomize a straightforward probabilistic algorithm, **Approx**, which is based upon the *random sampling* technique [14]. We do this using the *restricted independence* technique [4, 32, 33], which assumes **Approx** uses random variables that are only k-wise independent, for some constant k. Thus, before we give our methods, let us review these concepts.

3.1 Random Sampling

The type of random sample we assume in this paper is that one is given a parameter $s \leq n$, and, for each $x_i \in X$, one defines a random variable X_i such that $X_i = 1$ with probability s/n and one uses the rule that $x_i \in Y$ if $X_i = 1$ [7]. In this case one is guaranteed a set of $|Y| = X_1 + X_2 + \cdots + X_n$ unique elements, but its size may not be equal to s, although it is easy to see, by the linearity of expectation, that $E(|Y|) = s$.

3.2 k-Wise Independence

In order to apply the restricted independence technique, we must restrict our set $\mathcal{X}$ of random variables to be only *k-wise independent*, i.e., the variables in any subset $\mathcal{Y} \subseteq \mathcal{X}$ are guaranteed to be mutually independent only if $|\mathcal{Y}| \leq k$. Given a set X of n objects and an integer parameter s, we define a *k-wise independent expected s-sample* of X to be a sample determined by n k-wise independent indicator random variables, $X_1^{(k)}, X_2^{(k)}, \ldots, X_n^{(k)}$, where $X_i^{(k)} = 1$ with probability $p = s/n$.

Unfortunately, restricting our attention to k-wise independent indicator random variables prevents us from directly using the well-known and powerful Chernoff bounds [6, 4, 13, 28] for bounding the tail of the distribution of their sum. Nevertheless, as shown by Rompel [41], we may derive something analogous:

Lemma 3.1 (Rompel [41]): *Let $X^{(k)}$ be the sum of n k-wise independent random variables taking on values in the range $[0, 1]$, with $\mu = E(X^{(k)})$, where k is a positive even integer. Then there is a fixed constant $c > 0$ such that*

$$\Pr(|X^{(k)} - \mu| \geq \lambda) \leq c \left(\frac{k\mu + k^2}{\lambda^2} \right)^{k/2},$$

for any $\lambda > 0$.

In some cases, we can simplify this:

Corollary 3.2: *Let $X^{(k)}$ be the sum of n k-wise independent random variables taking on values in the range $[0, 1]$, with $\mu = E(X^{(k)}) \geq 1$, where k is a positive even constant. Then there is a fixed constant $c > 0$ such that*

$$\Pr(|X^{(k)} - \mu| \geq \lambda) \leq c \left(\frac{\mu^{k/2}}{\lambda^k} \right),$$

for any $\lambda > 0$.

Incidentally, this corollary also seems to follow from a recent inequality of Schmidt, Siegel, and Srinivasan [42], which may yield a better constant factor.

3.3 Derandomization

We are now ready to review the *restricted independence* technique for derandomizing a probabilistic algorithm [4, 32, 33]. We use the formulation of Luby [32], which assumes we have a probabilistic algorithm, **Random**, which is designed so that all the randomization is contained in a single *choice step*. In addition, we assume the following:

1. **Random** succeeds with constant probability even if the underlying random variables are only k-wise independent, for some constant k.
2. Each random variable X_i takes on values $\{x_1, x_2, \ldots, x_m\}$, where m is bounded by a polynomial[6] in n.

[6] In our usage each X_i will take a value from $\{0, 1\}$.

3. There is a prime number q bounded by a polynomial in n, and integers $n_{i,1}, n_{i,2}, \ldots, n_{i,m}$, such that X_i takes on value x_j with probability $n_{i,j}/q$ (with $\sum_{j=1}^{m} n_{i,j} = q$).

Luby [32] shows that if **Random** satisfies all of these conditions, then one may construct a space of q^k points so that each point corresponds to an assignment of values to $X_1, X_2, \ldots, X_n$. Moreover, each $X_i = x_j$ with probability $n_{i,j}/q$ and the X_i's are k-wise independent. Since this space is polynomial in size, we may therefore derandomize **Random** by calling it on each of the q^k sample points in parallel. Since **Random** succeeds with constant probability, at least one of these calls succeed (in fact, a constant fraction succeed). The output is given by one of these successful calls (where one breaks ties arbitrarily).

4 $O((nr)^{O(1)})$-Work Approximation Finding

Our first method for finding approximating subsets of X is quite simple and can be implemented to run very fast in parallel, albeit with a rather large number of processors. Our method is based upon a quantification of how close a particular subset is to being a $(1/r)$-approximation or $(1/r)$-net of the given range space. So, let $(X, \mathcal{R})$ be a range space, let Y be a subset of X, and let $\mathcal{S}$ be a subset of $\mathcal{R}$. Let $\mathcal{A}_Y(r, \mathcal{S})$ denote the number of ranges $R \in \mathcal{S}$ that Y does not $(1/r)$-approximate, and let $\mathcal{N}_Y(r, \mathcal{S})$ denote the number of ranges $R \in \mathcal{S}$ such that $|R| \geq |X|/r$ but $Y \cap R = \emptyset$. Of course, we desire these "error functions" to be as small as possible. The next lemma explores how well a random Y achieves this goal when Y is defined using k-wise independent random variables.

Lemma 4.1: *Let $(X, \mathcal{R})$ be a range space. Given a parameter $1 < r \leq |X|$, a parameter s that is greater than some fixed constant $s_0 > 1$, and a fixed positive even constant k, let Y be a k-wise independent expected s-sample of X, and let $\mathcal{S}$ be a subset of $\mathcal{R}$. Then the following is true with probability at least $1/2$:*

1. $s - \Theta(s^{1/2}) \leq |Y| \leq s + \Theta(s^{1/2})$,
2. $\mathcal{A}_Y(r, \mathcal{S}) \leq cr^k |\mathcal{S}|/s^k$,
3. $\mathcal{N}_Y(r, \mathcal{S}) \leq cr^{k/2} |\mathcal{S}|/s^{k/2}$,

for some constant $c > 0$.

Proof: Since Y is an expected s-sample of X, it is determined by n k-wise independent indicator random variables $X_1^{(k)}, X_2^{(k)}, \ldots, X_n^{(k)}$; hence $|Y|$, $\mathcal{A}_Y(r, \mathcal{S})$ and $\mathcal{N}_Y(r, \mathcal{S})$ are themselves random variables. Let us, therefore, consider each in turn, beginning with $|Y| = \sum_{i=1}^{n} X_i$. Since $|Y|$ is a sum of n k-wise independent indicator random variables with mean $\mu_{|Y|} = s$, we may apply Corollary 3.2 to bound the probability that Y does not satisfy Condition 1 as

$$\Pr(||Y| - s| > 6cs^{1/2}) \leq \frac{cs^{k/2}}{(6cs)^{k/2}} \leq 1/6, \qquad (1)$$

where c is as in the corollary. Thus, with probability $5/6$, $s - 6cs^{1/2} \leq |Y| \leq s + 6cs^{1/2}$.

Let us next consider $\mathcal{A}_Y(r, \mathcal{S})$, then. We can write $\mathcal{A}_Y(r, \mathcal{S}) = \sum_{R \in \mathcal{S}} Y_R$, where Y_R is an indicator random variable for "Y does not $(1/r)$-approximate R". We bound $\mathcal{A}_Y(r, \mathcal{S})$ by considering its expectation, which, by the linearity of expectation, is

$$E(\mathcal{A}_Y(r, \mathcal{S})) = \sum_{R \in \mathcal{S}} E(Y_R) = \sum_{R \in \mathcal{S}} \Pr(Y_R = 1).$$

Let us therefore derive a bound for

$$\Pr(Y_R = 1) = \Pr(||Y \cap R| - |Y|(|R|/n)| > |Y|/r).$$

Define random variables $U = |Y \cap R| - |Y \cap R|(|R|/n)$ and $V = |Y \cap (X \backslash R)|(|R|/n)$. Then

$$\Pr(Y_R = 1) = \Pr(|U - V| > |Y|/r).$$

Let $\mu_U = E(U)$ and $\mu_V = E(V)$ and note that $\mu_U = \mu_V = (s|R|/n)(1 - |R|/n)$. Thus,

$$\Pr(Y_R = 1) = \Pr(|U - \mu_U + \mu_V - V| > |Y|/r).$$

It is easy to verify that this latter probability is bounded by

$$\Pr(|U - \mu_U| > |Y|/2r) + \Pr(|V - \mu_V| > |Y|/2r).$$

Note that $U = \sum_{i \in R} X_i(1 - |R|/n)$ and $V = \sum_{i \notin R} X_i(|R|/n)$. Thus, we may apply Lemma 3.1 to bound this probability by

$$c' \left(\frac{[k(s|R|/n) + k^2]^{k/2} r^k}{|Y|^k} \right) \leq cr^k s^{k/2}/|Y|^k,$$

for some constant c, since $s \geq s_0 > 1$. Therefore, $E(\mathcal{A}_Y(r, \mathcal{S})) \leq cr^k s^{k/2} |\mathcal{S}|/|Y|^k$. We may then apply Markov's inequality(which has no independence assumptions) to show $\Pr(\mathcal{A}_Y(r, \mathcal{S}) > 6cr^k s^{k/2} |\mathcal{S}|/|Y|^k) \leq 1/6$. Combining this with (1) gives us the claimed bound.

The bound for $\mathcal{N}_Y(r, \mathcal{S})$ is proved similarly. ■

Given this lemma we can then apply the restricted independence technique to derive a deterministic method for range spaces with bounded scaffold dimension. In this case we assume $\mathcal{F}(Y)$ is computable in $O(\log n)$ time using work proportional to a polynomial in $\sum_{R \in \mathcal{F}(Y)} |R|$ on an EREW PRAM.

Theorem 4.2: *Let $(X, \mathcal{R})$ be a range space with generator function $\mathcal{F}$ and scaffold dimension d, for some constant $d > 0$, and let $n = |X|$. Also, let $1 < r < n$ be a given parameter and let $\epsilon > 0$ be any fixed (small) constant. Then, in $O(\log n)$ time using $O((nr)^c)$ work in the EREW PRAM model, for some constant $c > 0$, one can construct any of the following:*

1. *a $(1/r)$-approx. A of $(X, \mathcal{R})$ of size $\Theta(n^\epsilon r^2)$,*
2. *a $(1/r)$-approx. B of $(X, \mathcal{F}(B))$ of size $\Theta(r^{2+\epsilon})$,*
3. *a $(1/r)$-net C of $(X, \mathcal{R})$ of size $\Theta(n^\epsilon r)$, or*
4. *a $(1/r)$-net D of $(X, \mathcal{F}(D))$ of size $\Theta(r^{1+\epsilon})$.*

Proof: (Sketch) Apply the restricted independence technique [32] with $k > 2d/\epsilon$, and $s = O(n^\epsilon r^2)$, so that $|A|$ is $\Theta(s)$ and $\mathcal{A}_A(r, \mathcal{R}) < 1$ with probability at least $1/2$. We use similar arguments for constructing B, C, and D. ■

5 $O(nr^{O(1)})$-Work Approximation Finding

As already mentioned, the method of the previous section is simple and can be implemented to run very fast in parallel. Its work complexity is quite high, however. In this section we show how to reduce this by using the well-known divide-and-conquer paradigm (e.g., see [22]). We also use another range-space property, which holds for the types of range spaces that arise in computational geometry applications.

Let $(X, \mathcal{R})$ be a range space with generator function $\mathcal{F}$, and let Y be a subset of X. For any $\mathcal{S} \subseteq \mathcal{R}$, we define the range space $\mathcal{S}$ *restricted* to Y to be the set $\mathcal{S}|_Y = \{Y \cap R : R \in \mathcal{S}\}$. We say that $\mathcal{F}$ is *factorable* if, for any $Y \subseteq Z \subseteq X$, $\mathcal{F}(Z)|_Y = \mathcal{F}(Y)|_Y$. Note that factorability implies that $\mathcal{F}(Y)|_Y = \mathcal{R}|_Y$.

Many of range spaces that arise in computational geometry are factorable. For example, in applications based upon VC-dimension arguments, one gets factorability "for free," since in this case one can give the range space $(X, \mathcal{R})$ the generator function $\mathcal{F}(Y) = \mathcal{R}|_Y$, which is called the *shatter function* [34, 35, 36, 37]. If $(X, \mathcal{R})$ has VC-dimension d, then $(X, \mathcal{R})$ with $\mathcal{F}$ has scaffold dimension d as well, as shown in [43, 44] (using different terminology of course). Or, for another example, consider the range space $(X, \mathcal{R})$ introduced earlier, where X is a set of n points in the plane and $\mathcal{R}$ is the set of all ranges of X determined by intersecting X with disks. In this case, if we take $Y \subseteq Z \subseteq X$, then, for any range $R \in \mathcal{F}(Z)$, $R \cap Y$ is determined by some disk D whose generating points are in Z. To show that this same range is generated by Y imagine "shrinking" D until its degrees of freedom are constrained as much as possible without removing any points of Y from its intersection with Y. This "smaller" disk is determined by at most three points of Y, and it exactly contains the set $R \cap Y$; hence, it is in $\mathcal{F}(Y)|_Y$.

In addition to this notion of factorability, we also need two simple observations, which are adaptations of observations made by Matoušek in a slightly different context [35].

Observation 5.1: *Let $(X, \mathcal{R})$ be a range space with factorable generator function $\mathcal{F}$, and suppose $Y_1, Y_2, \ldots, Y_m$ are, respectively, equal-cardinality $(1/r)$-approximations for range spaces $(X_1, \mathcal{F}(X_1)|_{X_1})$, $(X_2, \mathcal{F}(X_2)|_{X_2}), \ldots, (X_m, \mathcal{F}(X_m)|_{X_m})$, where the X_i's also have equal cardinality, and $X = X_1 \cup X_2 \cup \cdots \cup X_m$. Then $Y = Y_1 \cup Y_2 \cup \cdots \cup Y_m$ is a $(1/r)$-approximation for $(X, \mathcal{R})$.*

Observation 5.2: *Let $(X, \mathcal{R})$ be a range space with factorable generator function $\mathcal{F}$. If Y is an $(1/r_1)$-approximation for $(X, \mathcal{R})$ and Z is a $(1/r_2)$-approximation for $(Y, \mathcal{F}(Y)|_Y)$, then Z is a $(1/r_1 + 1/r_2)$-approximation for $(X, \mathcal{R})$.*

Given a range space $(X, \mathcal{R})$ with factorable generator function $\mathcal{F}$ and bounded scaffold dimension, we wish to apply these observations to a divide-and-conquer method for constructing a $(1/r)$-approximation Y of $(X, \mathcal{R})$ of size $O(n^\delta r^2)$ using only $O(nr^{O(1)})$ work, for any reasonably small constant $\delta > 0$, where $n = |X|$. We achieve this by designing an algorithm, **Approx**, which is a modification of an early simple divide-and-conquer approach of Matoušek [34]. Whereas his method was work-inefficient, however, ours will be work efficient.

We define **Approx** so that it produces a $(1/t)$-approximation, Y, of $(X, \mathcal{R})$, where $t = \log n / r(\log n - 1)$. This is, of course, a better approximation than a $(1/r)$-approximation would be, but this formulation will prove easier to work with.

Algorithm Approx$(r, (X, \mathcal{R}))$:

1. If $n \leq r^2$, then return X.
2. Otherwise, divide X equally into m subsets X_1, X_2, ..., X_m and recursively call **Approx**$(r', (X_i, \mathcal{F}(X_i)|_{X_i}))$ for each i in parallel, where $r' = r$ and $m = n^\delta$ with $0 < \delta < 1$ being a constant to be set in the analysis.
3. Let Y_i be the set returned by recursive call i, and let $Y' = Y_1 \cup Y_2 \cup \cdots \cup Y_m$. Apply Theorem 4.2 to find a $(1/t')$-approximation Y of $(Y', \mathcal{F}(Y')|_{Y'})$, where $t' = (1 - \delta) \log n / \delta r$.
4. Return Y.

In the following theorem we show that the algorithm **Approx** performs correctly in the bounds claimed above.

Theorem 5.3: *Given a range space $(X, \mathcal{R})$ with factorable generator function $\mathcal{F}$ and scaffold dimension bounded by a constant, one can produce a $(1/r)$-approximation of $(X, \mathcal{R})$ of size $O(n^{\delta} r^{2+\delta})$ using $O(nr^{c})$ work, where $c > 0$ is some constant and δ is any positive constant strictly less than the reciprocal of the constant c in Theorem 4.2. The time required is $O(\log n)$ in the EREW PRAM model.*

Proof: (Sketch) The proof follows by a straightforward analysis of the recurrence relations for time, size, and work. ■

This theorem immediately implies work-efficient methods for constructing other approximating subsets, for we may follow an application of Theorem 5.3 by another application of Theorem 4.2 to produce sets B, C, and D as in that theorem in $O(\log n)$ time using $O(nr^{c})$ work on an EREW PRAM, for some constant c. Moreover, if one is willing to take a slightly longer amount of time, then one can construct smaller sets A and C, as the following corollary shows:

Corollary 5.4: *Given a range space $(X, \mathcal{R})$ with factorable generator function $\mathcal{F}$ and scaffold dimension bounded by a constant, one can produce a $(1/r)$-approximation of $(X, \mathcal{R})$ of size $O(r^{2+\delta})$, or a $(1/r)$-net of $(X, \mathcal{R})$ of size $O(r^{1+\delta})$, using $O(nr^{c})$ work, where $c > 0$ is some constant and δ is any positive constant strictly less than the reciprocal of the constant in Theorem 4.2. The time required is $O(\log^2 n / \log r)$ in the EREW PRAM model.*

Proof: (Sketch) Imagine a modified version of **Approx** where we set $m = r$, $r' = 2r$, and $t' = 2r$, and we apply Theorem 4.2 with $\epsilon \leq \delta/(3+\delta)$. It is easy to show that this algorithm runs in $O(\log^2 n / \log r)$ time and, more importantly, using an argument similar to that used by Matoušek [34], that this modified algorithm produces a $(1/r)$-approximation of size $O(r^{2+\delta})$ using work that is $O(n(r \log n)^{c})$ for some constant c. The $(1/r)$-approximation result follows, then, if we follow an application of Theorem 5.3 to find a $(1/\hat{r})$-approximation by a call to this modified version of **Approx** to find a $(1/\hat{r})$-approximation of this set, using $\hat{r} = 2r$. Following this by yet another call to Theorem 5.3 to construct a $(1/2r)$-net of this set gives us the $(1/r)$-net result (except for this result, we use $\hat{r} = 4r$ in the previous calls). ■

Incidentally, Chazelle and Matoušek [11] recently give a result similar to this corollary (using a different approach), but the running time and work bounds for their method are larger than those above.

In the next section we give an application of Theorem 5.3 to an important instance of the geometric partitioning problem.

6 Constructing a $(1/r)$-Cutting

Suppose we are given a set X of n hyperplanes in $\mathbb{R}^d$. A $(1/r)$-*cutting* for X is a partition of $\mathbb{R}^d$ into simplices such that each simplex intersects at most n/r hyperplanes. In this section we show how to apply Theorems 4.2 and 5.3 to construct an $O(r^d)$-sized $(1/r)$-cutting in $O(\log n)$ time using $O(nr^{O(1)})$ work on an EREW PRAM. We then show how to adapt a recursive-subdivision technique of Chazelle [8] so as to construct such a cutting in $O(\log n \log r)$ time using $O(nr^{d-1})$ work on an EREW PRAM, which is optimal if one must also output all the hyperplane-simplex adjacencies. Our method computes these adjacencies as a by-product at no additional cost.

6.1 An $O((nr)^{O(1)})$-Work Method

Our first method is based on an adaptation of an approach used by Chazelle and Friedman [9]. Their approach is to construct an optimal-sized $(1/r)$-cutting by using the conditional probabilities technique to derandomize a random sampling algorithm for finding a set Y such that the number of ranges of size tn/r not intersected by Y decreases geometrically in $1/t$.

In our case we will use Lemma 4.1 to get a similar set Y, except that the number of ranges of size tn/r not intersected by our Y will decrease quadratically in $1/t$. In addition, our set Y will only be a net for the ranges generated by Y, not for all of $\mathcal{R}$. Fortunately, this will still prove sufficient, as we will now show.

Suppose we are given a set X of n hyperplanes in $\mathbb{R}^d$. Let Y be a k-wise independent sample of X. If we compute the arrangement of Y [26, 27] and form a canonical triangulation [9, 14, 15, 34] of this arrangement (which is easily implemented in parallel), then we define $O(|Y|^d)$ simplices. Let $\mathcal{F}(Y)$ be the ranges determined by these simplices, where each simplex σ generates the range consisting of all the hyperplanes of X that intersect the interior of σ. The range space induced by $\mathcal{F}$ clearly has scaffold dimension[7] d, since there are $O(|Y|^d)$ such ranges determined by Y. For any range $R \in \mathcal{F}(Y)$, with $|R| = tn/r$, we desire that $Y \cap R \neq \emptyset$ if $t \geq 1$. We note that $\mathcal{N}_Y(r/t, \mathcal{F}(Y))$ denotes the number of such ranges in $\mathcal{F}(Y)$ that are "missed" by Y. We can choose $s = r$ in Lemma 4.1 so that, with probability at least $1/2$, $|Y|$ is $O(r)$ and $\mathcal{N}_Y(r/t, \mathcal{F}(Y)) \leq cr^{k/2}/(t^{k/2}s^{k/2-d}) = cr^d/t^{k/2}$, where k is a positive even integer (which we will set in the analysis below).

By a simple application of the restricted independence derandomization technique we may find such a Y in $O(\log n)$ time using $O(n^{d+5})$ work on an EREW PRAM. Given such a Y, we construct its arrangement and canonical triangulation in $O(\log^2 r)$ time using $O(r^d)$ work by the parallel EREW PRAM method

[7] This F is not factorable, however.

of the author [27]. We then compute, for each simplex σ in this subdivision, the set, R_σ, of hyperplanes intersecting σ. This can easily be done in $O(\log n)$ time using $O(nr^d)$ processors. We follow this, for each simplex σ, by constructing a $(1/t_\sigma)$-net Y_σ of the range space $(R_\sigma, \mathcal{F}(Y_\sigma)|_{R_\sigma})$ by Theorem 4.2, where $t_\sigma = |R_\sigma|r/n$. We can choose $\epsilon = 1$ in that Theorem so that each such $(1/t_\sigma)$-net will be of size $O(t_\sigma^2)$. Constructing the arrangement of each such Y_σ, restricted to σ, gives us a $(1/r)$-cutting, and this can be done in $O(\max_{t_\sigma}\{\log^2 t_\sigma\}) = O(\log^2 r)$ time using work that is polynomial in n and r. Since the size of the arrangement of Y_σ in σ is $O(t_\sigma^{2d})$, the total size of this cutting is proportional to

$$\sum_{R_\sigma \in \mathcal{F}(Y)} t_\sigma^{2d}.$$

We may re-write this sum as

$$\int_1^r \Big(\sum_{R_\sigma \in \mathcal{F}(Y) \text{ s.t. } t_\sigma = t} t^{2d} \Big) dt,$$

which, by the above discussion, can be bounded by

$$\int_1^r c(r^d/t^{k/2-2d})dt \;=\; cr^d \int_1^r t^{-2}dt,$$

for some constant $c > 0$, if we choose $k = 4d+4$. This is clearly $O(r^d)$. We summarize:

Lemma 6.1: *Let X be a set of n hyperplanes in $\mathbb{R}^d$, and let $1 < r < n$ be a given parameter. One can construct an $O(r^d)$-sized $(1/r)$-cutting of X in time $O(\log n + \log^2 r)$ using $O((nr)^c)$ work on an EREW PRAM, where c is some constant.*

6.2 An $O(nr^{O(1)})$-Work Method

The above method gives an optimal sized $(1/r)$-cutting, but it clearly does so at great expense. Nevertheless, it can be used to derive a more efficient method. Such a method is based upon the following simple observations (which are analogues of observations made by others in different contexts [8, 35]).

Observation 6.2: *Let $(X, \mathcal{R})$ be a range space with factorable generator function $\mathcal{F}$. If Y is an $(1/r_1)$-approximation for $(X, \mathcal{R})$ and Z is a $(1/r_2)$-net for $(Y, \mathcal{F}(Y)|_Y)$, then Z is a $(1/r_1 + 1/r_2)$-net for $(X, \mathcal{R})$.*

Observation 6.3: *Let X be a set of n hyperplanes in $\mathbb{R}^d$. If Y is an $(1/r)$-net for $(X, \mathcal{R})$ and C is the cutting formed by a canonical triangulation of Y's arrangement, then C is a (d/r)-cutting for X.*

We may "combine" these two observations to note

Observation 6.4: *Let X be a given set of n hyperplanes in $\mathbb{R}^d$. If Y is an $(1/r_1)$-approximation for $(X, \mathcal{R})$ and C is a $(1/r_2)$-cutting for Y, then C is a $d(1/r_1 + 1/r_2)$-cutting for X.*

Given these observations, we may then derive

Theorem 6.5: *Let X be a set of n hyperplanes in $\mathbb{R}^d$. One can construct an $O(r^d)$-sized $(1/r)$-cutting of X in time $O(\log n + \log^2 r)$ using $O(nr^c)$ work on an EREW PRAM, for any $1 < r < n$ and some constant $c > 0$.*

Proof: Let X be as given. By Observation 6.4, we may form a $(1/r)$-cutting by first applying Theorem 5.3 to construct a $(1/2dr)$-approximation of X and then apply Lemma 6.1 to form a $(1/2dr)$-cutting of this. The application of Theorem 5.3 takes $O(\log n)$ time using $O(nr^c)$ work, for some constant c. Since we assume d is a constant, the $(1/2dr)$-approximation produced can be made to be of size $O(n^\epsilon r^2)$, for any fixed constant $\epsilon > 0$. If we choose this ϵ to be the reciprocal of the constant c in Lemma 6.1, we may apply this lemma to construct a $(1/2dr)$-cutting of this approximation in $O(\log n + \log^2 r)$ time using $O(nr^c)$ work. This will be a $(1/r)$-cutting of size $O(r^d)$. ■

This is clearly an improvement in the work bound for constructing a $(1/r)$-cutting, but we can do even better. Our approach to finding an optimal-sized $(1/r)$-cutting efficiently in parallel is based upon an elegant recursive-subdivision idea due to Chazelle [8]. In this approach one forms an $O(1)$-sized $(1/r_0)$-cutting of X, for some constant r_0, and then one recurses on each simplex. Recursing $\log_{r_0} r$ levels gives us a $(1/r)$-cutting. Unfortunately, a simple-minded application of this approach using Theorem 6.5 would not yield an optimal-sized cutting. This is because the constant "hiding" behind the big-Oh in the size bound of Theorem 6.5 would grow with each level of the recursion, so that after $\log_{r_0} r$ levels the total size of the cutting could be as large as $\Theta(r^{d+\delta})$, for some constant δ, and not $O(r^d)$ as we would desire.

6.3 Achieving Intersection Sensitivity

In the generic recursive call, one has a set Y of hyperplanes from X and a simplex σ that they all intersect. The elegance in Chazelle's implementation of this approach [8] is that he shows how find each recursive $(1/r_0)$-cutting in σ so that its size depends on r_0 and on the number of hyperplane intersections that occur inside σ. In this way he is able to avoid the size blow-up problem mentioned above. Unfortunately, for our purposes, his approach again uses the conditional probabilities technique. Thus, if we are to adapt his approach to our parallel setting, we must derive an intersection-sensitive method for constructing a $(1/r)$-cutting that does not use this technique.

So, suppose we are given a set X of n hyperplanes that all intersect a given simplex σ. We can define a range space for X by considering all combinatorially distinct ways that the hyperplanes in X can be intersected by segment inside σ. Given a subset $Y \subseteq X$, we define $\mathcal{I}_\sigma(Y)$ to be the set of intersection points inside σ that are determined by the hyperplanes in Y. We can define a generator function $\mathcal{F}$ for such a Y by considering all combinatorially distinct segments that could have as an endpoint a point in $\mathcal{I}_\sigma(Y)$ or the point determined by the intersection of σ and $d-1$ hyperplanes in Y. This range space clearly has bounded scaffold dimension (it is $2d$). It is also easy to see that this range space is factorable.

Given $Y \subseteq X$, we say that Y is *sensitive*[8] to X in σ if $|\mathcal{I}_\sigma(Y)| \leq 4(|Y|/n)^d|\mathcal{I}_\sigma(X)|$.

Lemma 6.6: *Let X be a set of n hyperplanes in $\mathbb{R}^d$ that all intersect a given simplex σ. Then, in time $O(\log n)$ using $O((nr)^c)$ work on an EREW PRAM, one can construct an $O(r^{1+\epsilon})$-sized $(1/r)$-net Y for $(X, \mathcal{F}(Y))$ that is sensitive to X in σ, for any fixed constant $\epsilon > 0$.*

Proof: The proof is essentially the same as for Lemma 4.1 in Theorem 4.2, except in this case the chosen subset Y must also satisfy the geometric condition $|\mathcal{I}_\sigma(Y)| \leq 4(|Y|/n)^d|\mathcal{I}_\sigma(X)|$. So, let us consider the probability of this occurring if Y is a k-wise independent sample of X. If $k \geq d$, then $E(|\mathcal{I}_\sigma(Y)|) = (|Y|/n)^d|\mathcal{I}_\sigma(X)|$. Thus, by Markov's inequality,

$$\Pr(|\mathcal{I}_\sigma(Y)| > 4(|Y|/n)^d|\mathcal{I}_\sigma(X)|) \leq 1/4.$$

Adding this as an extra condition on Y, therefore, does not violate our application of the restricted independence derandomization technique, and increases the time and work bounds of Theorem 4.2 by at most a constant factor. ■

We can use this to derive an efficient method for finding an intersection-sensitive $(1/r)$-cutting by prefacing it by the construction of an appropriate $(1/r)$-approximation. The following composition lemma due to Chazelle [8] will prove to be useful in this regard:

Lemma 6.7 (Chazelle [8]): *Let X be a set of n hyperplanes in $\mathbb{R}^d$ that intersect a given simplex σ. If Y is an $(1/2dr)$-approximation for $(X, \mathcal{R})$ and Z is a $(1/2dr)$-net for Y that is sensitive to Y in σ, then Z is a $(1/r)$-net for X and $|\mathcal{I}_\sigma(Z)|$ is $O((|Z|/|X|)^d|\mathcal{I}_\sigma(X)| + |Z|^d/r)$.*

This leads to

Lemma 6.8: *Let X be a set of n hyperplanes in $\mathbb{R}^d$ that all intersect a given simplex σ, and let $1 < r < n$ be a given parameter. Then, in time $O(\log n + \log^2 r)$ using $O(nr^c)$ work on an EREW PRAM, one can construct a $(1/r)$-cutting C for X in σ of size $O(r^{(1+\epsilon)d-1} + (r^{1+\epsilon}/n)^d|\mathcal{I}_\sigma(X)|)$, for any fixed constant $\epsilon > 0$.*

Proof: Apply Theorem 5.3 to find a $(1/2dr)$-approximation Y of X, and then apply Lemma 6.6 to construct Z, a $(1/2dr)$-net of Y that is sensitive to Y in σ. The set Y can be made to be of size $(n^\epsilon r^2)$ and the set Z can be made to be of size $O(r^{1+\epsilon})$, and, by Observation 6.2, Z is a $(1/dr)$-net for X. The constant ϵ can be chosen so that the total work needed is $O(nr^c)$ for some constant c. By Observation 6.3, we can construct the arrangement and canonical triangulation of Z to form a $(1/r)$-cutting, C, of X in σ. The size of C is $O(|Z|^{d-1} + |\mathcal{I}_\sigma(Z)|) = O(r^{(1+\epsilon)(d-1)} + |\mathcal{I}_\sigma(Z)|)$, since $|C|$ is proportional to the number of intersection points determined by σ and the hyperplanes in Z. By Lemma 6.7, then, we have that $|C|$ is $O(r^{(1+\epsilon)d-1} + (r^{1+\epsilon}/n)^d|\mathcal{I}_\sigma(X)|)$. ■

6.4 An $O(nr^{d-1})$-Work Method

We are now ready to describe our algorithm for efficiently constructing a $(1/r)$-cutting, which is based upon the simple recursive subdivision technique of Chazelle [8]. Suppose we are given a set X of n hyperplanes in $\mathbb{R}^d$ and a parameter $r < n$ such that r is greater than some constant $r_0 \geq 2$. We begin by applying Theorem 6.5 to construct a $(1/r_0)$-cutting C_0 of size $O(r_0^d)$ in $O(\log n)$ time using a linear amount of work. For any simplex σ, let X_σ denote the set of hyperplanes in X that intersect σ's interior. For $i = 1$ to $\log_{r_0} r$ we refine C_{i-1} into C_i by applying Lemma 6.8 to form a $(1/\rho_0)$-cutting of each simplex σ in C_{i-1} such that $|X_\sigma| > n/r_0^i$, where $\rho_0 = r_0^i|X_\sigma|/n$. Each such cutting requires $O(\log n)$ time using $O(|X_\sigma|)$ work. Clearly, $\rho_0 \leq r_0$, and C_i is a $(1/r_0^i)$-cutting. Thus, when the computation completes (i.e., when $i = \log_{r_0} r$), C_i is a $(1/r)$-cutting. By Lemma 6.8, we can bound $|C_i|$ by

$$|C_i| \leq \sum_{\sigma \in C_{i-1}} c\left(r_0^{(1+\epsilon)d-1} + (r_0^{1+\epsilon}/|X_\sigma|)^d|\mathcal{I}_\sigma(X_\sigma)|\right),$$

where c is some constant. Since $\sum_{\sigma \in C_{i-1}} |\mathcal{I}_\sigma(X_\sigma)|$ is $O(n^d)$ and $|X_\sigma| \leq n/r_0^{i-1}$, we can re-write this

$$|C_i| \leq cr_0^{(1+\epsilon)d-1}|C_{i-1}| + cr_0^{(i+\epsilon)d}.$$

A simple induction proof shows that r_0 can be chosen large enough and ϵ small enough so that $|C_i|$ is $O(r_0^{id})$. Thus, $|C|$ is $O(r^d)$. The running time is $O(\log n \log r)$ and the total work used is proportional to

$$\sum_{i=1}^{\log_{r_0} r} n|C_i|/r_0^i \leq \sum_{i=1}^{\log_{r_0} r} nr_0^{i(d-1)},$$

which is $O(nr^{d-1})$. Thus, we have the following:

[8] Chazelle [8] refers to such a Y as a *strong* net for X.

Theorem 6.9: *Let X be a set of n hyperplanes in $\mathbb{R}^d$, and let $r < n$ be a given parameter larger than some constant $r_0 \geq 2$. One can construct an $O(r^d)$-sized $(1/r)$-cutting of X in time $O(\log n \log r)$ using $O(nr^{d-1})$ work on an EREW PRAM.*

6.5 Even More-Efficient Constructions

Matoušek [36] shows how to use results analogous to Theorems 5.3 and 6.9 to derive a recursive refinement method for constructing $(1/r)$-cuttings and $(1/r)$-approximations in $O(n \log r)$ time if $r \leq n^\alpha$ for some constant $\alpha > 0$. Substituting our theorems into his method (and implementing it in parallel) immediately implies the following:

Theorem 6.10: *Let X be a set of n hyperplanes in $\mathbb{R}^d$, and let r be such that $2 < r_0 < r \leq n^\alpha$, for some constant r_0 and some constant $\alpha > 0$ depending on d. Then, in time $O(\log n \log r)$ using $O(n \log r)$ work on an EREW PRAM, one can construct an $O(r^d)$-sized $(1/r)$-cutting of X or an $O(r^{d+\epsilon})$-sized $(1/r)$-approximation of X, for any constant $\epsilon > 0$.*

7 Applications

In this section we give a few consequences of our work.

7.1 Constructing 3-D Convex Hulls

Let X be a set of n points in $\mathbb{R}^3$. The 3-dimensional convex hull problem is to construct a representation of the boundary of the smallest convex set containing X. The best previous deterministic parallel algorithms for this problem run in $O(\log^2 n)$ time using $O(n \log^2 n)$ work on a CREW PRAM [5, 21, 23]. In the full version we show how to improve the work bound to $\Theta(n \log n)$ while keeping the time at $O(\log^2 n)$. This gives the first optimal-work *NC* algorithm for this problem, solving an open problem posed by Amato and Preparata [5]. Our approach is to derandomize the optimal-work randomized parallel algorithm of Reif and Sen [40], which is based on the random sampling technique. This work bound is optimal in the worst case, assuming the output size is $\Theta(n)$. In the general case, however, we believe one should be able to achieve $O(n \log h)$ work by using the iterative-squaring approach of Chazelle and Matoušek [10].

7.2 Distance Selection

The next application we give shows that our methods can actually improve the sequential complexity of a problem if it is used in conjunction with the parametric searching technique, which was introduced by Megiddo [38] and extended by several others [3, 12, 16, 17, 19, 20]. The particular problem we address is *distance selection*, where one is given a set X of n points in the plane and one wishes to find a pair of points in X that realize the k-th smallest distance.

In the full version we show how to apply the parametric searching technique and the methods from previous sections to derandomize the method of Agarwal *et al.* [2], deriving the following:

Theorem 7.1: *Given a set X of n points in the plane, one can determine a pair (p, q) in X that determine the k-th smallest distance in $O(n^{4/3} \log^{8/3} n)$ time.*

Acknowledgements

We would like to thank Chee Yap and Günter Rote for noticing some problems with an earlier version of this paper. We would also like to thank Bonnie Berger, Jiří Matoušek, and John Rompel for several helpful e-mail exchanges concerning the topics of this paper.

References

[1] P.K. Agarwal, "Geometric Partitioning and Its Applications," TR CS-1991-27, Dept. of Comp. Sci., Duke Univ., 1991.

[2] P.K. Agarwal, B. Aronov, M. Sharir, and S. Suri, "Selecting Distances in the Plane," *Proc. 6th ACM Symp. on Computational Geometry*, 1990, 321–331.

[3] P.K. Agarwal, M. Sharir, and S. Toledo, "New Applications of Parametric Searching in Computational Geometry," *Proc. 3rd ACM-SIAM Symp. on Disc. Algorithms*, 1992, 72–82.

[4] N. Alon, J.H. Spencer, and P. Erdös, *The Probabilistic Method*, Wiley-Interscience (New York: 1992).

[5] N.M. Amato and F.P. Preparata, "The Parallel 3D Convex Hull Problem Revisited," *Int. J. of Computational Geometry & Appl.*, **2**(2), 1992, 163–173.

[6] D. Angluin and L.G. Valiant, "Fast Probabilistic Algorithms for Hamiltonian Circuits and Matchings," *J.C.S.S.*, **18**, 1979, 155–193.

[7] B. Berger, J. Rompel, and P. Shor, "Efficient NC algorithms for set cover with applications to learning and geometry," *Proc. 30th IEEE Symp. on Found. Comp. Sci.*, 1989, 54–59.

[8] B. Chazelle, "An optimal convex hull algorithm and new results on cuttings," *Proc. 32nd IEEE Symp. on Found. Comp. Sci.*, 1991, 29–38.

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

[10] B. Chazelle and J. Matoušek, 'Derandomizing an Output-Sensitive Convex Hull Algorithm in Three Dimensions," *Computational Geometry: Theory and Applications*, submitted.

[11] B. Chazelle and J. Matoušek, "On Linear-Time Deterministic Algorithms for Optimization Problems in Fixed Dimension," *Proc. 4th ACM-SIAM Symp. on Disc. Algorithms*, 1993, 281–289.

[12] B. Chazelle, H. Edelsbrunner, L. Guibas, and M. Sharir, "Diameter, Width, Closest Line Pair, and Parametric Searching," *Proc. 8th ACM Symp. on Computational Geometry*, 1992.

[13] H. Chernoff, "A Measure of Asymptotic Efficiency for Tests of a Hypothesis Based on the Sum of Observations," *Annals of Math. Stat.*, **23**, 1952, 493–507.

[14] K.L. Clarkson, "A randomized algorithm for closest-point queries," *SIAM J. Comput.*, **17**, 1988, 830–847.

[15] K.L. Clarkson and P.W. Shor, "Applications of random sampling in computational geometry, II," *Disc. & Comp. Geom.*, **4**, 1989, 387–421.

[16] R. Cole, "Slowing Down Sorting Networks to Obtain Faster Sorting Algorithms," *J. ACM*, 1987, 200–208.

[17] R. Cole, "Parallel Merge Sort," *SIAM J. Comput.*, **17**(4), 1988, 770–785.

[18] R. Cole and M.T. Goodrich, "Optimal Parallel Algorithms for Point-Set and Polygon Problems," *Algorithmica*, **7**, 1992, 3–23.

[19] R. Cole, J. Salowe, W. Steiger, and E. Szemerédi, "An Optimal-Time Algorithm for Slope Selection," *SIAM J. Comput.*, **18**, 1989, 792–810.

[20] R. Cole, M. Sharir, and C. Yap, "On k-Hulls and Related Problems," *SIAM J. Comput.*, **16**, 1987, 61–77.

[21] R. Cole and O. Zajicek, "An Optimal Parallel Algorithm for Building a Data Structure for Planar Point Location," *J. Par. and Dist. Comput.*, **8**, 1990, 280–285.

[22] T.H. Cormen, C.E. Leiserson, and R.L. Rivest, *Introduction to Algorithms*, MIT Press (Cambridge, Mass.: 1990).

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

[24] M.T. Dickerson and R.S. Drysdale, "Enumerating k Distances for n Points in the Plane," *Proc. 7th ACM Symp. on Computational Geometry*, 1991, 234–238.

[25] P.W. Dymon and S.A. Cook, "Hardware Complexity and Parallel Comp.," *Proc. 21st IEEE Symp. on Found. of Comp. Sci.*, 1980, 360–372.

[26] H. Edelsbrunner, *Algorithms in Combinatorial Geometry*, Springer-Verlag (New York: 1987).

[27] M.T. Goodrich, "Constructing Arrangements Optimally in Parallel," *Disc. & Comp. Geom.*, to appear.

[28] T. Hagerup and C. Rüb, "A Guided Tour of Chernoff Bounds," *Info. Proc. Let.*, **33**, 1989/90, 305–308.

[29] D. Haussler and E. Welzl, "Epsilon-nets and simplex range queries," *Disc. & Comp. Geom.*, **2**, 1987, 127–151.

[30] J. JáJá, *An Introduction to Parallel Algorithms*, Addison-Wesley, 1992.

[31] Karp, R.M., and Ramachandran, V., "Parallel Algorithms for Shared-Memory Machines," in *Handbook of Theoretical Computer Science, A: Algorithms and Complexity*, MIT Press, 1990, 869–942.

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

[33] M. Luby, "Removing randomness in parallel computation without a processor penalty," *Proc. 29th IEEE Symp. on Found. Comp. Sci.*, 1988, 162–173.

[34] J. Matoušek, "Cutting hyperplane arrangements," *Disc. & Comp. Geom.*, **6**, 1991.

[35] J. Matoušek, "Approximation and geometric divide and conquer," *Proc. 23rd Annu. ACM Sympos. Theory Comput.*, 1991, 505–511.

[36] J. Matoušek, "Efficient partition trees," *Disc. & Comp. Geom.*, **8**, 1992, 315–334.

[37] J. Matoušek, E. Welzl, L. Wernisch, "Discrepancy and ϵ-Approximations for Bounded VC-Dimension," *Proc. 32nd IEEE Symp. on Found. Comp. Sci.*, 1991, 424–430.

[38] N. Megiddo, "Applying Parallel Computation Algorithms in the Design of Serial Algorithms," *J. ACM*, **30**, 1983, 852–865.

[39] R. Motwani, J. Naor, and M. Naor, "The probabilistic methods yields deterministic parallel algorithms," *Proc. 30th IEEE Symp. on Found. Comp. Sci.*, 1989, 8–13.

[40] J.H. Reif and S. Sen, "Optimal Parallel Randomized Algorithms for Three-Dimensional Convex Hulls and Related Problems," *SIAM J. Comput.*, **21**(3), 1992, 466–485.

[41] J.T. Rompel, *Techniques for Computing with Low-Independence Randomness*, Ph.D. Thesis, Dept. of Electrical Engineering and Computer Science, M.I.T., 1990.

[42] J.P. Schmidt, A. Siegel, and A. Srinivasan, "Chernoff-Hoeffding Bounds for Applications with Limited Independence," *Proc. 4th ACM-SIAM Symp. on Disc. Algorithms*, 1993, 331–340.

[43] N. Sauer, "On the density of families of sets," *J. of Combin. Theory Ser. A*, **13**, 1972, 145-147.

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

[45] C.K. Yap, "Parallel Triangulation of a Polygon in Two Calls to the Trapezoidal Map," *Algorithmica*, **3**, 1988, 279–288.