Chapter 16
Fixed-Dimensional Parallel Linear Programming via Relative ϵ-Approximations

Michael T. Goodrich*

Abstract

We show that linear programming in $\mathbb{R}^d$ can be solved deterministically in $O((\log\log n)^d)$ time using linear work in the PRAM model of computation, for any fixed constant d. Our method is developed for the CRCW variant of the PRAM parallel computation model, and can be easily implemented to run in $O(\log n(\log\log n)^{d-1})$ time using linear work on an EREW PRAM. A key component in these algorithms is a new, efficient parallel method for constructing ϵ-nets and ϵ-approximations (which have wide applicability in computational geometry). In addition, we introduce a new deterministic set approximation for range spaces with finite VC-exponent, which we call the *δ-relative ϵ-approximation*, and we show how such approximations can be efficiently constructed in parallel.

1 Introduction

The linear programming problem is central in the study of discrete algorithms. It has been applied to a host of combinatorial optimization problems since the first efficient algorithms for solving it were developed in the 1940's (e.g., see [17, 22, 37, 54]). Geometrically, it can be viewed as the problem of locating a point that is maximal in a given $\vec{v}$ direction in the polyhedral region P defined by the intersection of n halfspaces in $\mathbb{R}^d$. This viewpoint is particularly useful for the case when the dimensionality, d (corresponding to the number of variables), is fixed, as occurs, for example, in several applications of linear programming in geometric computing (e.g., see [15, 20, 27, 50, 51, 55]) and machine learning (e.g., see [9, 10]). Indeed, a major contribution of computational geometry research has been to show that fixed-dimensional linear programming can be solved in linear time, starting with the seminal work of Dyer [25] and Megiddo [50, 51], and following with subsequent work in the sequential domain concentrated primarily on reducing the constant "hiding behind" the big-oh in these results (e.g., see [15, 18, 20, 26, 36, 46, 60]) or on building data structures for linear programming queries (e.g., see [28, 49]).

In the parallel domain, Alon and Megiddo [3] give analogous results, showing that through the use of randomization one can solve a fixed-dimensional linear program in $O(1)$ time with very high probability using n processors in a randomized CRCW PRAM model[1]. The existing deterministic parallel algorithms are not as efficient, however. Ajtai and Megiddo [2] give a deterministic $O((\log\log n)^d)$ time method, but it has a sub-optimal $O(n(\log\log n)^d)$ work[2] bound and it is defined for the very powerful parallel model that only counts "comparison" steps [62]. The only work-optimal deterministic PRAM result we are familiar with is a method by Deng [23] for 2-dimensional linear programming that runs in $O(\log n)$ time using $O(n)$ work on a CRCW PRAM. Recently, Dyer [24] has given an $O(\log n(\log\log n)^{d-1})$ time method that uses $O(n\log n(\log\log n)^{d-1})$ work in the EREW PRAM model. In addition, we have recently learned that Sen [61] has independently discovered a CRCW PRAM LP method that runs in $O((\log\log n)^{d+1})$ time using $O(n)$ work.

1.1 Our results for parallel LP. In this paper we give a deterministic parallel method for fixed dimensional linear programming that runs in $O((\log\log n)^d)$ time using $O(n)$ work in the CRCW PRAM model. Thus, our method improves the work bound and the computational model of the Ajtai-Megiddo method while matching their running time, which is also an improvement over the time bound of Deng's method for $d = 2$. (It is also slightly faster than the recent result by Sen, which uses an approach that is considerably different than that for our method.) In addition, our method can be implemented in the EREW PRAM

*This research supported by the National Science Foundation under Grants IRI-9116843 and CCR-9300079.

[1] Recall that this is the synchronous shared-memory parallel model where processors are allowed to perform concurrent reads and concurrent writes, with concurrent writes being resolved, say, by requiring all writing processors to be writing the same *common* value (this standard resolution rule is the one we use in this paper). Alternatively, in the weaker EREW PRAM model processors may not concurrently access the same memory location.

[2] 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.

model to run in $O(\log n(\log\log n)^{d-1})$ time using $O(n)$ work, which improves the work bound of the parallel method by Dyer.

At a high level our method is actually quite simple: we efficiently derandomize a simple recursive procedure. In order to avoid the large time and processor bounds that come from known derandomization methods, however, we have had to develop a new approach to the parallel derandomization of geometric algorithms—one that is more "approximate" than previous approaches.

1.2 Derandomization. Randomized algorithms in computational geometry most often exploit small-sized random samples, and the derandomization of such algorithms is then done by (1) quantifying the combinatorial properties needed by random samples, and (2) showing that sets having these combinatorial properties can be constructed efficiently without using randomization. Interestingly, most of the combinatorial properties needed by geometric random samples can be characterized by two notions—the *ϵ-approximation* [45, 63] and the *ϵ-net* [34, 45]. These concepts are defined for very general frameworks, where one is given a set system $(X, \mathcal{R})$ consisting of a finite ground set, X, and a set, $\mathcal{R}$, of subsets of X. The subsets in $\mathcal{R}$ are often referred to as *ranges*, for $\mathcal{R}$ typically is defined in terms of some well-structured geometry or combinatorics. A subset Y is an ϵ-approximation for $(X, \mathcal{R})$ if, for each range $R \in \mathcal{R}$,

$$\left|\frac{|Y \cap R|}{|Y|} - \frac{|R|}{|X|}\right| \leq \epsilon.$$

That is, Y is such that the absolute error between $|Y \cap R|/|Y|$ and $|R|/|X|$ is at most ϵ. Relaxing this error requirement a bit, Y is said to be an *ϵ-net* [34, 45] of $(X, \mathcal{R})$ if $Y \cap R \neq \emptyset$ for each $R \in \mathcal{R}$ such that $|R| > \epsilon|X|$. This is clearly a weaker notion than that of an ϵ-approximation, for any ϵ-approximation is automatically an ϵ-net, but the converse need not be true.

We generalize the ϵ-approximation definition to say that, given non-negative parameters $\delta < 1$ and $\epsilon < 1$, a subset Y is a *δ-relative ϵ-approximation* if, for each range $R \in \mathcal{R}$,

$$\left|\frac{|Y \cap R|}{|Y|} - \frac{|R|}{|X|}\right| \leq \delta\frac{|R|}{|X|} + \epsilon.$$

This notion is a combined measure of the absolute and relative error between $|Y \cap R|/|Y|$ and $|R|/|X|$, and it is somewhat similar to a notion Brönnimann *et al.* [13] refer to as a "sensitive" ϵ-approximation. Note that this notion also subsumes that of an ϵ-net, for any δ-relative ϵ-approximation is automatically an $(\epsilon/(1-\delta))$-net.

Of course, our specific interest in this paper is in the design of fast and efficient deterministic methods for constructing small-sized δ-relative ϵ-approximations in parallel and applying these methods to fixed-dimensional linear programming. Our methods have other applications as well, including fixed-dimensional convex hull and geometric partition construction [5, 6], but these are beyond the scope of this paper.

1.3 Previous work on derandomizing geometric algorithms. Before we describe our results, however, let us review some related previous work. The study of random sampling in the design of efficient computational geometry methods really began in earnest with some outstanding early work of Clarkson [19], Haussler and Welzl [34], and Clarkson and Shor [21]. One general type of geometric structure that has motivated much of the derandomization research, and one that motivated the development of the ϵ-approximation and ϵ-net notions for computational geometry, is the *geometric partition* (e.g., see [1, 45]). In this problem, one is given a collection X of n hyperplanes in $\mathbb{R}^d$, and a parameter r, and one wishes to construct a partition of $\mathbb{R}^d$ into $O(r^d)$ constant-sized cells so that each cell intersects as few hyperplanes as possible. It is easy to see, for example, that an $O(r)$-sized ϵ-net of X can be used to construct such a partitioning so that each cell intersects at most ϵn hyperplanes. Moreover, one can apply random sampling to construct such a geometric partitioning of space for $\epsilon = \log r/r$ [21, 34]. Chazelle and Friedman [14] show that one can construct such a partitioning with $\epsilon = 1/r$ deterministically in polynomial time, and Berger, Rompel, and Shor [11] and Motwani, Naor, and Naor [52] show that one can construct similar geometric partitions for $\epsilon = \log r/r$ in NC. 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).

One can improve the running time of the Chazelle and Friedman algorithm for the case when the range space $(X, \mathcal{R})$, where $\mathcal{R}$ is defined as the set of combinatorially distinct ways to intersect X by "cells," has bounded Vapnik-Chervonenkis [63] (VC)-dimension. Letting $\mathcal{R}|_A$ denote the set $\{A \cap R : R \in \mathcal{R}\}$, the *VC-dimension* of $(X, \mathcal{R})$ is defined as the maximum size of a subset A of X such that $\mathcal{R}|_A = 2^A$ (e.g., see [45]). A related and simpler notion, however, is based upon the *shatter function*

$$\pi_{\mathcal{R}}(m) = \{|\mathcal{R}|_A| : A \subseteq X, |A| = m\}.$$

In particular, we say that $(X, \mathcal{R})$ has *VC-exponent* [7, 12] e if $\pi_{\mathcal{R}}(m)$ is $O(m^e)$[3]. For example, in the *hyperplane set system*, where X is a set of n hyperplanes in $\mathbb{R}^d$ and $\mathcal{R}$ is the set of all combinatorially distinct ways of intersecting hyperplanes with simplices, it is easy to see that the VC-exponent is bounded by $d(d+1)$. Interestingly, the VC-exponent definition subsumes that of the VC-dimension, for if $(X, \mathcal{R})$ has VC-dimension e, then it has VC-exponent e as well [58, 63]. There are several recent results that show that one can construct a $(1/r)$-approximation of size $O(r^2 \log r)$ for any range space with bounded VC-exponent e in time $O(nr^c)$ for some constant c depending on e (e.g., see [13, 15, 44, 43, 48, 47]). In addition, the author [30] has shown that one can construct such sets of size $O(n^\epsilon r^2)$ in parallel in $O(\log n)$ time using $O(nr^c)$ work on an EREW PRAM. Chazelle and Matoušek [15] give slower NC algorithms using $O(nr^c)$ work that construct such sets of size $O(r^{2+\alpha})$ for any fixed constant $\alpha > 0$.

1.4 Our results on parallel geometric derandomization. We give fast and efficient efficient parallel algorithms for constructing ϵ-nets and δ-relative ϵ-approximations. For example, our methods can be implemented in the CRCW PRAM model to run in $O(\log \log n)$ time using $O(nr^c)$ work to produce $(\log \log r)^{-b}$-relative $(1/r)$-approximations of size $O(r^{2+\alpha})$ for any fixed constants $\alpha > 0$ and $b > 0$, and some constant $c \geq 1$. We also show how to find such approximations of size $O(r^2 \log r)$ using more time and work. In addition, our methods can be implemented in the EREW PRAM model to run in $O(\log n)$ time using $O(nr^c)$ work to produce (0-relative) $(1/r)$-approximations of size $O(r^{2+\alpha})$ for any fixed constant $\alpha > 0$. Thus, our methods improve the previous size bounds from those achieved previously by the author [30] while also improving the time bounds from those achieved previously by Chazelle and Matoušek [15]. We also derive similar bounds for constructing $(1/r)$-nets, and this is the result we use to design a new efficient parallel method for fixed-dimensional linear programming.

2 Linear Programming in Fixed Dimensions

Recall the geometric view of fixed-dimensional linear programming. For simplicity of expression, let us assume that the optimal point p exists and is defined by the intersection of exactly d halfspace boundaries. Let us also assume that the origin, o, is contained in P, the polytope defined by the linear constraints. These assumptions can be removed with minor modifications to our method (similar to those used, for example, by Seidel [60]). Without loss of generality, we may additionally assume that $\vec{v} = (0, 0, \ldots, 0, -1)$, i.e., we are interested in the "lowest" vertex in P. Our method for finding p is inspired by the methods of Ajtai and Megiddo [2] and Dyer [24], but is nevertheless quite different. We find the optimal solution p by calling the following recursive procedure as $\mathbf{ParLP}_d(X, 2n)$.

Procedure $\mathbf{ParLP}_d(X, w)$:
Output: An optimal solution p for X (using work that is $O(w)$).

1. Let $n = |X|$. If $n \leq n_0$, find the optimal solution by any "brute force" method, where n_0 is a constant set in the analysis, and return. Likewise, if $d = 1$, then compute the minimum of the numbers in X and return.
2. Compute a $(1/r)$-net Y for X of size $O(r^{1+\epsilon})$ (in the hyperplane set system), where $r = (w/n)^{1/c}$ such that c is a constant to be set in the analysis and ϵ is a sufficiently small constant. (As we will show in the sections that follow, the time needed for this step is $O(\log \log n)$ in a CRCW PRAM implementation; the work needed for this step can be made $O(w)$ if c is a large enough constant (larger than the constant of Theorem 4.1).)
3. Compute the intersection of the halfspaces in Y and a canonical triangulation $\mathcal{T}$ [14] of this polyhedral region (with the origin as base apex), using a "brute force" method that uses $O(r^c)$ work. (In a CRCW implementation this can be done in $O(\log \log r)$ time; an EREW implementation takes $O(\log r)$ time. Both implementations are simple applications of parallel minimum-finding [35, 39, 56] and are left to the reader.)
4. Using $\mathbf{ParLP}_{d-1}$ as a subroutine, determine the simplex σ in $\mathcal{T}$ that contains p. This is implemented as follows:
   (a) For each simplex σ in $\mathcal{T}$ compute the intersection of the halfspaces in X with each of σ's $(d-1)$-dimensional boundary faces. This takes $O(1)$ time with $O(nr^{1+\epsilon})$ work, which is $O(w)$ if $c \geq 1 + \epsilon$.
   (b) For each simplex boundary face f we use $\mathbf{ParLP}_{d-1}$ to solve the linear program defined by f and the halfspaces that intersect f. Assuming that $\mathbf{ParLP}_{d-1}$ uses linear work, this step can be implemented using $O((n/r)r^{(1+\epsilon)\lfloor d/2 \rfloor})$ work, which is $O(w)$ if $c \geq (1+\epsilon)\lfloor d/2 \rfloor - 1$.

[3] Strictly speaking, we should define e as the infimum of all numbers s such that $\pi_{\mathcal{R}}(m)$ is $O(m^s)$, but this definition will suffice for our purposes.

(c) Each point that forms a solution to the linear program for a boundary face f of simplex σ belongs to a line L_f that intersects σ. The simplex that contains the true optimal point p can therefore be determined in $O(1)$ time by examining, for each simplex σ, how the L_f lines for its faces intersect σ. Since d is a fixed constant, this step can be implemented using $O(n)$ work.

Thus, if c is a large enough constant (which may depend upon d), then this step can be implemented using $O(w)$ work.

5. Compress the array of halfspaces whose boundary intersects this simplex σ and recursively call $\mathbf{ParLP}_d$ on this set of at most n/r halfspaces. The work bound we pass to this recursive call is w, unless this level in the recursion is equal to $ci + 1$, for some integer $i \geq 1$, in which case we pass the work bound $w/2^{1/c}$. (To implement this step in the CRCW PRAM model we use *λ-approximate compaction* [29, 32, 42], where one is given an array A with m of its locations "occupied" and one wishes to map these m distinguished elements to an array B of size $(1+\lambda)m$. The time bound is $O(\log\log n)$ [29] using linear work. Of course, in the EREW PRAM model this step can easily be implemented in $O(\log n)$ time via a parallel prefix computation [35, 39, 56].)

Since this method always recurses in a region σ guaranteed to contain the optimal point and we include in the subproblem all halfspaces whose boundary intersects σ, we will eventually find the optimal point p. To analyze the time complexity observe that every c levels in the recursion the problem size will go from n/r to n/r^2. Thus, the total depth in the recursion tree is $O(\log\log n)$. For $d = 2$, therefore, the running time in a CRCW PRAM implementation is $O((\log\log n)^2)$; hence, the running time for $d > 2$ is $O((\log\log n)^d)$ in this model. An EREW PRAM implementation would take $O(\log n \log\log n)$ time for $d = 2$; hence, the running time for $d > 2$ would be $O(\log n(\log\log n)^{d-1})$ in this model. As we have already observed, we can set c so that the work needed in each level of the recursion is $O(w)$. Moreover, since we decrease w by a constant factor every c levels in the recursion, the total work needed is $O(n)$. This gives us the following:

THEOREM 2.1. *Linear programming in $\mathbb{R}^d$ can be solved using $O(n)$ work and $O((\log\log n)^d)$ time on a CRCW PRAM, or, alternatively, using $O(n)$ work and $O(\log n(\log\log n)^{d-1})$ time on an EREW PRAM, for fixed d.*

Of course, this theorem depends upon our demonstrating a fast and efficient method for computing $(1/r)$-nets. Before we describe such a work-efficient method, however, we first describe some algorithms for constructing $(1/r)$-nets and $(1/r)$-approximations that are fast but not work-efficient.

3 $O((nr)^{O(1)})$-Work Approximation Finding

Our approach to constructing small-sized approximations and nets of range spaces with bounded VC-exponent is to derandomize a straightforward probabilistic algorithm, **Approx**, which is based upon the *random sampling* technique [19].

3.1 Probabilistic analysis. We do this using the *limited independence* technique [4, 38, 40, 41], which assumes **Approx** uses random variables that are only k-wise independent. The generic situation is that one is given a set X of n objects and an integer parameter s, and one wishes to construct a subset $Y \subset X$ of size s. In this paper we assume such a sample is chosen by defining, for each element x_i in X in parallel, a random variable X_i that is 1 with probability s/n; we use the rule that $x_i \in Y$ if $X_i = 1$ [11]. Note that one is guaranteed a set of $|Y| = X_1 + X_2 + \cdots + X_n$ unique elements, which we call an *expected s-sample*, for its size may not be equal to s, although it is easy to see, by the linearity of expectation, that $E(|Y|) = s$. We also restrict the X_i's to be only k-wise independent for some integer parameter k. Unfortunately, restricting our attention to k-wise independent indicator random variables prevents us from directly using the well-known and powerful Chernoff bounds [4, 16, 33, 53] for bounding the tail of the distribution of their sum. Nevertheless, as shown by Rompel [57], we may derive something analogous:

LEMMA 3.1. ([57]) *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 $\lambda > 0$.

Incidentally, this also seems to follow from an inequality of Schmidt, Siegel, and Srinivasan [59], which may yield a better constant factor.

We can easily derandomize such algorithms in parallel by using the the *limited independence* technique [4, 40, 41]. In applying this technique we construct an $O(n^{O(k)})$-sized k-wise independent probability space such that each vector in this space represents

an assignment of 0's and 1's to the underlying random variables. Then, we can deterministically simulate the running of the randomized algorithm using each vector in this space. At least one must succeed, and we can then take our output to be that of one of these succeeding simulations. We review the details of this technique in the full version.

3.2 Geometric random samples. Our first methods for finding approximating subsets of X are derived directly from the limited-independence approach and can be implemented to run very fast in parallel, albeit with a rather large number of processors (we will subsequently show how to improve these processor bounds).

Let $(X, \mathcal{R})$ be a given range space with bounded VC-exponent, e. Given a parameter $1 < r \leq |X|$, a parameter s that is greater than some fixed constant $s_0 > 1$, and a positive even integer k, let Y be a k-wise independent expected s-sample of X. Let us explore the probability that Y is an $O(s)$-sized (0-relative) $(1/r)$-approximation or $(1/r)$-net under various assumptions about s and k. The first lemma establishes the probability that $|Y|$ is $O(s)$.

LEMMA 3.2. *Let Y be defined as above. Then $s-\beta cs^{1/2} \leq |Y| \leq s+\beta cs^{1/2}$, with probability $1-1/\beta^{k/2}$, for some constant $c > 0$.*

Proof. Omitted in this extended abstract. ■

Let us therefore next bound the probability that Y is a $(1/r)$-net. Like previous arguments, which are based upon mutual independence (e.g., see [4, 34]), our k-wise independent analysis is based upon a double-sampling technique. Rather than define Y directly as an expected s-sample of X, we instead define Y to be an expected s-sample of a set, Z, which is an expected $(2s)$-sample of X, with both samples being defined by k-wise independent indicator random variables. That is, each member of Z is defined by a random variable $X_i^{(k)}$ that is 1 with probability $2s/n$ and each member of Y is defined by a random variable $\hat{X}_i^{(k)}$ that is 1 with probability 1/2 if $X_i^{(k)} = 1$. Let us further assume that $|Y| = s \pm \Theta(\sqrt{s})$ and $|Z| = 2s \pm \Theta(\sqrt{s})$, since, by Lemma 3.2, this can be made to occur with probability $1 - 1/c_0$ for any fixed constant $c_0 > 0$.

Let A be the event that there exists a set $R \in \mathcal{R}$ such that $|R| > n/r$ but $R \cap Y = \emptyset$. We wish to prove that $\Pr(A) \leq 1/2$. To do so we further define B to be the event that there exists a set $R \in \mathcal{R}$ such that $|R| > n/r$ but $R \cap Y = \emptyset$ *and* $|R \cap Z| \geq s/2r$.

LEMMA 3.3. *If $s \geq 8r$, then* $\Pr(B) \geq \Pr(A)/2$.

Proof. Clearly, $\Pr(B) = \Pr(A)\Pr(B|A)$. Thus, it suffices to show that $\Pr(B|A) \geq 1/2$. So, suppose event A occurs, i.e., there is a set $R \in \mathcal{R}$ such that $|R| > n/r$ but $R \cap Y = \emptyset$. The probability of B occurring, given A, is at least the probability that, for this particular R, $|R \cap Z| \geq s/2r$. Note that the quantity $|R \cap Z| = |R \cap Z \backslash Y|$ is defined by the sum of $|R|$ indicator random variables, each being 1 with probability $\frac{1}{2}(2s/n) = s/n$, i.e., it is a binomial random variable, with variance $|R|(s/n)(1-(s/n)) \leq |R|(s/n)$. Therefore, by Chebychev's inequality (which does not depend upon any independence assumptions),

$$\begin{aligned}\Pr(|R \cap Z \backslash Y| < s/2r) &\leq \Pr(|R \cap Z \backslash Y| < |R|s/2n) \\ &\leq \frac{|R|(s/n)}{(|R|s/2n)^2} \leq \frac{4r}{s}.\end{aligned}$$

Taking $s \geq 8r$, then, establishes the lemma. ■

For any set $R \in \mathcal{R}$, with $|R| \geq n/r$, let B_R denote the event that $R \cap Y = \emptyset$ and $|R \cap Z| \geq s/2r$. A crucial observation is that, having fixed the set Z, two events B_R and $B_{R'}$ are identical if $R \cap Z = R' \cap Z$. The occurrence of B_R depends only upon the intersection $R \cap Z$. Therefore, for any fixed Z, the number of distinct B_R events is bounded by $|\mathcal{R}|_Z|$. Since X has VC-exponent e, this is in turn bounded by $c|Z|^e$ for some constant c. Thus, $\Pr(B)$ is bounded by $c|Z|^e$ times the probability, for any range $R \in \mathcal{R}$, that B_R occurs.

LEMMA 3.4. *For any set $R \in \mathcal{R}$, given Z as above,* $\Pr(B_R) \leq C2^k(rk/s)^{k/2}$, *where C is some fixed constant.*

Proof. $\Pr(B_R)$ is equal to $\Pr(R \cap Y = \emptyset)$, given that $|R \cap Z| \geq s/2r$ for the set Z, which now is fixed. Note that $|R \cap Y|$ is the sum of $|R \cap Z|$ k-wise independent indicator random variables, each of which is 1 with probability 1/2. Moreover, $\mu = \mathbf{E}(|R \cap Y|) = |R \cap Z|/2 \geq s/4r$. Thus, we can use Lemma 3.1, assuming $\mu \geq k$, to bound $\Pr(R \cap Y = \emptyset) \leq \Pr(||R \cap Y| - \mu| \geq \mu) \leq C(k/\mu)^{k/2} \leq C(4rk/s)^{k/2}$, where C is the constant from Lemma 3.1. This completes the proof. ■

Therefore, we have the following:

LEMMA 3.5. $\Pr(B) \leq C(2s)^e 2^k (rk/s)^{k/2}$, *where e is the VC-exponent of $(X, \mathcal{R})$ and C is some fixed constant.*

Some immediate corollaries, then, are as follows.

COROLLARY 3.1. *Given a parameter $2 \leq r \leq n$ and any fixed constant $0 < \epsilon < 1$, there exist constants c_0 and k_0 (depending only upon ϵ and e), such that if $c_0 r^{1+\epsilon} \leq s \leq n$ and k is an even integer larger than k_0, then Y, chosen as above, is an $O(s)$-sized $(1/r)$-net of X with probability at least 1/2.*

COROLLARY 3.2. *Given a parameter $2 \leq r \leq n$, there exist constants b_1 and c_1 (depending only upon e), such that if $b_1 r \log r \leq s \leq n$ and k is an even integer*

larger than $c_1 \log r$, then Y, chosen as above, is an $O(s)$-sized $(1/r)$-net of X with probability at least $1/2$.

Having established the assumptions on k and s needed to allow Y to be a $(1/r)$-net with constant probability, we next turn to an analysis of the conditions needed for Y to be a $(1/r)$-approximation. Our analysis is similar in structure to that used to establish the above bounds for Y being a $(1/r)$-net, although the arguments are more intricate. Nevertheless, we give the details in the full version so as to derive the following corollaries:

COROLLARY 3.3. *Given a parameter $2 \leq r \leq n$ and any fixed constant $0 < \epsilon < 1$, there exist constants c_0 and k_0 (depending only upon ϵ and e), such that if $c_0 r^{2+\epsilon} \leq s \leq n$ and k is an even integer larger than k_0, then Y, chosen as above, is an $O(s)$-sized $(1/r)$-approximation of X with probability at least $1/2$.*

COROLLARY 3.4. *Given a parameter $2 \leq r \leq n$, there exist constants b_1 and c_1 (depending only upon e), such that if $b_1 r^2 \log r \leq s \leq n$ and k is an even integer larger than $c_1 \log r$, then Y, chosen as above, is an $O(s)$-sized $(1/r)$-approximation of X with probability at least $1/2$.*

3.3 CRCW PRAM algorithms. Unfortunely, we cannot immediately derive **Poly**($\log\log n$)-time methods for the CRCW PRAM from the above analysis, for checking if a given Y satisfies the condition for being a $(1/r)$-approximation requires $\Omega(\log n/\log\log n)$ time using a polynomial number of processor, by a simple reduction from the parity problem [8]. We can avoid this lower bound, however, by checking this condition approximately rather than exactly.

To do this we use a fast method for λ-approximate parallel prefix sums computation [29, 31], where one wishes to consistently compute all prefix sums of a sequence $(a_1, a_2, \ldots, a_n)$ with a relative error of λ.

LEMMA 3.6. ([29]) *λ-approximate parallel prefix sums can be computed in $O(1)$ time using polynomial work on a CRCW PRAM, with $\lambda = (\log\log n)^{-b}$, for any fixed constant $b > 0$.*

This lemma is crucial to our fast CRCW derandomization procedures, for we use it to estimate the sizes $|Y \cap R|$, $|Y|$, and $|R|$ that are needed in the definition of $(1/r)$-approximations and $(1/r)$-nets. In particular, for any such value x it allows us to derive an estimate x' such that $x/(1+\lambda) \leq x' \leq (1+\lambda)x$, for $\lambda \geq (\log\log n)^{-b}$ for any fixed constant $b > 0$. Let us therefore denote each of the estimates we need as $|Y \cap R|'$, $|Y|'$, and $|R|'$, respectively. (We may assume that $|X|$ is known explicitly.) Say that a set Y is *λ-estimated* to be a δ-relative ϵ-approximation if

$$\left| \frac{|Y \cap R|'}{|Y|'} - \frac{|R|'}{|X|} \right| \leq \delta \frac{|R|'}{|X|} + \epsilon.$$

LEMMA 3.7. *If Y is λ-estimated to be a δ-relative ϵ-approximation, then Y is a $(6\lambda + 3\delta)$-relative 2ϵ-approximation, provided $\lambda \leq 1/4$.*

Proof. Suppose Y is λ-estimated to be a δ-relative ϵ-approximation. Observe that $|Y \cap R|'/|Y|' \leq (1+\lambda)^2 |Y \cap R|/|Y|$ and that $|R|'/|X| \leq (1+\lambda)|R|/|X|$. Thus, by the definition of Y, we can derive the following bound on $\left| |Y \cap R|/|Y| - |R|/|X| \right|$:

$$\begin{aligned} & \left| \frac{|Y \cap R|'}{|Y|'} - \frac{|R|'}{|X|} \right| + \left| \frac{|Y \cap R|}{|Y|} - \frac{|Y \cap R|'}{|Y|'} \right| + \left| \frac{|R|'}{|X|} - \frac{|R|}{|X|} \right| \\ \leq\ & \delta(1+\lambda)\frac{|R|}{|X|} + ((1+\lambda)^2 - 1)\frac{|Y \cap R|}{|Y|} + \lambda \frac{|R|}{|X|} + \epsilon \\ =\ & (\lambda + (1+\lambda)\delta)\frac{|R|}{|X|} + \lambda(2+\lambda)\frac{|Y \cap R|}{|Y|} + \epsilon. \end{aligned}$$

We also know that

$$\begin{aligned} \frac{|Y \cap R|}{|Y|} &\leq (1+\lambda)^2 \frac{|Y \cap R|'}{|Y|'} \\ &\leq (1+\lambda)^2 \left((1+\delta)\frac{|R|'}{|X|} + \epsilon \right) \\ &\leq (1+\lambda)^3(1+\delta)\frac{|R|}{|X|} + (1+\lambda)^2\epsilon. \end{aligned}$$

Thus, we can combine the above bounds to derive the following bound on $\left| |Y \cap R|/|Y| - |R|/|X| \right|$:

$$\begin{aligned} & (\lambda + (1+\lambda)\delta)\frac{|R|}{|X|} + \lambda(2+\lambda)\left((1+\lambda)^3(1+\delta)\frac{|R|}{|X|} + (1+\lambda)^2\epsilon \right) + \epsilon \\ =\ & (\lambda + (1+\lambda)\delta + \lambda(2+\lambda)(1+\lambda)^3(1+\delta))\frac{|R|}{|X|} + (1 + \lambda(2+\lambda)(1+\lambda)^2)\epsilon \\ \leq\ & (6\lambda + 3\delta)\frac{|R|}{|X|} + 2\epsilon, \end{aligned}$$

provided $\lambda \leq 1/4$. ■

Likewise, we have the following:

LEMMA 3.8. *If Y is an ϵ-approximation, then Y will be λ-estimated to be a 4λ-relative 2ϵ-approximation, if $\lambda \leq 1/4$.*

Proof. Suppose Y is an ϵ-approximation. Then we can bound $\left| |Y \cap R|'/|Y|' - |R|'/|X| \right|$ by

$$\begin{aligned} & \left| \frac{|Y \cap R|}{|Y|} - \frac{|R|}{|X|} \right| + \left| \frac{|Y \cap R|'}{|Y|'} - \frac{|Y \cap R|}{|Y|} \right| + \left| \frac{|R|}{|X|} - \frac{|R|'}{|X|} \right| \\ \leq\ & \epsilon + \lambda(2+\lambda)\frac{|Y \cap R|}{|Y|} + \frac{\lambda|R|'}{|X|} \\ \leq\ & \epsilon + \lambda(2+\lambda)\left(\frac{|R|}{|X|} + \epsilon \right) + \frac{\lambda|R|'}{|X|} \\ \leq\ & \epsilon + \lambda(2+\lambda)\left(\frac{(1+\lambda)|R|'}{|X|} + \epsilon \right) + \frac{\lambda|R|'}{|X|} \\ \leq\ & \epsilon(1 + \lambda(2+\lambda)) + (\lambda(2+\lambda)(1+\lambda) + \lambda)\frac{|R|'}{|X|} \end{aligned}$$

$$\leq \quad 2\epsilon + 4\lambda\frac{|R|'}{|X|},$$

provided $\lambda \leq 1/4$. ■

These two lemmas together imply the following:

THEOREM 3.1. *Let $(X, \mathcal{R})$ be a range space with VC-exponent e, for some constant $e > 0$, and let $n = |X|$. Also, let $1 < r < n$ be a given parameter and let $\epsilon > 0$ be any fixed (small) constant. Then, in the CRCW PRAM model, for some constant $c > 0$, one can construct any of the following in the bounds claimed:*

1. *a $(\log\log n)^{-b}$-relative $(1/r)$-approximation A of $(X, \mathcal{R})$ of size $\Theta(r^{2+\epsilon})$ in $O(1)$ time using $O((nr)^c)$ work,*
2. *a $(\log\log n)^{-b}$-relative $(1/r)$-approximation B of $(X, \mathcal{R})$ of size $\Theta(r^2 \log r)$ in $O(1)$ time using $O((nr)^{c\log r})$ work,*
3. *a $(1/r)$-net C of $(X, \mathcal{R})$ of size $\Theta(r^{1+\epsilon})$ in $O(1)$ time using $O((nr)^c)$ work, or*
4. *a $(1/r)$-net D of $(X, \mathcal{R})$ of size $\Theta(r \log r)$ in $O(1)$ time using $O((nr)^{c\log r})$ work.*

Proof. Let us begin with the set A. We can set the constant in Corollary 3.1 high enough so that any s-sample Y is a $(1/4r)$-approximation with probability at least $1/2$, for $s = O(r^{2+\epsilon})$. By Lemma 3.8, this implies that in applying the limited independence derandomization technique there will be some Y λ-estimated to be a 4λ-relative $(1/2r)$-approximation. But, by Lemma 3.7, this in turn implies that Y is a (18λ)-relative $(1/r)$-approximation. By taking $\lambda = (\log\log n)^{-(b+1)}$, we make Y a $(\log\log n)^{-b}$-relative $(1/r)$-approximation (for n larger than some constant). The methods for constructing the other sets are similar applications of the limited-independence technique using Corollaries 3.2, 3.3, and 3.4, and Lemmas 3.7 and 3.8. ■

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

As already mentioned, the methods of the previous section are simple and can be implemented to run very fast in parallel. Their work complexities are quite high, however. In this section we show how to reduce this significantly.

Let $(X, \mathcal{R})$ be a range space with bounded VC-exponent e. We need some simple lemmas, which are adaptations of observations made by Matoušek [43].

LEMMA 4.1. *Suppose $Y_1, Y_2, \ldots, Y_m$ are δ-relative ϵ-approximations for disjoint range spaces $(X_1, \mathcal{R}|_{X_1})$, $(X_2, \mathcal{R}|_{X_2})$, ..., $(X_m, \mathcal{R}|_{X_m})$, respectively, where the X_i's 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 δ-relative ϵ-approximation for $(X, \mathcal{R})$.*

Proof. For any $R \in \mathcal{R}$, we can write

$$\begin{aligned}\left|\frac{|Y\cap R|}{|Y|} - \frac{|R|}{|X|}\right| &= \frac{1}{m}\left|\sum_{i=1}^{m}\frac{|Y_i\cap R|}{|Y_i|} - \frac{|R\cap X_i|}{|X_i|}\right| \\ &\leq \frac{1}{m}\sum_{i=1}^{m}\left|\frac{|Y_i\cap R|}{|Y_i|} - \frac{|R\cap X_i|}{|X_i|}\right|.\end{aligned}$$

Moreover, $R \cap X_i$ is a range in $\mathcal{R}|_{X_i}$. Therefore, for $i = 1, 2, \ldots, m$,

$$\left|\frac{|Y_i\cap R|}{|Y_i|} - \frac{|R\cap X_i|}{|X_i|}\right| \leq \delta\frac{|R\cap X_i|}{|X_i|} + \epsilon.$$

Thus,

$$\begin{aligned}\left|\frac{|Y\cap R|}{|Y|} - \frac{|R|}{|X|}\right| &\leq \frac{1}{m}\sum_{i=1}^{m}\left(\delta\frac{|R\cap X_i|}{|X_i|} + \epsilon\right) \\ &= \delta\frac{|R|}{|X|} + \epsilon,\end{aligned}$$

which establishes the lemma. ■

LEMMA 4.2. *If Y is a δ_1-relative ϵ_1-approximation for $(X, \mathcal{R})$ and Z is a δ_2-relative ϵ_2-approximation for $(Y, \mathcal{R}|_Y)$, then Z is a $(\delta_1 + \delta_2 + \delta_1\delta_2)$-relative $(\epsilon_1(1 + \delta_2) + \epsilon_2)$-approximation for $(X, \mathcal{R})$.*

Proof. Let R be a range in $\mathcal{R}$. We can write

$$\begin{aligned}\left|\frac{|Z\cap R|}{|Z|} - \frac{|R|}{|X|}\right| &\leq \left|\frac{|Z\cap R|}{|Z|} - \frac{|Y\cap R|}{|Y|}\right| + \left|\frac{|Y\cap R|}{|Y|} - \frac{|R|}{|X|}\right| \\ &\leq \delta_1\frac{|R|}{|X|} + \epsilon_1 + \delta_2\frac{|Y\cap R|}{|Y|} + \epsilon_2 \\ &\leq \delta_1\frac{|R|}{|X|} + \epsilon_1 + \delta_2\left((1+\delta_1)\frac{|R|}{|X|} + \epsilon_1\right) + \epsilon_2 \\ &= (\delta_1 + \delta_2 + \delta_1\delta_2)\frac{|R|}{|X|} + \epsilon_1(1+\delta_2) + \epsilon_2,\end{aligned}$$

which establishes the lemma. ■

In addition, we will make use of the following observation.

FACT 4.1. *If Y is a δ-relative ϵ_1-approximation for $(X, \mathcal{R})$ and Z is an ϵ_2-net $(Y, \mathcal{R}|_Y)$, then Z is a $(\epsilon_1 + \epsilon_2)/(1 - \delta)$-net for $(X, \mathcal{R})$.*

Given a range space $(X, \mathcal{R})$ with bounded VC-exponent, and a parameter $1 \leq r \leq n$, we wish to apply these lemmas to an efficient divide-and-conquer method for constructing a δ_0-relative $(1/r)$-approximation Y of $(X, \mathcal{R})$ of size $O(r^{2+\alpha})$ using only $O(nr^{O(1)})$ work, for any small constants $\delta_0 > 0$ and $\alpha > 0$, where $n = |X|$. We achieve this by designing an algorithm, **Approx**, which is a modification of earlier simple divide-and-conquer methods of Matoušek [44] and Goodrich [30].

We define **Approx** in terms of potential functions, $\delta(n)$ and $\epsilon(n)$, that dictate the relative error and absolute error of the approximation that we return. Specif-

ically, given any fixed constant $\delta_0 \leq 1/4$, **Approx** produces a $\delta(n)$-relative $\epsilon(n)$-approximation, Y, of $(X, \mathcal{R})$, where

$$\delta(n) \leq \delta_0 - \frac{1}{\log\log n} \tag{4.1}$$

and

$$\epsilon(n) \leq \frac{1}{2}\left(\frac{\log n - 1}{\log n}\right)\left(1 + \frac{\log\log n - 1}{\log\log n}\right)\frac{1}{r}. \tag{4.2}$$

This is, of course, a slightly stronger approximation than a δ_0-relative $(1/r)$-approximation would be, but this formulation will prove easier to work with in our recursive algorithm.

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

1. If $n \leq r^2$, then return X.
2. Otherwise, divide X into m equal-sized subsets $X_1, X_2, \ldots, X_m$ and call **Approx**$(r', (X_i, \mathcal{R}|_{X_i}))$ recursively for each i in parallel, where $r' = r$ and $m = n^\gamma$ with $0 < \gamma < 1$ being a constant to be set in the analysis. (Note: if $\log\log n^{1-\gamma} \leq 1/\delta_0$, then we do not recurse, but simply return X, so as to preserve the invariant of Equation (4.1).)
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 3.1 to find a $\delta'(n)$-relative $\epsilon'(n)$-approximation Y of $(Y', \mathcal{R}|_{Y'})$, where

$$\delta'(n) = \frac{-\log(1-\gamma)}{2(\log\log n^{1-\gamma})\log\log n}$$

and

$$\epsilon'(n) = \left(\frac{\gamma}{2\log n^{1-\gamma}}\right)\frac{1}{r}.$$

4. Return Y.

LEMMA 4.3. **Approx** *produces a $\delta(n)$-relative $\epsilon(n)$-approximation Y of X of size $O((r\log n)^{2+\alpha})$, for any fixed constant $\alpha > 0$. The work bound is $O(nr^c)$, for some constant $c \geq 1$, and the running time is $O(\log\log n)$ in the CRCW PRAM model.*

Proof. Our proof is an inductive argument based upon Lemmas 4.1 and 4.2. The number of levels in the recursion is clearly $O(\log\log n)$, so the time bound for a CRCW PRAM implementation is $O(\log\log n)$. By Theorem 3.1, the size of the approximation produced can be made to be $O(r^{2+\alpha})$ for any fixed constant $\alpha > 0$. The work complexity, $W(r, n)$, is therefore bounded by the recurrence equation

$$n^\gamma W(r, n^{1-\gamma}) + O(n^\gamma(r\log n^{1-\gamma})^{2+\alpha} + (r\log n)^{c(2+\alpha)}),$$

where c is the constant in the work bound of Theorem 3.1. If we choose γ to be a constant strictly less than $1/c$, then $W(r, n)$ will be $O(nr^{2c+1})$. There are clearly $O(\log\log n)$ levels in this recursive algorithm, each of which can be implemented in $O(1)$ time by Theorem 3.1; hence, the total running time is $O(\log\log n)$. ■

This lemma can in turn be used to derive work-efficient methods for constructing approximating subsets, as the following theorem shows:

THEOREM 4.1. *Let $(X, \mathcal{R})$ be a range space with bounded VC-exponent, e. Also, let α be any positive constant strictly less than the reciprocal of the constant in Theorem 3.1, Then, for some constant $c > 0$, one can produce the following sets in the bounds claimed in the CRCW PRAM:*

1. *a $(1/4)$-relative $(1/r)$-approximation A of $(X, \mathcal{R})$ of size $O(r^{2+\alpha})$ in $O(\log\log n)$ time using $O(nr^c)$ work,*
2. *a $(1/4)$-relative $(1/r)$-approximation B of $(X, \mathcal{R})$ of size $O(r^2\log r)$ in $O(\log\log n)$ time using $O(nr^{c\log r})$ work,*
3. *a $(1/r)$-net of $(X, \mathcal{R})$ C of size $O(r^{1+\alpha})$ in $O(\log\log n)$ time, using $O(nr^c)$ work,*
4. *a $(1/r)$-net of $(X, \mathcal{R})$ D of size $O(r\log r)$ in $O(\log\log n)$ time using $O(nr^{c\log r})$ work.*

Proof. The result for A follows by using the algorithm **Approx** to find a $(1/4 - 1/\log\log n)$-relative $(1/2r)$-approximation of size $O((r\log n)^{2+\epsilon})$ and follow that by an application of Theorem 3.1.1 to find a $(\log\log n)^{-1}$-relative $(1/4r)$-approximation of that. The set B is constructed similarly, using Theorem 3.1.2. The sets C and D are constructed by using **Approx** to find a $(1/4)$-relative $(1/4r)$-approximation of size $O(r^{2+\epsilon})$ and then applying Theorem 3.1.3 or 3.1.4 to find a $(1/4r)$-net of that, which, by Fact 4.1 will be a $(1/r)$-net of $(X, \mathcal{R})$. ■

Thus, we have established the needed result to complete the proof of Theorem 2.1 for the CRCW PRAM model. For analogous results for the EREW PRAM model, we may use the following theorem:

THEOREM 4.2. *Let $(X, \mathcal{R})$ be a range space with bounded VC-exponent, e. Also, let α be any positive constant strictly less than the reciprocal of the constant in Theorem 3.1, Then, for some constant $c > 0$, one can produce the following sets in the bounds claimed in the EREW PRAM:*

1. *a $(1/r)$-approximation A of $(X, \mathcal{R})$ of size $O(r^{2+\alpha})$ in $O(\log n)$ time using $O(nr^c)$ work,*
2. *a $(1/r)$-approximation B of $(X, \mathcal{R})$ of size $O(r^2\log r)$ in $O(\log n)$ time using $O(nr^{c\log r})$ work,*

3. *a* $(1/r)$*-net of* $(X, \mathcal{R})$ C *of size* $O(r^{1+\alpha})$ *in* $O(\log n)$ *time, using* $O(nr^c)$ *work,*

4. *a* $(1/r)$*-net of* $(X, \mathcal{R})$ D *of size* $O(r \log r)$ *in* $O(\log n)$ *time using* $O(nr^{c \log r})$ *work.*

Proof. The method is similar to that used to derive the CRCW PRAM bounds, expect that in this case we use Theorem 4.2 (in Step 3) and define **Approx** to produce a (0-relative) $\epsilon(n)$-approximation where

$$\epsilon(n) = \left(\frac{\log n - 1}{\log n}\right)\frac{1}{r},$$

by defining

$$\epsilon'(n) = \left(\frac{\gamma}{\log n^{1-\gamma}}\right)\frac{1}{r}.$$

The time bound for such an EREW PRAM implementation can be characterized by the recurrence $T(r, n) \leq T(r, n^{1-\gamma}) + O(\log n)$, which is $O(\log n)$. ■

Acknowledgements. We would like to thank Bonnie Berger, Jiří Matoušek, and John Rompel for several helpful comments concerning the topics of this paper. We would also like to especially thank Edgar Ramos for several comments that ultimately led to improvements to our constructions of $(1/r)$-nets and $(1/r)$-approximations.

References

[1] P. K. Agarwal. Geometric partitioning and its applications. In J. E. Goodman, R. Pollack, and W. Steiger, editors, *Computational Geometry: Papers from the DIMACS special year*. Amer. Math. Soc., 1991.

[2] M. Ajtai and N. Megiddo. A deterministic *poly*(log log n)-time n-processor algorithm for linear programming in fixed dimension. In *Proc. 24th Annu. ACM Sympos. Theory Comput.*, pages 327–338, 1992.

[3] N. Alon and N. Megiddo. Parallel linear programming in fixed dimension almost surely in constant time. In *Proc. 31st Annu. IEEE Sympos. Found. Comput. Sci.*, pages 574–582, 1990.

[4] N. Alon and J. Spencer. *The probabilistic method*. J. Wiley & Sons, 1993.

[5] N. M. Amato, M. T. Goodrich, and E. A. Ramos. Parallel algorithms for higher-dimensional convex hulls. In *Proc. 35th Annu. IEEE Sympos. Found. Comput. Sci.*, pages 683–694, 1994.

[6] N. M. Amato, M. T. Goodrich, and E. A. Ramos. Computing faces in segment and simplex arrangements. In *Proc. 27th Annu. ACM Sympos. Theory Comput.*, pages 672–682, 1995.

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

[8] P. Beame and J. Hastad. Optimal bounds for decision problems on the CRCW PRAM. *Journal of the ACM*, 36(3):643–670, 1989.

[9] K. P. Bennett. Decision tree construction via linear programming. In *Proceedings of the 4th Midwest Artificial Intelligence and Cognitive Science Society Conference*, pages 97–101, 1992.

[10] K.P. Bennett and O.L. Mangasarian. Multicategory discrimination via linear programming. *Optimization Methods and Software*, 3:29–39, 1994.

[11] B. Berger, J. Rompel, and P. W. Shor. Efficient NC algorithms for set cover with applications to learning and geometry. In *Proc. 30th Annu. IEEE Sympos. Found. Comput. Sci.*, volume 30, pages 54–59, 1989.

[12] H. Brönnimann and M. T. Goodrich. Almost optimal set covers in finite vc-dimension. In *Proc. 10th Annu. ACM Sympos. Comput. Geom.*, pages 293–302, 1994.

[13] 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.

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

[15] B. Chazelle and J. Matoušek. On linear-time deterministic algorithms for optimization problems in fixed dimension. In *Proc. 4th ACM-SIAM Sympos. Discrete Algorithms*, pages 281–290, 1993.

[16] H. Chernoff. A measure of asymptotic efficiency for tests of a hypothesis based on the sum of the observations. *Annals of Math. Stat.*, 23:493–509, 1952.

[17] V. Chvátal. *Linear Programming*. W. H. Freeman, New York, NY, 1983.

[18] K. L. Clarkson. Linear programming in $O(n3^{d^2})$ time. *Inform. Process. Lett.*, 22:21–24, 1986.

[19] K. L. Clarkson. New applications of random sampling in computational geometry. *Discrete Comput. Geom.*, 2:195–222, 1987.

[20] K. L. Clarkson. A Las Vegas algorithm for linear programming when the dimension is small. In *Proc. 29th Annu. IEEE Sympos. Found. Comput. Sci.*, pages 452–456, 1988.

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

[22] G. B. Dantzig. *Linear Programming and Extensions*. Princeton University Press, Princeton, NJ, 1963.

[23] X. Deng. An optimal parallel algorithm for linear programming in the plane. *Inform. Process. Lett.*, 35:213–217, 1990.

[24] M. Dyer. A parallel algorithm for linear programming in fixed dimension. In *Proc. 11th ACM Symp. on Computational Geometry*, 1995.

[25] M. E. Dyer. Linear time algorithms for two- and three-variable linear programs. *SIAM J. Comput.*, 13:31–45, 1984.

[26] M. E. Dyer. On a multidimensional search technique and its application to the Euclidean one-center problem. *SIAM J. Comput.*, 15:725–738, 1986.

[27] H. Edelsbrunner. *Algorithms in Combinatorial Geometry*, volume 10 of *EATCS Monographs on Theoretical*

Computer Science. Springer-Verlag, Heidelberg, West Germany, 1987.

[28] D. Eppstein. Dynamic three-dimensional linear programming. *ORSA J. Comput.*, 4:360–368, 1992.

[29] T. Goldberg and U. Zwick. Optimal deterministic approximate parallel prefix sums and their applications. In *Proc. 4th IEEE Israel Symp. on Theory of Computing and Systems*, pages 220–228, 1995.

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

[31] M. T. Goodrich, Y. Matias, and U. Vishkin. Optimal parallel approximation for prefix sums and integer sorting. In *Proc. 5th ACM-SIAM Symp. on Discrete Algorithms*, pages 241–250, 1994.

[32] T. Hagerup. Fast deterministic processor allocation. In *4th ACM-SIAM Symposium on Discrete Algorithms*, pages 1–10, 1993.

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

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

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

[36] G. Kalai. A subexponential randomized simplex algorithm. In *Proc. 24th Annu. ACM Sympos. Theory Comput.*, pages 475–482, 1992.

[37] H. Karloff. *Linear Programming*. Birkhauser, Boston, 1991.

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

[39] R. M. Karp and V. Ramachandran. Parallel algorithms for shared memory machines. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, pages 869–941. Elsevier/The MIT Press, Amsterdam, 1990.

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

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

[42] 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.

[43] 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.*

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

[45] J. Matoušek. Epsilon-nets and computational geometry. In J. Pach, editor, *New Trends in Discrete and Computational Geometry*, volume 10 of *Algorithms and Combinatorics*, pages 69–89. Springer-Verlag, 1993.

[46] J. Matoušek, M. Sharir, and E. Welzl. A subexponential bound for linear programming. In *Proc. 8th Annu. ACM Sympos. Comput. Geom.*, pages 1–8, 1992.

[47] J. Matoušek, E. Welzl, and L. Wernisch. Discrepancy and ε-approximations for bounded VC-dimension. *Combinatorica*, 13:455–466, 1993.

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

[49] 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.

[50] N. Megiddo. Linear-time algorithms for linear programming in R^3 and related problems. *SIAM J. Comput.*, 12:759–776, 1983.

[51] N. Megiddo. Linear programming in linear time when the dimension is fixed. *J. ACM*, 31:114–127, 1984.

[52] R. Motwani, J. Naor, and M. Naor. The probabilistic method yields deterministic parallel algorithms. In *Proc. 30th Annu. IEEE Sympos. Found. Comput. Sci.*, pages 8–13, 1989.

[53] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, New York, 1995.

[54] C. H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Prentice Hall, Englewood Cliffs, NJ, 1982.

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

[56] J. H. Reif. *Synthesis of Parallel Algorithms*. Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1993.

[57] J. T. Rompel. *Techniques for Computing with Low-Independence Randomness*. Ph.D. thesis, Dept. of EECS, M.I.T., 1990.

[58] N. Sauer.. On the density of families of sets. *Journal of Combinatorial Theory*, 13:145–147, 1972.

[59] J. P. Schmidt, A. Siegel, and A. Srinivasan. Chernoff-Hoeffding bounds for applications with limited independence. In *Proc. 4th ACM-SIAM Symp. on Discrete Algorithms*, pages 331–340, 1993.

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

[61] S. Sen. A deterministic poly($\log \log n$) time optimal CRCW PRAM algorithm for linear programming in fixed dimension. Technical Report 95-08, Dept. of Computer Science, University of Newcastle, 1995.

[62] L. Valiant. Parallelism in comparison problems. *SIAM J. Comput.*, 4(3):348–355, 1975.

[63] 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.