Discrete Comput Geom 18:397–420 (1997)

Discrete & Computational
Geometry

Bounded-Independence Derandomization of Geometric Partitioning with Applications to Parallel Fixed-Dimensional Linear Programming*

M. T. Goodrich[1] and E. A. Ramos[2]

[1]Center for Geometric Computing, Johns Hopkins University,
Baltimore, MD 21218, USA
goodrich@cs.jhu.edu

[2]DIMACS, Rutgers University,
Piscataway, NJ 08855-1179, USA
ramose@dimacs.rutgers.edu

Abstract. We give fast and efficient methods for constructing ε-nets and ε-approximations for range spaces with bounded VC-exponent. These combinatorial structures have wide applicability to geometric partitioning problems, which are often used in divide-and-conquer constructions in computational geometry algorithms. In addition, we introduce a new deterministic set approximation for range spaces with bounded VC-exponent, which we call the *δ-relative ε-approximation*, and we show how such approximations can be efficiently constructed in parallel. To demonstrate the utility of these constructions we show how they can be used to solve the linear programming problem in $\mathbb{R}^d$ 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.

* This research was announced in preliminary form in *Proc. 9th ACM Symposium on Computational Geometry* (*SCG*), 1993, pp. 73–82, and in *Proc. 7th ACM–SIAM Symposium on Discrete Algorithms* (*SODA*), 1996, pp. 132–141. The research of M. T. Goodrich was supported by the National Science Foundation under Grants IRI-9116843, CCR-9300079, and CCR-9625289, and by ARO under Grant DAAH04-96-1-0013. The research of E. A. Ramos was supported by a DIMACS Postdoctoral Fellowship. DIMACS is a cooperative project of Rutgers University, Princeton University, AT&T Research, Bell Labs, and Bellcore. DIMACS is an NSF Science and Technology Center, funded under Contract STC-91-19999; and also receives support from the New Jersey Commission on Science and Technology. The current address of E. A. Ramos is Max-Planck-Institut für Informatik, Im Stadtwald, 66123 Saarbrücken, Germany. ramos@mpi-sb.mpg.de.

1. Introduction

The study of randomized algorithms and methods for reducing the amount of perfect randomness needed for geometric algorithms has proven to be a very rich area of research (e.g., see [1], [2], [4], [5], [13], [15], [22], [42], [57], and [58]). Indeed, randomized geometric algorithms are typically simpler and more efficient than their deterministic counterparts and studying the limitation of the randomness needed by such algorithms often yields insights into the specific properties of randomization that are needed to achieve this simplicity and efficiency.

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* [51], [68] and the *ε-net* [36], [51]. 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 $R \in \mathcal{R}$,

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

Relaxing this requirement a bit, Y is said to be an *ε-net* [36], [51] of $(X, \mathcal{R})$ if $Y \cap R \neq \emptyset$ for each $R \in \mathcal{R}$ such that $|R| > \varepsilon |X|$. 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 nonnegative parameters $\delta < 1$ and $\varepsilon < 1$, a subset Y is a *δ-relative ε-approximation* if, for each range $R \in \mathcal{R}$,

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

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.[1] Note that this notion also subsumes that of an ε-net, for any δ-relative ε-approximation is automatically an $(\varepsilon/(1-\delta))$-net.

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 [6], [7], but these are beyond the scope of this paper.

---

[1] Brönnimann *et al.* [13] call a subset $A \subseteq X$ a *sensitive ε-approximation* if $||A \cap R|/|A| - |R|/|X|| \leq (\varepsilon/2)(\varepsilon + \sqrt{|R|/|X|})$.

1.1. *Previous Work on Derandomizing Geometric Algorithms*

Before we describe our results, however, we 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 [20], Haussler and Welzl [36], and Clarkson and Shor [22]. 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 [2] and [51]). 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. Random sampling can be applied to construct such a partitioning so that each cell intersects at most εn hyperplanes, for $\varepsilon = \log r/r$ [22], [36]. Chazelle and Friedman [15] show that in fact such a partitioning with $\varepsilon = 1/r$ can be constructed deterministically in polynomial time, and Berger *et al.* [12] and Motwani *et al.* [56] show that similar geometric partitions can be constructed for $\varepsilon = \log r/r$ in NC. (Recall that NC denotes the class of problems solvable in polylogarithmic time using a polynomial number of processors [37], [43].) 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).

A general framework for geometric partitioning emerges from the framework when a range space $(X, \mathcal{R})$ has constant Vapnik–Chervonenkis [68] (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 [51]). 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* [8], [14] *bounded by* e if $\pi_{\mathcal{R}}(m)$ is $O(m^e)$. For example, if $(X, \mathcal{R})$ is 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, then $(X, \mathcal{R})$ has VC-exponent 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 bounded by e as well [63], [68]. 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 VC-exponent bounded by e in time $O(nr^c)$ for some constant c depending on e (e.g., see [13], [16], [47–[49], and [53]). Chazelle and Matoušek [16] give slower NC algorithms using $O(nr^c)$ work[2] that construct such sets of size $O(r^{2+\alpha})$ for any fixed constant $\alpha > 0$.

[2] Recall that the *work* done by a parallel algorithm is the total number of operations performed by all processors, and it is never more than the product of the running time and the number of processors needed to achieve that running time.

1.2. *Our Results on Parallel Geometric Derandomization*

We give fast and efficient parallel algorithms for constructing ε-nets and δ-relative ε-approximations. For example, our methods can be implemented in the CRCW PRAM model[3] to run in $O(\log\log n)$ time using $O(nr^c)$ work to produce δ-relative $(1/r)$-approximations of size $O(r^{2+\alpha})$ for any fixed constants $\delta > 0$ and $\alpha > 0$, 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 [32] while also improving the time bounds from those achieved previously by Chazelle and Matoušek [16]. We also derive similar bounds for constructing $(1/r)$-nets. To demonstrate the utility of this result, we show how it can be used to design a new efficient parallel method for fixed-dimensional linear programming.

1.3. *Fixed-Dimensional Linear Programming*

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 1940s (e.g., see [18], [23], [40], and [59]). 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 half-spaces in $\mathbb{R}^d$. Of particular interest is 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 [16], [21], [29], [54], [55], and [60]) and machine learning (e.g., see [10] and [11]). 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 [27] and Megiddo [54], [55], 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 [16], [19], [21], [28], [39], [52], and [65]) or on building data structures for linear programming queries (e.g., see [30] and [50]).

In the parallel domain, Alon and Megiddo [4] give analogous results, showing that through the use of randomization a fixed-dimensional linear program can be solved in $O(1)$ time with very high probability using n processors in a randomized CRCW PRAM model. The existing deterministic parallel algorithms are not as efficient, however. Ajtai and Megiddo [3] give a deterministic $O((\log\log n)^d)$-time method, but it has a suboptimal $\Theta(n(\log\log n)^d)$ work bound and it is defined for the very power-

[3] 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 common 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.

ful parallel model that only counts "comparison" steps [67]. The only work-optimal deterministic PRAM result we are familiar with is a method by Deng [24] for two-dimensional linear programming that runs in $O(\log n)$ time using $O(n)$ work on a CRCW PRAM. Recently, Dyer [26] 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 [66] has independently discovered a CRCW PRAM method that runs in $O((\log\log n)^{d+1})$ time using $O(n)$ work.

1.4. *Our Results for Parallel Linear Programming*

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 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 using our parallel procedure for ε-net construction.

The remainder of this paper is structured as follows. In the next section we review some of the probabilistic background we will be using in subsequent sections. Since a work-efficient parallel algorithm immediately implies an efficient sequential method, we describe all of our procedures as parallel algorithms. We begin this discussion in Section 3, where we give fast, but work-inefficient, parallel methods. In Section 4 we describe how to apply a divide-and-conquer strategy to make these methods work-efficient. We give applications of these methods to fixed-dimensional linear programming in Section 5, and we conclude in Section 6.

2. Probabilistic Preliminaries

Our approach to constructing small-sized $(1/r)$-nets and $(1/r)$-approximations of range spaces with bounded VC-exponent is to derandomize a straightforward probabilistic algorithm that is based upon the *random sampling* technique [20]. We perform this derandomization using the *bounded independence* derandomization technique [5], [41], [44], [45], [64], which assumes our algorithm uses random variables that are only k-wise independent. Thus, before we give our methods, we review these concepts (see also [5] and [57]).

2.1. *Random Sampling*

Since the probabilistic algorithm we wish to derandomize is based upon random sampling, we begin by saying a few words about this technique. 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 random subset $Y \subset X$ of size s. Sequentially, this is quite easy to do. 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 $X_i = 1$ [12]. Note that a set of $|Y| = X_1 + X_2 + \cdots + X_n$ unique elements is guaranteed, but its size may not be equal to s, although it is easy to see, by the linearity of expectation, that $E(|Y|) = s$.

2.2. *k-Wise Independence*

In order to apply the bounded-independence derandomization 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 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 $p = s/n$. Note that in this notation $X^{(n)} = X$; hence, we may omit the superscript if the underlying random variables are mutually independent.

Unfortunately, restricting our attention to k-wise independent indicator random variables prevents us from directly using the well-known and powerful Chernoff bounds [5], [17], [35], [57] for bounding the tail of the distribution of their sum. Nevertheless, as shown by Rompel [62] (see also [64]), we may derive something analogous:

Lemma 2.1 [62]. *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 $\lambda > 0$.

2.3. *Derandomization via Bounded Independence*

We are now ready to review the *bounded independence* technique for derandomizing a probabilistic algorithm [5], [41], [44], [45]. We use the parallel formulation of Luby [44], which is based upon a combinatorial construction of Joffe [38] (see also [41]). In this formulation, we assume we have a parallel 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.

2. Each random variable X_i takes on values $\{x_1, x_2, \ldots, x_m\}$, where m is bounded by a polynomial[4] in n.
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 $\sum_{j=1}^{m} n_{i,j} = q$). Of course, such a prime number can be easily found in $O(1)$ time in the CRCW PRAM model using a polynomial number of processors.

Luby [44] shows that if **Random** satisfies all of these conditions, then a space of q^k points may be constructed 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.[5] 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 succeeds (in fact, a constant fraction succeed). The output is given by one of these successful calls (where one breaks ties arbitrarily). The benefit of using this approach is that it is very simple, and, although the processor costs may be high, the speed of the algorithm is the same as that used in **Random** (plus an additional term for performing an "**or**" on all the results in parallel, which can be done in $O(1)$ time in the CRCW PRAM model and $O(\log n)$ time in the EREW PRAM model [37], [43], [61]).

Having reviewed the necessary probabilistic preliminaries, we now turn to the problem of constructing $(1/r)$-approximations and $(1/r)$-nets.

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

Before we describe our 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. This 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 [20].

3.1. *Geometric Random Samples*

Let $(X, \mathcal{R})$ be a given range space with VC-exponent bounded by e, for some constant $e > 0$. Given a parameter $2 \leq r \leq |X|$, 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. We 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 $\Theta(s)$.

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

[5] Recently, Dietzfelbinger [25] has given an alternative construction that does not make use of the availability of a prime q.

Lemma 3.1. *Let Y be defined as above, with $k \geq 2$ even. Then*

$$||Y| - s| < \max\{\beta c, 1\}(sk + k^2)^{1/2},$$

with probability at least $1 - 1/\beta$, for some constant $c > 0$. In particular, if $s \geq C(\beta)k$, for some constant $C(\beta) > 0$, then $||Y| - s| = \Theta(k^{1/2}s^{1/2})$ and also $||Y| - s| \leq s/2$ with probability at least $1 - 1/\beta$.

Proof. Y is an expected s-sample of X determined by n indicator random variables. Since $|Y|$ has mean $\mu_{|Y|} = s$, we may apply Lemma 2.1 to bound the probability that Y does not satisfy the above size condition as

$$\Pr(||Y| - s| \geq (\beta c)^{1/k}(sk + k^2)^{1/2}) \leq 1/\beta,$$

where c is as in the lemma. The bounds claimed follow from this one. □

We therefore bound the probability that Y is a $(1/r)$-net or a $(1/r)$-approximation. In particular, let $\mathcal{S}$ be a subset of $\mathcal{R}$, and let $\mathcal{A}_Y(r, \mathcal{S})$ denote the number of ranges $R \in \mathcal{S}$ that Y does not $(1/r)$-*approximate* (i.e., the number of ranges $R \in \mathcal{S}$ such that $||Y \cap R|/|Y| - |R|/|X|| > 1/r$), 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 3.2. *Let $(X, \mathcal{R})$ be a range space. Given a parameter $C \leq r \leq |X|$, for some $C > 0$, a positive even integer $k \leq n$, and a parameter $s \geq rk$, 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. *$s - \Theta(k^{1/2}s^{1/2}) \leq |Y| \leq s + \Theta(k^{1/2}s^{1/2})$ and in particular $s/2 \leq |Y| \leq 3s/2$.*
2. *$\mathcal{A}_Y(r, \mathcal{S}) \leq c4^k(ks + k^2)^{k/2}r^k|\mathcal{S}|/s^k$ for some constant $c > 0$.*
3. *$\mathcal{N}_Y(r, \mathcal{S}) \leq c(2k)^{k/2}r^{k/2}|\mathcal{S}|/s^{k/2}$ for some constant $c > 0$.*

Proof. Our proof is to show that properties 2 and 3 hold with probability at least $5/6$ each, given property 1, which also holds with probability at least $5/6$. We can choose $\beta = 6$ from Lemma 3.1 so that property 1 holds with probability $5/6$. So, we assume that $|Y|$ is $s \pm s/2$, and consider the quantity $\mathcal{A}_Y(r, \mathcal{S})$. 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

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

We 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 2.1 to bound this probability by

$$c\left(\frac{2^k[k(s|R|/n) + k^2]^{k/2} r^k}{|Y|^k}\right) \leq c(4^k(ks + k^2)^{k/2} r^k)/s^k$$

for some constant c, since $s \geq kr > 1$ and $|R| \leq n$. Therefore,

$$E(\mathcal{A}_Y(r, \mathcal{S})) \leq c(4^k(ks + k^2)^{k/2} r^k)|\mathcal{S}|/s^k.$$

We may then apply Markov's inequality (which has no independence assumptions) to show

$$\Pr(\mathcal{A}_Y(r, \mathcal{S}) > 6c(4^k(ks + k^2)^{k/2} r^k)|\mathcal{S}|/s^k) \leq 1/6.$$

The bound for $\mathcal{N}_Y(r, \mathcal{S})$ is proved similarly, but we give the details here for completeness. We can write

$$\mathcal{N}_Y(r, \mathcal{S}) = \sum_{R \in \mathcal{S} \& |R| > |X|/r} Z_R,$$

where Z_R is an indicator random variable for "$Y \cap R = \emptyset$ but $|R| \geq |X|/r$." By the linearity of expectation,

$$\begin{aligned} E(\mathcal{N}_Y(r, \mathcal{S})) &= \sum_{R \in \mathcal{S} \& |R| > n/r} E(Z_R) \\ &\leq \sum_{R \in \mathcal{S} \& |R| > n/r} \Pr(|\,|Y \cap R| - (s/n)|R|\,| \geq (s/n)|R|), \end{aligned}$$

where $n = |X|$. Note that $|Y \cap R| = \sum_{i \in R} X_i$. Thus, we may apply Lemma 2.1 to derive

$$\begin{aligned} \Pr(|\,|Y \cap R| - (s/n)|R|\,| \geq (s/n)|R|) &\leq c\left(\frac{k(s/n)|R| + k^2}{(s/n)^2|R|^2}\right)^{k/2} \\ &\leq c(2k)^{k/2} r^{k/2}/s^{k/2}, \end{aligned}$$

for some constant $c > 0$, since $|R| > n/r$ and $s \geq rk$. Therefore,

$$E(\mathcal{N}_Y(r, \mathcal{S})) \leq c(2k)^{k/2} r^{k/2}|\mathcal{S}|/s^{k/2},$$

and we may then apply Markov's inequality to show that

$$\mathcal{N}_Y(r, \mathcal{S}) \leq 6c(2k)^{k/2} r^{k/2}|\mathcal{S}|/s^{k/2}$$

with probability $5/6$. This completes the proof. □

3.2. *EREW PRAM Algorithms*

Given this lemma, we can apply the bounded-independence derandomization technique to derive deterministic $(1/r)$-net and $(1/r)$-approximation construction methods for range spaces with bounded VC-exponent. We assume $\mathcal{R}|_Y$ is computable in $O(1)$ time using work polynomial in $|Y|$ on a CRCW PRAM or in $O(\log n)$ time on a EREW PRAM. From the above lemma we can derive the following:

Theorem 3.3. *Let $(X, \mathcal{R})$ be a range space with VC-exponent bounded by e, for some constant $e > 0$, and let $n = |X|$. Also, let $2 \leq r < n$ be a given parameter, and let $k > 0$ be an even integer parameter. Then, in the EREW PRAM model, for some constant $c > 0$, the following can be constructed in the bounds claimed*:

1. *A $(1/r)$-approximation A of $(X, \mathcal{R})$ of size $\Theta(r^2 k n^{e/k})$ in $O((e+k)\log n)$ time using $O(2^k n^{e+k+1})$ work.*
2. *A $(1/r)$-net B of $(X, \mathcal{R})$ of size $\Theta(rkn^{e/k})$ in $O((e+k)\log n)$ time using $O(2^k n^{e+k+1})$ work.*

Proof. The methods for constructing these sets are straightforward applications of the bounded-independence derandomization technique using $\mathcal{S} = \mathcal{R}$ in Lemma 3.2. The main idea is to set the s parameter in Lemma 3.2 so that $\mathcal{N}_Y(r, S) < 1$ and $\mathcal{A}_Y(r, S) < 1$ (i.e., since they are integer values, $\mathcal{N}_Y(r, S) = 0$ and $\mathcal{A}_Y(r, S) = 0$), while $|Y|$ is $\Theta(s)$, with probability $1/2$, and then derandomize the implied construction by the bounded-independence derandomization technique. For example, each probability of the form s/n can be approximated by $\lceil sq/n \rceil / q$, and there is a simple, effective method for testing if a set satisfies the needed conditions to be a $(1/r)$-net or $(1/r)$-approximation in $O(\log n)$ time using a linear number of processors. Thus, since $|\mathcal{R}|$ is $O(n^e)$, and the probability space in the proof of Lemma 3.2 has size $q^k = O((2n)^k)$, then performing the $(1/r)$-net or $(1/r)$-approximation test for the set Y determined by each point in the probability space in parallel requires $O(2^k n^{e+k+1})$ processors. A constant fraction of these points are guaranteed to yield satisfactory results, so by taking one such successful test (arbitrarily) we can construct the desired set. Since all the test computations can be performed in $O(\log n)$ time and selecting a single successful outcome can be done in time $O(\log(2^k n^{e+k+1})) = O((e+k)\log n)$, the performance bounds of the theorem follow. □

This, in turn, implies the following:

Corollary 3.4. *Let $(X, \mathcal{R})$ be a range space with VC-exponent bounded by e, for some constant $e > 0$, and let $n = |X|$. Also, let $2 \leq r < n$ be a given parameter and let $\alpha > 0$ be any fixed (small) constant. Then, in the EREW PRAM model, for some constant $c > 0$, the following can be constructed in the bounds claimed*:

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

3. *A* $(1/r)$*-net* B *of* $(X, \mathcal{R})$ *of size* $\Theta(rn^{\alpha})$ *in* $O(\log n)$ *time using* $O(n^c)$ *work with* $c = e(1 + 1/\alpha) + 1$.
4. *A* $(1/r)$*-net* D *of* $(X, \mathcal{R})$ *of size* $\Theta(r \log n)$ *in* $O(\log^2 n)$ *time using* $O(n^{e(2+\log n)+1})$ *work*.

Proof. Simply apply Theorem 3.3. For A and B take $k = e/\alpha$. For C and D take $k = e \log n$. □

Actually, we can apply a simple "recursive refinement" technique to improve this to the following:

Theorem 3.5. *Let* $(X, \mathcal{R})$ *be a range space with VC-exponent bounded by* e*, for some constant* $e > 0$*, and let* $n = |X|$*. Also, let* $2 \leq r < n$ *be a given parameter and let* $\alpha > 0$ *be any fixed* (*small*) *constant. Then, in the EREW PRAM model, for some constant* $c > 0$*, the following can be constructed in the bounds claimed*:

1. *A* $(1/r)$*-approximation* A *of* $(X, \mathcal{R})$ *of size* $O(r^{2+\alpha})$ *in* $O(\log n)$ *time using* $O(n^c)$ *work with* $c = e(1 + (4 + 2\max\{2, \alpha\})/\alpha) + 1$.
2. *A* $(1/r)$*-approximation* C *of* $(X, \mathcal{R})$ *of size* $O(r^2 \log r)$ *in* $O(\log n + \log^2 r)$ *time using* $O(n^{9e+1} + r^{9e \log(cr)})$ *work.*
3. *A* $(1/r)$*-net* B *of* $(X, \mathcal{R})$ *of size* $O(r^{1+\alpha})$ *in* $O(\log n)$ *time using* $O(n^c)$ *work with* $c = e(1 + (4 + 2\max\{2, \alpha\})/\alpha) + 1$.
4. *A* $(1/r)$*-net* D *of* $(X, \mathcal{R})$ *of size* $O(r \log r)$ *in* $O(\log n + \log^2 r)$ *time using* $O(n^{9e+1} + r^{9e \log(cr)})$ *work.*

Proof. The structure of the proof is to apply the previous corollary to refine recursively our approximations to be of a size depending only on r, not n. The main idea of this approach is to take advantage of an observation of Matoušek [47] on an additive property of ε-approximations, which states that an ε-approximation of a δ-approximation of a set X is itself an $(\varepsilon + \delta)$-approximation of X. Thus, to construct the set A we proceed as follows: If $r \geq n^{1/8}$, then we construct A immediately using Corollary 3.4(1) to get a $1/r$-approximation of size $O(r^2 n^{\beta})$ where $\beta = 8\alpha$. This yields a set of size $O(r^{2+\alpha})$ in time $O(\log n)$, which for the sake of an inductive argument we characterize as being at most $b_0 \log n - b_1 \log r$, for constants $b_0 > b_1 \geq 1$. Otherwise, if $r < n^{1/8}$, then we recursively construct a $(1/r^2)$-approximation A' of $(X, \mathcal{R})$ of size at most $c_1(r^2)^{2+\alpha}$, for some constant c_1 (to be defined below). By induction, this recursive call takes time at most $b_0 \log n - b_1 \log(r^2)$. We then apply Corollary 3.4(1) to construct a $[(1/r) - (1/r^2)]$-approximation A of A' with size $c_0[r^2/(r-1)]^2 c_1 (r^2)^{(2+\alpha)\beta}$, for a constant $\beta = \alpha/(4+2\alpha) < 1/2$. By the additive property of ε-approximations, the set A will be a $(1/r)$-approximation of $(X, \mathcal{R})$. Moreover, if we choose $c_1 \geq (4c_0)^{1/(1-\beta)}$, then $|A| \leq c_1 r^{2+\alpha}$. This final call to the method of Corollary 3.4 takes time $O(\log |A|)$, which is at most $b_2 \log r$, for some constant $b_2 > 0$. Thus, the total time required is $b_0 \log n - b_1 \log(r^2) + b_2 \log r$, which is at most $b_0 \log n - b_1 \log r$, if $b_1 \geq b_2$. For the work, note that the computation is a sequence of applications of Corollary 3.4(1) on sets of size rapidly decreasing. At the bottom of the recursion (when the approximation size is largest),

Corollary 3.4(1) is used with $\beta = 8\alpha$, while at the other steps Corollary 3.4(1) is used with $\beta = \alpha/(4+2\alpha)$. Hence the work is $O(n^c)$ with $c = e(1+(4+2\max\{2,\alpha\})/\alpha)+1$.

The set C is constructed similarly, in that we first construct the set A as above to be a $(1/2r)$-approximation, with say $\alpha = 1$, and we then apply Corollary 3.4(2) to construct a $(1/2r)$-approximation of even smaller size (we leave the details to the reader). Likewise, for the sets B and D we first construct a $(1/2r)$-approximation and then find a $(1/2r)$-net of that, taking advantage of the additional property that an ε-net of a δ-approximation of a set X is an $(\varepsilon+\delta)$-net of X. □

Note that our methods for constructing A and B are in the complexity class NC for all values of r, but our methods for constructing C and D are in NC only for constant values of r.

3.3. *CRCW PRAM Algorithms*

Unfortunately, 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 [9]. 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 counting* [31], [33], where one wishes to compute the sum of an array of n bits with a relative error of λ. That is, if x is the number of 1's in the array, then we desire a value x' such that $x/(1+\lambda) \le x' \le (1+\lambda)x$.

Lemma 3.6 [31]. *Performing λ-approximate counting of an n-element Boolean array, with $\lambda = (\log N)^{-b}$, can be done in $O(1)$ time using $O((n+N)^{f(b)})$ work on a CRCW PRAM, for any fixed constant $b > 0$.*

We use this lemma to estimate the sizes $|Y \cap R|$, $|Y|$, and $|R|$, all of which involve computing the sum of $O(n)$ bits. We 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

$$\left|\frac{|Y\cap R|'}{|Y|'} - \frac{|R|'}{|X|}\right| \le \delta\,\frac{|R|'}{|X|} + \varepsilon.$$

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 $\lambda \le 1/4$.*

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

$$\left|\frac{|Y\cap R|'}{|Y|'} - \frac{|R|'}{|X|}\right| + \left|\frac{|Y\cap R|}{|Y|} - \frac{|Y\cap R|'}{|Y|'}\right| + \left|\frac{|R|'}{|X|} - \frac{|R|}{|X|}\right|$$

$$\leq \delta(1+\lambda)\frac{|R|}{|X|} + ((1+\lambda)^2 - 1)\frac{|Y \cap R|'}{|Y|'} + \lambda\frac{|R|}{|X|} + \varepsilon$$
$$= (\lambda + (1+\lambda)\delta)\frac{|R|}{|X|} + \lambda(2+\lambda)\frac{|Y \cap R|'}{|Y|'} + \varepsilon.$$

We also know that

$$\begin{aligned}\frac{|Y \cap R|'}{|Y|'} &\leq (1+\delta)\frac{|R|'}{|X|} + \varepsilon \\ &\leq (1+\lambda)(1+\delta)\frac{|R|}{|X|} + \varepsilon.\end{aligned}$$

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

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

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, observing that $|Y \cap R|'/|Y|' \leq (1+\lambda)^2|Y \cap R|/|Y|$, we can bound $|\,|Y \cap R|'/|Y|' - |R|'/|X|\,|$ by

$$\begin{aligned}&\left|\frac{|Y \cap R|}{|Y|} - \frac{|R|}{|X|}\right| + \left|\frac{|Y \cap R|'}{|Y|'} - \frac{|Y \cap R|}{|Y|}\right| + \left|\frac{|R|}{|X|} - \frac{|R|'}{|X|}\right| \\ &\quad \leq \varepsilon + \lambda(2+\lambda)\frac{|Y \cap R|}{|Y|} + \frac{\lambda|R|'}{|X|} \\ &\quad \leq \varepsilon + \lambda(2+\lambda)\left(\frac{|R|}{|X|} + \varepsilon\right) + \frac{\lambda|R|'}{|X|} \\ &\quad \leq \varepsilon + \lambda(2+\lambda)\left(\frac{(1+\lambda)|R|'}{|X|} + \varepsilon\right) + \frac{\lambda|R|'}{|X|} \\ &\quad \leq \varepsilon(1 + \lambda(2+\lambda)) + (\lambda(2+\lambda)(1+\lambda) + \lambda)\frac{|R|'}{|X|} \\ &\quad \leq 2\varepsilon + 4\lambda\frac{|R|'}{|X|},\end{aligned}$$

provided $\lambda \leq 1/4$. □

Say that Y is λ-estimated to be an ε-net if $Y \cap R \neq \emptyset$ for each R with $|R|' > \varepsilon|X|$. We make use of the following observation.

Lemma 3.9. *If Y is λ-estimated to be an ε-net, then Y is a $(1+\lambda)\varepsilon$-net. If Y is an ε-net, then Y will be λ-estimated to be a $(1+\lambda)\varepsilon$-net.*

These lemmas, together with previous results, imply the following:

Theorem 3.10. *Let $(X, \mathcal{R})$ be a range space with VC-exponent bounded by e, for some constant $e > 0$, and let $n = |X|$. Also, let $2 \leq r < n$ be a given parameter and let $\alpha > 0$ be any fixed (small) constant. Then, in the CRCW PRAM model, for some constant $c > 0$, any of the following can be constructed in the bounds claimed*:

1. *A $(\log N)^{-b}$-relative $(1/r)$-approximation A of $(X, \mathcal{R})$ of size $\Theta(r^2 n^\alpha)$ in $O(1)$ time using $O(n^c \cdot (n+N)^{f(b)})$ work with $c = e(1+1/\alpha)$.*
2. *A $(\log N)^{-b}$-relative $(1/r)$-approximation B of $(X, \mathcal{R})$ of size $\Theta(r^2 \log n)$ in $O(1)$ time using $O(n^{e(2+\log n)} \cdot (n+N)^{f(b)})$ work.*
3. *A $(1/r)$-net C of $(X, \mathcal{R})$ of size $\Theta(rn^\alpha)$ in $O(1)$ time using $O(n^c)$ work with $c = e(1+1/\alpha) + f(1)$.*
4. *A $(1/r)$-net D of $(X, \mathcal{R})$ of size $\Theta(r \log n)$ in $O(1)$ time using $O(n^{e(2+\log n)+f(1)})$ work.*

Proof. We begin with the set A. We can set the parameter $s = \Theta(r^2 n^\alpha)$ in Lemma 3.2 so that any expected s-sample Y is a $(1/4r)$-approximation with probability at least $1/2$. By Lemma 3.8, this implies that in applying the bounded independence derandomization technique there will be some Y that is λ-estimated to be a 4λ-relative $(1/2r)$-approximation. However, by Lemma 3.7, this in turn implies that Y is a (18λ)-relative $(1/r)$-approximation. By taking $\lambda = (\log N)^{-(b+1)}$, we therefore force such a Y to be a $(\log N)^{-b}$-relative $(1/r)$-approximation (for N larger than some constant). The rest of the construction, then, is a straightforward (CRCW PRAM) implementation of the bounded-independence derandomization technique following the argument of the proof of Theorem 3.3. For the set C, using Lemma 3.9, it suffices to use estimates within a constant factor (so N is a constant). The methods for constructing the other sets are similar applications of the bounded-independence technique. □

As in our EREW algorithms, we can apply a composition technique to improve the size bounds in the above constructions. Unlike our EREW methods, however, our CRCW PRAM size-efficient methods will not run quite as fast as the size-inefficient methods of Theorem 3.10. Our methods are based in part on the following additive property for δ-relative ε-approximations.

Lemma 3.11. *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 $(\varepsilon_1(1+\delta_2) + \varepsilon_2)$-approximation for $(X, \mathcal{R})$.*

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

$$\begin{aligned}\left|\frac{|Z\cap R|}{|Z|}-\frac{|R|}{|X|}\right| &\le \left|\frac{|Z\cap R|}{|Z|}-\frac{|Y\cap R|}{|Y|}\right|+\left|\frac{|Y\cap R|}{|Y|}-\frac{|R|}{|X|}\right| \\ &\le \delta_2\frac{|Y\cap R|}{|Y|}+\varepsilon_2+\delta_1\frac{|R|}{|X|}+\varepsilon_1 \\ &\le \delta_2\left((1+\delta_1)\frac{|R|}{|X|}+\varepsilon_1\right)+\varepsilon_2+\delta_1\frac{|R|}{|X|}+\varepsilon_1 \\ &= (\delta_1+\delta_2+\delta_1\delta_2)\frac{|R|}{|X|}+\varepsilon_1(1+\delta_2)+\varepsilon_2,\end{aligned}$$

which establishes the lemma. □

We also use the following observation:

Lemma 3.12. *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 an $(\varepsilon_1+\varepsilon_2)/(1-\delta)$-net for $(X,\mathcal{R})$.*

Our main CRCW PRAM result, then, is the following:

Theorem 3.13. *Let $(X,\mathcal{R})$ be a range space with VC-exponent bounded by e, for some constant $e>0$, and let $n=|X|$. Also, let $2\le r<n$ be a given parameter and let $\alpha>0$ be any fixed (small) constant. Then, in the CRCW PRAM model, for some constant $c>0$, any of the following can be constructed in the bounds claimed*:

1. *A $(\log N)^{-b}$-relative $(1/r)$-approximation A of $(X,\mathcal{R})$ of size $O(r^{2+\alpha})$ in $O(\log\log n)$ time using $O(n^c\cdot(n+N)^{f(b)})$ work with $c=e(1+(4+2\max\{2,\alpha\})/\alpha)$.*
2. *A $(\log N)^{-b}$-relative $(1/r)$-approximation C of $(X,\mathcal{R})$ of size $O(r^2\log r)$ in $O(\log\log n)$ time using $O(n^{ce}\cdot(n+N)^{f(b)}+r^{c\log r}\cdot(r+N)^{f(b)})$ work.*
3. *A $(1/r)$-net B of $(X,\mathcal{R})$ of size $O(r^{1+\alpha})$ in $O(\log\log n)$ time using $O(n^{ce/\alpha})$ work.*
4. *A $(1/r)$-net D of $(X,\mathcal{R})$ of size $O(r\log r)$ in $O(\log\log n)$ time using $O(n^{ce}+r^{c\log n})$ work.*

Proof. We address the construction of set A. We describe it as a recursive procedure. If $r\ge n^{1/8}$, then we apply Theorem 3.10 to construct a $(2\log\log n-\log\log r)(\log N)^{-(b+1)}$-relative $(1/r)$-approximation of size $\Theta(r^{2+\alpha})$ in $O(1)$ time using $O((n+N)^{f(b)})$ work. For the purposes of the recursion, we refer to the running time of this method as being $b_1\log\log n-b_2\log\log r$, for constants $b_1>b_2\ge 1$. If $r<n^{1/8}$, then we recursively construct a $(2\log\log n-\log\log(r^2))(\log N)^{-(b+1)}$-relative $(1/r^2)$-approximation A' of size at most $c_1(r^2)^{2+\alpha}$, for some constant $c_1\ge 1$ (which we set below). We inductively assume this takes time at most $b_1\log\log n-b_2\log\log(r^2)$. We then apply Theorem 3.10 to construct a $(\log N)^{-(b+2)}$-relative $[(1/r)-(1/r^2)(3/2)]$-approximation A of A' in $O(1)$ additional time using $O((n+N)^{f(b+2)})$ work. By Lemma 3.11 A is a $(2\log\log n-\log\log r)(\log N)^{-b}$-relative $(1/r)$-approximation of $(X,\mathcal{R})$. The size of A is at most $c_0[r^2/(r-3/2)]^2(c_1r^{4+2\alpha})^{\beta}]$, which is at most $c_1r^{2+\alpha}$, if we choose the constants $\beta\le\alpha/(4+2\alpha)$ and $c_1\ge(16c_0)^{1/(1-\beta)}$. Likewise, the total running time of constructing

A is $b_1 \log\log n - b_2 \log\log(r^2) + b_3$, for some constant $b_3 \geq 1$. This, of course, is $b_1 \log\log n - b_2 \log\log r$, if $b_2 \geq b_3$.

Our method for constructing C is first to construct A as a $(\log N)^{-(b+1)}$-relative $(1/2r)$-approximation and then construct a $(\log N)^{-(b+1)}$-relative $(1/3r)$-approximation of that. The sets B and D are constructed in a similar manner, in that we first find a $(1/5)$-relative $(2/5r)$-approximation and then form a $(2/5r)$-net of that, which will be a $(1/r)$-net for $(X, \mathcal{R})$ by Lemma 3.12 (we leave the details to the reader). □

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

As already mentioned, the methods of the previous section 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 VC-exponent bounded by e. We need another simple lemma, which is an adaptation of an observation made by Matoušek [47].

Lemma 4.1. *Suppose $Y_1, Y_2, \ldots, Y_m$ are δ-relative ε-approximations for disjoint range spaces $(X_1, \mathcal{R}|_{X_1}), (X_2, \mathcal{R}|_{X_2}), \ldots, (X_m, \mathcal{R}|_{X_m})$, 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 $i = 1, 2, \ldots, m$,

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

Thus,

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

which establishes the lemma. □

Given a range space $(X, \mathcal{R})$ with bounded VC-exponent, and a parameter $2 \leq r \leq n$, we wish to develop 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 almost achieves this goal, in that it has a good work bound, but does not quite achieve the size bound (the **Approx** procedure is a modification of earlier simple divide-and-conquer method of Matoušek [48]). We can then follow this by a call to Theorem 3.13 to improve the size bound, while keeping the work bound at $O(nr^{O(1)})$.

We define **Approx** in terms of potential functions, $\delta(n)$ and $\varepsilon(n)$, that dictate the relative error and absolute error of the approximation that we return. Specifically, given any fixed constant $\delta_0 \leq 1/4$, **Approx** produces a $\delta(n)$-relative $\varepsilon(n)$-approximation, Y, of $(X, \mathcal{R})$, where

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

and

$$\varepsilon(n) \leq \left(\frac{\log n - 1}{\log n}\right)\left(\frac{\mu \log n - 1}{\mu \log n}\right)\frac{1}{r}, \tag{2}$$

where μ is a constant strictly less than $1/2\,f(1)$, where the function f is as in Lemma 3.6. 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 $\mu \log n^{1-\gamma} \leq 1/\delta_0$, then we do not recurse, but simply return X, so as to preserve the invariant of (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 3.10 (not Theorem 3.13) to find a $\delta'(n)$-relative $\varepsilon'(n)$-approximation Y of $(Y', \mathcal{R}|_{Y'})$, where
$$\delta'(n) = \frac{\gamma}{2\mu(1-\gamma)\log n}$$
and
$$\varepsilon'(n) = \left(\frac{\gamma}{2(1-\gamma)\log n}\right)\frac{1}{r}.$$
4. Return Y.

Lemma 4.2. **Approx** *produces a $\delta(n)$-relative $\varepsilon(n)$-approximation Y of X of size $O(r^3 n^\gamma)$. The work bound can be made $O(nr^c)$, 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 3.11. In particular, we inductively assume, by Lemma 4.1, that Y' is a $\delta(n^{1-\gamma})$-relative $\varepsilon(n^{1-\gamma})$-approximation, where $\delta(n)$ and $\varepsilon(n)$ are defined as in (1) and (2). Moreover, we inductively assume $|Y'|$ is $O(n^\gamma r^3 n^{(1-\gamma)\gamma})$. By Lemma 3.11, Y will then be a $(\delta(n^{1-\gamma}) +$

$\delta'(n) + \delta(n^{1-\gamma})\delta'(n))$-relative $(\varepsilon(n^{1-\gamma})(1 + \delta(n^{1-\gamma})) + \varepsilon'(n))$-approximation. By our definition of $\delta'(n)$ we have

$$\begin{aligned}
\delta(n^{1-\gamma}) + \delta'(n) + \delta(n^{1-\gamma})\delta'(n) &\leq \delta(n^{1-\gamma}) + 2\delta'(n) \\
&\leq \delta_0 - \frac{1}{\mu \log n^{(1-\gamma)}\mu} + \frac{\gamma}{\mu(1-\gamma)\log n} \\
&= \delta_0 - \frac{1}{\mu \log n} \\
&= \delta(n).
\end{aligned}$$

In addition, by our definition of $\varepsilon'(n)$, we have that $\varepsilon(n^{1-\gamma})(1 + \delta(n^{1-\gamma})) + \varepsilon'(n)$ is bounded by

$$\begin{aligned}
&\left(\frac{\log n^{1-\gamma}-1}{\log n^{1-\gamma}}\right)\left(\frac{\mu \log n^{1-\gamma}-1}{\mu \log n^{1-\gamma}}\right)\frac{1}{r}\left(1+\frac{\gamma}{2(1-\gamma)\mu \log n^{1-\gamma}}\right)+\left(\frac{\gamma}{2(1-\gamma)\log n}\right)\frac{1}{r} \\
&\quad\leq \left(\frac{\log n^{1-\gamma}-1}{\log n^{1-\gamma}}\right)\left(\frac{\mu \log n^{1-\gamma}-1}{\mu \log n^{1-\gamma}}+\frac{\gamma}{\mu \log n^{1-\gamma}}\right)\frac{1}{r}+\left(\frac{\gamma}{2\log n^{1-\gamma}}\right)\frac{1}{r} \\
&\quad= \left(\frac{\log n^{1-\gamma}-1}{\log n^{1-\gamma}}\right)\left(\frac{\mu \log n-1}{\mu \log n}\right)\frac{1}{r}+\left(\frac{\gamma}{2\log n^{1-\gamma}}\right)\frac{1}{r} \\
&\quad\leq \left(\frac{\log n^{1-\gamma}-1}{\log n^{1-\gamma}}+\frac{\gamma}{\log n^{1-\gamma}}\right)\left(\frac{\mu \log n-1}{\mu \log n}\right)\frac{1}{r} \\
&\quad= \left(\frac{\log n-1}{\log n}\right)\left(\frac{\mu \log n-1}{\mu \log n}\right)\frac{1}{r} \\
&\quad= \varepsilon(n).
\end{aligned}$$

The running time of this algorithm is characterized by the recurrence

$$\mathcal{T}(n) = \mathcal{T}(n^{1-\gamma}) + b,$$

for some constant $b \geq 1$, which implies that $\mathcal{T}$ is $O(\log \log n)$. To analyze the size bound, we inductively assume that the size of the approximation returned by each recursive call is at most $c_1 r^3 n^{(1-\gamma)\gamma}$, for some constant $c_1 \geq 1$. Thus, by Theorem 3.10, the size of the approximation produced can be made to be at most $c_0(r \log n)^2(c_1 n^\gamma r^3 n^{(1-\gamma)\gamma})^{1/4}$. This is at most $c_1 r^3 n^\gamma$ if $c_1 \geq c_0^{4/3}$. The work complexity, $W(r, n)$, is therefore bounded by the recurrence equation

$$W(r, n) \leq n^\gamma W(r, n^{1-\gamma}) + O([n^\gamma r^3 n^{(1-\gamma)\gamma}]^c \cdot n^{\mu f(1)}),$$

where c is the constant in the work bound of Theorem 3.10 (note that in this case c depends only on e, the bound on the VC-exponent). If we choose γ to be a constant strictly smaller than $1/4c$, then $W(r, n)$ will be $O(nr^{3c})$. □

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

Theorem 4.3. *Let $(X, \mathcal{R})$ be a range space with VC-exponent bounded by e, for some constant $e > 0$. Also, let constants $\alpha > 0$ and $0 < \delta \leq 1/4$ be given. Then, for some constant $c > 0$, the following sets can be produced in the bounds claimed in the CRCW PRAM*:

1. *A δ-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 δ-relative $(1/r)$-approximation C 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 B of $(X, \mathcal{R})$ of size $O(r^{1+\alpha})$ in $O(\log\log n)$ time, using $O(nr^c)$ work.*
4. *A $(1/r)$-net D of $(X, \mathcal{R})$ 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 Lemma 4.2 to produce a $\delta/3$-relative $(1/2r)$-approximation of size $O(r^3 n^\beta)$, where β is the inverse of the constant in Theorem 3.13. We follow this by a call to Theorem 3.13 to find a $\delta/3$-relative $(1/3r)$-approximation of that. This set will be a δ-relative $(1/r)$-approximation of $(X, \mathcal{R})$, which is produced in $O(\log\log n)$ time using $O(nr^c)$ work. The sets B, C, and D are constructed similarly, using techniques that are now familiar. □

For analogous results for the EREW PRAM model, we may use the following theorem:

Theorem 4.4. *Let $(X, \mathcal{R})$ be a range space with VC-exponent bounded by e, for some constant $e > 0$. Also, let α be any positive constant. Then, for some constant $c > 0$, the following sets can be produced 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 C of $(X, \mathcal{R})$ of size $O(r^2 \log r)$ in $O(\log n + \log^2 r)$ time using $O(nr^{c\log r})$ work.*
3. *A $(1/r)$-net B of $(X, \mathcal{R})$ of size $O(r^{1+\alpha})$ in $O(\log n)$ time, using $O(nr^c)$ work.*
4. *A $(1/r)$-net D of $(X, \mathcal{R})$ of size $O(r \log r)$ in $O(\log n + \log^2 r)$ time using $O(nr^{c\log r})$ work.*

Proof. The method is similar to that used to derive the CRCW PRAM bounds, except that in this case we use Theorem 3.5 (in Step 3) and define **Approx** to produce a (0-relative) $\varepsilon(n)$-approximation where

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

by defining

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

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

In the next section we explore applications of these two theorems to fixed-dimensional linear programming.

5. Linear Programming in Fixed Dimensions

Recall the geometric view of fixed-dimensional linear programming. For simplicity of expression, we assume that the optimal point p exists and is defined by the intersection of exactly d half-space boundaries. We 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 [65]). 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 [3] and Dyer [26], 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 $O(r^{1+\alpha})$ (in the hyperplane set system), where $r = (w/n)^{1/c}$ such that c is a constant to be set in the analysis and α is a sufficiently small constant. By Theorem 4.3, the time needed for this step is $O(\log \log n)$ in a CRCW PRAM implementation or $O(\log n)$ time in an EREW PRAM implementation, by Theorem 4.4; the work needed for this step can be made $O(w)$ if c is a constant larger than the constants of Theorems 4.3 and 4.4.
3. Compute the intersection of the half-spaces in Y and a canonical triangulation $\mathcal{T}$ [15] 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 [37], [43], [61] 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 half-spaces in X with each of σ's $(d-1)$-dimensional boundary faces. This takes $O(1)$ time with $O(nr^{1+\alpha})$ work, which is $O(w)$ if $c \geq 1 + \alpha$.
   (b) For each simplex boundary face f we use $\mathbf{ParLP}_{d-1}$ to solve the linear program defined by f and the half-spaces that intersect f. Assuming that $\mathbf{ParLP}_{d-1}$ uses linear work, this step can be implemented using $O((n/r)r^{(1+\alpha)\lfloor d/2 \rfloor})$ work, which is $O(w)$ if $c \geq (1+\alpha)\lfloor d/2 \rfloor - 1$.
   (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 half-spaces whose boundary intersects this simplex σ and recursively call $\mathbf{ParLP}_d$ on this set of at most n/r half-spaces. 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* [31], [34], [46], 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 $O(\log\log n)$ [31] using linear work. Of course, in the EREW PRAM model this step can be easily implemented in $O(\log n)$ time via a parallel prefix computation [37], [43], [61].)

Since this method always recurses in a region σ guaranteed to contain the optimal point and we include in the subproblem all half-spaces whose boundary intersects σ, we will eventually find the optimal point p. To analyze the time complexity observe that for every $2c$ level in the recursion the problem size will go from n/r to at most 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 level in the recursion, the total work needed is $O(n)$. This gives us the following:

Theorem 5.1. *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.*

6. Conclusion

We have given a general scheme for derandomizing random sampling efficiently in parallel, and have shown how it can be used to solve the fixed-dimensional linear programming problem efficiently in parallel. Interestingly, Amato *et al.* [6], [7] have shown how to use such methods to derive efficient parallel algorithms for d-dimensional convex hull construction, planar segment intersection computation, $(1/r)$-cutting construction, and d-dimensional point location. We suspect that there may be other applications as well.

Acknowledgments

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

References

1. P. K. Agarwal. Partitioning arrangements of lines: I. An efficient deterministic algorithm. *Discrete Comput. Geom.*, 5:449–483, 1990.
2. 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.* American Mathematical Society, Providence, RI, 1991.
3. M. Ajtai and N. Megiddo. A deterministic *poly*(log log *n*)-time *n*-processor algorithm for linear programming in fixed dimension. In *Proc. 24th Ann. ACM Symp. on Theory of Computing*, pp. 327–338.
4. N. Alon and N. Megiddo. Parallel linear programming in fixed dimension almost surely in constant time. In *Proc. 31st Ann. IEEE Symp. on Foundations of Computer Science*, pp. 574–582, 1990.
5. N. Alon and J. Spencer. *The Probabilistic Method*. Wiley, New York, 1993.
6. N. M. Amato, M. T. Goodrich, and E. A. Ramos. Parallel algorithms for higher-dimensional convex hulls. In *Proc. 35th Ann. IEEE Symp. on Foundations of Computer Science*, pp. 683–694, 1994.
7. N. M. Amato, M. T. Goodrich, and E. A. Ramos. Computing faces in segment and simplex arrangements. In *Proc. 27th Ann. ACM Symp. on Theory of Computing*, pp. 672–682, 1995.
8. P. Assouad. Densité et dimension. *Ann. Inst. Fourier* (*Grenoble*). 3:232–282, 1983.
9. P. Beame and J. Hastad. Optimal bounds for decision problems on the CRCW PRAM. *J. Assoc. Comput. Mach.*, 36(3):643–670, 1989.
10. K. P. Bennett. Decision tree construction via linear programming. In *Proc. 4th Midwest Artificial Intelligence and Cognitive Science Society Conference*, pp. 97–101, 1992.
11. K. P. Bennett and O. L. Mangasarian. Multicategory discrimination via linear programming. *Optim. Methods Software*, 3:29–39, 1994.
12. B. Berger, J. Rompel, and P. W. Shor. Efficient NC algorithms for set cover with applications to learning and geometry. In *Proc. 30th Ann. IEEE Symp. on Foundations of Computer Science*, pp. 54–59, 1989. Also in *J. Comput. Systems Sci.* 49:454–477, 1994.
13. H. Brönnimann, B. Chazelle, and J. Matoušek. Produce range spaces, sensitive sampling, and derandomization. In *Proc. 34th Ann. IEEE Symp. on Foundations of Computer Science*, pp. 400–409, 1993.
14. H. Brönnimann and M. T. Goodrich. Almost optimal set covers in finite VC-dimension. *Discrete Comput. Geom.*, 14:463–479, 1995.
15. B. Chazelle and J. Friedman. A deterministic view of random sampling and its use in geometry. *Combinatorica*, 10(3):229–249, 1990.
16. B. Chazelle and J. Matoušek. On linear-time deterministic algorithms for optimization problems in fixed dimension. In *Proc. 4th ACM–SIAM Symp. on Discrete Algorithms*, pp. 281–290, 1993.
17. H. Chernoff. A measure of asymptotic efficiency for tests of a hypothesis based on the sum of the observations. *Ann. of Math. Statist.*, 23:493–509, 1952.
18. V. Chvátal. *Linear Programming*. Freeman, New York, 1983.
19. K. L. Clarkson. Linear programming in $O(n3^{d^2})$ time. *Inform. Process. Lett.*, 22:21–24, 1986.
20. K. L. Clarkson. New applications of random sampling in computational geometry. *Discrete Comput. Geom.*, 2:195–222, 1987.
21. K. L. Clarkson. A Las Vegas algorithm for linear programming when the dimension is small. In *Proc. 29th Ann. IEEE Symp. on Foundations of Computer Science*, pp. 452–456, 1988. Also in *J. Assoc. Comput. Mach.*, 42:488–499, 1995.
22. K. L. Clarkson and P. W. Shor. Applications of random sampling in computational geometry, II. *Discrete Comput. Geom.*, 4:387–421, 1989.
23. G. B. Dantzig. *Linear Programming and Extensions*. Princeton University Press, Princeton, NJ, 1963.
24. X. Deng. An optimal parallel algorithm for linear programming in the plane. *Inform. Process. Lett.*, 35:213–217, 1990.
25. M. Dietzfelbinger. Universal hashing and *k*-wise independent random variables via integer arithmetic without primes. In *Proc. 13th Ann. Symp. on Theoretical Aspects of Computer Science*, pp. 569–580, 1996. Lecture Notes in Computer Science, vol. 1046. Springer-Verlag, Berlin, 1996.
26. M. E. Dyer. A parallel algorithm for linear programming in fixed dimension. In *Proc. 11th ACM Symp. on Computational Geometry*, 1995.
27. M. E. Dyer. Linear time algorithms for two- and three-variable linear programs. *SIAM J. Comput.*, 13:31–45, 1984.

28. M. E. Dyer. On a multidimensional search technique and its application to the Euclidean one-center problem. *SIAM J. Comput.*, 15:725–738, 1986.
29. H. Edelsbrunner. *Algorithms in Combinatorial Geometry*. EATCS Monographs on Theoretical Computer Science, vol. 10. Springer-Verlag, Heidelberg, 1987.
30. D. Eppstein. Dynamic three-dimensional linear programming. *ORSA J. Comput.*, 4:360–368, 1992.
31. 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*, pp. 220–228, 1995.
32. M. T. Goodrich. Geometric partitioning made easier, even in parallel. In *Proc. 9th Ann. ACM Symp. on Computational Geometry*, pp. 73–82, 1993.
33. 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*, pp. 241–250, 1994.
34. T. Hagerup. Fast deterministic processor allocation. In *Proc. 4th ACM–SIAM Symp. on Discrete Algorithms*, pp. 1–10, 1993.
35. T. Hagerup and C. Rüb. A guided tour of Chernoff bounds. *Inform. Process. Lett.*, 33(10):305–308, 1990.
36. D. Haussler and E. Welzl. Epsilon-nets and simplex range queries. *Discrete Comput. Geom.*, 2:127–151, 1987.
37. J. Jájá. *An Introduction to Parallel Algorithms*. Addison-Wesley, Reading, MA, 1992.
38. A. Joffe. On a set of almost deterministic k-independent random variables. *Ann. Probab.*, 2:161–162, 1974.
39. G. Kalai. A subexponential randomized simplex algorithm. In *Proc. 24th Ann. ACM Symp. on Theory of Computing*, pp. 475–482, 1992.
40. H. Karloff. *Linear Programming*. Birkhäuser, Boston, 1991.
41. H. Karloff and Y. Mansour. On construction of k-wise independent random variables. In *Proc. ACM Symp. on Theory of Computing*, pp. 564–573, 1994.
42. H. Karloff and P. Raghavan. Randomized algorithms and pseudorandom numbers. *J. Assoc. Comput. Mach.*, 40(3):454–476, 1993.
43. R. M. Karp and V. Ramachandran. Parallel algorithms for shared memory machines. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, pp. 869–941. Elsevier/The MIT Press, Amsterdam/Cambridge, MA, 1990.
44. M. Luby. A simple parallel algorithm for the maximal independent set problem. *SIAM J. Comput.*, 15(4):1036–1053, 1986.
45. M. Luby. Removing randomness in parallel computation without a processor penalty. In *Proc. 29th IEEE Symp. on Foundations of Computer Science*, pp. 162–173, 1988.
46. Y. Matias and U. Vishkin. Converting high probability into nearly-constant time—with applications to parallel hashing. In *Proc. 23rd Ann. ACM Symp. on Theory of Computing*, pp. 307–316, 1991.
47. J. Matoušek. Approximations and optimal geometric divide-and-conquer. In *Proc. 23rd Ann. ACM Symp. on Theory of Computing*, pp. 505–511, 1991. Also in *J. Comput. System Sci.*, 50:203–208, 1995.
48. J. Matoušek. Cutting hyperplane arrangements. *Discrete Comput. Geom.*, 6:385–406, 1991.
49. J. Matoušek. Efficient partition trees. *Discrete Comput. Geom.*, 8:315–334, 1992.
50. 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 Symp. on Computational Geometry*, pp. 16–25, 1992.
51. J. Matoušek. Epsilon-nets and computational geometry. In J. Pach, editor, *New Trends in Discrete and Computational Geometry*, pp. 69–89. Algorithms and Combinatorics, Vol. 10. Springer-Verlag, Berlin, 1993.
52. J. Matoušek, M. Sharir, and E. Welzl. A subexponential bound for linear programming. In *Proc. 8th Ann. ACM Symp. on Computational Geometry*, pp. 1–8, 1992. Also in *Algorithmica*, 16:498–516, 1996.
53. J. Matoušek, E. Welzl, and L. Wernisch. Discrepancy and ε-approximation for bounded VC-dimension. *Combinatorica*, 13:455–466, 1993.
54. N. Megiddo. Linear-time algorithms for linear programming in R^3 and related problems. *SIAM J. Comput.*, 12:759–776, 1983.
55. N. Megiddo. Linear programming in linear time when the dimension is fixed. *J. Assoc. Comput. Mach.*, 31:114–127, 1984.
56. R. Motwani, J. Naor, and M. Naor. The probabilistic method yields deterministic parallel algorithms. In

Proc. 30*th Ann*. *IEEE Symp. on Foundations of Computer Science*, pp. 8–13, 1989. Also in *J. Comput. System Sci*., 49:478–516, 1994.
57. R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, New York, 1995.
58. K. Mulmuley. *Computational Geometry*: *An Introduction Through Randomized Algorithms*. Prentice-Hall, Englewood Cliffs, NJ, 1993.
59. C. H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization*: *Algorithms and Complexity*. Prentice-Hall, Englewood Cliffs, NJ, 1982.
60. F. P. Preparata and M. I. Shamos. *Computational Geometry*: *An Introduction*. Springer-Verlag, New York, 1985.
61. J. H. Reif. *Synthesis of Parallel Algorithms*. Morgan Kaufmann, San Mateo, CA, 1993.
62. J. T. Rompel. Techniques for Computing with Low-Independence Randomness. Ph.D. thesis, Dept. of EECS, M.I.T., 1990.
63. N. Sauer. On the density of families of sets. *J. Combin. Theory*, 13:145–147, 1972.
64. J. P. Schmidt, A. Siegel, and A. Srinivasan. Chernoff–Hoeffding bounds for applications with limited independence. In *Proc*. 4*th ACM–SIAM Symp. on Discrete Algorithms*, pp. 331–340, 1993.
65. R. Seidel. Small-dimensional linear programming and convex hulls made easy. *Discrete Comput. Geom*., 6:423–434, 1991.
66. 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.
67. L. Valiant. Parallelism in comparison problems. *SIAM J. Comput*., 4(3):348–355, 1975.
68. 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.

Received August 7, 1995, *and in revised form November* 11, 1996.