

Parallel Algorithms via the Probabilistic Method

	1.1 Introduction and Preliminaries	1-1 ■
	Introduction • Preliminaries	
	1.2 Limited Independence	1-7 ■
	The Method of Conditional Probabilities • k -Wise Independence and Small Sample Spaces • Almost k -Wise Independence.	
	1.3 Parallel Algorithms using k -Wise Independence ..	1-15 ■
	Functions over \mathbb{Z}_2^k • Multivalued Functions • Maximal Independent Sets in Graphs • Low Discrepancy Colorings	
	1.4 Fooling Automata	1-36 ■
	1.5 Parallel Algorithms for Packing Integer Programs	1-46 ■
	Impact of k -Wise Independence: Multidimensional Bin Packing • Impact of Automata Fooling: Srinivasan's Parallel Packing	
	1.6 Open Problems	1-56 ■
Lasse Kliemann		
<i>Christian-Albrechts Universität zu Kiel</i>		
Anand Srivastav		
<i>Christian-Albrechts Universität zu Kiel</i>		

1.1 Introduction and Preliminaries

1.1.1 Introduction

We give an introduction to the design of parallel algorithms with the probabilistic method. Algorithms of this kind usually possess a randomized sequential counterpart. Parallelization of such algorithms is inherently linked with derandomization, either with the Erdős-Spencer method of conditional probabilities, or exhaustive search in a polynomial sized sample space.

The key notation is the treatment of random variables with various concepts of only limited independence, leading to polynomial sized sample spaces with time and space efficient implementations of the conditional probability method.

Starting with the definition of limited independence in Section 1.2, we discuss in Section 1.3 algorithms using limited independence in a more or less direct way: the maximal independent set algorithms of Luby (1986) [Lub86], which marks the begin of the subject, and the algorithms due to Berger, Rompel (1989) [BR91] and Motwani, Naor, Naor (1989, 1994) [MNN94] for parallel computation of low discrepancy colorings of hypergraphs and its extension to the lattice approximation problem. Although the lattice approximation problem can be viewed as a packing problem, the direct application of limited independence could not achieve parallel counterparts of well-known approximation algorithms for

integer programs of packing type. An important step broadening the range of applicability has been the introduction of efficient parallelization by so called fooling of automata by Karger and Koller [KK97]¹, which is presented in Section 1.4. The idea is to simulate a randomized algorithm by deterministic finite automata on a probability distribution for which the number of strings with nonzero probability is polynomial in the input size, while the original probability distributions for the randomized algorithm typically lead to an exponential number of such strings. A similar approach giving a little stronger results for the lattice approximation problem has been presented by Mahajan, Ramos, and Subrahmanyam [MRS97]. Finally, in Section 1.5 we first show the impact of limited independence to a packing type problem, the multidimensional bin packing due to Srivastav and Stangier [SS97a, SS97b] and then proceed to Srinivasan's [Sri01a] parallel algorithm for general packing integer problems, which is based on the automata fooling and a nice compression of the (exponential) state space to polynomial size. A comprehensive treatment of all applications of the probabilistic method in parallelizing algorithms is beyond the scope of this chapter. Though, we have tried to include many topics in the bibliography and remarks at the end of the specific sections. The intention of this paper is to show the development of the method of k -wise independence in a straightforward way, starting with applications where constant independence is sufficient, passing applications of $\log n$ -wise independence like discrepancy problems and reaching the applications to integer programming. We give the proofs in the first three sections in full detail so that also non-experts can find a self-contained introduction to the subject with detailed proofs of processor and time bounds without tracing a list of papers with sometimes varying notations. The examinations of so far presented proofs lead to the revision of some processor and time bounds. The Sections 1.4 and 1.5 contain some state-of-the-art results on parallel algorithms via the probabilistic method. Our emphasis there is on methodical aspects rather than completeness. There are numerous open problems in the area. In Section 1.6 we give a selection of them which we find not only appealing, but important beyond the scope of parallelization.

1.1.2 Preliminaries

Vectors and Matrices

For every $n \in \mathbb{N}$ denote $[n] := \{1, \dots, n\}$. The logarithm \log is always taken to basis 2. Given a $(m \times n)$ matrix A over some field \mathbb{F} , denote by $\text{range } A$ its range $\{Ax; x \in \mathbb{F}^n\} \subseteq \mathbb{F}^m$ and set $\text{rank } A := \dim \text{range } A$. The kernel, i.e., elements in \mathbb{F}^n that are mapped to zero, is denoted by $\ker A$. For a vector $x \in \mathbb{F}^n$ and an index set $I \subseteq [n]$ denote by x_I the vector of entries from x with indices in I . We choose the same notation, x_α for vectors of indices, i.e., $\alpha \in [n]^k$ for some $k \in \mathbb{N}$. For a matrix A , the submatrix with rows from A according to I is A_I (according to a vector α , it is A_α). For two vectors $x, y \in \mathbb{F}^n$ denote the scalar (or inner) product $x \cdot y := \sum_{i \in [n]} x_i y_i$.

The i th component of a vector $v \in \mathbb{F}^n$ as usually is denoted by v_i , and if a family of vectors v_1, \dots, v_k is given, v_{ji} denotes the i th coordinate of vector v_j , $i \in [n]$, $j \in [k]$.

¹We note that a close inspection and analysis of the lattice approximation algorithms of [MNN94] gives error bounds leading to weaker bounds for the support size in the basis crashing algorithm of [KK97].

Binary Representations

We use two different ways for the binary representation of natural and rational numbers. Given a set $\Omega := \{0, \dots, d-1\}$ for some $d \in \mathbb{N}$ which is a power of 2, and a vector $x \in \Omega^n$ (or a sequence $x_1, \dots, x_n \in \Omega$) we write $\mathbf{x} \in \{0, 1\}^{n \log d}$ for the binary representation of x . More precisely, \mathbf{x} is chosen such that

$$x_i = \sum_{j=(i-1)\log(d)+1}^{i \log d} \mathbf{x}_j 2^{j-1-(i-1)\log d} \quad \text{for all } i \in [n]. \quad (1.1)$$

Given a number $x \in [0, 1]$ with $L+1$ bits of precision, $\text{bit}_l(x)$ denotes the l th bit, i.e., we have

$$x = \sum_{l=0}^L 2^{-l} \text{bit}_l(x).$$

Given a vector $p \in [0, 1]^n$, where each of the n entries is given with $L+1$ bits of precision, we sometimes need the l th bit of each of the entries. We write $\vec{\text{bit}}_l(p) = (\text{bit}_l(p_1), \dots, \text{bit}_l(p_n))$ for this.

Logarithmic Terms

Given parameters a_1, \dots, a_k and a function f depending on them, we write $f(a_1, \dots, a_k) = O(\log^{(*)}(a_1, \dots, a_k))$ if $f(a_1, \dots, a_k) = O(\log(g(a_1, \dots, a_k)))$, and g depends polynomially on a_1, \dots, a_k . For example, $\log(a_1 \log^2(a_2 \log(a_3^5) + a_4)) = O(\log^{(*)}(a_1, \dots, a_4))$. We drop the parameters (a_1, \dots, a_k) in some cases, if they are polynomially bounded in the input length of the given problem. This will be used to simplify the statement of some results.

Graphs and Hypergraphs

We use the standard notation of finite graphs and hypergraphs [Ber73]. A *graph* $G = (V, E)$ is a pair of a finite set V (the set of *vertices* or *nodes*) and a subset $E \subseteq \binom{V}{2}$, where $\binom{V}{2}$ denotes the set of all 2-element subsets of V . The elements of E are called *edges*. Denote the *neighbors* of a vertex $v \in V$ by $N(v) := \{w \in V; \exists \{v, w\} \in E\}$. For a subset $S \subseteq V$ of vertices denote their neighbors by $N(S) := \{w \in V \setminus S; \exists v \in S : \{v, w\} \in E\}$.

A *hypergraph* or set system $\mathcal{H} = (V, \mathcal{E})$ is a pair of a finite set V and a subset $\mathcal{E} \subseteq \mathcal{P}(V)$ of the powerset $\mathcal{P}(V)$. The elements of \mathcal{E} are called *hyperedges*. The *degree* of a vertex $v \in V$ in \mathcal{H} , denoted by $\deg(v)$ or $d(v)$, is the number of hyperedges containing v , and $\deg(\mathcal{H}) = \max_{v \in V} d(v)$ is the (*vertex-*)*degree* of \mathcal{H} . For a pair of vertices $u, v \in V$, $\text{codeg}(u, v)$ is the *co-degree* of u and v , and is the number of edges containing both u and v , and $\text{codeg}(\mathcal{H})$ is the maximum over all $\text{codeg}(u, v)$. \mathcal{H} is called *r-regular* if $\deg(v) = r$ for all $v \in V$. It is called *s-uniform* if $|E| = s$ for all $E \in \mathcal{E}$. It is convenient to order the vertices and hyperedges, $V = \{v_1, \dots, v_n\}$ and $\mathcal{E} = \{E_1, \dots, E_m\}$, and to identify vertices and edges with their indices. The *vertex-hyperedge incidence matrix* of a hypergraph $\mathcal{H} = (V, \mathcal{E})$, with $V = \{v_1, \dots, v_n\}$ and $\mathcal{E} = \{E_1, \dots, E_m\}$, is a matrix $A = (a_{ij}) \in \{0, 1\}^{n \times m}$, where $a_{ij} = 1$ if $v_i \in E_j$, and 0 else.

For a modern treatment of graph theory, we refer to the books of Bollobás [Bol98], Diestel [Die97], and West [Wes96].

Probabilistic Tools

Throughout this chapter we consider only finite probability spaces (Ω, \mathbb{P}) , where Ω is a finite set, usually a product of finite sets, and \mathbb{P} is a probability measure with respect to the powerset $\mathcal{P}(\Omega)$ as the sigma field. Let u_1, \dots, u_n and v_1, \dots, v_n be integers, and let X_1, \dots, X_n be mutually independent (briefly independent) random variables, where X_j takes the values u_j and v_j , $j \in [n]$ and

$$\mathbb{P}[X_j = u_j] = p_j, \quad \mathbb{P}[X_j = v_j] = 1 - p_j$$

for $p_j \in [0, 1]$ for all $j \in [n]$. For all $j \in [n]$ let w_j denote rational weights with

$$0 \leq w_j \leq 1 \quad \text{and let} \quad \psi := \sum_{j=1}^n w_j X_j$$

be the weighted sum. For $u_j = 1$, $v_j = 0$, $w_j = 1$ and $p_j = p$ for all $j \in [n]$, the sum $\psi = \sum_{j=1}^n X_j$ is the well-known binomially distributed random variable with mean np . The first large deviation inequality is implicitly given in Chernoff [Che52] in the binomial case. In explicit form it can be found in Okamoto [Oka58]. Its generalization to arbitrary weights is due to Hoeffding [Hoe63]:

THEOREM 1.1 (Hoeffding 1963)

Let $u_j = 1$, $v_j = 0$ for all $j \in [n]$ and let $\lambda > 0$. Then

- (i). $\mathbb{P}[\psi \geq \mathbb{E}[\psi] + \lambda] \leq \exp(-\frac{2\lambda^2}{n})$
- (ii). $\mathbb{P}[\psi \leq \mathbb{E}[\psi] - \lambda] \leq \exp(-\frac{2\lambda^2}{n})$.

In the literature Theorem 1.1 is well known as the Chernoff bound. A proof of a stronger version of it can be found in the Book of Habib, McDiarmid, Ramirez-Alfonsin, and Reed [HMRAR98]. The following theorem is also known as Chernoff bound. Its proof for the unweighted case can be found at several places, e.g., in the book of Motwani and Raghavan [MR95]. For convenience, we give a proof for the general case.

THEOREM 1.2 Let $u_j = 1$, $v_j = 0$. Then

- (i). $\mathbb{P}[\psi \geq (1 + \beta)\mathbb{E}[\psi]] \leq \left(\frac{\exp(\beta)}{(1+\beta)^{(1+\beta)}}\right)^{\mathbb{E}[\psi]}$ for all $0 \leq \beta$, and
- (ii). $\mathbb{P}[\psi \leq (1 - \beta)\mathbb{E}[\psi]] \leq \left(\frac{\exp(-\beta)}{(1-\beta)^{(1-\beta)}}\right)^{\mathbb{E}[\psi]}$ for all $0 \leq \beta \leq 1$.

Proof Fix some $t > 0$, which will be specified precisely later. By the Markov inequality and independence we have

$$\begin{aligned} \mathbb{P}[\psi \geq (1 + \beta)\mathbb{E}[\psi]] &= \mathbb{P}[\exp(t\psi) \geq \exp(t(1 + \beta)\mathbb{E}[\psi])] \\ &\leq \frac{\mathbb{E}[\exp(t\psi)]}{\exp(t(1 + \beta)\mathbb{E}[\psi])} \\ &= \frac{\prod_{j=1}^n \mathbb{E}[\exp(tw_j X_j)]}{\exp(t(1 + \beta)\mathbb{E}[\psi])}. \end{aligned} \tag{1.2}$$

For all $j \in [n]$ we can bound the expectation $\mathbb{E}[\exp(tw_j X_j)] = p_j \exp(tw_j) + (1 - p_j) \leq \exp(p_j \exp(tw_j) - p_j)$, because $1 + x \leq \exp(x)$ holds for all $x \geq 0$. Plugging in $t = \ln(1 + \beta)$ yields $\mathbb{E}[\exp(tw_j X_j)] \leq \exp(p_j \exp(\ln(1 + \beta)w_j) - p_j) = \exp(p_j(1 + \beta)^{w_j} - p_j)$. Because for all $x \geq 0$ and $y \in [0, 1]$ we have $(1 + x)^y \leq 1 + xy$, we find $\exp(p_j(1 + \beta)^{w_j} - p_j) \leq \exp(p_j(1 + \beta w_j) - p_j) = \exp(\beta p_j w_j)$. So we have

$$\prod_{j=1}^n \mathbb{E}[\exp(tw_j X_j)] \leq \prod_{j=1}^n \exp(\beta p_j w_j) = \exp\left(\sum_{j=1}^n \beta p_j w_j\right) = \exp(\beta \mathbb{E}[\psi]).$$

Using this in (1.2) and further simplifying with $t = \ln(1 + \beta)$ yields the assertion (i) of the theorem. The assertion (ii) can be proven along the same lines, using that $(1 - x)^y \leq 1 - xy$ for all $x, y \in [0, 1]$. \blacksquare

The right-hand-sides of these bounds are known as functions G and H resp. in the literature:

$$G(\mu, \beta) := \left(\frac{\exp(\beta)}{(1 + \beta)^{(1+\beta)}}\right)^\mu, \quad 0 \leq \mu, \beta,$$

$$H(\mu, \beta) := \left(\frac{\exp(-\beta)}{(1 - \beta)^{(1-\beta)}}\right)^\mu, \quad 0 \leq \mu, 0 \leq \beta \leq 1.$$

By real analysis, one can show that for $0 \leq \beta \leq 1$ we have $G(\mu, \beta) \leq \exp(-\frac{\beta^2 \mu}{3})$ and $H(\mu, \beta) \leq \exp(-\frac{\beta^2 \mu}{2})$. This gives the following bound, originally due to Angluin and Valiant [AV79].

THEOREM 1.3 (Angluin, Valiant 1979)

Let $u_j = 1, v_j = 0$ for all $j \in [n]$ and let $0 \leq \beta \leq 1$. Then

- (i). $\mathbb{P}[\psi \geq (1 + \beta)\mathbb{E}[\psi]] \leq \exp(-\frac{\beta^2 \mathbb{E}[\psi]}{3})$,
- (ii). $\mathbb{P}[\psi \leq (1 - \beta)\mathbb{E}[\psi]] \leq \exp(-\frac{\beta^2 \mathbb{E}[\psi]}{2})$.

For random variables with zero expectation there are several inequalities which can be found in the book of Alon, Spencer, and Erdős [ASE92], and Alon and Spencer [AS00].

THEOREM 1.4 (Hoeffding 1963)

Let $u_j = 1, v_j = -1, w_j = 1$ for all $j \in [n]$. For $\lambda > 0$ we have

- (i). $\mathbb{P}[\psi \geq \lambda] \leq \exp(-\frac{\lambda^2}{2n})$,
- (ii). $\mathbb{P}[\psi \leq -\lambda] \leq \exp(-\frac{\lambda^2}{2n})$.

The following, for the unweighted case (i.e., $w_j = 1$ for all $j \in [n]$), is [AS00, Th. A.1.6, A.1.7]. Their proof can be extended in a straight-forward way to the general case.

THEOREM 1.5 Let $u_j = 1 - p_j, v_j = -p_j$ for all $j \in [n]$ and let $\lambda > 0$. Then

- (i). $\mathbb{P}[\psi \geq \lambda] \leq \exp(-\frac{2\lambda^2}{\sum_j w_j^2})$,
- (ii). $\mathbb{P}[\psi \leq -\lambda] \leq \exp(-\frac{2\lambda^2}{\sum_j w_j^2})$.

Hence $\mathbb{P}[|\psi| \geq \lambda] \leq 2 \exp(-\frac{2\lambda^2}{\sum_j w_j^2})$.

Proof Assertion (ii) follows from (i) via symmetry (apply (i) to $-\psi$) and the assertion about $|\psi|$ clearly is a direct consequence of (i) and (ii).

We show (i). The idea is the same as in the proof of Theorem 1.2. Fix $t > 0$ to be specified later. By the Markov inequality and by independence we have

$$\mathbb{P}[\psi \geq \lambda] = \mathbb{P}[\exp(t\psi) \geq \exp(t\lambda)] \leq \frac{\mathbb{E}[\exp(t\psi)]}{\exp(t\lambda)} = \frac{\prod_{j=1}^n \mathbb{E}[\exp(tw_j X_j)]}{\exp(t\lambda)}. \quad (1.3)$$

For all $j \in [n]$ we have $\mathbb{E}[\exp(tw_j X_j)] = p_j \exp(tw_j(1-p_j)) + (1-p_j) \exp(-tw_j p_j) \leq \exp(\frac{t^2 w_j^2}{8})$, because $x \exp(t(1-x)) + (1-x) \exp(-tx) \leq \exp(\frac{t^2}{8})$ holds for all $x \in [0, 1]$, see [AS00, Lem. A.1.6]. So we have

$$\prod_{j=1}^n \mathbb{E}[\exp(tw_j X_j)] \leq \prod_{j=1}^n \exp(\frac{t^2 w_j^2}{8}) = \exp(\frac{t^2}{8} \sum_{j=1}^n w_j^2).$$

With (1.3), we see

$$\mathbb{P}[\psi \geq \lambda] \leq \exp(\frac{t^2}{8} \sum_{j=1}^n w_j^2 - t\lambda).$$

Choosing $t := \frac{4\lambda}{\sum_{j=1}^n w_j^2}$ yields the bound in (i). Note that we may assume that not all weights are zero, since otherwise the statement of the theorem is trivial. ■

Alon and Spencer improved the Hoeffding bound to $\exp(-\frac{2\lambda^2}{n})$ replacing n by $pn = p_1 + \dots + p_n$.

THEOREM 1.6 (Alon, Spencer 1992)

Let $u_j = 1 - p_j$, $v_j = -p_j$, $w_j = 1$ for all $j \in [n]$ and let $\lambda > 0$. Set $p = \frac{1}{n}(p_1 + \dots + p_n)$. Then

- (i). $\mathbb{P}[\psi \geq \lambda] \leq \exp(-\frac{\lambda^2}{2pn} + \frac{\lambda^3}{2(pn)^2})$,
- (ii). $\mathbb{P}[\psi \leq -\lambda] \leq \exp(-\frac{\lambda^2}{2pn})$.

Computational Model

In this chapter we consider the synchronous parallel random access machine model, the PRAM model. The various specifications for the PRAM are EREW, ERCW, CREW, and CRCW, indicating exclusive read/exclusive write, exclusive read/common write, common read/exclusive write, and common read/common write into the global memory, respectively. We consider in this chapter only the EREW PRAM model, thus do not specify in the statements of processor bounds and running times the parallel model explicitly. We sometimes write \mathcal{NC} algorithm instead of *parallel algorithm*, referring to the class \mathcal{NC} (for a definition see [Pap94, Chapter 15]). For model details we refer the reader to standard text books, e.g., Papadimitriou [Pap94]. A quick introduction is given in the book of Motwani and Raghavan [MR95] in the chapter on parallel and distributed computing.

1.2 Limited Independence

1.2.1 The Method of Conditional Probabilities

The method of conditional probabilities has been invented by P. Erdős and J.L. Selfridge [ES73]. Further developments were given by Beck and Fiala [BF81], Beck and Spencer [BS83], Spencer [Spe87], and introduced as a derandomization technique in integer programming by Raghavan [Rag88]. With Raghavan's paper the technique became a common and popular tool in the design of derandomized algorithms.

We consider the probability space (Ω, \mathbb{P}) where $\Omega = \prod_{i=1}^n \Omega_i$, $\Omega_i = [N]$, the powerset $\mathcal{P}(\Omega)$ is the σ -field and \mathbb{P} is a product, i.e., $\mathbb{P} = \bigotimes_{i=1}^n \mathbb{P}_i$, where \mathbb{P}_i is a probability measure on Ω_i . Let $E \subseteq \Omega$ be an event with $\mathbb{P}[E] > 0$ and let E^c denote the complement of E . For $\omega_1, \dots, \omega_l \in [N]$, $l \in [n]$ denote

$$\mathbb{P}[E^c \mid \omega_1, \dots, \omega_l] := \frac{\mathbb{P}[E^c \cap \{y \in \Omega; y_1 = \omega_1, \dots, y_l = \omega_l\}]}{\mathbb{P}_1[\{\omega_1\}] \cdot \dots \cdot \mathbb{P}_l[\{\omega_l\}]}.$$

The following simple procedure constructs a vector in E .

Algorithm 1: CONDPROB

Input: An event $E \subseteq \Omega$ with $\mathbb{P}[E] > 0$.

Output: A vector $x \in E$.

choose x_1 as a minimizer of the function $[N] \rightarrow [0, 1]$, $\omega \mapsto \mathbb{P}[E^c \mid \omega]$;

for $l \leftarrow 2$ **to** n **do**

choose x_l as a minimizer of the function $[N] \rightarrow [0, 1]$, $\omega \mapsto \mathbb{P}[E^c \mid x_1, \dots, x_{l-1}, \omega]$;

end

A similar algorithm works for conditional expectations.

Algorithm 2: CONDEXP

Input: A function $F : \Omega \mapsto \mathbb{Q}$.

Output: A vector $x \in \Omega$ with $F(x) \geq \mathbb{E}[F]$.

choose x_1 as a minimizer of the function $[N] \rightarrow [0, 1]$, $\omega \mapsto \mathbb{E}[F \mid \omega]$;

for $l \leftarrow 2$ **to** n **do**

choose x_l as a minimizer of the function $[N] \rightarrow [0, 1]$, $\omega \mapsto \mathbb{E}[F \mid x_1, \dots, x_{l-1}, \omega]$;

end

PROPOSITION 1.1 The algorithms CONDPROB and CONDEXP are correct.

Proof The argument is due to Erdős and Selfridge [ES73]. Let $x_1, \dots, x_{l-1} \in [N]$, $l \in [n]$. Conditional probabilities can be written as a convex combination:

$$\mathbb{P}[E^c \mid x_1, \dots, x_{l-1}] = \sum_{\omega \in [N]} \mathbb{P}_l[\{\omega\}] \cdot \mathbb{P}[E^c \mid x_1, \dots, x_{l-1}, \omega].$$

By the choice of the x_1, \dots, x_n and the assumption $\mathbb{P}[E] > 0$,

$$1 > \mathbb{P}[E^c] \geq \mathbb{P}[E^c \mid x_1] \geq \dots \geq \mathbb{P}[E^c \mid x_1, \dots, x_n] \in \{0, 1\},$$

so $\mathbb{P}[E^c \mid x_1, \dots, x_n] = 0$ and $x \in E$. The proof for the algorithm CONDEXP is similar. ■

Efficiency of these algorithms depends on the efficient computation of the conditional probabilities and expectations, respectively. In general, it seems to be hopeless to compute conditional probabilities directly. But for the purpose of derandomization it suffices to compute upper bounds for the conditional probabilities, which then play the role of the conditional probabilities. Such upper bounds have been introduced by Spencer [Spe87] in the hyperbolic cosine algorithm, and later defined in a rigorous way as so-called pessimistic estimators by Raghavan [Rag88]. Comprehensive treatment of sequential derandomization in combinatorial optimization can be found in the handbook of randomized computing [Sri01b].

1.2.2 k -Wise Independence and Small Sample Spaces

The conditional probability method can be viewed as a binary search for a “good” vector in the sample space Ω . If the size of Ω is polynomial in N and n , an exhaustive search is already a polynomial-time algorithm. Small sample spaces correspond to random variables with limited rather than full independence. Limited independence is applicable not only for derandomization, but also for parallelization.

Let $(\Omega, \Sigma, \mathbb{P})$ be a probability space. In this chapter we consider only random variables X_1, \dots, X_n with finite ranges D_1, \dots, D_n .

DEFINITION 1.1 (k -Wise Independence)

X_1, \dots, X_n are k -wise independent if for any $J \subseteq [n]$ with $|J| \leq k$ and any choice of $\alpha_j \in D_j, j \in J$ we have

$$\mathbb{P}[X_j = \alpha_j \text{ for all } j \in J] = \prod_{j \in J} \mathbb{P}[X_j = \alpha_j]. \quad (1.4)$$

The definition usually given in papers from the theoretical computer science community is the following special case of Definition 1.1:

REMARK 1.1 For uniformly distributed 0/1 random variables, k -wise independence reads as follows: X_1, \dots, X_n are k -wise independent if for any $r \in [k]$, $\alpha \in \{0, 1\}^r$ and any choice of r variables X_{i_1}, \dots, X_{i_r} , $1 \leq i_1 < i_2 < \dots < i_r \leq n$,

$$\mathbb{P}[(X_{i_1}, \dots, X_{i_r}) = \alpha] = 2^{-r}.$$

A motivation for k -wise independence from the computational point of view is the following observation. If X_1, \dots, X_n are independent 0/1 random variables, then the generation of a vector $(X_1, \dots, X_n) = (\omega_1, \dots, \omega_n)$ requires n bits (n independent Bernoulli trials). Now suppose that there is a $(n \times l)$ matrix B such that

$$BY = X,$$

where $Y = (Y_1, \dots, Y_l)$, $X = (X_1, \dots, X_n)$, the Y_1, \dots, Y_l are independent and X_1, \dots, X_n are k -wise independent 0/1 random variables. If l is smaller than n , we need only $l < n$ random bits in order to generate X_1, \dots, X_n . In other words, the sample space is $\{0, 1\}^l$ instead of $\{0, 1\}^n$ corresponding to the X_1, \dots, X_n .

Alon, Babai, and Itai [ABI86] showed that k -wise independent 0/1 random variables can be constructed from mutually independent random variables using a matrix over $GF(2)$.

For simplicity, we make two assumptions.

- We assume that $n = 2^{n'} - 1$ for some $n' \in \mathbb{N}$, $n' \geq 2$. If n is not of this form, we take the next larger number having this property. This enlarges the quantity n by at most a factor of 2. This will change the size of the sample space, see Theorem 1.7. It will not, however, change the bounds on the quantity l , which later will play the most important role. In all bounds on l , see (1.7) and (1.11), there is only logarithmic dependence on n , and so bounds only change by a constant factor (hidden in the $O(\cdot)$ notation).
- We assume k to be odd, i.e., $k = 2k' + 1$ for some $k' \in \mathbb{N}$. If k is even, take $k + 1$ instead of k . This does not change the bound on the size of the sample space in Theorem 1.7.

Let b_1, \dots, b_n be the n non-zero elements of $GF(2^{n'})$ and let B be the following $(n \times (1 + \frac{k-1}{2}))$ matrix over $GF(2^{n'})$

$$B := \begin{bmatrix} 1 & b_1 & b_1^3 & \dots & b_1^{k-2} \\ 1 & b_2 & b_2^3 & & b_2^{k-2} \\ 1 & b_3 & b_3^3 & & b_3^{k-2} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & b_n & b_n^3 & & b_n^{k-2} \end{bmatrix}. \tag{1.5}$$

In coding theory B is well-known as the parity check matrix of binary *BCH* codes [MS77].

PROPOSITION 1.2 Any k rows of B are linearly independent over $GF(2)$.

Proof Let $I \subseteq n$ be the set of indices of any k rows of B . The idea is to extend the submatrix defined by I to a Vandermonde matrix. This is done by inserting columns with even powers 2 to $k - 1$. Let $\lambda_i \in GF(2)$, $i \in I$ be such that

$$\sum_{i \in I} \lambda_i b_i^j = 0 \quad \text{for all } j \in \{1, 3, \dots, k - 2\}. \tag{1.6}$$

Let $r \in \{2, 4, \dots, k - 1\}$. Then we can write $r = 2^{r'} j$ with an odd $j \in \{1, 3, \dots, k - 2\}$. We have

$$\begin{aligned} 0 &= \left(\sum_{i \in I} \lambda_i b_i^j \right)^{2^{r'}} && \text{due to (1.6)} \\ &= \sum_{i \in I} \lambda_i^{2^{r'}} b_i^{j \cdot 2^{r'}} && GF(2^{n'}) \text{ has characteristic 2} \\ &= \sum_{i \in I} \lambda_i b_i^r && \text{since } \lambda_i \in \{0, 1\}. \end{aligned}$$

Hence

$$\sum_{i \in I} \lambda_i b_i^j = 0 \quad \text{for all } j \in [k - 1].$$

As the Vandermonde matrix is non-singular, it follows that $\lambda_i = 0$ for all $i \in I$. ■

The additive group of $GF(2^{n'})$ is isomorphic to the vector space $\bigotimes_{i=1}^{n'} GF(2)$, which has dimension n' over $GF(2)$. Hence the elements of $GF(2^{n'})$ (and so the entries in B) can be represented by 0/1 vectors of length n' , and their componentwise addition is the addition in $GF(2^{n'})$. Put

$$l := 1 + \frac{k-1}{2}n' = 1 + \frac{k-1}{2}\log(n+1) = O(k \log n). \quad (1.7)$$

In the following, we will regard B as a $(n \times l)$ matrix, where all columns except the first one are expanded to their 0/1 representation. It is clear that Proposition 1.2 still holds for this expanded version of B .

Let Y_1, \dots, Y_l be independent and uniformly distributed. Let Y be the vector $Y = (Y_1, \dots, Y_l)$. Define $\Omega := \{0, 1\}$ -valued random variables X_1, \dots, X_n by

$$X_i := (BY)_i \quad \text{for all } i \in [n], \quad (1.8)$$

where the computation is done over $GF(2)$. The following theorem was proved by Alon, Babai, and Itai [ABI86].

THEOREM 1.7 *The random variables X_1, \dots, X_n as defined in (1.8) are k -wise independent and uniformly distributed. The size of the sample space is*

$$O((n+1)^{\lfloor \frac{k}{2} \rfloor}).$$

Proof Choose $I \subseteq [n]$ with $k_0 := |I| \leq k$. Let $x \in \Omega^{k_0}$ be an arbitrarily chosen but fixed vector ($\Omega = \{0, 1\}$). Set $X_I = (X_i)_{i \in I}$. For k -wise independence we must show $\mathbb{P}[X_I = x] = 2^{-k_0}$. Let B_I be the submatrix of B with row indices from I . B_I is a $(k_0 \times l)$ matrix over $GF(2)$. Because of Proposition 1.2 we can extend B_I to an invertible $(l \times l)$ matrix C over $GF(2)$. Define

$$\Omega_x = \{x' \in \Omega^l; x'_i = x_i \text{ for all } i \in [k]\}. \quad (1.9)$$

Then $|\Omega_x| = 2^{l-k}$, and we have (all calculations done in $GF(2)$)

$$\begin{aligned} \mathbb{P}[X_I = x] &= \mathbb{P}[B_I Y = x] \\ &= \mathbb{P}[CY \bmod d \in \Omega_x] \\ &= \sum_{x' \in \Omega_x} \mathbb{P}[CY - x' = 0] \\ &= \sum_{x' \in \Omega_x} \mathbb{P}[Y = C^{-1}x'] \\ &= \sum_{x' \in \Omega_x} 2^{-l} \quad (\text{the } Y_1, \dots, Y_l \text{ are independent}) \\ &= \frac{|\Omega_x|}{2^l} = \frac{2^{l-k_0}}{2^l} = 2^{-k_0}. \end{aligned}$$

The random variables X_i constructed in this theorem can be viewed as mappings $X_i : \{0, 1\}^l \rightarrow \{0, 1\}, \omega \mapsto (B\omega)_i$ and the sample space corresponding to the X_1, \dots, X_n is $\{0, 1\}^l$. According to (1.7) its size is

$$2^l = O((n+1)^{\lfloor \frac{k}{2} \rfloor}). \quad \blacksquare$$

In view of the lower bound $\Omega(n^k)$ for sample spaces of certain k -wise independent random variables given by Chor, Friedmann, Goldreich, Håstad, Rudich, and Smolensky [CFG⁺85] this is best possible up to constants in the exponent. Note that the sample space constructed by Alon, Babai, and Itai is polynomial only if k is constant. In applications k is often not constant, but $k = O(\log^c n)$ with some constant $c > 0$. Derandomization in such sample spaces can be done combining $O(\log^c n)$ -wise random variables with the conditional probability method. A significant reduction of the size of the sample space was achieved by Naor and Naor [NN93]. The heart of their construction is the notation of almost k -wise independent random variables discussed in the next section.

The construction described above can be extended to multivalued random variables. Fix $d \in \mathbb{N}$ which is a power of 2, and set $\Omega := \{0, \dots, d-1\}$. The aim is to construct k -wise independent uniformly distributed random variables X_1, \dots, X_n with values in Ω and a “small” sample space. To this end, use the above construction to find $N := n \log d$ uniformly distributed binary random variables $\mathbf{X}_1, \dots, \mathbf{X}_N$ that are $(k \log d)$ -wise independent². These are interpreted as binary representations of the to construct random variables. For every $i \in [n]$ we then put

$$X_i := \sum_{j=(i-1)\log(d)+1}^{i\log d} \mathbf{X}_j 2^{j-1-(i-1)\log d}. \quad (1.10)$$

It can easily be verified that X_1, \dots, X_n are k -wise independent and uniformly distributed in Ω . This is summarized in the following corollary.

COROLLARY 1.1 Let $n, d, k \in \mathbb{N}$ and $\Omega := \{0, \dots, d-1\}$. Uniformly distributed Ω -valued and k -wise independent random variables X_1, \dots, X_n can be constructed from

$$l := O(k \log d \cdot \log(n \log d)) \quad (1.11)$$

binary random variables Y_1, \dots, Y_l with a sample space of size

$$2^l = O((1 + n \log d)^{\lceil \frac{k \log d}{2} \rceil}).$$

Proof We show that the random variables X_1, \dots, X_n defined by (1.10) are k -wise independent and uniformly distributed. Let $I \subseteq [n]$ with $|I| \leq k$ and $x \in \{0, \dots, d-1\}^n$. We only need the values x_i for $i \in I$, however having x as an n dimensional vector allows easier notation of the binary representation of its entries, see (1.1). We have

$$\begin{aligned} & \mathbb{P}[\forall i \in I : X_i = x_i] \\ &= \mathbb{P}[\forall i \in I \forall j \in [\log d] : \mathbf{X}_{j+(i-1)\log d} = \mathbf{x}_{j+(i-1)\log d}] \\ &= \prod_{i \in I} \prod_{j \in [\log d]} \mathbb{P}[\mathbf{X}_{j+(i-1)\log d} = \mathbf{x}_{j+(i-1)\log d}] \quad \text{by } (k \log d)\text{-wise indep.} \\ &= \prod_{i \in I} \mathbb{P}[\forall j \in [\log d] : \mathbf{X}_{j+(i-1)\log d} = \mathbf{x}_{j+(i-1)\log d}] \quad \text{(} k \log d\text{)-wise indep.} \\ & \quad \text{implies } (\log d)\text{-wise indep.} \\ &= \prod_{i \in I} \mathbb{P}[X_i = x_i]. \end{aligned}$$

²We slightly abuse notation here, see (1.1).

To see that X_1, \dots, X_n are uniformly distributed, fix $i \in [n]$ and let $a \in \{0, \dots, d-1\}$. We see easily that

$$\begin{aligned} \mathbb{P}[X_i = a] &= \mathbb{P}\left[\sum_{j=(i-1)\log(d)+1}^{i\log d} \mathbf{X}_j 2^{j-1-(i-1)\log d} = a\right] \\ &= \mathbb{P}[\forall j \in [\log d] : \mathbf{X}_{j+(i-1)\log d} = \mathbf{a}_j] \\ &= \prod_{j \in [\log d]} \mathbb{P}[\mathbf{X}_{j+(i-1)\log d} = \mathbf{a}_j] && (k \log d)\text{-wise indep.} \\ & && \text{implies } (\log d)\text{-wise indep.} \\ &= \prod_{j \in [\log d]} \frac{1}{2} = \frac{1}{2^{\log d}} = \frac{1}{d}. \quad \blacksquare \end{aligned}$$

In some applications, a uniform distribution is not enough. Instead, one is given probabilities p_1, \dots, p_n and the task is to construct random variables X_1, \dots, X_n such that $\mathbb{P}[X_j = 1] = p_j$ for all $j \in [n]$. This is, in fact, approximately possible.

THEOREM 1.8 (Luby [Lub86])

Given probabilities p_1, \dots, p_n one can construct 2-wise independent 0/1 random variables X_1, \dots, X_n such that

$$\mathbb{P}[X_j = 1] = \frac{\lfloor p_j \cdot q \rfloor}{q} \quad \text{for all } j \in [n].$$

for a prime number $n \leq q \leq 2n$. The sample space has size $q^2 = O(n^2)$.

1.2.3 Almost k -Wise Independence.

Almost k -wise independence is an approximate version of equation (1.4). For uniformly distributed 0/1 random variables it reads as follows:

DEFINITION 1.2 (Almost k -Wise Independence)

Let $\varepsilon, \delta > 0$ and $k \in [n]$, $n \in \mathbb{N}$.

- (i). X_1, \dots, X_n are (ε, k) -independent, if for any $r \in [k]$, $\alpha \in \{0, 1\}^r$ and any choice of indices $1 \leq i_1 < i_2 < \dots < i_r \leq n$, we have

$$|\mathbb{P}[(X_{i_1}, \dots, X_{i_r}) = \alpha] - 2^{-r}| \leq \varepsilon.$$

- (ii). X_1, \dots, X_n are δ -away from k -wise independence, if for any $r \in [k]$ and any choice of indices $1 \leq i_1 < i_2 < \dots < i_r \leq n$, we have

$$\sum_{\alpha \in \{0,1\}^r} |\mathbb{P}[(X_{i_1}, \dots, X_{i_r}) = \alpha] - 2^{-r}| \leq \delta.$$

(ε, k) -independence measures the deviation from the uniform distribution in the maximum norm, while Definition 1.2 (ii) describes statistical closeness to the uniform distribution in the L^1 -norm.

(ε, k) -independent random variables are at most $2^k\varepsilon$ -away from k -wise independence, whereas if they are δ -away from k -wise independence, they are (δ, k) -independent. For a set $S \subseteq [n]$ let $X_S := \sum_{i \in S} X_i$ and put $X := \sum_{i=1}^n X_i$. The number $X_S \bmod 2$ is 0 iff X_S is even and 1 else. It can be viewed as the parity of S . If the X_1, \dots, X_n are independent, and $\mathbb{P}[X_i = 1] = \mathbb{P}[X_i = 0]$ for all $i = 1, \dots, n$, then

$$\mathbb{P}[X_S \bmod 2 = 1] = \mathbb{P}[X_S \bmod 2 = 0]. \tag{1.12}$$

An approximative version of (1.12) leads to the concept of ε -biased random variables (see Vazirani [Vaz86], Naor and Naor [NN93] and Peralta [Per90]). The bias of S is

$$\text{bias}(S) := |\mathbb{P}[X_S \bmod 2 = 0] - \mathbb{P}[X_S \bmod 2 = 1]|. \tag{1.13}$$

So for independent uniform random variables, $\text{bias}(S) = 0$ according to (1.12).

DEFINITION 1.3 (ε -Bias)

- (i). X_1, \dots, X_n are ε -biased, if $\text{bias}(S) \leq \varepsilon$ for all $S \subseteq [n]$.
- (ii). X_1, \dots, X_n are k -wise ε -biased, if $\text{bias}(S) \leq \varepsilon$ for all $S \subseteq [n]$, $|S| \leq k$.

k -wise ε -biased random variables and almost k -wise independence are closely related. To see this, let $D : \Omega \rightarrow [0, 1]$ be the probability distribution induced by the measure \mathbb{P} , so $D(\omega) := \mathbb{P}[(X_1 \dots, X_n) = \omega]$, and let U be the uniform distribution, i.e., $U(\omega) = 2^{-n}$ for all $\omega \in \Omega$. The variation distance $\|D - U\|$ between D and U is the L^1 -Norm of $D - U$,

$$\|D - U\| := \sum_{\omega \in \Omega} |D(\omega) - U(\omega)| = \sum_{\omega \in \Omega} |D(\omega) - 2^{-n}|.$$

$\|D - U\|$ is a measure for the distance of D from the uniform distribution. Let $D(S)$ be the restriction of D and $U(S)$ the restriction of U to a subset $S \subseteq [n]$.

DEFINITION 1.4 X_1, \dots, X_n are k -wise δ -dependent, if for all subsets $S \subseteq [n]$ with $|S| \leq k$,

$$\|D(S) - U(S)\| \leq \delta.$$

Note that if the X_1, \dots, X_n are k -wise δ -dependent, then they are δ -away from k -wise independence. Taking the Fourier transform $\widehat{D - U}$, Diaconis and Shahashahani [Dia88] proved

$$\|D - U\|^2 = \|\widehat{D - U}\|^2 \leq \sum_{S \subseteq [n]} \text{bias}_D(S).$$

This inequality immediately implies a corollary due to U. Vazirani (PhD thesis 1986 [Vaz86], see also the papers of Vazirani, Vazirani [VV84] and Chor et al. [CFG⁺85]).

COROLLARY 1.2 If X_1, \dots, X_n are δ -biased, then they are k -wise $2^{\frac{k}{2}}\delta$ -dependent and $(2^{\frac{k}{2}}\delta, k)$ -independent.

In conclusion, ε -biased random variables for small $\varepsilon > 0$ should behave as k -wise independent variables. For derandomization the hope is to replace k -wise independence by the weaker notion of almost k -wise independence and to obtain an even smaller sample space. Naor and Naor [NN93] proved that ε -biased random variables can be constructed with only “few” random bits:

THEOREM 1.9 (Naor and Naor 1993)

Let $\varepsilon > 0$ and $n \in \mathbb{N}$, $k \leq n$.

- (i). Uniformly distributed 0/1-valued random variables X_1, \dots, X_n which are ε -biased can be constructed using $O\left(\log n + \log \frac{1}{\varepsilon}\right)$ random bits. The size of the corresponding sample space is $2^{O\left(\log n + \log \frac{1}{\varepsilon}\right)} = \left(\frac{n}{\varepsilon}\right)^c$ for a constant $c > 0$.
- (ii). Uniformly distributed 0/1-valued random variables X_1, \dots, X_n which are k -wise ε -dependent can be constructed using $O\left(\log \log n + k + \log \frac{1}{\varepsilon}\right)$ random bits. The size of the corresponding sample space is $\left(\frac{2^k \log n}{\varepsilon}\right)^{O(1)}$.

The constant c in Theorem 1.9 (i) depends on the expansion rate of an expander graph and is slightly larger than 4 for the asymptotically optimal expanders of Lubotzky, Phillips, and Sarnak [LPS88]. For polynomially large $\frac{1}{\varepsilon}$, i.e., $\frac{1}{\varepsilon} = O(\text{poly}(n))$, the size of the sample space is polynomial in n . In applications of the method of almost k -wise independence one would like to have k large and ε small while the size of the sample space should remain polynomial. The bound $\left(\frac{2^k \log n}{\varepsilon}\right)^{O(1)}$ limits the growth of k , in fact for $\frac{1}{\varepsilon} = O(\text{poly}(n))$, k must be about $O\left(\log \frac{1}{\varepsilon}\right) = O(\log n)$. Seminal papers managed to reduce the size of sample spaces corresponding to ε -biased random variables. (Azar, Motwani, Naor [AMN98], Even, Goldreich, Luby, Nisan, Veličković [EGL⁺98] and Alon, Goldreich, Håstad, Peralta [AGHP92]). Alon, Goldreich, Håstad, and Peralta achieved a size of roughly $\left(\frac{n}{\varepsilon}\right)^2$ for ε -biased random variables. In particular, they generate n random variables which are ε -away from k -wise independence with only $(2 + o(1))\left(\log \log n + \frac{k}{2} + \log k + \log \frac{1}{\varepsilon}\right)$ random bits. This beats the bound of Naor & Naor as long as $\varepsilon < \frac{1}{k \log n}$. There are two critical aspects in all of these results.

1. The random variables X_1, \dots, X_n are 0/1-valued and uniform, i.e., $\mathbb{P}[X_i = 1] = \mathbb{P}[X_i = 0] = \frac{1}{2}$ for all i . In applications, this may not be the case.
2. The usual strategy for derandomization using k -wise independence is to construct a small sample space *a priori* and to simulate a randomized algorithm for a specific problem in such a space, if possible. Thus the sample space is chosen independently of the problem!

Schulman [Sch92] gave an interesting, different approach for problem 2: He observed that for concrete problems only certain sets of d random variables, so called d -neighborhoods, among the n random variables need to be independent. Thus the choice of the magnitude of independence is *driven by the problem*. Koller and Megiddo [KM94] further developed Schulman's approach covering also multivalued, non-uniformly distributed random variables. Koller and Megiddo showed that the sample space of k -wise independent random variables X_1, \dots, X_n with non-uniform probabilities $\mathbb{P}[X_i = 1] = p_i$, $i \in [n]$ correspond to a sample space of size at most $m(n, k) = \binom{n}{k} + \binom{n}{k-1} + \dots + \binom{n}{0}$. Karloff and Mansour [KM97] showed

the existence of p_1, \dots, p_n such that the size of any k -wise independent 0/1 probability space over p_1, \dots, p_n is at least $m(n, k)$. An interesting connection between small hitting sets for combinatorial rectangles in high dimension and the construction of small sample spaces for general multivalued random variables is given in the paper of Linial, Luby, Saks, and Zuckerman [LLSZ97].

A different approach to the construction of small sample spaces than k -wise independence was taken by Karger and Koller [KK97]. In Section 1.4 we describe their technique. It will be used later in Srinivasan's parallel algorithms for packing problems (Section 1.5.2).

Bibliography and Remarks. In probability theory, k -wise independence was already used by Joffe [Jof74]. This concept was adapted to combinatorics and the design of algorithms in the years 1985/86, where the fundamental papers of Karp and Widgerson [KW85] and Luby [Lub86] on 2-wise independence, the work of Alon, Babai, and Itai on k -wise independence [ABI86] and the lower bound proof for the size of k -wise independent sample space by Chor et al. [CFG⁺85] were published. Derandomization of space bounded computations is treated by Armoni [Arm98] and Saks [Sak96]. A survey on parallel derandomization techniques is given in the paper of Han [Han92].

In computational geometry, k -wise independence was applied by Berger et al. [BRS94] and later by Goodrich [Goo93b, Goo93a, Goo96], and by Amato et al. [AGR94]. For the parallelization of derandomized geometric algorithms Mulmuley [Mul96] extended earlier work of Karloff and Raghavan [KR93] on limiting random resources using bounded independence distributions and demonstrated that a polylogarithmic number of random bits suffices for guaranteeing a good expected performance of many randomized incremental algorithms.

The method of k -wise independence has been applied successfully to reduce or remove randomness from probabilistic constructions and from algorithms in various fields, like hashing, pseudo-random generators, one-way functions, circuit and communication complexity and Boolean matrix multiplication. Many of these aspects are treated in the lecture notes of Luby and Widgerson [LW95]. Applications to learning algorithms are discussed in [NSS95]. Sitharam and Straney [SS01] applied derandomization to learn Boolean functions.

Yao [Yao82] introduced the concept of a pseudo-random generator, see also [Nie92]. An excellent book on pseudorandom generators and applications to cryptography (and other areas) are the lectures of Luby [Lub96]. Blum and Micali [BM84] showed that the problem of constructing a pseudo-random generator in based on the concept of computational indistinguishability introduced by Goldwasser and Micali [GM84]. Pseudo-random generators have been designed by Karp, Pippenger, Sipser [KPS88], Ajtai, Komlos, Szemerédi [AKS87], Chor and Goldreich [CG89], Nisan [Nis91, Nis92], Hastad, Impagliazzo, Zuckerman [IZ90], Impagliazzo, Levin, Luby [HILL99], Sipser [Sip88], Hastad, Impagliazzo, Levin, Luby [HILL97] and Impagliazzo, Wigderson [IW97] and Andreev, Clementi, and Rolim [ACR98], to name some of the researchers.

Santha [San87] showed how to sample with a small number of random points. Feder, Kushilevitz, and Naor [FKN95] applied almost k -wise independence to amortize the communication complexity. Application to hashing and Boolean matrix multiplication were discussed by Alon, Galil, Margalit, Naor [AGMN92], Alon, Galil [AG97], Alon, Naor [AN96] and Naor, Naor [NN93]. Approximation of DNF via derandomization was carried out by Luby and Velickovic [LV96] and approximate counting of depth-2 circuits is shown by Luby, Wigderson, and Velickovic [LWV93].

Blum et al. [BEG⁺94] applied hash functions to authenticate memories. An interesting technique for deterministic construction of small sample spaces for general multivalued random variables via the construction of point sequences with a discrepancy or ε -net property was introduced by Linial, Luby, Saks, and Zuckerman [LLSZ97].

1.3 Parallel Algorithms using k -Wise Independence

In this section we show the algorithmic impact of k -wise independence, in particular its power for parallelizing algorithms.

1.3.1 Functions over \mathbb{Z}_2^k

For convenience, we formulate the derandomization problem in the following setting. Let \mathbb{Z}_2^k denote the k -dimensional vector space over $GF(2)$. We describe a class of functions $F : \mathbb{Z}_2^n \rightarrow \mathbb{Q}$ for which the derandomization problem can be solved in parallel. Let $N = N(n, m, k)$ be an integer valued function. At the moment we do not assume that N is polynomially bounded in m and n . Let F be of the form:

$$F(x_1, \dots, x_n) = \sum_{i=1}^N c_i f_i(x_{i_1}, \dots, x_{i_k}) \quad (1.14)$$

with $c_i \in \mathbb{Q}$ and $f_i(x_{i_1}, \dots, x_{i_k}) = (-1)^{\sum_{j=1}^k x_{i_j}}$. The so given function $[k] \rightarrow [n]$, $j \mapsto i_j$ will occur in the following.

THEOREM 1.10 *Let $k, n \in \mathbb{N}$, $k \leq n$, and X_1, \dots, X_n be k -wise independent uniformly distributed 0/1 random variables as in Theorem 1.7. Let $F : \mathbb{Z}_2^n \rightarrow \mathbb{Q}$ be a function as in (1.14). With $O(N)$ parallel processors we can construct $x_0 \in \{0, 1\}^n$ in time $O(k \log n \log N)$ such that $F(x_0) \leq \mathbb{E}[F(X_1, \dots, X_n)]$. The same result holds for the construction of $x'_0 \in \{0, 1\}^n$ such that $F(x'_0) \geq \mathbb{E}[F(X_1, \dots, X_n)]$.*

Proof The proof is based on [BR91]. The random variables X_1, \dots, X_n by definition have the form $X_i = (BY)_i$, where $Y = (Y_1, \dots, Y_l)$, the Y_1, \dots, Y_l are uniformly distributed 0/1 random variables, $l = 1 + \frac{k-1}{2} \log(n+1)$, B is the (expanded version of the) matrix in (1.5), which is of dimension $(n \times l)$. Therefore it suffices to give assignments for the Y_1, \dots, Y_l . The conditional probability method then goes as follows:

Suppose that for some $t \in [l]$ we have computed the assignments

$$Y_1 = y_1, \dots, Y_{t-1} = y_{t-1}.$$

Then we choose for Y_t the value $y_t \in \{0, 1\}$ which minimizes the function

$$w \mapsto \mathbb{E}[F(X_1, \dots, X_n) \mid Y_1 = y_1, \dots, Y_{t-1} = y_{t-1}, Y_t = w]. \quad (1.15)$$

Obviously, after l steps we get $(Y_1, \dots, Y_l) = y$ for some $y \in \{0, 1\}^l$. Thus $x_0 := By$ is a solution according to the correctness of the algorithm CONDEXP in Section 1.2.1. We are done, if the conditional expectations

$$\mathbb{E}[F(X_1, \dots, X_n) \mid Y_1 = y_1, \dots, Y_{t-1} = y_{t-1}, Y_t = y_t]$$

can be computed within the claimed processor and time bounds for every t . Fix now $t \in [l]$. By linearity of expectation, it is sufficient to compute for each $i \in [N]$

$$\mathbb{E}[f_i(X_{i_1}, \dots, X_{i_k}) \mid Y_1 = y_1, \dots, Y_{t-1} = y_{t-1}, Y_t = y_t]. \quad (1.16)$$

Let B_i be the i th row of B and put $b := \sum_{j=1}^k B_{i_j}$, computed over $GF(2)$. Note that b is a vector – a sum of rows of matrix B . Because in the exponent of f_i , only the parity is

relevant, we have that $f_i(X_{i_1}, \dots, X_{i_k}) = (-1)^{b \cdot Y}$. Let s be the last position in the vector b which contains a 1. To shorten notation put $\vec{Y}_t = (Y_1, \dots, Y_t)$ and $\vec{y}_t = (y_1, \dots, y_t)$. Now

$$\mathbb{E} [(-1)^{b \cdot Y} \mid \vec{Y}_t = \vec{y}_t] = \begin{cases} (-1)^{\sum_{j=1}^s b_j y_j} & \text{if } t \geq s \\ 0 & \text{if } t < s. \end{cases} \quad (1.17)$$

(1.17) follows from the following observations. If $t < s$ then

$$b \cdot Y = \underbrace{\sum_{j=1}^t b_j y_j}_{=: a_1} + \underbrace{\sum_{j=t+1}^s b_j Y_j}_{=: a_2}$$

Hence $\mathbb{E} [(-1)^{a_1+a_2}] = (-1)^{a_1} \mathbb{E} [(-1)^{a_2}] = (-1)^{a_1} \cdot 0$, because the Y_{t+1}, \dots, Y_s are independent and uniformly distributed. If $t \geq s$, then obviously $b \cdot Y = \sum_{j=1}^s b_j y_j$, due to condition $\vec{Y}_t = \vec{y}_t$. Hence, for fix i and t we can compute $\mathbb{E} [f_i \mid \vec{Y}_t = \vec{y}_t]$ in constant time. The total running time is computed as follows: we assign to each $f_i(X_{i_1}, \dots, X_{i_k})$ one processor, so we have N processors in total. In the t th step, $t \in [l]$, we can compute the sum $\sum_{i=1}^N \mathbb{E} [f_i \mid \vec{Y}_t = \vec{y}_t]$ in $O(\log N)$ time using N parallel processors. Summing up over the l steps we get a total running time of $O(l \log N)$. ■

Theorem 1.10 gives a parallel derandomized algorithm, provided $N(n, m, k)$ is polynomially bounded in n and m . Its extension to uniformly distributed multivalued variables is presented in the next section.

1.3.2 Multivalued Functions

We will again make use of k -wise independent random variables X_1, \dots, X_n constructed via the matrix B , but this time with values in $\Omega := \{0, \dots, d-1\}$. For the conditional probability method to apply, we need to compute conditional probabilities for subsets of the X_1, \dots, X_n . We first show that this can be done efficiently.

PROPOSITION 1.3 Let Y_1, \dots, Y_l , the matrix B , and X_1, \dots, X_n be as in Section 1.2.2. For every $t \in [l]$ denote $\vec{Y}_t := (Y_1, \dots, Y_t)$. Now fix $t \in [l]$ and a vector $y \in \{0, 1\}^t$, as well as $s \in [n]$ and a vector $x \in \{0, 1\}^{[s]}$, $I \subseteq [n]$, and set $\hat{B} := B_I$. Let \hat{B}_1 be the first t columns of \hat{B} and \hat{B}_2 the last $l-t$ ones. Then

$$\mathbb{P}[X_I = x \mid \vec{Y}_t = y] = \begin{cases} 2^{-\text{rank } \hat{B}_2} & \text{if } x - \hat{B}_1 y \in \text{range } \hat{B}_2 \\ 0 & \text{else.} \end{cases}$$

This value can be computed using $O(n)$ parallel processors in $O(l^3 \log n)$ time. A similar result extends to $\{0, \dots, d-1\}$ -valued random variables, d a power of 2, with $l = O(k \log d \cdot \log(n \log d))$.

Proof Because Y_1, \dots, Y_l are independent, we have

$$\begin{aligned} \mathbb{P}[X_I = x \mid \vec{Y}_t = y] &= \mathbb{P}[\hat{B}Y = x \mid \vec{Y}_t = y] \\ &= \mathbb{P}[\hat{B}_1 y + \hat{B}_2(Y_{t+1}, \dots, Y_l) = x] = \mathbb{P}[\hat{B}_2(Y_{t+1}, \dots, Y_l) = x - \hat{B}_1 y]. \end{aligned}$$

We have $l - t = \dim \ker \hat{B}_2 + \dim \text{range } \hat{B}_2$, hence $\dim \ker \hat{B}_2 = l - t - \dim \text{range } \hat{B}_2$. The case of $x \notin \text{range } \hat{B}_2$ is clear. If $x \in \text{range } \hat{B}_2$, then the set of all preimages is a subspace of dimension $\dim \ker \hat{B}_2$. Because the field has two elements, there are $2^{\dim \ker \hat{B}_2}$ preimages of x . On the other hand, there are 2^{l-t} possible settings for the Y_{t+1}, \dots, Y_l , and so the claim follows:

$$\mathbb{P}[\hat{B}_2(Y_{t+1}, \dots, Y_l) = x - \hat{B}_1 y] = \frac{2^{\dim \ker \hat{B}_2}}{2^{l-t}} = 2^{l-t - \dim \text{range } \hat{B}_2 - (l-t)} = 2^{-\dim \text{range } \hat{B}_2}.$$

It is possible to compute $\dim \text{range } \hat{B}_2 = \text{rank } \hat{B}_2$ within the stated processor and time bounds by a simple divide-and-conquer application of Gauss elimination. The same can be used for testing $x - \hat{B}_1 y \in \text{range } \hat{B}_2$.

For the extension to $\{0, \dots, d-1\}$ -valued random variables, use Corollary 1.1. \blacksquare

Now we consider a more general class of functions than in the previous section. They are of a type later used in packing integer programs, Section 1.5.1. We give a \mathcal{NC} algorithm for finding points below or above the expectation. Let $d, k, m, n \in \mathbb{N}$ and for each $(i, z, j) \in [m] \times \Omega \times [n]$ let $g_{izj} : \mathbb{R} \rightarrow \mathbb{R}$ such that for every $r \in \mathbb{R}$, the value $g_{izj}(r)$ can be computed on a single processor in constant time. For each $(i, z) \in [m] \times \Omega$ define the functions

$$f_{iz} : \Omega^n \rightarrow \mathbb{R}, (x_1, \dots, x_n) \mapsto \sum_{j=1}^n g_{izj}(x_j),$$

and finally put them together

$$F : \Omega^n \rightarrow \mathbb{R}, (x_1, \dots, x_n) \mapsto \sum_{(i,z) \in [m] \times \Omega} (f_{iz}(x_1, \dots, x_n))^k.$$

THEOREM 1.11 *Let $d, k, m, n \in \mathbb{N}$, $d \geq 2$ a power of 2, and $N := mdn^k$. Let X_1, \dots, X_n be k -wise independent random variables with values in $\Omega = \{0, \dots, d-1\}$ as defined in Corollary 1.1 and let the g_{izj} , f_{iz} , and F as above. Then with $O(\max\{nd^k, N\})$ parallel processors we can construct $x_1, \dots, x_n \in \Omega$ and $y_1, \dots, y_n \in \Omega$ in time*

$$O(\log(n \log d) \cdot (k^4 \log^4 d \log^3(n \log d) \log n + k \log d \log N))$$

such that $F(x_1, \dots, x_n) \geq \mathbb{E}[F(X_1, \dots, X_n)]$, resp. $F(y_1, \dots, y_n) \leq \mathbb{E}[F(X_1, \dots, X_n)]$.

Proof We prove the first assertion as the second follows from it. Fix $(i, z) \in [m] \times \Omega$ for a moment. We write the k th power of f_{iz} in a different way. For every multiindex $\alpha \in [n]^k$ and $(x_1, \dots, x_k) \in \Omega^k$ define

$$f_{\alpha}^{(iz)}(x_1, \dots, x_k) := \prod_{j=1}^k g_{iz\alpha_j}(x_j),$$

and observe that this function depends on k variables only. It can be evaluated with k processors in $O(\log k)$ time. It is easy to see that for all $x = (x_1, \dots, x_n) \in \Omega^n$ we have, recalling that $x_{\alpha} = (x_{\alpha_1}, \dots, x_{\alpha_k})$, $f_{iz}(x_1, \dots, x_n) = \sum_{\alpha \in [n]^k} f_{\alpha}^{(iz)}(x_{\alpha})$, and so

$$F(x_1, \dots, x_n) = \sum_{(i,z) \in [m] \times \Omega} \sum_{\alpha \in [n]^k} f_{\alpha}^{(iz)}(x_{\alpha}).$$

By construction of the X_1, \dots, X_n , it suffices to give an assignment to 0/1 random variables Y_1, \dots, Y_l with $l = O(k \log d \cdot \log(n \log d))$, see Corollary 1.1 and the discussion preceding it. The conditional probability method then goes as follows: Suppose that for some $t \in [l]$ we have computed the the values

$$Y_1 = y_1, \dots, Y_{t-1} = y_{t-1},$$

then choose for Y_t a value $y_t \in \Omega$ which maximizes the function

$$w \mapsto \mathbb{E} [F(X_1, \dots, X_n) \mid y_1, \dots, y_{t-1}, Y_t = w]. \quad (1.18)$$

After l steps we have $Y = y$ with some $y \in \{0, 1\}^l$ and the solution can be computed from it by (1.10) and (1.8). Let $\vec{Y}_t = (Y_1, \dots, Y_t)$ and $\vec{y}_t = (y_1, \dots, y_t)$. We are done, if we can compute the conditional expectations

$$\mathbb{E} [F(X_1, \dots, X_n) \mid \vec{Y}_t = \vec{y}_t]$$

within the claimed processor and time bound. By linearity of expectation, it is sufficient to compute for each triple $(i, z, \alpha) \in [m] \times \Omega \times [n]^k$ the conditional expectation

$$\mathbb{E} [f_\alpha^{(iz)}(X_\alpha) \mid \vec{Y}_t = \vec{y}_t].$$

To this end, write

$$\begin{aligned} \mathbb{E} [f_\alpha^{(iz)}(X_\alpha) \mid \vec{Y}_t = \vec{y}_t] &= \sum_{x \in \Omega^k} f_\alpha^{(iz)}(x) \mathbb{P} [X_\alpha = x \mid \vec{Y}_t = \vec{y}_t] \\ &= \sum_{x \in \Omega^k} f_\alpha^{(iz)}(x) \mathbb{P} [\mathbf{X}_\alpha = \mathbf{x} \mid \vec{Y}_t = \vec{y}_t]. \end{aligned}$$

According to Proposition 1.3, for a fixed $x \in \Omega^k$, the conditional probability can be computed using $O(n)$ processors in time $O(l^3)$. Since we have d^k such vectors x , and each $f_\alpha^{(iz)}(x)$ uses k processors and $O(\log k)$ time, we can compute $\mathbb{E} [f_\alpha^{(iz)}(X_\alpha) \mid \vec{Y}_t = \vec{y}_t]$ for every $t \in [l]$ and $(i, z, \alpha) \in [m] \times \Omega \times [n]^k$ with $O(d^k(k+n)) = O(nd^k)$ parallel processors (recall $d \geq 2$) in $O(l^3 \log n + \log d^k + \log k) = O(l^3 \log n + k \log d)$ time. Then we compute for every y_t , $t \in [l]$, the expectation

$$\mathbb{E} [F(X_1, \dots, X_n) \mid \vec{Y}_t = \vec{y}_t]$$

in $O(\log N)$ time using $O(N)$ parallel processors. Finally, a $y_t \in \Omega$ which maximizes (1.18) can be computed finding the maximum of the d conditional expectations in $O(\log d)$ time with $O(d)$ processors. The maximum number of processors used is $O(\max\{nd^k, N\})$ and the total running time over all l steps is

$$\begin{aligned} &O(l \cdot (l^3 \log n + k \log d + \log N + \log d)) \\ &= O(\log(n \log d) \cdot (k^4 \log^4 d \log^3(n \log d) \log n + k^2 \log^2 d + k \log d \log N + k \log^2 d)) \\ &= O(\log(n \log d) \cdot (k^4 \log^4 d \log^3(n \log d) \log n + k \log d \log N)). \quad \blacksquare \end{aligned}$$

1.3.3 Maximal Independent Sets in Graphs

We start with the celebrated parallel algorithm of Luby computing a maximal independent set in graphs. Historically, it was the first striking application of limited dependence to the design of a \mathcal{NC} algorithm.

We consider a graph $G = (V, E)$ with $V = [n]$ and $|E| = m$. A subset I of vertices is called *independent*, if the induced subgraph on I contains no edges. A *maximal independent set* (MIS) has the additional property that adding an arbitrary vertex destroys independence. The following simple algorithm computes the lexicographically first maximal independent set in linear time.

Algorithm 3: LFMIS

Input: A Graph $G = (V, E)$.
Output: A maximal independent set $I \subseteq V$.
 $I \leftarrow \emptyset$;
for $i \leftarrow 1$ **to** n **do**
 $I \leftarrow I \cup \{i\}$;
 $V \leftarrow V \setminus (\{i\} \cup N(i))$;
end

Cook [Coo85] proved that deciding whether a vertex belongs to the lexicographically first maximal independent set is logspace-complete in \mathcal{P} . Thus there is no hope to parallelize LFMIS. Yet, the MIS problem is not inherently sequential. We will present a parallel randomized algorithm of Luby [Lub86] with expected running-time $O(\log^2 n)$ on $O(n + m)$ processors. Removing the randomness by means of searching a small sample space will reveal that $O(n^2 m)$ processors can solve MIS in $O(\log^2 n)$ time. The key idea is to successively add to the MIS under construction not just a single vertex i but an independent set S that is computed in parallel.

Algorithm 4: PARALLELMIS (High Level)

Input: A Graph $G = (V, E)$.
Output: A maximal independent set $I \subseteq V$.
 $I \leftarrow \emptyset$;
while $V \neq \emptyset$ **do**
 select an independent set $S \subseteq V$;
 $I \leftarrow I \cup S$;
 $V \leftarrow V \setminus (S \cup N(S))$;
end

The high level description makes clear that PARALLELMIS terminates with a maximal independent set I . The parallelization is hidden in the selection of S . Luby analyzed the following randomized selection procedure.

Algorithm 5: PARALLELMIS (Details of the selection step)

Input: A Graph $G = (V, E)$.
Output: An independent set $S \subseteq V$.

```

1 in parallel forall  $i \in V$  do
2   if  $\deg(i) = 0$  then mark  $i$ ;
3   else mark  $i$  with probability  $\frac{1}{2 \cdot \deg(i)}$ ;
4 end
5 in parallel forall  $\{i, j\} \in E$  do
6   if both  $i$  and  $j$  are marked then unmark the vertex with smaller degree;
7 end
8  $S \leftarrow \emptyset$ ;
9 in parallel forall  $i \in V$  do
10  if  $i$  is marked then  $S \leftarrow S \cup \{i\}$ ;
11 end

```

One can show that each iteration of PARALLELMIS can be implemented to run in $O(\log n)$ time on an EREW PRAM with $O(n + m)$ processors. We will prove that the algorithm terminates after an expected total number of $O(\log n)$ iterations, since a constant fraction of edges are expected to get removed in each update step in line 5 in Algorithm 4. We call a vertex $i \in V$ *good* if the degree of at least $\frac{1}{3}$ of its neighbors is at most $\deg(i)$ and *bad* otherwise. Now consider an arbitrary iteration.

LEMMA 1.1 Let $i \in V$ be a good vertex. The probability that $i \in N(S)$ is at least

- (a) $\frac{1 - \exp(-\frac{1}{6})}{2}$ if the random choices in line 3 in Algorithm 5 are completely independent,
- (b) $\frac{1}{24}$ if the random choices are only pairwise independent.

Proof Observe that if a vertex j is marked, then it is selected into S with probability at least $1 - \mathbb{P}[\exists \text{ marked } k \in N(j) \text{ with } \deg(k) \geq \deg(j)] \geq 1 - \sum_{k \in N(j)} \frac{1}{2 \deg(j)} = \frac{1}{2}$.

(a) Since the vertex i is good, it has at least $\frac{\deg(i)}{3}$ neighbors with degree at most $\deg(i)$. Due to complete independence, the probability that none of these is marked is at most $\left(1 - \frac{1}{2 \deg(i)}\right)^{\frac{\deg(i)}{3}} \leq \exp(-\frac{1}{6})$. Hence, with probability at least $\frac{1 - \exp(-\frac{1}{6})}{2}$ a neighbor of i is selected into S , which means that i is in $N(S)$.

(b) Let X_j be the event that $j \in N(i)$ is marked in line 3, and let Y_j be the event that $j \in N(i)$ is selected into S . Since $\mathbb{P}[Y_j] \geq \frac{\mathbb{P}[X_j]}{2}$ and $Y_j \subseteq X_j$ we have by the principle of inclusion-exclusion that the probability for $i \in N(S)$ is

$$\begin{aligned}
\mathbb{P}\left[\bigcup_{j \in N(i)} Y_j\right] &\geq \sum_{j \in N(i)} \mathbb{P}[Y_j] - \sum_{j \neq k \in N(i)} \mathbb{P}[Y_j \cap Y_k] \\
&\geq \frac{1}{2} \sum_{j \in N(i)} \mathbb{P}[X_j] - \sum_{j \neq k \in N(i)} \mathbb{P}[X_j \cap X_k] \\
&\geq \frac{1}{2} \left(\sum_{j \in N(i)} \mathbb{P}[X_j] - \left(\sum_{j \in N(i)} \mathbb{P}[X_j] \right)^2 \right)
\end{aligned}$$

$$= \frac{1}{2} \left(\sum_{j \in N(i)} \mathbb{P}[X_j] \right) \left(1 - \sum_{j \in N(i)} \mathbb{P}[X_j] \right).$$

Since i is a good vertex,

$$\sum_{j \in N(i)} \mathbb{P}[X_j] \geq \frac{\deg(i)}{3} \cdot \frac{1}{2 \deg(i)} = \frac{1}{6},$$

and assuming that $\sum_{j \in N(i)} \mathbb{P}[X_j] \leq \frac{1}{2}$ ³ we get $\mathbb{P}[\bigcup_{j \in N(i)} Y_j] \geq \frac{1}{24}$ as claimed. ■

We call an edge *good*, if at least one of its endpoints is good.

LEMMA 1.2 In each iteration at least half of the edges are good.

Proof We direct the edges of the graph such that $(i, j) \in E$ implies $\deg(i) \leq \deg(j)$. Let V_g and V_b denote the sets of good and bad vertices, respectively, and let $E(X, Y)$ abbreviate the cardinality of the set of edges from $E \cap (X \times Y)$. Let $\deg^-(i)$ be the in-degree of a vertex i , i.e., the number of edges incident with i and pointing to i . Let $\deg^+(i)$ be the out-degree of a vertex i , i.e., the number of edges incident with i and pointing away from i . Using the fact that for every bad vertex i , $2 \deg^-(i) \leq \deg^+(i)$ we get that

$$\begin{aligned} 2E(V_b, V_b) + E(V_b, V_g) + E(V_g, V_b) &= \sum_{i \in V_b} (\deg^+(i) + \deg^-(i)) \\ &\leq 3 \sum_{i \in V_b} (\deg^+(i) - \deg^-(i)) = 3 \sum_{i \in V_g} (\deg^-(i) - \deg^+(i)) \\ &= 3(E(V_b, V_g) - E(V_g, V_b)) \leq 3(E(V_b, V_g) + E(V_g, V_b)). \end{aligned}$$

This gives $E(V_b, V_b) \leq E(V_b, V_g) + E(V_g, V_b)$, implying that the number of bad edges is at most the number of good edges, so that at least half of the edges are good. ■

Since an endpoint of a good edge is in $N(S)$ with probability at least $\frac{1}{24}$, we can conclude that a fraction of at least $\frac{1}{48}$ edges are expected to be deleted in each iteration. Hence the algorithm terminates after $O(\log n)$ iterations. Since only pairwise independence is needed, we can derandomize the algorithm by exhaustively searching a sample space of size $O(n^2)$, see Theorem 1.8. Thus we get a deterministic algorithm for solving MIS in time $O(\log^2 n)$ with $O(n^2 m)$ processors. Goldberg and Spencer [GS89a] have shown that the number of processors can be reduced to $O\left(\frac{n+m}{\log n}\right)$ on cost of an increased running-time of $O(\log^3 n)$.

Bibliography and Remarks. Karp, Upfal, and Wigderson [KUW88] developed a randomized algorithm for finding a maximal independent set in an independence system. Their algorithm can be adapted to compute a MIS in a general hypergraph in time $O(\sqrt{n}(\log n + \log m))$ on $O(m \cdot n)$ processors. Faster parallel algorithms have been described by Goldberg and T. Spencer [GS89b], [GS89a]. Dahlhaus and Karpiński [DK89], and, independently, Kelsen [DKK92a] have given efficient \mathcal{NC} algorithms for computing MIS in 3-uniform hypergraphs (journal version in

³Otherwise can choose an appropriate subset of neighbors s.t. $\sum \mathbb{P}[X_j] \approx \frac{1}{2}$ and get $\mathbb{P}[\bigcup Y_j] \approx \frac{1}{8}$.

[DKK92b]). Beame and Luby [BL90] studied the MIS problem for hypergraphs where the size of each edge is bounded by a fixed absolute constant c . They presented a generalized version of the randomized PARALLEL MIS-algorithm running on $O(n+c)$ processors with running time polynomial in $\log(n+c)$, as verified by Kelsen [Kel92]. They also gave a similar algorithm for the general case of k -uniform hypergraphs, and on $O(n \cdot m)$ processors conjectured a running-time polynomial in $\log(n+m)$. Luczak and Szymańska [LS97] observed that a linear hypergraph, i.e., a hypergraph in which each pair of edges has at most one vertex in common, contains a large subhypergraph without vertices of large degree. They proved that the randomized algorithm of Beame and Luby finds a maximal independent set in a linear hypergraph in polylogarithmic expected running-time if a preprocessing step is added to make the algorithm perform on such an “equitable” subhypergraph. A derandomized version of their result has been given in the PhD thesis of Edita Szymańska [Szy98].

1.3.4 Low Discrepancy Colorings

One root for derandomization certainly is the application of the probabilistic method to combinatorial discrepancy theory. Combinatorial discrepancy theory deals with the problem of partitioning the vertices of a hypergraph such that all hyperedges are split into about equal parts by the partition classes. Discrepancy measures the deviation of an optimal partition from an ideal one, that is one where all edges contain the same number of vertices in any partition class. An introduction to combinatorial discrepancy theory is given in the book of Alon, Spencer, and Erdős [ASE92] and the survey article of Beck and Sós [BS95]. Excellent sources covering many aspects (and proofs) of combinatorial, geometric as well as classical discrepancy theory are the books of Matoušek [Mat99] and Chazelle [Cha00] for the connection of derandomization and discrepancy theory. For an extension of 2-color discrepancy theory to c colors see Doerr and Srivastav [DS99, DS03].

Let $\mathcal{H} = (V, \mathcal{E})$ be a hypergraph, $\mathcal{E} = \{E_1, \dots, E_m\}$, and let $\chi : V \rightarrow \{-1, +1\}$ be a function. Identifying -1 and $+1$ with colors, say red and blue, χ is a 2-coloring of V . The sets $\chi^{-1}(-1)$ and $\chi^{-1}(+1)$ build the partition of V induced by χ . The imbalance of a hyperedge $E \in \mathcal{E}$ with respect to χ can be expressed by

$$\chi(E) := \sum_{x \in E} \chi(x). \quad (1.19)$$

The *discrepancy* of \mathcal{H} with respect to χ is

$$\text{disc}(\mathcal{H}, \chi) := \max_{E \in \mathcal{E}} |\chi(E)|, \quad (1.20)$$

and the *discrepancy* of \mathcal{H} is

$$\text{disc}(\mathcal{H}) := \min_{\chi: V \rightarrow \{-1, +1\}} \text{disc}(\mathcal{H}, \chi). \quad (1.21)$$

THEOREM 1.12 (Spencer 1987)

A 2-coloring of V with

$$\text{disc}(\chi) \leq \sqrt{2n \log(4m)}$$

can be constructed in $O(mn^2 \log(mn))$ time.

Proof Put $\alpha := \sqrt{2n \log(4m)}$. Let χ_1, \dots, χ_n be independent random variables defined by $\mathbb{P}[\chi_j = +1] = \mathbb{P}[\chi_j = -1] = \frac{1}{2}$ for all j . The vector $\chi = (\chi_1, \dots, \chi_n)$ is a random

2-coloring of V . For each $i \in [m]$ let F_i be the event “ $|\chi(E_i)| \leq \alpha$ ”. From Hoeffding’s inequality (Theorem 1.4) we infer $\mathbb{P}[F_i^c] \leq \frac{1}{2m}$, thus

$$\mathbb{P}\left[\bigcup_{i=1}^m F_i^c\right] \leq \frac{1}{2},$$

and a coloring with discrepancy at most α exists. With the algorithmic version of the Chernoff-Hoeffding inequality [SS96] it can be constructed in $O(mn^2 \log(mn))$ time. ■

The original proof of Spencer is done with the hyperbolic cosine algorithm instead of the algorithmic Chernoff-Hoeffding bound. It is known that the discrepancy bound $O(\sqrt{n \ln m})$ is not optimal. The celebrated “six-standard-deviation” theorem of Spencer [Spe85] proves the existence of a 2-coloring with discrepancy at most $6\sqrt{n}$ (for $m = n$). In general, the bound $O(\sqrt{n})$ for discrepancy is optimal up to a constant factor, since the discrepancy of the set system induced by an $(n \times n)$ Hadamard matrix is at least $\frac{1}{2}\sqrt{n}$ (see [ASE92]). It is a challenging open problem to give a randomized or deterministic polynomial-time algorithm that finds such a coloring. The main hinderance to the transformation of this existence result into an algorithm is the use of the pigeonhole principle in the proof.

We proceed to the parallelization of the discrepancy algorithm in Theorem 1.12. Berger, Rompel [BR91] and Motwani, Naor, Naor [MNN94] in 1989 presented the first parallel algorithm for this problem. The key for the success of their approach is the computation of k th moments of the discrepancy function over a k -wise independent distribution.

Define $\Delta_i := \max\{|E_i|, 3\}$ for each $i \in [m]$. Let $A = (a_{ij})$ be the hyperedge-vertex incidence matrix of \mathcal{H} . Let X_1, \dots, X_n be $-1/1$ random variables with $\mathbb{P}[X_j = +1] = \mathbb{P}[X_j = -1] = \frac{1}{2}$ and set

$$\psi_i := \sum_{j=1}^n a_{ij} X_j \quad \text{for each } i \in [m], \quad \text{and } \Delta_0 := \min_{i \in [m]} \Delta_i (\geq 3), \quad \Delta := \max_{i \in [m]} \Delta_i.$$

In the following, we will always assume that all hyperedges have cardinality at least 3 (and so $|E_i| = \Delta_i$ and each ψ_i is the sum of Δ_i random variables). The established bounds will also hold in the general case, however, because they are always at least 2, which is an upper bound on the discrepancy of hyperedges of cardinality 1 and 2 regardless of the coloring.

For all $i \in [m]$ let k_i be a non-negative integer and put

$$k := \max_{i \in [m]} k_i. \tag{1.22}$$

We assume that the X_1, \dots, X_n are k -wise independent. For every multiindex $\alpha \in [n]^{k_i}$, $\alpha = (\alpha_1, \dots, \alpha_{k_i})$, we define

$$a_{i\alpha} := \prod_{l=1}^{k_i} a_{i\alpha_l} \quad \text{and} \quad \phi_\alpha := \prod_{l=1}^{k_i} X_{\alpha_l}.$$

We consider the k_i th powers $\psi_i^{k_i}$, $i \in [m]$. Let $\mathbb{E}_{\text{indep}}[\psi_i^{k_i}]$ be the expectation under the assumption of (completely) independent random variables X_1, \dots, X_n .

LEMMA 1.3 $\mathbb{E}[\psi_i^{k_i}] \leq 2(k_i \Delta_i)^{\frac{k_i}{2}}$ for all $i \in [m]$.

Proof We have $\psi_i^{k_i} = \sum_{\alpha \in [n]^{k_i}} a_{i\alpha} \phi_\alpha$, hence $\mathbb{E}[\psi_i^{k_i}] = \mathbb{E}_{\text{indep}}[\psi_i^{k_i}]$, because there occur only $k_i \leq k$ random variables in each summand and the expectation is linear. So we may apply Theorem 1.4 to get

$$\begin{aligned} \mathbb{E}[\psi_i^{k_i}] &= \mathbb{E}_{\text{indep}}[\psi_i^{k_i}] = \int_0^\infty \mathbb{P}_{\text{indep}}[|\psi_i|^{k_i} \geq x] dx \\ &= \int_0^\infty \mathbb{P}_{\text{indep}}[|\psi_i| \geq x^{\frac{1}{k_i}}] dx \\ &\leq \int_0^\infty 2 \cdot \exp\left(-\frac{x^{\frac{2}{k_i}}}{2\Delta_i}\right) dx && \text{by Theorem 1.4} \\ &= 2\left(\frac{k_i}{2}\right)!(2\Delta_i)^{\frac{k_i}{2}} \\ &\leq 2(k_i\Delta_i)^{\frac{k_i}{2}}. \quad \blacksquare \end{aligned}$$

Let $0 < \varepsilon < 1$ be a fixed parameter. For each $i \in [m]$ let $\beta_i > \frac{1}{\varepsilon}$ be such that

$$k_i := \frac{\beta_i \log(2m)}{\log \Delta_i} \tag{1.23}$$

is the smallest even integer with $k_i \in \left\{ \left\lceil \frac{\log(2m)}{\varepsilon \log \Delta_i} \right\rceil, \left\lceil \frac{\log(2m)}{\varepsilon \log \Delta_i} \right\rceil + 1, \frac{\log(2m)}{\varepsilon \log \Delta_i} + 2 \right\}$. Define bounds for the discrepancies of each hyperedge E_i

$$\lambda_i := \Delta_i^{\frac{1}{2} + \varepsilon} \sqrt{\frac{\beta_i \log(2m)}{\log \Delta_i}}. \tag{1.24}$$

Note that because $\Delta_i \geq 3$, we have $\log \Delta_i > 1$.

LEMMA 1.4 For all $i \in [m]$ we have

$$\mathbb{P}\left[|\psi_i| \geq \Delta_i^{\frac{1}{2} + \varepsilon} \sqrt{\frac{\log(2m)}{\varepsilon \log \Delta_i}} + 2\right] \leq \mathbb{P}[|\psi_i| \geq \lambda_i] \leq \frac{\mathbb{E}[\psi_i^{k_i}]}{\lambda_i^{k_i}} < \frac{1}{m}.$$

Hence, with positive probability $|\psi_i| \leq \lambda_i \leq \Delta_i^{\frac{1}{2} + \varepsilon} \sqrt{\frac{\log(2m)}{\varepsilon \log \Delta_i}} + 2$ for all $i \in [m]$.

Proof We fix an arbitrary $i \in [m]$. The first inequality is trivial. The k_i th moment inequality [Fel67] gives

$$\begin{aligned} \mathbb{P}[|\psi_i| \geq \lambda_i] &\leq \frac{\mathbb{E}[|\psi_i|^{k_i}]}{\lambda_i^{k_i}} = \frac{\mathbb{E}[\psi_i^{k_i}]}{\lambda_i^{k_i}} \\ &\leq \frac{2(k_i\Delta_i)^{\frac{k_i}{2}}}{\lambda_i^{k_i}} && \text{by Lemma 1.3.} \end{aligned} \tag{1.25}$$

The term in (1.25) is

$$2\left(\frac{k_i\Delta_i}{\lambda_i^2}\right)^{\frac{k_i}{2}} = 2\left(\frac{1}{\Delta_i^{2\varepsilon}}\right)^{\frac{k_i}{2}}$$

$$\begin{aligned}
&= 2 \cdot 2^{-2\varepsilon \log \Delta_i \beta_i \log(2m) / (2 \log \Delta_i)} \\
&= 2 \cdot 2^{-\beta_i \varepsilon \log(2m)} \\
&< 2 \cdot 2^{-\log(2m)} = \frac{1}{m} \qquad \text{since } \beta_i > \frac{1}{\varepsilon}.
\end{aligned}$$

The final assertion now follows with the union-bound. \blacksquare

COROLLARY 1.3 If $k := \max_{i \in [m]} k_i \leq \log(2m)$, then for any $i \in [m]$, we have $\varepsilon \geq \frac{1}{\beta_i} \geq \frac{1}{\log \Delta_i}$, and with positive probability

$$|\chi(E_i)| = |\psi_i| \leq \Delta_i^{\frac{1}{2} + \varepsilon} \sqrt{\log(2m) + 2} \leq \Delta_i^{\frac{1}{2} + \varepsilon} \sqrt{\log(2m) + 2},$$

holds for all $i \in [m]$, hence $\text{disc}(\mathcal{H}, \chi) \leq \Delta_i^{\frac{1}{2} + \varepsilon} \sqrt{\log(2m) + 2}$.

Proof The assumption $k \leq \log(2m)$ gives $\beta_i \leq \log \Delta_i$, so $\varepsilon \geq \frac{1}{\beta_i} \geq \frac{1}{\log \Delta_i}$, and further $(\varepsilon \log \Delta_i)^{-1} \leq 1$ and $\frac{\beta_i}{\log \Delta_i} \leq 1$, thus $\lambda_i \leq \Delta_i^{\frac{1}{2} + \varepsilon} \sqrt{\log(2m) + 2}$ for all $i \in [m]$. The union-bound together with Lemma 1.4 implies the assertion of the corollary.

REMARK 1.2 Note that the discrepancy bound $\Delta_i^{\frac{1}{2} + \varepsilon} \sqrt{\log(2m) + 2}$ in Corollary 1.3 holds only for all i if ε is sufficiently large, i.e., $\varepsilon \geq \frac{1}{\log \Delta_0}$. We can also take a different point of view: first we can assume $\Delta_i \geq \log(2m)$ for all i . Otherwise, the discrepancy bounds in Corollary 1.3 become $\Delta_i^{\frac{1}{2} + \varepsilon} \sqrt{\log(2m) + 2} < \Delta_i$, which is a trivial bound. So, the lower bound condition for ε now says $\varepsilon \geq \frac{1}{\log \log(2m)}$, which is certainly true for large enough m provided that ε is fix. Thus, Corollary 1.3 is true for any $\varepsilon > 0$ and hypergraphs where $2m \geq 2^{2^{\frac{1}{\varepsilon}}}$.

The following theorem in a similar version is due to Motwani, Naor, Naor [MNN94]. Here we compute the exact processor and time bounds and give a compact proof using the derandomization stated in Theorem 1.10.

THEOREM 1.13

Let $\mathcal{H} = (V, \mathcal{E})$ be a hypergraph with $|V| = n$, $|\mathcal{E}| = m$. Let $0 < \varepsilon < 1$ fix and k_i for each $i \in [m]$ as in (1.23), $k = \max_{i \in [m]} k_i$.

- (i). Using k -wise independence, a 2-coloring χ of \mathcal{H} with $|\chi(E_i)| \leq \Delta_i^{\frac{1}{2} + \varepsilon} \sqrt{\frac{\log(2m)}{\varepsilon \log \Delta_i} + 2}$ can be constructed with $O(n^2 m^{1 + \frac{1}{\varepsilon}})$ processors in time $O\left(\frac{1}{\varepsilon^2 \log \Delta_0} \log^2 n \log^2 m\right)$.
- (ii). If $k \leq \log(2m)$, then $\varepsilon \geq \frac{1}{\beta_i} \geq \frac{1}{\log \Delta_i}$ and the discrepancy bounds reduce to $\Delta_i^{\frac{1}{2} + \varepsilon} \sqrt{\log(2m) + 2}$.

Proof (ii) is an immediate consequence of (i) and Corollary 1.3. To prove (i), we invoke Theorem 1.10. Since we need 0/1 random variables, we introduce k -wise independent Z_1, \dots, Z_n , think of each X_j as $X_j = (-1)^{Z_j}$ and regard ψ_i as a function of the Z_1, \dots, Z_n , $\psi_i(Z_1, \dots, Z_n) = \sum_{j=1}^n a_{ij} X_j(Z_j) = \sum_{j=1}^n a_{ij} (-1)^{Z_j}$. For every multiindex $\alpha \in [n]^{k_i}$,

$\alpha = (\alpha_1, \dots, \alpha_{k_i})$, we have $\phi_\alpha(Z_1, \dots, Z_n) = (-1)^{\sum_{j=1}^{k_i} Z_{\alpha_j}}$. Therefore

$$\begin{aligned} \sum_{i=1}^m \lambda_i^{-k_i} \psi_i^{k_i}(Z_1, \dots, Z_n) &= \sum_{i=1}^m \lambda_i^{-k_i} \sum_{\alpha \in [n]^{k_i}} a_{i\alpha} \phi_\alpha(Z_1, \dots, Z_n) \\ &= \sum_{i=1}^m \lambda_i^{-k_i} \sum_{\alpha \in [n]^{k_i}} a_{i\alpha} (-1)^{\sum_{j=1}^{k_i} Z_{\alpha_j}} \\ &=: F(Z_1, \dots, Z_n). \end{aligned} \tag{1.26}$$

The function F has the form as in Theorem 1.10. For fixed i , the number of non-zero terms in the sum

$$\sum_{\alpha \in [n]^{k_i}} a_{i\alpha} (-1)^{\sum_{j=1}^{k_i} Z_{\alpha_j}}$$

is at most $\Delta_i^{k_i} \leq \Delta_i^{\frac{\log(2m)}{\varepsilon \log \Delta_i} + 2} = (2m)^{\frac{1}{\varepsilon}} \Delta_i^2 = O(m^{\frac{1}{\varepsilon}} n^2)$. The number $N = N(n, m, k)$ of terms in (1.26) thus is at most $N = m \cdot \max_{i \in [m]} \Delta_i^{k_i} = O(n^2 m^{1+\frac{1}{\varepsilon}})$. By Theorem 1.10 we can construct $y \in \{0, 1\}^n$ using $O(N) = O(n^2 m^{1+\frac{1}{\varepsilon}})$ parallel processors in $O(k \log n \log N) = O\left(k \log n \cdot (\log n + \frac{1}{\varepsilon} \log m)\right) = O\left(\frac{1}{\varepsilon^2 \log \Delta_0} \log^2 n \log^2 m\right)$ time such that $F(y) \leq \mathbb{E}[F(X_1, \dots, X_n)]$. The vector $x = (x_1, \dots, x_n)$, where $x_i = (-1)^{y_i}$, defines a 2-coloring χ .

Let A_i be the event that $|\psi_i| \geq \lambda_i$, for $i \in [m]$. Then $A := \bigcup_{i \in [m]} A_i$ is the event that there exists some i such that A_i holds. We fix an arbitrary i . According to Lemma 1.4, the k_i th moment inequality, which in fact is the Markov inequality, states that

$$\mathbb{P}[A_i] = \mathbb{P}[|\psi_i| \geq \lambda_i] \leq \lambda_i^{-k_i} \mathbb{E}[\psi_i^{k_i}]. \tag{1.27}$$

Let $t \in [l]$, $\vec{y}_t \in \{0, 1\}^t$ and with $\vec{Y}_t = (Y_1, \dots, Y_t)$ consider the event $\vec{Y}_t = \vec{y}_t$. It is easily checked that (1.27) is also true conditioned on $\vec{Y}_t = \vec{y}_t$:

$$\mathbb{P}[A_i \mid \vec{Y}_t = \vec{y}_t] = \mathbb{P}[|\psi_i| \geq \lambda_i \mid \vec{Y}_t = \vec{y}_t] \leq \lambda_i^{-k_i} \mathbb{E}[\psi_i^{k_i} \mid \vec{Y}_t = \vec{y}_t]. \tag{1.28}$$

Then

$$\mathbb{P}[A \mid \vec{Y}_t = \vec{y}_t] \leq \sum_{i=1}^m \lambda_i^{-k_i} \mathbb{E}[\psi_i^{k_i} \mid \vec{Y}_t = \vec{y}_t] = \mathbb{E}[F(Z_1, \dots, Z_n) \mid \vec{Y}_t = \vec{y}_t].$$

Hence

$$\mathbb{P}[A \mid Y = y] \leq \mathbb{E}[F(Z_1, \dots, Z_n) \mid Y = y] = F(y). \tag{1.29}$$

Since by Lemma 1.4, $\mathbb{E}[F(Z_1, \dots, Z_n)] < 1$, (1.29) and $F(y) \leq \mathbb{E}[F(Z_1, \dots, Z_n)]$ yield $\mathbb{P}[A \mid Y = y] < 1$, so $\mathbb{P}[A \mid Y = y] = 0$, because $\mathbb{P}[A \mid Y = y] \in \{0, 1\}$. In other words, the 2-coloring defined by y has the claimed discrepancy $|\chi(E_i)| \leq \lambda_i$ for all $i \in [m]$. ■

Bibliography and Remarks. Beck and Fiala [BF81] showed for a general hypergraph \mathcal{H} that $\text{disc}(\mathcal{H}) \leq 2 \deg(\mathcal{H})$. A long-standing conjecture of great interest is whether a bound of $O(\sqrt{\deg(\mathcal{H})})$ is valid. Beck [Bec81] showed for any hypergraph \mathcal{H} of degree t , $\text{disc}(\mathcal{H}) = O(\sqrt{t \ln t \ln n})$. Srinivasan [Sri97] improved the bound of Beck to $O(\sqrt{t \ln n})$ and gave better bounds for the lattice approximation problem (see Beck and Fiala [BF81] and Raghavan [Rag88] for bounds derived with Chernoff-Hoeffding type inequalities). The best known bound $O(\sqrt{t \log n})$ is due to Banaszczyk [Ban98].

Combinatorial Discrepancies in more than two colors were introduced by Doerr and Srivastav [DS99, DS01, DS03]. They show that most of the important results for two colors hold also in arbitrary numbers of colors. The c -color discrepancy, $c \geq 2$, of a hypergraph on n points with n hyperedges is at most $O(\sqrt{\frac{n}{c} \ln c})$, and there are examples of hypergraphs with c -color discrepancy at least $\Omega\left(\sqrt{\frac{n}{c}}\right)$. Parallel algorithms for the computation of multicolorings with low discrepancy are not known, but we are confident that this should be possible. They also observed (see also Doerr [Doe02b]) that there are hypergraphs having very different discrepancies in different numbers of colors. A deep result of Doerr [Doe02a, Doe04] shows that this dichotomy is structurally inherent. Further insight was given by Doerr, Gnewuch, and Hebbinghaus [DGH06] and recently Doerr and Fouz [DF06]. One particular application of multi-color discrepancies is the declustering problem in computer science, which can be modelled as a discrepancy problem of the hypergraph of rectangles in the higher-dimensional grid. Doerr, Hebbinghaus, and Werth [DHW04] give some of the current best results in this direction.

Among hypergraphs of special interest is the hypergraph of arithmetic progressions, where $[N]$ is the node set and all arithmetic progressions in $[N]$ are the hyperedges. For the hypergraph AP of arithmetic progressions in the first N integers Roth [Rot64] proved $\text{disc}(AP) \geq cN^{\frac{1}{4}}$. In 1994, Matoušek and Spencer [MS96] showed $\text{disc}(AP) \leq c'N^{\frac{1}{4}}$, resolving the problem ($c, c' > 0$ are constants). For the hypergraph of cartesian products of arithmetic progressions AP^d in $[N]^d$ Doerr, Srivastav, and Wehr [DSW04] proved $\text{disc}(AP^d) = \Theta(N^{\frac{d}{4}})$, with constants depending only on d . For extension of these results to multicolor discrepancy we refer to Doerr, Srivastav [DS03]. For the hypergraph of arithmetic progressions over $[N]$ the c -color discrepancy is at most $O(c^{-0.16} \sqrt[4]{N})$, and is at least $O(c^{-0.5} \sqrt[4]{N})$. Another problem dealing with cartesian products was studied by Matoušek [Mat00], who proved – answering a question of Beck and Chen – that the discrepancy of the family of cartesian products of circular discs in the plane in \mathbb{R}^4 is $O(n^{\frac{1}{4}+\varepsilon})$ for an arbitrarily small constant $\varepsilon > 0$, thus it is essentially the same as for circular discs in the plane. **But all these results lack for efficient parallel algorithms!**

A breakthrough for 2-coloring uniform hypergraphs was achieved by Beck in 1991 [Bec91] via the Lovász-Local-Lemma. Alon [Alo91], using Beck's idea, gave a probabilistic algorithm and derandomized it with the method of almost k -wise independence for hypergraphs of degree at most $2^{\frac{n}{500}}$.

The lattice approximation problem is stated as follows:

Lattice Approximation Problem

Input: An $(m \times n)$ matrix $A = (a_{ij})$, $a_{ij} \in [0, 1]$ for all entries, vector $p \in [0, 1]^n$.

Task: Find a vector $q \in \{0, 1\}^n$ such that $\|Ap - Aq\|_\infty$ is minimum.

We call it *half lattice approximation problem* or *vector balancing problem* if all entries of p are $\frac{1}{2}$. For convenience we define $c := Ap$. The construction of an \mathcal{NC} algorithm for the lattice approximation problem in [MNN94] uses the \mathcal{NC} algorithm for the discrepancy problem. In the first step we consider the half lattice approximation problem and then generalize with a bit-by-bit rounding technique.

All our algorithms make use of the binary representation of the entries in A and p . It is necessary to limit the precision, so that we have a binary representation with a known number of bits. We assume that each a_{ij} and each p_j , $i \in [m]$, $j \in [n]$ has binary encoding length $L' = L + 1$. If $A' = (a'_{ij})$ is the matrix where each a'_{ij} denotes the entry a_{ij} truncated off to $L + 1$ bits of precision, and p' is the truncated version of p , it is straightforward to see that

$$\max \{ \|Ap - A'p'\|_\infty, \|A'q - Aq\|_\infty \} \leq 2n2^{-L'} = n2^{-L}. \quad (1.30)$$

Hence to push this error below some $\delta > 0$, it is sufficient to choose $L = \left\lceil \log(n \frac{1}{\delta}) \right\rceil$. As

long as we are only interested in bounds of the kind $\|Ap - Aq\|_\infty = O(f(m, n, \varepsilon))$, it is sufficient to choose $L = \Theta(\log n)$, because then

$$\max\{\|Ap - A'p'\|_\infty, \|A'q - Aq\|_\infty\} = O(1).$$

Thus, in the following we may indeed assume that

$$a_{ij} = \sum_{k=0}^L 2^{-k} \text{bit}_k(a_{ij}) \quad \text{and} \quad p_j = \sum_{k=0}^L 2^{-k} \text{bit}_k(p_j) \quad \text{for all } i \in [m] \text{ and } j \in [n],$$

knowing that L can be chosen logarithmic in n/δ or just in n , depending on the application, to achieve useful approximations.

(a) Half Lattice Approximation

Let A, p be an instance of the half lattice approximation problem. We show its equivalence to a discrepancy problem. Let B be the $((L'm) \times n)$ matrix over $\{0, 1\}$ drawn from A by replacing each a_{ij} by the 0/1 vector⁴ $(\text{bit}_0(a_{ij}), \dots, \text{bit}_L(a_{ij}))$. We denote by $\mathcal{H}_A = (V, \mathcal{E})$ the hypergraph on $|V| = n$ nodes and $|\mathcal{E}| = L'm$ hyperedges $E_1, \dots, E_{L'm}$ whose hyperedge-vertex incidence matrix is B . Let $\chi : V \rightarrow \{-1, 1\}$ be a 2-coloring of \mathcal{H}_A and $q \in \{0, 1\}^n$ be the vector with entries $q_j = \frac{1+\chi(j)}{2}$, $j \in [n]$. (Hence, q is a 0/1 representation of the coloring χ .)

To shorten notations, define

$$h := h(m, L', \Delta_0, \varepsilon) := \frac{\log(2L'm)}{\varepsilon \log \Delta_0} + 2, \tag{1.31}$$

which is part of the discrepancy bound of Theorem 1.13 for the given hypergraph \mathcal{H}_A .

THEOREM 1.14 *Let $0 < \varepsilon \leq \frac{1}{3}$ and as before let $\Delta_0 = \min_{i \in [m]} \Delta_i$, where $\Delta_i = \max\{|E_i|, 3\}$.*

(i). *If for all $E \in \mathcal{E}$, $|\chi(E)| \leq |E|^{\frac{1}{2}+\varepsilon} \sqrt{\frac{\log(2L'm)}{\varepsilon \log |E|}} + 2$, then for all $i \in [m]$:*

$$|(Ap - Aq)_i| = O\left(c_i^{\frac{1}{2}+\varepsilon} \sqrt{h}\right).$$

(ii). *There is a parallel algorithm for the half lattice approximation problem using at most $O(n^2(L'm)^{1+\frac{1}{\varepsilon}})$ processors and running in $O\left(\frac{1}{\varepsilon^2 \log \Delta_0} \log^2 n \log^2(L'm)\right)$ time, achieving the approximation guarantee stated above.*

⁴Note about our notation: This is not something like “ $\vec{\text{bit}}_l(a_{ij})$ ”.

Proof (i). With $Q_1 := \{j; q_j = 1\}$ we have

$$\begin{aligned}
|(Bq - Bp)_i| &= \left| |E_i \cap Q_1| - \frac{|E_i|}{2} \right| \\
&= \frac{1}{2} |\chi(E_i)| \\
&\leq \frac{1}{2} |E_i|^{\frac{1}{2} + \varepsilon} \sqrt{\frac{\log(2L'm)}{\varepsilon \log \Delta_i}} + 2 \quad \text{by assumption} \\
&\leq \frac{1}{2} |E_i|^{\frac{1}{2} + \varepsilon} \sqrt{h}.
\end{aligned} \tag{1.32}$$

Now, for $i \in [m]$:

$$\begin{aligned}
c_i &= \sum_{j=1}^n \frac{1}{2} a_{ij} = \sum_{k=0}^L 2^{-k} \sum_{j=1}^n \frac{1}{2} \text{bit}_k(a_{ij}) \\
&= \sum_{k=0}^L 2^{-k} (Bp)_{(i-1)L+k} \\
&= \sum_{k=0}^L 2^{-k} \frac{1}{2} |E_{(i-1)L+k}|,
\end{aligned} \tag{1.33}$$

and

$$\begin{aligned}
|(Ap - Aq)_i| &= \left| \sum_{j=1}^n \frac{1}{2} a_{ij} - \sum_{j=1}^n a_{ij} q_j \right| \\
&= \left| \sum_{j=1}^n \frac{1}{2} \sum_{k=0}^L 2^{-k} \text{bit}_k(a_{ij}) - \sum_{j=1}^n \sum_{k=0}^L 2^{-k} \text{bit}_k(a_{ij}) q_j \right| \\
&= \left| \sum_{k=0}^L 2^{-k} \left(\sum_{j=1}^n \frac{1}{2} \text{bit}_k(a_{ij}) - \sum_{j=1}^n \text{bit}_k(a_{ij}) q_j \right) \right| \\
&= \left| \sum_{k=0}^L 2^{-k} (Bp - Bq)_{(i-1)L+k} \right| \\
&= \sum_{k=0}^L 2^{-k-1} |E_{(i-1)L+k}|^{\frac{1}{2} + \varepsilon} \sqrt{h} \quad \text{by (1.32)} \\
&= \sqrt{h} \sum_{k=0}^L 2^{(k+1)(\varepsilon - \frac{1}{2})} \underbrace{(2^{-k-1} |E_{(i-1)L+k}|)}_{\leq c_i \text{ by (1.33)}}^{\frac{1}{2} + \varepsilon} \\
&= O\left(c_i^{\frac{1}{2} + \varepsilon} \sqrt{h}\right),
\end{aligned}$$

as $\varepsilon \leq \frac{1}{3}$ and so the geometric sum $\sum_{k=0}^L 2^{(k+1)(\varepsilon - \frac{1}{2})}$ is $O(1)$.

(ii). The construction of B can be done with $O(L'mn)$ processors in constant time. The computation of a coloring χ with discrepancy

$$|\chi(E_i)| \leq |E_i|^{\frac{1}{2} + \varepsilon} \sqrt{\frac{\log(2L'm)}{\varepsilon \log |E_i|}} + 2 \quad \text{for all } i \in [L'm]$$

can be done according to Theorem 1.13 within the claimed processor and time bounds. ■

(b) Lattice Approximation

In the general problem the input is given by a $(m \times n)$ matrix $A \in [0, 1]^{m \times n}$ and a vector $p \in [0, 1]^n$. We describe the bit-by-bit randomized rounding. In the following algorithm, after k iterations, each p_j , $j \in [n]$ is represented by at most $L + 1 - k$ bits of precision.

Algorithm 6: BIT-RR

Input: $p = (p_1, \dots, p_n) \in [0, 1]^n$, with $L + 1$ bits of precision in each component.
Output: Lattice point in $\{0, 1\}^n$.

```

for  $l \leftarrow L$  downto 1 do
  forall  $j \in [n]$  do
    if  $\text{bit}_l(p_j) = 1$  then
      with probability  $\frac{1}{2}$  do  $p_j \leftarrow p_j - 2^{-l}$ ;           /* round down */
      else  $p_j \leftarrow p_j + 2^{-l}$ ;                             /* round up */
    end
  end
end
return  $\vec{\text{bit}}_0(p)$ 

```

It is clear that the algorithm terminates after at most L iterations with the desired lattice point, which is the result of successively rounding the bits in p , until at most the most significant bit can be set to 1. Reviewing this rounding process, it is straightforward to prove that the probability of rounding the j th entry of p to 1 is exactly p_j . We proceed to the parallel derandomization of the algorithm BIT-RR.

Algorithm 7: DERAND-BIT

Input: $p = (p_1, \dots, p_n) \in [0, 1]^n$ with $L + 1$ bits of precision in each component and parameter $0 < \varepsilon \leq \frac{1}{4}$.
Output: Lattice point in $\{0, 1\}^n$.

```

 $P^{(L)} \leftarrow p$ ;
for  $l \leftarrow L$  downto 1 do
   $q^{(l)} \leftarrow$  solution to half LAP with input  $\frac{1}{2}\vec{\text{bit}}_l(P^{(l)})$  and  $\varepsilon$ ;
   $P^{(l-1)} \leftarrow P^{(l)} + 2^{-(l-1)}(q^{(l)} - \frac{1}{2}\vec{\text{bit}}_l(P^{(l)}))$ ;
end
return  $P^{(0)}$ 

```

Recall the definition and the role of h ; see (1.31).

A similar version of the next theorem is a main result of [MNN94]. In the following formulation we state the approximation as well as processor and time bounds with all necessary parameters explicitly.

THEOREM 1.15

Let $q \in \{0, 1\}^n$ be the lattice point computed by DERAND-BIT and $i \in [m]$. There is a

constant⁵ $\gamma \geq 1$ such that the following holds.

(i). If $c_i \geq \gamma h^{\frac{1}{1-2\varepsilon}}$, then

$$|(Ap - Aq)_i| = O\left(c_i^{\frac{1}{2}+\varepsilon} \sqrt{h}\right).$$

(ii). If $c_i < \gamma h^{\frac{1}{1-2\varepsilon}}$, then

$$|(Ap - Aq)_i| = O\left(h^{\frac{1}{1-2\varepsilon}}\right).$$

The algorithm uses $O(n^2(L'm)^{1+\frac{1}{\varepsilon}})$ processors and runs in $O\left(\frac{L'}{\varepsilon^2 \log \Delta_0} \log^2 n \log^2(L'm)\right)$ time. (Δ_0 is taken for the hypergraph induced by A , as before. We always have $\log \Delta_0 > 1$, because $\Delta_0 \geq 3$.)

Proof Denote $C^{(l)} := AP^{(l)}$ for all $l \in \{0, \dots, L\}$ and

$$D_i^{(l)} := |C_i^{(l+1)} - C_i^{(l)}| \quad \text{for all } i \in [m] \text{ and } l \in \{0, \dots, L-1\}.$$

So, we have in particular $C^{(L)} = c$, and $D_i^{(l)}$ denotes the error in the i th component introduced when rounding to l bits of precision (from $l+1$ bits). We can prove a bound on these errors, regardless in which of the two cases (i) or (ii) we are.

Claim. There is a constant $\alpha > 0$ such that

$$D_i^{(l)} \leq \alpha \frac{(C_i^{(l+1)})^{\frac{1}{2}+\varepsilon} \sqrt{h}}{(2^{\frac{1}{2}-\varepsilon})^l} \quad \text{for all } l \in \{0, \dots, L-1\}. \quad (1.34)$$

For the proof, fix $l \in \{0, \dots, L-1\}$. We have

$$\begin{aligned} 2^l C_i^{(l+1)} &= 2^l (AP^{(l+1)})_i = 2^l \left(A \sum_{k=0}^{l+1} 2^{-k} \vec{\text{bit}}_k(P^{(l+1)}) \right)_i = \sum_{k=0}^{l+1} 2^{l-k} \left(A \vec{\text{bit}}_k(P^{(l+1)}) \right)_i \\ &\geq 2^{l-(l+1)} \left(A \vec{\text{bit}}_{l+1}(P^{(l+1)}) \right)_i = \left(A \frac{1}{2} \vec{\text{bit}}_{l+1}(P^{(l+1)}) \right)_i. \end{aligned} \quad (1.35)$$

Let α be the constant from the first $O(\cdot)$ in Theorem 1.14, (i). We get

$$\begin{aligned} D_i^{(l)} &= |C_i^{(l+1)} - C_i^{(l)}| = |(AP^{(l+1)})_i - (AP^{(l)})_i| \\ &= \left| (AP^{(l+1)})_i - \left(A \left(P^{(l+1)} + 2^{-l} (q^{(l+1)} - \frac{1}{2} \vec{\text{bit}}_{l+1}(P^{(l+1)})) \right) \right)_i \right| \\ &= 2^{-l} \left| \left(A q^{(l+1)} - A \frac{1}{2} \vec{\text{bit}}_{l+1}(P^{(l+1)}) \right)_i \right| \\ &\leq 2^{-l} \alpha \cdot \left(A \frac{1}{2} \vec{\text{bit}}_{l+1}(P^{(l+1)}) \right)_i^{\frac{1}{2}+\varepsilon} \sqrt{h} && \text{by Theorem 1.14} \\ &\leq 2^{-l} \alpha \cdot (2^l C_i^{(l+1)})^{\frac{1}{2}+\varepsilon} \sqrt{h} && \text{by (1.35)} \\ &= \alpha \frac{(C_i^{(l+1)})^{\frac{1}{2}+\varepsilon} \sqrt{h}}{(2^{\frac{1}{2}-\varepsilon})^l}. \end{aligned}$$

⁵I.e., independent of the LAP instance and of ε .

This proves the claim.

We need β such that

$$\beta \geq \sum_{k=1}^L \frac{2\alpha}{(2^{\frac{1}{2}-\varepsilon})^k}.$$

Because $\varepsilon \leq \frac{1}{4}$, we have

$$\frac{1}{1-2^{-\frac{1}{4}}} \geq \sum_{k=1}^L \frac{1}{(2^{\frac{1}{4}})^k} \geq \sum_{k=1}^L \frac{1}{(2^{\frac{1}{2}-\varepsilon})^k},$$

and hence choosing $\beta := 2\alpha \frac{1}{1-2^{-\frac{1}{4}}}$, which is a constant, is sufficient. Define then

$$\gamma := \max \left\{ \left(\frac{\beta^2}{2^{\frac{4}{3}} - 1} \right)^2, 1 \right\}.$$

Consider first the case of (i), i.e., $c_i \geq \gamma h^{\frac{1}{1-2\varepsilon}}$. We do a slightly complicated induction from L down to 0. Since $C^{(l)} = c$, the following inequality holds trivially for $l = L$:

$$C_i^{(l)} \leq c_i + \beta c_i^{\frac{1}{2}+\varepsilon} \sqrt{h}. \quad (1.36)$$

We will show that if it holds for some $l \in [L]$, then it also holds for $l-1$. At the same time, we will prove a better bound on the errors $D_i^{(l)}$. Now fix $l \in [L]$ and assume that (1.36) holds. We have by (1.34) that

$$\begin{aligned} D_i^{(l-1)} &\leq \alpha \frac{(C_i^{(l)})^{\frac{1}{2}+\varepsilon} \sqrt{h}}{(2^{\frac{1}{2}-\varepsilon})^{l-1}} \\ &\leq \alpha \frac{\left(c_i + \beta c_i^{\frac{1}{2}+\varepsilon} \sqrt{h} \right)^{\frac{1}{2}+\varepsilon} \sqrt{h}}{(2^{\frac{1}{2}-\varepsilon})^{l-1}} && \text{by (1.36)} \\ &\leq \frac{2\alpha c_i^{\frac{1}{2}+\varepsilon} \sqrt{h}}{(2^{\frac{1}{2}-\varepsilon})^{l-1}} && \text{because } c_i \geq \gamma h^{\frac{1}{1-2\varepsilon}} \\ &&& \text{and the def. of } \gamma. \end{aligned}$$

To verify the last inequality, see that for it to be true it is sufficient that $c_i^{\frac{1}{2}-\varepsilon} \geq \frac{\beta \sqrt{h}}{2^{1+2\varepsilon}-1}$ and the fact that γ was chosen in a way that this holds when $c_i \geq \gamma h^{\frac{1}{1-2\varepsilon}}$.

Consider now for some $r \in \{0, \dots, L\}$ the following statement, of which we will show that it holds for all r :

$$\begin{aligned} C_i^{(r)} &\leq c_i + \sum_{k=1}^{L-r} D^{(L-k)} \\ &\leq c_i + \sum_{k=1}^{L-r} \frac{2\alpha c_i^{\frac{1}{2}+\varepsilon} \sqrt{h}}{(2^{\frac{1}{2}-\varepsilon})^{L-k}} && (1.37) \\ &= c_i + c_i^{\frac{1}{2}+\varepsilon} \sqrt{h} \sum_{k=1}^{L-r} \frac{2\alpha}{(2^{\frac{1}{2}-\varepsilon})^{L-k}} \\ &\leq c_i + \beta c_i^{\frac{1}{2}+\varepsilon} \sqrt{h}. \end{aligned}$$

This is obviously true for $r = L$. To see that it is true also for $r = L - 1$, we use that (1.36) holds for $l = L$ and the analysis following equation (1.36). Then, (1.37) establishes (1.36) for $l = L - 1$. Hence, we can prove the above bound on $D_i^{(l-1)}$ for this l and gain (1.37) for $r = L - 2$. Proceeding in this manner gives (1.37) for all r , in particular for $r = 0$. This gives

$$|(Ap - Aq)_i| \leq \sum_{k=1}^L D_i^{(L-k)} \leq \beta c_i^{\frac{1}{2} + \varepsilon} \sqrt{h}.$$

The theorem is proved for the first case.

Consider now the case of (ii), i.e., $c_i < \gamma h^{\frac{1}{1-2\varepsilon}}$. As long as the values $C_i^{(l)}$ stay below $\gamma h^{\frac{1}{1-2\varepsilon}}$, the total error (so far) does so as well. If they always stay below this value, we are done. Otherwise, let l_0 be maximal such that $C_i^{(l_0)} \geq \gamma h^{\frac{1}{1-2\varepsilon}}$. We can now apply the analysis from case (i), however⁶ with $C_i^{(l_0)}$ instead of c_i . It is therefore necessary to upper-bound $C_i^{(l_0)}$. Fortunately, because $C_i^{(l_0+1)} < \gamma h^{\frac{1}{1-2\varepsilon}}$, (1.34) gives us

$$|C_i^{(l_0+1)} - C_i^{(l_0)}| = D_i^{(l_0)} \leq \alpha \frac{(C_i^{(l_0+1)})^{\frac{1}{2} + \varepsilon} \sqrt{h}}{(2^{\frac{1}{2} - \varepsilon})^{l_0}} < \underbrace{\alpha \gamma^{\frac{3}{4}}}_{=: \alpha'} \frac{h^{\frac{1}{1-2\varepsilon}}}{(2^{\frac{1}{2} - \varepsilon})^{l_0}} \leq \alpha' h^{\frac{1}{1-2\varepsilon}},$$

using $\varepsilon \leq \frac{1}{4}$ and the identity

$$\frac{1}{1-2\varepsilon} \cdot \left(\frac{1}{2} + \varepsilon\right) + \frac{1}{2} = \frac{1}{1-2\varepsilon}. \quad (1.38)$$

Hence $C_i^{(l_0)} \leq (1 + \alpha') h^{\frac{1}{1-2\varepsilon}}$. Using that $(1 + \alpha')^{\frac{1}{2} + \varepsilon} \leq (1 + \alpha')^{\frac{3}{4}}$, the result for case (i) with (1.38) finally proves the assertion for case (ii).

The statement about processors and running time follows from Theorem 1.14 because we have L' iterations. ■

REMARK 1.3 If we choose $L' = \lceil \log n \rceil$, the number of processors in Theorem 1.14 and Theorem 1.15 is $O(n^2(L'm)^{1+\frac{1}{\varepsilon}}) = O(n^2(m \lceil \log n \rceil)^{1+\frac{1}{\varepsilon}})$, which is close to the estimation implicitly given in [MNN94]. The running times become $O\left(\frac{1}{\varepsilon^2 \log \Delta_0} \log^2 n \log^2(m \log n)\right)$ and $O\left(\frac{1}{\varepsilon^2 \log \Delta_0} \log^3 n \log^2(m \log n)\right)$ respectively.

If the entries in A are small compared to the right-hand-side c , say $a_{ij} \leq \delta c_i$, for a sufficiently small δ , by a scaling technique one can construct a lattice point q with a relative error of $O(\delta^{\frac{1}{2}-n} \sqrt{h})$, and there is even a good upper bound on the number of positive entries in q . This will later be used in the application of basis crashing, which is an important building-block for automata fooling.

We write $\text{supp}(q) := \{j; q_j > 0\}$ and call it the *support* of the vector q . The cardinality of the support is denoted by $|q| := |\text{supp}(q)|$. The sum of all entries in a vector p is denoted by \bar{p} . We have:

⁶This point was not clearly stated in the original proof.

THEOREM 1.16 Let $A \in [0, 1]^{m \times n}$, $p \in [0, 1]^n$ be a lattice approximation instance, $c := Ap$, $c \in \mathbb{R}_{>0}^m$. Put $L := \lceil \log n \rceil$, $h := \frac{\log(2L^m)}{\eta \log \Delta_0} + 2$, $0 < \eta \leq \frac{1}{4}$, γ as in Theorem 1.15. Assume that there exists $\delta > 0$ such that

- $\delta \leq (\gamma h^{\frac{1}{1-2\eta}} + 1)^{-1}$ and
- $a_{ij} \leq \delta c_i$ for all i .

Then we can construct in \mathcal{NC} a vector $q \in \{0, 1\}^n$ such that

$$(Aq)_i \in (1 \pm O(\delta^{\frac{1}{2}-\eta} \sqrt{h}))(Ap)_i$$

and $|q| = \bar{p} + O(\bar{p}^{\frac{1}{2}+\eta} \sqrt{h}) + 1$. The construction can be done with $O(n^2 O(m \log n)^{1+\frac{1}{\eta}})$ parallel processors in $O\left(\frac{1}{\eta^2} \log^3 n \log^2(m \log n)\right)$ time. The entries in A and p may be of any precision.

Proof Scale each row i of A by $\frac{1}{\delta c_i}$. The resulting matrix \tilde{A} is in $[0, 1]^{m \times n}$ and $\tilde{c} := \tilde{A}p$ has the property that $\tilde{c}_i = \frac{1}{\delta}$ for all i . In addition, we augment \tilde{A} by an extra row of 1s at the bottom.

Truncate the entries in \tilde{A} and p to $L+1$ bits of precision, yielding matrix A' and vector p' . We know from (1.30) that for all vectors q and all $i \in [m]$ with $c' := A'p'$

$$|(\tilde{c} - c')_i| = |(\tilde{A}p - A'p')_i| \leq 1 \text{ and } |(\tilde{A}p - \tilde{A}q)_i| \leq |(A'p' - A'q)_i| + 2.$$

We have $c'_i \geq \tilde{c}_i - 1 = \frac{1}{\delta} - 1 \geq \gamma h^{\frac{1}{1-2\eta}}$ for all $i \in [m]$. Hence we can use case (i) of Theorem 1.15 to solve the LAP instance defined by A' and p' with parameter η yielding a vector $q \in \{0, 1\}$ with

$$|(A'p' - A'q)_i| = O\left(c'_i^{\frac{1}{2}+\eta} \sqrt{h}\right) = O\left(\left(\frac{1}{\delta}\right)^{\frac{1}{2}+\eta} \sqrt{h}\right) \text{ for all } i \in [m], \quad (1.39)$$

using that trivially $c'_i \leq \tilde{c}_i = \frac{1}{\delta}$. The relative error for the original problem so is

$$\begin{aligned} \frac{|(Ap - Aq)_i|}{(Ap)_i} &= \frac{(\delta c_i)^{-1} |(Ap - Aq)_i|}{(\delta c_i)^{-1} (Ap)_i} \\ &= \frac{|(\tilde{A}p - \tilde{A}q)_i|}{\delta^{-1}} \\ &\leq \delta(O\left(\left(\frac{1}{\delta}\right)^{\frac{1}{2}+\eta} \sqrt{h}\right) + 2) \\ &= O(\delta^{\frac{1}{2}-\eta} \sqrt{h}) + 2\delta \\ &= O(\delta^{\frac{1}{2}-\eta} \sqrt{h}). \end{aligned}$$

For the support of q , consider the last entry in c' . For this $c'_{m+1} = \sum_{j=1}^n p'_j = \bar{p}'$ we know because of $c_1 = \sum_{j=1}^n a_{1j} p_j \leq \delta c_1 \sum_{j=1}^n p_j = \delta c_1 \bar{p}$ that we have $c'_{m+1} = \bar{p}' \geq \bar{p} - 1 \geq \frac{1}{\delta} - 1 \geq \gamma h^{\frac{1}{1-2\eta}}$. Hence we can also use case (i) of Theorem 1.15 for this row of the matrix. We get $|\bar{p} - |q|| \leq |\bar{p}' - |q|| + 1 = |(A'p' - A'q)_{m+1}| + 1 = O(\bar{p}^{\frac{1}{2}+\eta} \sqrt{h}) + 1$. The claimed bound follows.

The bounds for processors and running time follow from Theorem 1.15. ■

Bibliography and Remarks. Srinivasan [Sri97] gave better bounds for the lattice approximation problem (see Beck and Fiala [BF81] and Raghavan [Rag88] for bounds derived with Chernoff-Hoeffding type inequalities). The quadratic lattice approximation problem, a generalization of the

(linear) lattice approximation problem, has been studied by Srivastav and Stangier [SS93], and a derandomized algorithm using Azuma's martingale was obtained. Here, parallel algorithms are not known.

Closely related to the lattice approximation problem is the concept of linear discrepancy of hypergraphs. Solving a long-standing open problem, Doerr [Doe01] showed that linear discrepancy of a unimodular hypergraph is at most $\frac{n}{n+1}$. This problem is a particular case of another open problem, namely the relation of hereditary and linear discrepancies of a hypergraph. The classical result due to Beck and Spencer [BS84] as well as Lovász, Spencer, and Vesztegombi [LSV86] is $\text{disc}(\mathcal{H}) \leq 2 \text{herdisc}(\mathcal{H})$. The latter paper also gives a class of hypergraphs fulfilling $\text{disc}(\mathcal{H}) = 2 \frac{n}{n+1} \text{herdisc}(\mathcal{H})$ and conjectures that this should be the right factor. Doerr [Doe00] showed $\text{disc}(\mathcal{H}) \leq 2 \frac{2^m-1}{2^m} \text{herdisc}(\mathcal{H})$. This is still the best upper bound known for this problem.

1.4 Fooling Automata

Many randomized algorithms can be modelled by a set of deterministic finite automata. If the input strings for these automata are drawn randomly according to a certain distribution ζ , the automata simulate the randomized algorithm as it works on some fixed input – the automata themselves may depend on the input to the algorithm. Such a representation of an algorithm allows a special kind of derandomization. The idea is to modify the distribution ζ to a second distribution ρ such that the size of the support, i.e., the number of strings with nonzero probability, becomes polynomial, and that the automata approximately still simulate the algorithm. This is called *automata fooling*, because the automata are presented with a different distribution but still behave essentially as before. If the support of ρ is sufficiently small, all strings can be checked efficiently (in parallel), and hence we have a deterministic algorithm. We present the automata fooling technique from the paper of Karger and Koller [KK97].

Automata are represented by regular, directed multigraphs.

DEFINITION 1.5 An automaton \mathcal{A} is a triple (V, E, s) , where V is the set of states (the nodes or vertices of the graph) and E are labeled edges, which define possible transitions between two states, and $s \in V$ is a special state called the *starting state* of the automaton. Each state has outgoing edges labeled from 1 to r for some fixed number r . (In practice, automata will have ending states, from where there are no outgoing edges. To fit with this definition, for such a node v simply think of r loops (v, v) .) Inputs for \mathcal{A} are strings from $[r]^*$, i.e., concatenations of the numbers from 1 to r .

Giving to \mathcal{A} a word $w = (w_1, w_2, \dots, w_l) \in [r]^*$ as input means that starting from s , the automaton changes its state by traversing the edges as specified by w . In other words: w defines a walk in the graph (V, E) starting at node s .

Throughout this section, we fix a set $\mathcal{A}_1, \dots, \mathcal{A}_m$ of m automata, write $\mathcal{A}_i = (V_i, E_i, s_i)$ for each $i \in [m]$, each of them taking inputs from the same set $[r]^*$.

Let X_1, \dots, X_n be independent random variables each taking values in $[r]$. They define a distribution on $W := [r]^n$. We denote this distribution by $\zeta : W \rightarrow [0, 1]$, i.e., $\zeta(w)$ is the probability that (X_1, \dots, X_n) takes on the value $w \in W$. (This probability is equal to $\mathbb{P}[X_1 = w_1] \cdot \dots \cdot \mathbb{P}[X_n = w_n]$.)

The automata $\mathcal{A}_1, \dots, \mathcal{A}_m$ when provided with inputs drawn from W according to ζ are assumed to simulate some randomized algorithm \mathbf{A} under consideration. The final states of the automata correspond to the output of the algorithm. The automata themselves may depend on the input for the algorithm \mathbf{A} . We fix an input instance \mathcal{I} for our presentation.

It is assumed that the automata have polynomially many states, in the length of \mathcal{I} .

DEFINITION 1.6 The *support* of a distribution ξ on a set U is

$$\text{supp}(\xi) := \{w \in U; \xi(w) > 0\}.$$

For the cardinality of the support, also called the *size* of the distribution, we write

$$|\xi| := |\text{supp}(\xi)|.$$

If we know that with positive probability (w.r.t. ζ), the algorithm (simulated by the automata) outputs a good solution⁷, it suffices to check the output of the automata for every $w \in \text{supp}(\zeta)$. However, this set may be too large. If, on the other hand, we are able to replace ζ by some distribution ρ such that $\text{supp}(\rho)$ is small and the automata still (approximately) simulate the algorithm, we can check every word in $\text{supp}(\rho)$ instead. This can be done in parallel.

There are methods to reduce the support of a distribution while maintaining the probabilities of certain events. The challenge is that in general, the running times of these algorithms depend (polynomially) on the size of the original support. The solution to this is a divide-and-conquer approach. It will be described in the following. First, we look at the problem of reducing the support of a given distribution directly, and then consider intermediate transition probabilities, which help in the design of the divide-and-conquer technique.

Reducing Support

Let ξ be a distribution and B_1, \dots, B_k events (e.g., transitions in the automata). The events are also called *constraints* in this context, and we denote the set of all constraints by \mathcal{C} .

For every $\varepsilon > 0$, by $(1 \pm \varepsilon)$ we denote the interval $(1 - \varepsilon, 1 + \varepsilon)$. Consequently, for a real number x , by $(1 \pm \varepsilon)x$ the interval $(x - \varepsilon x, x + \varepsilon x)$ is denoted.

DEFINITION 1.7 Let $1 > \varepsilon \geq 0$. Given a distribution ξ , a distribution ξ' is said to *absolute ε -fool* constraints B_1, \dots, B_k , if

$$|\mathbb{P}_{\xi'}[B_j] - \mathbb{P}_{\xi}[B_j]| < \varepsilon \quad \text{for all } j \in [k].$$

It is said to *relative ε -fool* constraints B_1, \dots, B_k , if

$$\mathbb{P}_{\xi'}[B_j] \in (1 \pm \varepsilon)\mathbb{P}_{\xi}[B_j] \quad \text{for all } j \in [k].$$

Obviously, because we are dealing with probabilities, which are in $[0, 1]$, relative ε -fooling is stronger than absolute ε -fooling. In [KK97], only relative ε -fooling is considered. We include absolute ε -fooling in our analysis, because even in this case, we get interesting results, and there may be algorithms for reducing support with an absolute error that require fewer work (processors and time) than those known for relative fooling.

One technique for reducing the support of a distribution so that the resulting one absolute or relative ε -fools a given set of constraints is known as *basis crashing*. We speak of *absolute*

⁷E.g., a solution approximating some objective function within a certain margin.

approximate basis crashing or *relative approximate basis crashing* depending on the kind of error we intend to achieve. We speak of *basis crashing* if $\varepsilon = 0$ (in which case relative and absolute error coincides). Basis crashing generally works by expressing the constraints as a set of linear equations and then looking for a basic solution, which, by the properties of a basic solution, leads to a support of cardinality $\leq k$, see [KK97, Thm. 2.2]. To our knowledge, the best algorithm for basis crashing ($\varepsilon = 0$) is due to Beling and Megiddo [BM98] and runs in $O(k^{1.62} |\xi|)$. Hence, if ξ has a large support (e.g., exponential in the input length), this technique alone does not yield a polynomial (or an \mathcal{NC} algorithm) for the original problem.

As a building-block for the divide-and-conquer approach, it suffices however to have an \mathcal{NC} implementation for the reduction of the support, even if the number of processors and running time depend (polynomially and logarithmically, respectively) on the size of the original distribution, provided that the support is reduced strongly enough.

We will give an algorithm for relative approximate basis crashing. (This, of course, yields one for absolute basis crashing.) However, we will also show how the divide-and-conquer algorithm works if only an algorithm for absolute approximate basis crashing is available.

The nice idea of [KK97] is to formulate relative approximate basis crashing as a lattice approximation problem. The latter can then be solved using the parallel algorithm from Theorem 1.16. Consider some sample space Ω , e.g., the set of all words W . The given distribution is a mapping $\xi : \Omega \rightarrow [0, 1]$ and the constraints are subsets of Ω , i.e., $B_i \subseteq \Omega$ for all $i \in [k]$. Denote the elements of the support $\text{supp}(\xi) = \{\omega_1, \dots, \omega_S\}$ and consider the constraints as subsets of $\text{supp}(\xi)$. If not already the case, we include the event B_0 consisting of all the elements from $\text{supp}(\xi)$ in the set of constraints. This will help in ensuring that we really get a distribution out of our construction. So we may have in fact $k + 1$ constraints. We have to pay attention here, because the term $O((k + 1)^{1 + \frac{1}{\eta}})$, later occurring in the processor bounds, is not $O(k^{1 + \frac{1}{\eta}})$, unless η is appropriately lower-bounded.

We use the algorithm from Theorem 1.16. Set $\varepsilon' := \frac{\varepsilon}{3}$, fix some $0 < \eta \leq \frac{1}{4}$ and choose δ such that $\delta \leq (\varepsilon'^2 h^{-1})^{\frac{1}{1-2\eta}}$ and δ is small enough as required in the theorem. Reviewing the requirement of the theorem, we see that if ε is small enough compared to γ^{-1} (which we may assume), a setting of δ such that

$$\frac{1}{\delta} = (\varepsilon'^{-2} h)^{\frac{1}{1-2\eta}} = O\left(\left(\frac{h}{\varepsilon^2}\right)^{\frac{1}{1-2\eta}}\right) \quad (1.40)$$

suffices.

We construct the lattice approximation instance. The constraint-sample matrix associated to $(B_i)_{i=1, \dots, k}$ is

$$\tilde{A} = (\tilde{a}_{ij})_{\substack{i=1, \dots, k \\ j=1, \dots, S}}, \quad \text{where } \tilde{a}_{ij} := \begin{cases} 1 & \text{if } \omega_j \in B_i \\ 0 & \text{else.} \end{cases}$$

Define a vector $b \in [0, 1]^k$ by

$$b_i := \mathbb{P}_\xi[B_i] \quad \text{for all } i \in [k],$$

and vectors $r, p \in [0, 1]^S$ by

$$\begin{aligned} r_j &:= \xi(\omega_j) \quad \text{for all } j \in [S], \\ p_j &:= \min \left\{ \max_{i \in [k]} \frac{\tilde{a}_{ij} r_j}{\delta b_i}, 1 \right\} \quad \text{for all } j \in [S], \end{aligned}$$

and finally set

$$A = (a_{ij})_{\substack{i=1,\dots,k \\ j=1,\dots,S}}, \quad \text{where } a_{ij} := \frac{\tilde{a}_{ij}r_j}{p_j} \quad i \in [k], j \in [S].$$

The lattice approximation problem is defined by A and p , meaning that $c := Ap$ is the vector of constraint probabilities b . By the algorithm from Theorem 1.16 we get a vector $q \in \{0, 1\}^S$ with a support bounded as in the theorem such that $|(Ap - Aq)_i| \leq \varepsilon'(Ap)_i$, i.e.,

$$(Aq)_i \in (1 \pm \varepsilon')(Ap)_i \quad \text{for all } i \in [k]. \quad (1.41)$$

Define $\tilde{\xi}'(\omega_j) := \frac{r_j}{p_j} q_j$ for all $j \in [S]$ and $\lambda := \sum_{j \in [S]} \tilde{\xi}'(\omega_j)$. The following setting yields a distribution

$$\xi'(\omega) := \begin{cases} \frac{1}{\lambda} \tilde{\xi}'(\omega) & \text{if } \omega \in \{\omega_1, \dots, \omega_S\} \\ 0 & \text{else} \end{cases} \quad \text{for all } \omega \in \Omega, \quad (1.42)$$

which is not far from the ‘‘pseudo’’ distribution $\tilde{\xi}'$, because, by the assumption that the event B_0 is among the constraints, we have

$$\lambda \in (1 \pm \varepsilon'). \quad (1.43)$$

THEOREM 1.17 *Let ξ be a distribution, $0 < \varepsilon < 1$, and consider constraints (i.e., events) B_1, \dots, B_k . A distribution ξ' can be constructed such that the given constraints are relative ε -fooled and*

$$|\xi'| = O\left(k \left(\frac{\log(k \log S)}{\eta \varepsilon^2}\right)^{\frac{1}{1-2\eta}}\right).$$

The construction can be done with $O(S^2 O(k \log S)^{1+\frac{1}{\eta}})$ processors in time

$$O\left(\frac{1}{\eta^2} \log^3 S \log^2(k \log S)\right).$$

Proof Let ξ' be as in (1.42). By (1.41), we see that the constraints are relative ε -fooled by ξ' : for $i \in [k]$ we have $(Ap)_i = b_i = \mathbb{P}_\xi[B_i]$ and $(Aq)_i = \lambda \mathbb{P}_{\xi'}[B_i]$. Hence $\lambda \mathbb{P}_{\xi'}[B_i] \in (1 \pm \varepsilon) \mathbb{P}_\xi[B_i]$. The choice of ε' and (1.43) give the desired $\mathbb{P}_{\xi'}[B_i] \in (1 \pm \varepsilon) \mathbb{P}_\xi[B_i]$.

We proceed to bound the support size of ξ' , which is the same as the support size of q . We know from Theorem 1.16 that $|q| \leq \bar{p} + O(\bar{p}^{\frac{1}{2} + \eta} \sqrt{h}) + 1$. Hence, we need an upper bound on \bar{p} . We say that $i \in [k]$ *owns* $j \in [S]$ if the term $\frac{\tilde{a}_{ij}r_j}{\delta b_i}$ is maximal (over all possible i). Fix by $\text{own}(j)$ an arbitrary i which owns j . We then have

$$\begin{aligned} \bar{p} &= \sum_{j \in [S]} p_j = \sum_{i \in [k]} \sum_{\substack{j \in [S] \\ \text{s.t. } i = \text{own}(j)}} p_j \leq \sum_{i \in [k]} \sum_{\substack{j \in [S] \\ \text{s.t. } i = \text{own}(j)}} \frac{\tilde{a}_{ij}r_j}{\delta b_i} \\ &= \sum_{i \in [k]} \frac{1}{\delta b_i} \sum_{\substack{j \in [S] \\ \text{s.t. } i = \text{own}(j)}} \tilde{a}_{ij}r_j \leq \sum_{i \in [k]} \frac{1}{\delta b_i} b_i = \sum_{i \in [k]} \frac{1}{\delta} = \frac{k}{\delta}. \end{aligned}$$

The claimed bound for the support follows from a calculation using (1.40) and the definition of h from Theorem 1.16.

The bounds on the number of processors and running time follow directly from 1.16. ■

REMARK 1.4 Karger and Koller [KK97, Th. 4.6] state a bound of $O\left(\frac{k \log k}{\varepsilon^2}\right)$ for the support size. We cannot see how to establish this bound for the following reasons. Our bound $O\left(k\left(\frac{\log(k \log S)}{\eta \varepsilon^2}\right)^{\frac{1}{1-2\eta}}\right)$ in the previous theorem relies on our rigorous analysis of the discrepancy algorithm, Theorem 1.13. The error parameter ε in Theorem 1.13, which is η here, can be found in the exponent as well as in the denominator (under the square root) in the discrepancy bound of our version of the theorem, while it only appears in the exponent in [MNN94]. This factor of $\frac{1}{\varepsilon}$ persists also in the bounds for the lattice approximation, and thus also must appear in the bounds for the support size (at first hidden in the quantity h). Hence one cannot simply substitute η by $o(1)$, as it is done in [KK97], where presumably the discrepancy result of [MNN94] was used which did not show the factor of $\frac{1}{\varepsilon}$ (which is $\frac{1}{\eta}$ here). In any case putting $\eta = o(1)$ is questionable, because the number of processors depends exponentially on $\frac{1}{\eta}$, and the running time depends on it polynomially. Our analysis of the discrepancy algorithm also introduces a dependence on the number of variables, which in this application is the size of the support, since we cannot assume “ $\Theta(\log n) = \Theta(\log m)$ ” in general. However, for $\eta = \frac{1}{16}$ we get the bound $O\left(k\frac{\log^{1.15}(k \log S)}{\varepsilon^{2.3}}\right)$. We have not optimized η here, so there might be room for some improvements, which we expect to be minor. Note that the constant hidden in the $O(\cdot)$ notation of this bound goes to infinity as η goes to zero.

In the following, we will assume that some deterministic \mathcal{NC} algorithm RED is given which can be used to reduce the support of a distribution while fooling a set of constraints. For example, we may use the basis crashing algorithm from Theorem 1.17.

Denote the output distribution of RED by $\text{RED}(\xi, \mathcal{C}, \varepsilon)$. The number of processors used is denoted by $\text{RED}_{\text{proc}}(|\xi|, k, \varepsilon)$ and the running time by $\text{RED}_{\text{time}}(|\xi|, k, \varepsilon)$. Define

$$\text{RED}_{\text{ratio}}(S, k, \varepsilon) := \sup_{\substack{\xi, \mathcal{C} \\ \text{s.t. } |\xi|=S \\ \text{and } |\mathcal{C}|=k}} \frac{|\text{RED}(\xi, \mathcal{C}, \varepsilon)|}{k}.$$

We assume RED_{proc} , RED_{time} , and $\text{RED}_{\text{ratio}}$ to be non-decreasing in the first two arguments, and non-increasing in the third argument. All these quantities are dependent on the particular choice for RED. If we choose the algorithm from Theorem 1.17 as RED, we have $\text{RED}_{\text{ratio}}(S, k, \varepsilon) = O\left(\left(\frac{\log(k \log S)}{\eta \varepsilon^2}\right)^{\frac{1}{1-2\eta}}\right)$, while for the choice of the exact algorithm of Beling and Megiddo, it would be $\leq 1 + o(1)$.

For the algorithm RED to be efficient, $\text{RED}_{\text{ratio}}$ must be small enough and appropriately dependent on S , so that the size of the support behaves well under a certain iteration. To explain this, fix k and ε and define

$$f(S) := k \cdot \text{RED}_{\text{ratio}}(S^2, k, \varepsilon). \quad (1.44)$$

The following scheme will later be realized in Algorithm 8. Think of some distribution of size S_0 . It is modified such that its support grows to at most S_0^2 . Its support is then reduced to size S_1 using RED, and then it is modified again so that its support grows to at most S_1^2 . Then it is treated with RED again, and so on. Doing so t times by definition never

(meaning: in no step during the iteration) gives a distribution of size larger than

$$\text{RED}_{\text{iter}}^t(S_0, k, \varepsilon) := \max_{\underbrace{t' \leq t}_{t' \text{ times}}} \{f(f(\dots f(S_0)))\}. \quad (1.45)$$

For fixed t , $\text{RED}_{\text{iter}}^t(S_0, k, \varepsilon)$ is non-decreasing in its first two arguments and non-increasing in the third.

In Corollary 1.4 we will give an analysis of the iteration behavior of the algorithm from Theorem 1.17, showing that the support grows not too much.

Transition Probabilities

Let $i \in [m]$ and $a, b \in V_i$. Denote by $\mathbb{P}_\xi[a \rightarrow_i b]$ the probability that starting from state a , the automaton \mathcal{A}_i will reach state b when it is given as input a word $w \in W$ generated according to distribution ξ . We call this the *transition probability* of i from a to b under ξ .

The transition probabilities where a is the initial state are the important properties of the automata. They are to be maintained approximately during the construction of a new distribution.

For $l \leq h \leq n$ we need the *intermediate transition probability* $\mathbb{P}_\xi[a \rightarrow_i b]^{l,h}$ defined as the probability for a transition from a to b when the input is generated according to the random variables X_l, \dots, X_h only. The extreme cases are $l = h$, where only one random variable is used, and $l = 1$ and $h = n$, where all random variables are used. We introduce the notation $W^{l,h} := [r]^{h-l+1}$.

A simple but important observation is that we can construct intermediate transition probabilities for larger words from smaller ones. If $l \leq g \leq h$, then

$$\mathbb{P}_\xi[a \rightarrow_i b]^{l,h} = \sum_{v \in V_i} \mathbb{P}_\xi[a \rightarrow_i v]^{l,g} \cdot \mathbb{P}_\xi[v \rightarrow_i b]^{g+1,h}. \quad (1.46)$$

Note that this extends to the case when ξ is any distribution with the property that

$$\xi(w_l, \dots, w_h) = \xi(w_l, \dots, w_g) \cdot \xi(w_{g+1}, \dots, w_h) \quad \forall (w_l, \dots, w_h) \in W^{l,h}. \quad (1.47)$$

DEFINITION 1.8 Let $\varepsilon \geq 0$. A distribution ρ is said to absolute ε -fool a set $\mathcal{A}_1, \dots, \mathcal{A}_m$ of automata w.r.t. a distribution ζ , if

$$|\mathbb{P}_\rho[s_i \rightarrow_i v] - \mathbb{P}_\zeta[s_i \rightarrow_i v]| < \varepsilon \quad \forall v \in V_i \quad \forall i \in [m].$$

It is said to relative ε -fool the automata if

$$\mathbb{P}_\rho[s_i \rightarrow_i v] \in (1 \pm \varepsilon)\mathbb{P}_\zeta[s_i \rightarrow_i v] \quad \forall v \in V_i \quad \forall i \in [m].$$

In other words, it ε -fools the automata if it ε -fools all the relevant transition events. (The divide-and-conquer algorithm will, as a side-effect, ε -fool all transition events, not only those starting at s_i .)

Divide-and-Conquer Algorithm

We present an algorithm that constructs a distribution that ε -fools a given set of automata w.r.t. a given distribution. The algorithm works for absolute as well as for relative errors, provided that an adequate algorithm RED is given. The analysis for the case of relative

error is taken from [KK97]. We have added the analysis in the case of an absolute error, which requires some more work.

For each $l \leq h \leq n$ we denote by $\mathcal{C}^{l,h}$ the set of constraints consisting of transition events of all automata under inputs generated according to X_l, \dots, X_h . We have

$$|\mathcal{C}^{l,h}| \leq \sum_{i \in [m]} |V_i|^2 =: k. \quad (1.48)$$

Note that k is polynomial in the length of the input instance \mathcal{I} , because the automata by assumption have only polynomially many states.

Define $V_{\max} := \max_{i \in [m]} |V_i|$ and

$$\epsilon(\varepsilon) := \begin{cases} \min \left\{ \frac{\varepsilon}{4}, \left(\frac{\varepsilon}{4V_{\max}} \right)^{\frac{1}{2}} \right\} & \text{for an absolute error} \\ \frac{\varepsilon}{4} & \text{for a relative error.} \end{cases}$$

The algorithm has to use smaller and smaller ε -values in each iteration. The function ϵ describes how these values decrease. It is crucial that the reciprocal of the ε -values stays polynomial. That is why we first take a look at the behavior of ϵ under iteration. Define for each $t \in \mathbb{N}$ the t th iterate by $\epsilon^{(t)}(\varepsilon) := \epsilon(\epsilon^{(t-1)}(\varepsilon))$ and $\epsilon^{(0)}(\varepsilon) := \varepsilon$.

PROPOSITION 1.4 Let $\varepsilon > 0$ and write $V := 4V_{\max}$. Then $\epsilon^{(\lceil \log n \rceil)}(\varepsilon) \geq \frac{\varepsilon}{4n^2 V}$ in the case of an absolute error and $\epsilon^{(\lceil \log n \rceil)}(\varepsilon) \geq \frac{\varepsilon}{4n^2}$ in the case of a relative error, hence the reciprocal is polynomial in the input length of \mathcal{I} .

Proof The case of relative error is easy; there we have $\epsilon^{(\lceil \log n \rceil)}(\varepsilon) = \frac{\varepsilon}{4^{\lceil \log n \rceil}} \geq \frac{\varepsilon}{4n^2}$.

Now consider the absolute error. By the assumption that the automata have polynomially many states, V is polynomial. Assume that starting from some value ε_1 , for t_1 iterations, the minimum is always attained by the first expression in the definition of ϵ . Then it is, for some t_2 iterations, attained by the second expression. For some t_3 iterations, it is again assumed by the first one, and so on. We have a sequence of numbers t_1, \dots, t_r such that $\sum_{s=1}^r t_s = \lceil \log n \rceil$, and the numbers with an odd index mark periods where the first expression matters and those with an even index mark periods where the second expression matters. A tedious calculation – the exponent of V is already significantly simplified in the second term of the following equation – shows that

$$\epsilon^{(\lceil \log n \rceil)}(\varepsilon) \geq \varepsilon^{(\frac{1}{2})^{\sum_{s \text{ even}} t_s}} 4^{-\sum_{s \text{ odd}} t_s} V^{-\sum_{s=1}^{\lceil \log n \rceil} \frac{1}{2^s}} \geq \varepsilon 4^{-\lceil \log n \rceil} V^{-1} \geq \frac{\varepsilon}{4n^2 V}. \quad \blacksquare$$

Algorithm 8: FOOL

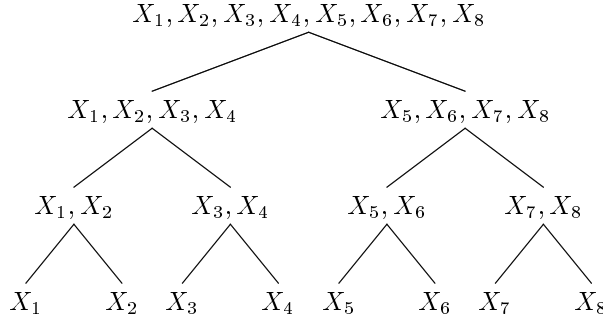
Input: Random variables X_1, \dots, X_h (with indices), parameter ε .

Output: Distribution ρ that absolute or relative ε -fools the automata, depending on ε and RED.

```

1  if  $l = h$  then
2    forall  $w \in W^{l,h}$  do                               /* note that here  $W^{l,h} = [r]$  */
3       $\tilde{\rho}(w) \leftarrow \mathbb{P}[X_l = w]$ ;
4    end
5  end
6  else
7     $g \leftarrow \lceil \frac{l+h}{2} \rceil$ ;
8     $\rho' \leftarrow \text{FOOL}(X_1, \dots, X_g, \varepsilon(\varepsilon))$  in parallel with  $\rho'' \leftarrow \text{FOOL}(X_{g+1}, \dots, X_h, \varepsilon(\varepsilon))$ ;
9    in parallel forall  $(w_l, \dots, w_h) \in W^{l,h}$ 
10   such that  $\rho'(w_l, \dots, w_g) > 0$  and  $\rho''(w_{g+1}, \dots, w_h) > 0$  do
11      $\tilde{\rho}(w_l, \dots, w_h) \leftarrow \rho'(w_l, \dots, w_g) \cdot \rho''(w_{g+1}, \dots, w_h)$ ;
12   end
13    $\rho \leftarrow \text{RED}(\tilde{\rho}, \mathcal{C}^{l,h}, \varepsilon)$ ;
14 end
15 return  $\rho$ ;
    
```

The following tree illustrates how FOOL works for eight random variables X_1, \dots, X_8 . At the bottom, distributions are constructed for each random variable; this is what happens in lines 1 to 5. Moving upwards in the tree means, for each branch, to “merge” two distributions; this is what happens in line 11.



The depth of the tree is $\log n$. Note that without the calls to RED in line 13, the size of the support could be $r^{2^{\log n}} = r^n$ at the top, which is superpolynomial hence making it impractical to reduce it by just one final call to RED.

THEOREM 1.18 Let $1 > \varepsilon > 0$ and set $\varepsilon_{\min} := \varepsilon^{\lceil \log n \rceil}(\varepsilon)$.

(i). The algorithm FOOL constructs a distribution ρ that ε -fools the automata and has size at most $S_{\max} := \text{RED}_{\text{iter}}^{\lceil \log n \rceil}(r, k, \varepsilon_{\min})$.

(ii). It runs using

$$O(n \cdot \max\{O(S_{\max}^2), \text{RED}_{\text{proc}}(S_{\max}, k, \varepsilon_{\min})\})$$

parallel processors in time

$$O(\log n \cdot \text{RED}_{\text{time}}(S_{\max}, k, \varepsilon_{\min})).$$

Because RED is an \mathcal{NC} algorithm, FOOL is so as well – as long as $\text{RED}_{\text{iter}}^{\lceil \log n \rceil}$ is polynomial in its three arguments.

Proof (i). The recursion depth is $\lceil \log n \rceil$. The statement about the size of the final distribution is clear by (1.45), noting that the treatment of the distribution in line 11 is expressed by taking the size of the support to the power of 2 in (1.44), that the size of the starting distributions is bounded by r , and that $\text{RED}_{\text{iter}}^{\lceil \log n \rceil}$ is assumed to be non-increasing in its third argument. So we can argue with ε_{\min} in $\text{RED}_{\text{iter}}^{\lceil \log n \rceil}(r, k, \varepsilon_{\min})$ and do not need to consider the intermediate values of “ ε ”.

We now show by induction on $h - l$ that the algorithm has the required fooling property. If $h - l = 0$, this is easily seen. The distribution $\tilde{\rho}$ constructed in lines 2 to 4 exactly reflects the transition probabilities (of length one) for the random variable in question.

First consider an absolute error, let $h - l > 0$ and assume that the algorithm works correctly on smaller inputs. This means that ρ' and ρ'' have the property to absolute $\varepsilon(\varepsilon)$ -fool the (intermediate) transition events for X_l, \dots, X_g and X_{g+1}, \dots, X_h respectively. The distribution $\tilde{\rho}$ defined by the multiplications in line 11 by this very definition has the property (1.47). Hence we can use (1.46). This yields for all $i \in [m]$ and $a, b \in V_i$

$$\begin{aligned}
\mathbb{P}_{\tilde{\rho}}[a \rightarrow_i b] &= \sum_{v \in V_i} \mathbb{P}_{\rho'}[a \rightarrow_i v] \cdot \mathbb{P}_{\rho''}[v \rightarrow_i b] \\
&\leq \sum_{v \in V_i} (\mathbb{P}_{\zeta}[a \rightarrow_i v]^{l,g} + \varepsilon(\varepsilon)) \cdot (\mathbb{P}_{\zeta}[v \rightarrow_i b]^{g+1,h} + \varepsilon(\varepsilon)) \\
&\leq \left(\sum_{v \in V_i} \mathbb{P}_{\zeta}[a \rightarrow_i v]^{l,g} \cdot \mathbb{P}_{\zeta}[v \rightarrow_i b]^{g+1,h} \right) \\
&\quad + \underbrace{\varepsilon(\varepsilon) \sum_{v \in V_i} \mathbb{P}_{\zeta}[a \rightarrow_i v]^{l,g}}_{=1} + \underbrace{\varepsilon(\varepsilon) \sum_{v \in V_i} \mathbb{P}_{\zeta}[v \rightarrow_i b]^{g+1,h}}_{=1} + |V_i| \varepsilon(\varepsilon)^2 \\
&\leq \mathbb{P}_{\zeta}[a \rightarrow_i b]^{l,h} + 2 \frac{\varepsilon}{4} + |V_i| \frac{\varepsilon}{4V_{\max}} \\
&= \mathbb{P}_{\zeta}[a \rightarrow_i b]^{l,h} + \frac{3}{4} \varepsilon.
\end{aligned}$$

A corresponding lower bound can be established in the same way. The call to RED introduces another absolute error of at most $\frac{\varepsilon}{4}$, resulting in the distribution ρ to absolute ε -fool the (intermediate) transition events.

Consider now the case of a relative error. The distributions ρ' and ρ'' have the property to relative $\varepsilon(\varepsilon)$ -fool the (intermediate) transition events for X_l, \dots, X_g and X_{g+1}, \dots, X_h respectively, where $\varepsilon(\varepsilon) = \frac{\varepsilon}{4}$. The distribution $\tilde{\rho}$ defined by the multiplications in line 11 by this very definition has the property (1.47). Hence we can use (1.46). This yields for all $i \in [m]$ and $a, b \in V_i$

$$\begin{aligned}
\mathbb{P}_{\tilde{\rho}}[a \rightarrow_i b] &= \sum_{v \in V_i} \underbrace{\mathbb{P}_{\rho'}[a \rightarrow_i v]}_{\in (1 \pm \frac{\varepsilon}{4}) \mathbb{P}_{\zeta}[a \rightarrow_i v]^{l,g}} \cdot \underbrace{\mathbb{P}_{\rho''}[v \rightarrow_i b]}_{\in (1 \pm \frac{\varepsilon}{4}) \mathbb{P}_{\zeta}[v \rightarrow_i b]^{g+1,h}} \\
&\in \left(1 \pm \frac{\varepsilon}{4}\right)^2 \sum_{v \in V_i} \mathbb{P}_{\zeta}[a \rightarrow_i v]^{l,g} \cdot \mathbb{P}_{\zeta}[v \rightarrow_i b]^{g+1,h} \\
&= \left(1 \pm \frac{\varepsilon}{4}\right)^2 \mathbb{P}_{\zeta}[a \rightarrow_i b]^{l,h}.
\end{aligned}$$

The call to RED introduces another relative error of $\frac{\varepsilon}{4}$, resulting in

$$\mathbb{P}_\rho [a \rightarrow_i b] \in (1 \pm \frac{\varepsilon}{4})^3 \mathbb{P}_\zeta [a \rightarrow_i b]^{l,h}.$$

Because (calculations omitted)

$$1 - \varepsilon \leq 1 - \varepsilon + \frac{27}{64}\varepsilon^2 \leq (1 - \frac{\varepsilon}{4})^3 \quad \text{and} \quad (1 + \frac{\varepsilon}{4})^3 \leq 1 + \frac{61}{64}\varepsilon \leq 1 + \varepsilon,$$

the distribution ρ thus relative ε -fools the (intermediate) transition events.

(ii). First consider the time complexity. The statement follows from the fact that the recursion depth is $\lceil \log n \rceil$, the only expensive task in each recursion is the call to RED, and the definition of ε_{\min} . Keep in mind that we assumed RED_{time} to be non-decreasing in the first two, and non-increasing in the third argument.

Now consider the number of processors. There will only be n instances of FOOl running in parallel at a time. There are only $O(S_{\max}^2)$ multiplications to do in line 11, and with the same argument as above, a call to RED never uses more than RED_{proc}($S_{\max}, k, \varepsilon_{\min}$) processors. ■

COROLLARY 1.4 Consider relative ε -fooling. If RED is chosen as the algorithm from Theorem 1.17, algorithm FOOl yields a distribution of size

$$S_{\max} = O \left(k \left(n^4 \frac{\log(k \log^{(*)})}{\eta \varepsilon^2} \right)^{\frac{1}{1-2\eta}} \right), \tag{1.49}$$

using

$$O \left(nk^2 \left(n^4 \frac{\log(k \log^{(*)})}{\eta \varepsilon^2} \right)^{\frac{2}{1-2\eta}} (O(k) \log^{(*)})^{1+\frac{1}{\eta}} \right)$$

parallel processors in $O \left(\frac{1}{\eta^2} \log^{(*)} \right)$ time.

To prove this, we need an upper bound on RED_{iter} for the case of the algorithm from Theorem 1.17. Recall (1.44) and (1.45). The following technical proposition examines this iteration in a general setting.

PROPOSITION 1.5 Let a, b, c, p, S be large enough⁸ such that $6p + 1 \leq b$, and define $f(x) := a \log^p(b \log(cx^2))$. Then, for the t th iterate of f , for all $t \in \mathbb{N}$, we have

$$\begin{aligned} f^{(t)}(S) &\leq a \log^p(b^3(\log(ca^2) + \log(cS^2))) = a \log^p(2b^3(\log(acS))) \\ &= O(a \log^p(b \log(acS))). \end{aligned} \tag{1.50}$$

Proof Set $A := \log(ca^2) + \log(cS^2)$. We clearly have $f(S) \leq a \log^p(b^3 A)$ and furthermore

$$f(a \log^p(b^3 A)) = a \log^p(b \log(ca^2 \log^{2p}(b^3 A)))$$

⁸It is sufficient to choose $a \geq 2$, $c \geq 1$, $b \geq 12$, $S \geq 2$.

$$\begin{aligned}
&= a \log^p(b \log(ca^2) + b \log \log^{2p}(b^3 A)) \\
&\leq a \log^p(bA + 2pb \log \log(b^3 A)) \\
&\leq a \log^p(bA + 2pb(\log(\log(b^3)) + \log(A))) \\
&\leq a \log^p(bA + 4pb^2 + 2pb \log(A)) \\
&\leq a \log^p((b + 4pb^2 + 2pb)A) \leq a \log^p((1 + 6p)b^2 A) \leq a \log^p(b^3 A).
\end{aligned}$$

The second and third assertion is a simple calculation. \blacksquare

The bound of the proposition can certainly be improved. We proceed to prove the Corollary.

Proof (of Corollary 1.4)

In terms of the previous proposition, we have $a = O(k(\eta \varepsilon_{\min}^2)^{-\frac{1}{1-2\eta}})$, $b = k$, $c = 1$, and $p = \frac{1}{1-2\eta}$. The starting value is $S = r$, and we have $\varepsilon_{\min}^{-1} \leq \frac{4n^2}{\varepsilon}$. A calculation using the previous proposition leads to stated bound on S_{\max} . Using the equations for RED_{proc} and RED_{time} from Theorem 1.17 and $\frac{1}{1-2\eta} = O(1)$ we get the stated bounds for processors and running time. \blacksquare

REMARK 1.5 Fix one automaton A and let its final states give the values of some d -valued random variable Y . If we have a relative error, then

$$\mathbb{E}_{\zeta}[Y] = \sum_{y \in [d]} y \underbrace{\mathbb{P}_{\zeta}[Y=y]}_{\in (1 \pm \varepsilon) \mathbb{P}_{\rho}[Y=y]} \in (1 \pm \varepsilon) \sum_{y \in [d]} y \mathbb{P}_{\rho}[Y=y] = (1 \pm \varepsilon) \mathbb{E}_{\rho}[Y].$$

So, in other words, expectations are fooled just like probabilities. In the case of an absolute error, we have

$$\begin{aligned}
\mathbb{E}_{\zeta}[Y] &= \sum_{y \in [d]} y \mathbb{P}_{\zeta}[Y=y] \leq \sum_{y \in [d]} y(\mathbb{P}_{\rho}[Y=y] + \varepsilon) \\
&= \sum_{y \in [d]} y \mathbb{P}_{\rho}[Y=y] + \sum_{y \in [d]} y \varepsilon \leq \mathbb{E}_{\rho}[Y] + d^2 \varepsilon,
\end{aligned}$$

and a similar lower bound. Hence, in case of absolute error, if expectations are to be ε -fooled, the fooling of the probabilities has to be done with an “ ε ” not larger than $\frac{\varepsilon}{d^2}$. If the automata have polynomially many states, d is polynomial, and hence such a requirement is reasonable.

1.5 Parallel Algorithms for Packing Integer Programs

1.5.1 Impact of k -Wise Independence: Multidimensional Bin Packing

We choose the multidimensional bin packing problem as an example for a packing type problem for which a \mathcal{NC} algorithm with the method of k -wise independence can be given. The material for this subsection is based on Srivastav, Stangier [SS97a]. In the next section we will see the impact of the approach of Srinivasan [Sri01a] for general integer programs of packing type.

DEFINITION 1.9 (Bin Packing Problem $\text{BIN}(\vec{b}, m)$)

Let $m, n \in \mathbb{N}$, and let $\vec{b} = (b_1, \dots, b_m) \in \mathbb{N}^m$, called the *bin size vector*. Given vectors $\vec{v}_1, \dots, \vec{v}_n \in [0, 1]^m$, pack⁹ all vectors in a minimum number of bins such that in each bin B and for each coordinate $i \in [m]$ the condition $\sum_{\vec{v}_j \in B} v_{ji} \leq b_i$ holds. Define

$$C := \left\lceil \max_{i \in [m]} \frac{1}{b_i} \sum_{j=1}^n v_{ji} \right\rceil. \quad (1.51)$$

Observe that C is the minimum number of bins, if fractional packing is allowed. Let C_{opt} be the number of bins in an optimal integer solution.

$\text{BIN}(1, m)$ is the multidimensional bin packing problem, and $\text{BIN}(1, 1)$ is the classical bin packing problem. The intention behind the formulation with a bin size vector \vec{b} is to analyze the relationship between bin sizes and polynomial-time approximability of the problem.

The known polynomial-time approximation algorithms for resource constrained scheduling applies also to multidimensional bin packing and are due to Garey, Graham, Johnson, Yao [GGJY76] and Röck and Schmidt [RS83]. Garey et al. constructed with the First-Fit-Decreasing heuristic a packing with C_{FFD} number of bins which asymptotically is a $(m + \frac{1}{3})$ -factor approximation, i.e., there is a non-negative integer C_0 such that $C_{FFD} \leq C_{opt}(m + \frac{1}{3})$ for all instances with $C_{opt} \geq C_0$. De la Vega and Lueker [dlVL81] improved this result presenting for every $\varepsilon > 0$ a linear-time algorithm which achieves an asymptotic approximation factor of $m + \varepsilon$. For restricted bin packing and scheduling Błazewicz and Ecker gave a linear-time algorithm (see [BESW93]). The first polynomial-time approximation algorithm for resource constrained scheduling and multidimensional bin packing with a constant factor approximation guarantee has been given in [SS97b]: For every $\varepsilon > 0$, $\frac{1}{\varepsilon} \in \mathbb{N}$, a bin packing of size at most $\lceil (1 + \varepsilon)C_{opt} \rceil$ can be constructed in strongly polynomial time, provided that

$$b_i \geq \frac{3(1 + \varepsilon)}{\varepsilon^2} \log(8nd). \quad (1.52)$$

This approximation guarantee is independent of the dimension m . For $\varepsilon = 1$ this gives a sequential 2-factor approximation algorithm for multidimensional bin packing under the above assumption on b .

There is no obvious way to parallelize this algorithm. In this section we will show that at least in some special cases there is an \mathcal{NC} approximation algorithm. The algorithm is based on the method of $\log^c n$ -wise independence. The parallel bin packing algorithm has 4 steps.

Step 1: *Finding an optimal fractional bin packing in parallel.* We wish to apply randomized rounding and therefore have to generate an optimal fractional solution. The following integer program is equivalent to the multidimensional bin packing problem.

$$\begin{aligned} \min D \\ \sum_j v_{ji} x_{jz} &\leq b_i && \forall i \in [m], z \in [D] \\ \sum_z x_{jz} &= 1 && \forall j \in [n] \\ x_{jz} &= 0 && \forall j \text{ and } z > D \\ x_{jz} &\in \{0, 1\}. \end{aligned}$$

⁹Packing simply means to find a partitioning of vectors in a minimum number of sets so that the vectors in each partition (= bin) satisfy the upper bound conditions of the definition.

Note that this integer program is not an integer linear program, thus we cannot solve it with standard methods. But, fortunately, its optimal fractional solution is given by C , see (1.51). This is easily checked by setting $x_{jz} = \frac{1}{C}$ for all vectors j and all bins $z \in [C]$.

Step 2: *Enlarged fractional bin packing.* In order to define randomized rounding we enlarge the fractional bin packing. The reason for such an enlargement is that for the analysis of the rounding procedure we need some room. Fix $\alpha > 1$ and define

$$d := 2^{\lceil \log(\alpha C) \rceil}, \quad (1.53)$$

hence $d \leq 2\alpha C$.

The new fractional assignments of vectors to bins are $\tilde{x}_{jz} = \frac{1}{d}$ for all $j \in [n]$, $z \in [d]$. Note that this assignment defines a valid fractional bin packing, even with tighter bounds

$$\sum_{j=1}^n v_{ji} \tilde{x}_{jz} \leq \frac{b_i}{\alpha} \quad (1.54)$$

for all $i \in [m]$, $z \in [d]$.

Step 3: *Randomized rounding.* The sequential randomized rounding procedure is to assign randomly and independently vector j to bin z with uniform probability $\frac{1}{d}$. With the method of limited independence we parallelize and derandomize this rounding scheme in the next step.

Step 4: *Derandomized rounding in \mathcal{NC} .* We apply the parallel conditional probability method for multivalued random variables, Theorem 1.11.

First, C can be computed in parallel with standard methods:

LEMMA 1.5 C can be computed with $O(nm)$ processors in $O(\log(nm))$ time.

Recall the definitions preceding Theorem 1.11. For integers $r, z \in [d]$ define the indicator function

$$[r = z] := \begin{cases} 1 & \text{if } r = z \\ 0 & \text{else.} \end{cases}$$

For all $(i, z, j) \in [m] \times [d] \times [n]$ define the functions g_{izj} by

$$g_{izj} : \mathbb{R} \rightarrow \mathbb{R}, \quad r \mapsto v_{ji}([r = z] - \frac{1}{d}),$$

and, for an even k to be specified later, f_{iz} and F as in Theorem 1.11. We then have

$$F(x_1, \dots, x_n) = \sum_{(i,z) \in [m] \times [d]} \underbrace{\left| \sum_{j=1}^n v_{ji}([x_j = z] - \frac{1}{d}) \right|^k}_{= f_{iz}(x_1, \dots, x_n)}$$

for all $(x_1, \dots, x_n) \in [d]^n$. Furthermore, we need estimates of the k th moments.

LEMMA 1.6 Let $\frac{1}{\log n} \leq \tau < \frac{1}{2}$ and $k := 2 \left\lceil \frac{\log(6\alpha nm)}{2\tau \log n} \right\rceil$. Then

$$\mathbb{E} [F(X_1, \dots, X_n)]^{\frac{1}{k}} \leq n^{\frac{1}{2} + \tau} (\log(6\alpha nm) + 2)^{\frac{1}{2}}$$

for k -wise independent $[d]$ -valued random variables X_1, \dots, X_n .

Proof Using Theorem 1.5 we can show as in the proof of Lemma 1.3

$$\mathbb{E} [|f_{iz}(X_1, \dots, X_n)|^k] \leq 2\left(\frac{k}{2}\right)! \left(\frac{n}{2}\right)^{\frac{k}{2}}. \quad (1.55)$$

Robbins exact Stirling formula shows the existence of a constant γ_n with $\frac{1}{12n+1} \leq \gamma_n \leq \frac{1}{12n}$ so that $n! = \left(\frac{n}{e}\right)^n \sqrt{2\pi n} e^{\gamma_n}$ ([Bol98], page 4), thus $n! \leq 3\left(\frac{n}{e}\right)^n \sqrt{n}$. Furthermore, for all $k \geq 2$, we have $2\sqrt{k}(4e)^{-\frac{k}{2}} \leq 1$. With these bounds the right hand side of inequality (1.55) becomes $3(kn)^{\frac{k}{2}}$. The claimed bound now follows by summing over all $i \in [m]$, $z \in [d]$, using the trivial bound $C \leq n$ and taking the $\frac{1}{k}$ th root. \blacksquare

The main result is:

THEOREM 1.19 (Srivastav, Stangier 1997 [SS97a])

Let $\frac{1}{\log n} \leq \tau < \frac{1}{2}$ and $\alpha > 1$ and suppose that $b_i \geq \alpha(\alpha - 1)^{-1} n^{\frac{1}{2} + \tau} (\log(6\alpha nm) + 2)^{\frac{1}{2}}$ for all $i \in [m]$. Then there is an \mathcal{NC} algorithm that runs on $O(n^3(nm)^{\frac{1}{\tau} + 1})$ parallel processors and finds in $O\left(\left(\frac{\log^2 n \log(nm)}{\tau}\right)^4\right)$ time a bin packing of size at most $2^{\lceil \log(\alpha C) \rceil} \leq 2\alpha C$.

Proof Set $k = 2 \left\lceil \frac{\log(6\alpha nm)}{2\tau \log n} \right\rceil$ and let d be as defined in (1.53), so $d \leq 2\alpha C \leq 2\alpha n$. Thus for fixed α we have $d^k = O(n^k)$ (we are not interested in large values of α). Also, because of $\frac{1}{\log n} \leq \tau$, we have $O(1)^{\frac{1}{\tau}} = O(1)$. By Theorem 1.11 we can construct integers $\hat{x}_1, \dots, \hat{x}_n$, where $\hat{x}_j \in [d]$ for all j such that

$$F(\hat{x}_1, \dots, \hat{x}_n) \leq \mathbb{E}[F(X_1, \dots, X_n)] \quad (1.56)$$

holds, using

$$O(n^{k+2}m) = O(n^3(nm)^{\frac{1}{\tau} + 1})$$

parallel processors in time

$$\begin{aligned} & O\left(\underbrace{\log(n \log d)}_{=O(\log n)} \cdot \underbrace{(k^4)}_{=O(\log^4 n)} \cdot \underbrace{\log^3(n \log d) \log n + k \log d \log(mdn^k)}_{=O(\log^4 n)}\right) \\ &= O\left(\log^4 n \cdot \left(\frac{\log(6\alpha nm)}{\tau}\right)^4 \frac{\log^4 n}{\log^4 n} \log n + \left(\frac{\log(6\alpha nm)}{\tau}\right)^2 \log(nm)\right) \\ &= O\left(\left(\frac{\log^2 n \log(nm)}{\tau}\right)^4\right). \end{aligned}$$

To complete the proof we must show that the vector $(\hat{x}_1, \dots, \hat{x}_n)$ defines a valid bin packing. This is seen as follows. By definition of the function F we have for all coordinates i and all bins $z \in [d]$

$$\begin{aligned} \left| \sum_{j=1}^n v_{ji} \left(x_{jz} - \frac{1}{d}\right) \right| &\leq F(x_1, \dots, x_n)^{\frac{1}{k}} \leq \mathbb{E}[F(X_1, \dots, X_n)]^{\frac{1}{k}} \\ &\leq n^{\frac{1}{2} + \tau} (\log(6\alpha nm) + 2)^{\frac{1}{2}}. \end{aligned} \quad (1.57)$$

(the last inequality follows from Lemma 1.6). Hence, using (1.57), (1.54), and the assumption $b_i \geq \alpha(\alpha - 1)^{-1}n^{\frac{1}{2}+\tau}(\log(6\alpha nm) + 2)^{\frac{1}{2}}$, we get for all coordinates i and all bins $z \in [d]$

$$\begin{aligned} \sum_{j=1}^n v_{ji}x_{jz} &\leq \left| \sum_{j=1}^n v_{ji}(x_{jz} - \frac{1}{d}) \right| + \sum_{j=1}^n v_{ji} \frac{1}{d} \\ &\leq n^{\frac{1}{2}+\tau}(\log(6\alpha nm) + 2)^{\frac{1}{2}} + \frac{b_i}{\alpha} \\ &\leq (1 - \frac{1}{\alpha})b_i + \frac{b_i}{\alpha} = b_i, \end{aligned}$$

and the theorem is proved. \blacksquare

Note that we cannot achieve a polynomial-time approximation better than a factor of $\frac{4}{3}$, even if $b_i = \Omega(\log(nm))$ for all bounds b_i and $\tau < \frac{1}{2}$.

THEOREM 1.20 (Srivastav, Stangier 1997 [SS97a])

Let $0 < \tau < \frac{1}{2}$ and $3 \leq \Delta$ be a fixed integer. Under the assumption $b_i = \Omega(n^{\frac{1}{2}+\tau}\sqrt{\log(nm)})$ for all i , it is \mathcal{NP} -complete to decide whether or not there exists a multidimensional bin packing of size Δ .

1.5.2 Impact of Automata Fooling: Srinivasan's Parallel Packing

Let $A \in ([0, 1] \cap \mathbb{Q})^{m \times n}$, $b \in ((1, \infty) \cap \mathbb{Q})^m$, and $w \in ([0, 1] \cap \mathbb{Q})^n$ such that $\max_{i \in [n]} w_i = 1$. We consider the *packing integer program*

$$\begin{aligned} \max w \cdot x \\ Ax \leq b \\ x \in \{0, 1\}^n \end{aligned} \tag{PIP}$$

This is an \mathcal{NP} -hard problem, and hence good efficient approximation algorithms are sought. Let $x^* \in ([0, 1] \cap \mathbb{Q})^n$ be an optimal solution to the LP relaxation of the PIP, i.e., to the following linear program:

$$\begin{aligned} \max w \cdot x \\ Ax \leq b \\ x \in [0, 1]^n \end{aligned}$$

By OPT^* we denote the value of x^* , i.e., $w \cdot x^*$. Set $B := \min\{b_i; i \in [m]\}$ (which is > 1).

We briefly review previous results for PIPs. Let $0 < \varepsilon < 1$. If $B \geq 12\varepsilon^{-2} \ln(2m)$, then a vector $x \in \{0, 1\}^n$ with $Ax \leq b$ and $w \cdot x \geq (1 - \varepsilon)\text{OPT}^*$ can be constructed in deterministic polynomial time [SS96]. This result is based on an efficient derandomization of Chernoff-Hoeffding bounds. Srinivasan's work [Sri99] extends the range for the right-hand side b where a polynomial time approximation algorithm does exist.

THEOREM 1.21 (Srinivasan 1995, 1999)

There are constants K_0 and K_1 such that the following holds. A solution $x \in \{0, 1\}^n$ for a given PIP can be constructed in polynomial time with

$$w \cdot x \geq K_0 \min \left\{ \text{OPT}^*, \left(K_1 \frac{\text{OPT}^*}{m^{\frac{1}{B}}} \right)^{\frac{B}{B-1}} \right\}.$$

REMARK 1.6 (Asymptotics of the approximation bounds)

Let us briefly compare Srinivasan’s bound with the bounds of [SS96]. If $B = B(m)$ is a function of m with $B(m) \rightarrow \infty$, we may assume that $\frac{B}{B-1}$ is $1 + o(1)$. So

$$\left(K_1 \frac{\text{OPT}^*}{m^{\frac{1}{B}}}\right)^{\frac{B}{B-1}} = (K_1 \text{OPT}^*)^{1+o(1)} e^{-\frac{\ln m}{B-1}} \approx K_1 \text{OPT}^* e^{-\frac{\ln m}{B-1}}.$$

Then the approximation quality is governed by $e^{-\frac{\ln m}{B-1}}$. For $B = c \ln(m) + 1$, c a constant, we have $e^{-\frac{\ln m}{B-1}} = e^{-c}$, leading to a constant approximation factor $K_1 e^{-c}$. For any $c = \Omega(\varepsilon^{-2})$, $\varepsilon > 0$, e^{-c} is exponentially small, thus inferior to the $1 - \varepsilon$ factor of [SS96]. But Srinivasan’s bound gives a constant factor approximation for any constant $c > 0$, while in [SS96], the “constant” is at least $12\varepsilon^{-2}$. On the other hand, if $\frac{\ln m}{B-1}$ tends to ∞ , for example if $B = O(\ln \ln(m) + 1)$, we have that $e^{-\frac{\ln m}{B-1}} = e^{-\omega(m)} = o(1)$, leading to an asymptotically zero approximation factor. In this range of B , we have essentially no approximation guarantee.

In the rest of this section, we present Srinivasan’s derandomized \mathcal{NC} approximation algorithm [Sri01a]. We first describe a randomized approximation algorithm but will not try to make a statement about its success probability. Instead we will right away derandomize it using the automata fooling technique from Section 1.4. The randomized version merely serves as an illustration of the idea behind the to be constructed automata.

The Idea

To make the fractional optimum x^* an integral solution, consider the following rounding and alteration scheme. Fix a parameter $\lambda \geq 3$, which will be specified more precisely later. For each $j \in [n]$, set

$$y_j := \begin{cases} 1 & \text{with probability } \frac{x_j^*}{\lambda} \\ 0 & \text{with probability } 1 - \frac{x_j^*}{\lambda}. \end{cases} \tag{1.58}$$

This way we get an integral vector y with

$$\mathbb{E}[w \cdot y] = \frac{\text{OPT}^*}{\lambda}. \tag{1.59}$$

However, y might be infeasible – some of the m constraints in “ $Ay \leq b$ ” might be violated. To repair this, y is modified in a greedy manner to fulfill all the constraints. Initialize $x := y$. For each row $i \in [m]$ of the matrix A sort the entries $(a_{ij})_{j=1, \dots, m}$ in decreasing order

$$a_{i,j_{1,i}} \geq a_{i,j_{2,i}} \geq \dots a_{i,j_{m,i}}.$$

Then, starting with the first position $j = j_{1,i}$, set x_j to 0 until the constraint “ $a_i \cdot x \leq b_i$ ” is fulfilled¹⁰. This is done in parallel for all rows i , without any communication between the processes. Hence, there might in fact occur unnecessary alterations.

Let the random variables Y_1, \dots, Y_m denote the number of alterations for each row. (They are called this way because they are fully determined by the intermediate solution y .) Let I

¹⁰ a_i denotes the i th row of A .

denote the input length of the PIP (clearly, $\max\{n, m\} \leq I$). Let $C_0 \geq 1$ be a large enough constant and set

$$B' := \min\{C_0 \log I, B\} (> 1) \quad (1.60)$$

and take

$$\lambda := K_0 \max\left\{1, \left(\frac{K_1 m}{\text{OPT}^*}\right)^{1/(B'-1)}\right\}, \quad (1.61)$$

where $K_1 > 1$ and $K_0 \geq 1$ are some constants. Then, the following bounds can be proven using the Chernoff bound from Theorem 1.2.

$$\mathbb{E}[Y_i] = O\left(\left(\frac{e}{\lambda}\right)^{B'+1}\right) \quad \forall i \in [m]. \quad (1.62)$$

This allows an estimation for the expected value of the objective function after the alteration process.

$$\mathbb{E}[w \cdot x] = \frac{\text{OPT}^*}{\lambda} - \sum_{i=1}^m \mathbb{E}[Y_i] = \frac{\text{OPT}^*}{\lambda} - m \cdot O\left(\left(\frac{e}{\lambda}\right)^{B'+1}\right).$$

Derandomization

Derandomization is done via the construction of finite deterministic automata $\mathcal{A}_1, \dots, \mathcal{A}_{m+1}$ with polynomially many states which simulate the randomized algorithm described above. We need certain assumptions on the precision of the occurring numbers.

All entries in A , w , x^* , and $\frac{x^*}{\lambda}$ are assumed to be rationals with denominator 2^d , where $d = \Theta(\log I)$, and all entries in A are assumed to be non-positive powers of 2. (1.63)

This can be achieved by appropriate modifications of these numbers, and a simple observation how we can get a feasible solution for the original instance from a solution for the modified one.

The automata (together with an appropriate distribution ζ) will simulate the randomized rounding, where y is constructed from x^* , as follows.

- As inputs, all possible vectors from $[0, 1]^n$ with rational entries, each with denominator 2^d are accepted. They are represented as elements from $\{0, \dots, 2^d\}^n$.
- Inputs are created randomly according to ζ , which is chosen to be the uniform distribution.
- Let t_1, \dots, t_n be such an input. For each l , the automata check whether $t_l \leq 2^d \mathbb{E}[y_l] - 1$ and if so, behave as if $y_l = 1$. Otherwise, they behave as if $y_l = 0$.

The automata are layered with $n + 1$ layers each, numbered from 0 to n . Transitions are allowed only from one layer to the next. Layer 0 has one state only – the starting state. The automata being in level l intuitively means that variables y_1, \dots, y_l (and so x_1, \dots, x_l) have been fixed. This will become clearer in the following. Automata $\mathcal{A}_1, \dots, \mathcal{A}_m$ correspond to the alteration process (where x is constructed from y) and \mathcal{A}_{m+1} corresponds to the randomized rounding (where y is constructed from x^*).

We start by describing \mathcal{A}_{m+1} , which is the simplest. For each layer $l \in \{0, \dots, n\}$, although there are exponentially many choices for a 0/1-vector (y_1, \dots, y_l) , the following set Λ_l is of polynomial size only

$$\Lambda_l := \left\{ \left(\sum_{j=1}^l w_j y_j, l \right); (y_1, \dots, y_l) \in \{0, 1\}^l \right\} \subseteq \mathbb{Q}_{\geq 0} \times \{l\}.$$

This is true, because all these values are between 0 and n and all have denominator $2^d = 2^{\Theta(\log I)} = \Theta(I)$. Hence, only $O(nI)$ values are possible. The states in layer l are the elements from the set Λ_l .

For the transitions, assume that $l < n$ and consider a state $(\omega, l) \in \Lambda_l$. There are $2^d \mathbb{E}[y_{l+1}]$ edges going from (ω, l) to $(\omega + w_{l+1}, l+1) \in \Lambda_{l+1}$, with labels $0, \dots, 2^d \mathbb{E}[y_{l+1}] - 1$. This corresponds to the case “ $y_{l+1} = 1$ ”. Then, there are $2^d - (2^d \mathbb{E}[y_{l+1}] - 1)$ edges from (ω, l) to $(\omega, l+1) \in \Lambda_{l+1}$ with labels $2^d \mathbb{E}[y_{l+1}], \dots, 2^d$. This corresponds to the case “ $y_{l+1} = 0$ ”.

Note that by assumption, $\mathbb{E}[y_{l+1}] = \frac{x^*}{\lambda}$ is a fraction with denominator 2^d .

We now turn to automata $\mathcal{A}_1, \dots, \mathcal{A}_m$. Their states have to express the number of variables which have to be altered (i.e., set to 0). \mathcal{A}_i will essentially express Y_i for $i \in [m]$. Fix an $i \in [m]$ now. For given y_1, \dots, y_l , we collect variables x_j that correspond to matrix entries a_{ij} of certain magnitudes and sum them up:

$$U_k(y_1, \dots, y_l) := \sum_{\substack{j \leq l \\ \text{s.t. } a_{ij} = 2^{-k}}} x_j \quad \text{for all } k \in \{0, \dots, d\}. \quad (1.64)$$

Denote the vectors

$$T(y_1, \dots, y_l) := (U_0(y_1, \dots, y_l), \dots, U_d(y_1, \dots, y_l)). \quad (1.65)$$

From $T(y_1, \dots, y_n)$ the value of Y_i for $y = (y_1, \dots, y_n)$ can be inferred. A first idea is to define the states in layer l of \mathcal{A}_i to be all possible values of $T(y_1, \dots, y_l)$ with (y_1, \dots, y_l) ranging over all elements of $\{0, 1\}^l$. This, however, can lead to superpolynomial state sets.

The solution is, intuitively, to summarize unimportant information so that the relevant information can be expressed more succinct. Before we start with that, one more assumption on the PIP has to be made. We need that $b_i \leq C_0 \log I$ (for all i). This can be achieved by multiplying the offending constraints “ $a_i \cdot x \leq b_i$ ” with $\frac{C_0 \log I}{b_i}$. Note that this motivates (1.60). If this reduces the value of B , i.e., if $B' = \frac{C_0 \log I}{b_i}$, the value of λ increases (reducing the approximation guarantee) but is upper-bounded by a constant. This can be seen by a calculation, using that we may assume¹¹ $\text{OPT}^* \geq 1$.

We now turn to the more succinct representation of information. This is done in two steps. First, values $U_k(\cdot)$ for large k are put together, resulting in shorter vectors T' , which take the role of the vectors T . Set

$$k_0 := \lceil \log(C_0 \log I) \rceil \quad \text{and} \quad U'(y_1, \dots, y_l) := \sum_{\substack{j \leq l \\ \text{s.t. } a_{ij} \leq 2^{-k_0}}} x_j.$$

¹¹A solution with value 1 can always be constructed by setting $x_j := 1$ for some j where $w_j = 1$ and all other entries to 0.

Then define

$$T'(y_1, \dots, y_l) := (U_0(y_1, \dots, y_l), \dots, U_{k_0-1}(y_1, \dots, y_l), U'(y_1, \dots, y_l)).$$

The second step is to characterize the interesting values the random variable T' can assume. This will be a polynomially sized set and hence our choice for the automata states (with one small addition of a “bad” state, see below). The following function will be useful as an upper bound on the interesting values:

$$g(k) := b_i 2^k + C_0 2^k \log I \log \log I.$$

Define the states V_l of layer l (for automata \mathcal{A}_i) as follows.

$$\begin{aligned} V_l := & \{(u_0, \dots, u_{k_0-1}, u', l) \in \mathbb{N}_0^{k_0+1} \times \{l\}; \forall k \in \{0, \dots, k_0-1\}: u_k \leq g(k) \text{ and } u' \leq b_i\} \\ & \cup \{\text{bad}_l\} \end{aligned} \tag{1.66}$$

Transitions from layer l to $l+1$, similarly as in \mathcal{A}_{m+1} , roughly correspond to the cases of “ $y_{l+1} = 0$ ” and “ $y_{l+1} = 1$ ”. However, more distinctions have to be made. First fix a state $u = (u_0, \dots, u_{k_0-1}, u', l) \in V_l$ (which is not the bad state). Edges with labels $0, \dots, 2^d \mathbb{E}[y_{l+1}] - 1$ go from u to $(u_0, \dots, u_{k_0-1}, u', l+1) \in V_{l+1}$. This corresponds to the case “ $y_{l+1} = 0$ ”, in which obviously all values in u have to be maintained (except the indication for the level in the last entry).

For the case “ $y_{l+1} = 1$ ”, there are edges with labels $2^d \mathbb{E}[y_{l+1}], \dots, 2^d$, which go from u to...

- $(u_0, \dots, u_k + 1, \dots, u_{k_0-1}, u', l+1) \in V_{l+1}$ if there exists $k \in \{0, \dots, k_0-1\}$ s.t. $a_{il} = 2^{-k}$ and $u_k + 1 \leq g(k)$,
- $(u_0, \dots, u_{k_0-1}, u' + 1, l+1) \in V_{l+1}$ if $a_{il} \leq 2^{-k_0}$ and $u' + 1 \leq b_i$,
- bad_l otherwise.

From a bad state, all $2^d + 1$ transitions go only to the next bad state. This is consistent with the meaning of the states, because the entries in vectors used as the states are non-decreasing in l , and hence once one of the bounds (“ $u_k \leq g(k)$ ” and “ $u' \leq b_i$ ”) is violated, it will so be further on. It is also clear that the value of Y_i can be read off the final states (in layer n), except from the bad state. If the automaton slips into the bad states, we have no information on Y_i , and hence to assume the worst, which is $Y_i = n$. Fortunately, we can bound the probability that this happens. The following lemma can be proven again using Theorem 1.2.

LEMMA 1.7 If the constant C_0 is chosen large enough, the probability that the automaton reaches a bad state is at most $\frac{1}{I^3}$.

Let Y'_i be the random variable expressed by \mathcal{A}_i . Then, the expected value of Y'_i differs from that of Y_i by at most $\frac{1}{I^2}$ (because $n \leq I$). Hence the bound in (1.62) is only weakened to

$$\mathbb{E}[Y'_i] = O\left(\left(\frac{e}{\lambda}\right)^{B'+1} + \frac{1}{I^2}\right).$$

Finally, we have to show that V_l has polynomial size. This can be checked by using the definition of the bounding function g and that $O(\log I)^{O(\log \log I)}$ can be bounded by a polynomial in I .

The construction is now ready for automata fooling. The approximation guarantee will be reduced depending on the ε used for the fooling. It suffices to do absolute fooling, so we give the analysis for this. A later remark will show a similar bound for the case of relative fooling.

THEOREM 1.22 (Srinivasan 2001)

(i). *There is a derandomized \mathcal{NC} algorithm for packing integer programs which fulfill (1.63) that computes for every $1 > \varepsilon > 0$ a solution $x \in \{0, 1\}^n$ for a PIP of input length I with value*

$$w \cdot x \geq \frac{\text{OPT}^*}{\lambda} - m \cdot O\left(\left(\frac{e}{\lambda}\right)^{B'+1} + \frac{1}{I^2}\right) - (m+1)\varepsilon.$$

(ii). *The following bound holds for the general case, when (1.63) is not necessarily given*

$$w \cdot x \geq \frac{1}{2} \left(\frac{\text{OPT}^* - \frac{2n}{2^d}}{\lambda} - m \cdot O\left(\left(\frac{e}{\lambda}\right)^{B'+1} + \frac{1}{I^2}\right) - (m+1)\varepsilon \right).$$

Proof (i). There are \mathcal{NC} algorithms in the literature to construct a fractional solution with an arbitrary small relative error [LN93]. For simplicity, we have not noted this error in the given bounds, nor will do in the following.

Let X be the random variable denoting the final state of \mathcal{A}_{m+1} , which corresponds to the value of the constructed solution. Let $K_3 \geq 2$ be a constant such that X and Y_1, \dots, Y_m do not take on more than I^{K_3} different values. Such a constant – meaning: independent of the given PIP – exists, since already the automata expressing these random variables have polynomially many states. We use $\varepsilon' = \frac{\varepsilon}{I^{K_3}}$ for automata fooling with an absolute error, resulting in the expectations of X and Y_1, \dots, Y_m having absolute error ε (see Remark 1.5). Let ζ be the original distribution (which is the uniform distribution on $\{0, \dots, 2^d\}^n$) and ρ the distribution created by the automata fooling algorithm. We have $\mathbb{E}_\rho[X] \geq \mathbb{E}_\zeta[X] - \varepsilon = \frac{\text{OPT}^*}{\lambda} - \varepsilon$, and also $\mathbb{E}_\rho[Y_i] \leq \mathbb{E}_\zeta[Y_i] + \varepsilon = O\left(\left(\frac{e}{\lambda}\right)^{B'+1} + \frac{1}{I^2}\right) + \varepsilon$.

(ii). If a given instance is not of the form in (1.63), we have to modify it first. This can be done by rounding down all entries in w to the next real in $\left\{\frac{k}{2^d}; k \in \{0, \dots, 2^d\}\right\}$. And we only allow fractional solutions $x^* \in \left\{\frac{k}{2^d}; k \in \{0, \dots, 2^d\}\right\}^n$. For the fractional optimum of the new instance OPT_0^* it holds that $\text{OPT}_0^* \geq \text{OPT}^* - \frac{2n}{2^d}$. Next, we consider the impact of rounding the entries in A . They are rounded down to the next non-negative power of 2. If x is feasible for this new instance, then $\frac{1}{2}x$ is feasible for the original one. ■

REMARK 1.7 If we use relative ε -fooling, we can prove in a similar manner

$$w \cdot x \geq (1 - \varepsilon) \left(\frac{\text{OPT}^*}{\lambda} - m \cdot O\left(\left(\frac{e}{\lambda}\right)^{B'+1} + \frac{1}{I^2}\right) \right).$$

The number of parallel processors used by this algorithm and the running time depend on the algorithm used for automata fooling, which – for the case of the fooling algorithm presented in this chapter – in turn depends on the algorithm RED. Bounds for this are

stated in Corollary 1.4 and Theorem 1.17. The polynomial size of the state space is essential, because the quantity k depends on it, see (1.48).

Srinivasan claims in [Sri01a] that a result similar to Theorem 1.22 holds for covering integer programs and refers to the full version. The full version had not appeared at the time this chapter was written.

REMARK 1.8 Let $B' = \Omega(\log m)$, say $B' = C_1 \log m$ with $C_1 \geq 2$, and $K_0 \geq 2e$. Then the first bound of Theorem 1.22 can be simplified to

$$w \cdot x \geq \frac{\text{OPT}^*}{2K_0(2 + o(1))} - o(1) - (m + 1)\varepsilon.$$

Proof The assumption $B' = \Omega(\log m)$ leads to $K_1^{\frac{1}{B'-1}} = 1 + o(1)$ and $m^{\frac{1}{B'-1}} \leq 2 + o(1)$, thus

$$\lambda \leq K_0 \max \left\{ 1, (2 + o(1)) \text{OPT}^{*- \frac{1}{C_1 \log m - 1}} \right\} \leq K_0(2 + o(1)),$$

because $\text{OPT}^* \geq \frac{1}{2}$. Furthermore $mO\left(\left(\frac{\varepsilon}{\lambda}\right)^{B'+1}\right) = mO(m^{-C_1}) = o(1)$, and trivially $\frac{m}{T^2} = o(1)$. The stated bound follows. ■

At the end of this section, a very legitimate question is whether Theorem 1.22 can help to improve the result for multidimensional bin packing (Theorem 1.19) in a way that the quite restrictive lower bound condition on the bin sizes $b_i = \Omega(n^{\frac{1}{2} + \tau} \sqrt{\log(nm)})$ can be weakened to $b_i = \Omega(\log(nm))$ for all i . We think that this should be possible by adapting the proof of Theorem 1.22 to the scenario of randomized rounding with multivalued random variables with equality constraints.

Bibliography and Remarks. Earlier derandomized \mathcal{NC} algorithms for packing integer programs were developed by Alon and Srinivasan [AS97]. For the case of $b_i = O(\log(m + n))$ for all $i \in [m]$ their algorithm gives results within a $1 + o(1)$ factor of the sequential bound $\Omega\left(\frac{\text{OPT}^*}{m^{1/B}}\right)$ achieved before [RT87, Rag88]. By a scaling technique, for any $c > 1$, they can also solve general PIPs, but only within a factor $\frac{1}{c}$ of the sequential bound.

1.6 Open Problems

Parallel Computation of Low Discrepancy Colorings. The perhaps most challenging algorithmic problem in combinatorial discrepancy theory is to give an efficient algorithm which computes for a hypergraph $\mathcal{H} = (V, \mathcal{E})$, say, with $|V| = n = |\mathcal{E}|$, a 2-coloring with discrepancy at most $O(\sqrt{n})$, matching the celebrated six standard-deviations bound of Spencer [Spe85]. It would be interesting to explore the impact of automata fooling and state space compression in this context.

Furthermore, 2-colorings for special hypergraphs with optimal or near-optimal discrepancy can be obtained by the *partial coloring* method of Beck [Bec81]. A \mathcal{NC} algorithm here is not known. A very interesting example for research in this direction would be the hypergraph of all arithmetic progressions in $[n]$. Its discrepancy is $\Theta(\sqrt[4]{n})$ (Matoušek, Spencer [MS96], Roth [Rot64]). The probabilistic method gives only $O(\sqrt{n^2 \log n})$ and the parallel algorithm from Section 1.3.4 computes colorings of discrepancy $O(n^{\frac{1}{2} + \varepsilon} \sqrt{\log n})$, $\frac{1}{2} \leq \varepsilon \leq 1$. Any progress towards $O(\sqrt[4]{n})$, sequential or parallel, is of great interest.

Sum of Dependent Random Variables. In the theoretical setting as well as applications of \mathcal{NC} algorithm design with the probabilistic method, the underlying randomized sequential algorithm is usually analyzed with concentration inequalities for sums of independent random variables. But many combinatorial optimization problems show dependencies among the variables in a linear programming model in a natural way. An interesting example is the partial covering problem in graphs and hypergraphs (see [GKS04]). Here \mathcal{NC} algorithms are not known.

Parallelizing the Random Hyperplane Method. The celebrated algorithms of Goemans and Williamson [GW95] for the max-cut problem can be efficiently derandomized (Mahajan, Ramesh [MR99]). An efficient parallelization (\mathcal{RNC} or \mathcal{NC} algorithm) for the random hyperplane method is *terra incognita* and a formidable challenge.

Practicability. Today's research in the area of efficient algorithms is more and more challenged by the legitimate question of practicability. Algorithms based on *randomized rounding* are often practical, both because they are fast, easy to implement and of very good performance of repeated several times [NRT87], [BS04]. \mathcal{NC} algorithms designed with the probabilistic method use a polynomial number of processors, and thus seem to be impractical in view of real-world applications, where we have access to only a constant number of parallel processors. Besides that, questions about communication overhead (e.g., synchronization, data communication between processors) or bottleneck situations when accessing a shared memory arise. Interdisciplinary work together with computer engineers will be necessary to get from theoretically efficient to practically efficient parallel algorithms.

This list is certainly incomplete. But we think that effort in these directions is sought to advance the design of parallel algorithms through an interaction of theoretical and practical issues.

Acknowledgements. We thank Andreas Baltz for his help in preparing the material for the paragraph on maximal independent sets (Section 1.3.3), and David Karger for clarifying proofs given in [KK97]. We thank our students Sandro Esquivel and Carsten Krapp for setting parts of the chapter in L^AT_EX. The first author thanks the Deutsche Forschungsgemeinschaft, priority program 1126 “Algorithmics of Large and Complex Networks” for the support through the grant Sr7-3. Last but not least we sincerely thank the editors for the invitation to contribute to the handbook, their patience and cooperation.

References

References

- [ABI86] N. Alon, L. Babai, and A. Itai. A fast and simple randomized algorithm for the maximal independent set problem. *Journal of Algorithms*, 7:567–583, 1986.
- [ACR98] A.F. Andreev, A.E.F. Clementi, and J.D.P. Rolim. A new general derandomization method. *Journal of the ACM*, 45(1):179–213, 1998.
- [AG97] N. Alon and Z. Galil. On the exponent of all pairs shortest path problem. *Computer System Sciences*, 54(2):255–262, 1997. Preliminary Version in: Proceedings of the 32nd IEEE Symposium on Foundations of Computer Science 1991 (FOCS ’91), pp. 569–575.
- [AGHP92] N. Alon, O. Goldreich, J. Håstad, and R. Peralta. Simple constructions for almost k-wise independent random variables. *Random Structures & Algorithms*, 3(3):289–304, 1992.
- [AGMN92] N. Alon, Z. Galil, O. Margalit, and M. Naor. Witnesses for matrix multiplication and for shortest paths. In *Proceedings of the 33rd IEEE-Symposium on Foundations of Computer Science 1992 (FOCS ’92)*, pages 417–426, 1992.
- [AGR94] N.M. Amato, M.T. Goodrich, and E.A. Ramos. Parallel algorithms for higher dimensional convex hulls. In *Proceedings of the 35th IEEE Symposium on Foundation of Computer Science 1994 (FOCS ’94)*, pages 683–694, 1994.
- [AKS87] M. Ajtai, J. Komlós, and E. Szemerédi. Deterministic simulation in logspace. In *Proceedings of the 19th Annual ACM Symposium on the Theory of Computing 1987 (STOC ’87)*, pages 132–140, 1987.
- [Alo91] N. Alon. A parallel algorithmic version of the Local Lemma. *Random Structures & Algorithms*, 2(4):367–378, 1991.
- [AMN98] Y. Azar, R. Motwani, and J. Naor. Approximating probability distributions using small sample spaces. *Combinatorica*, 18(2):151–171, 1998.
- [AN96] N. Alon and M. Naor. Derandomization, witnesses for boolean matrix multiplication and construction of perfect hash functions. *Algorithmica*, 16(4–5):434–449, 1996.
- [Arm98] R. Armoni. On the derandomization of space-bounded computations. In *Proceedings of the 2nd international workshop on Randomization and approximation techniques in computer science (RANDOM ’98)*, pages 47–59, Barcelona, Spain, October 1998.
- [AS97] N. Alon and A. Srinivasan. Improved parallel approximation of a class of integer programming problems. *Algorithmica*, 17:449–462, 1997.
- [AS00] N. Alon and J. Spencer. *The probabilistic method*. John Wiley & Sons, Inc., 2000.

- [ASE92] N. Alon, J. Spencer, and P. Erdős. *The probabilistic method*. John Wiley & Sons, Inc., 1992.
- [AV79] D. Angluin and L.G. Valiant. Fast probabilistic algorithms for hamiltonian circuits and matchings. *Computer System Sciences*, 18:155–193, 1979.
- [Ban98] W. Banaszczyk. Balancing vectors and gaussian measure of n -dimensional convex bodies. *Random Structures & Algorithms*, 12:351–360, 1998.
- [Bec81] J. Beck. Roth’s estimate of the discrepancy of integer sequences is nearly sharp. *Combinatorica*, 1(4):319–325, 1981.
- [Bec91] J. Beck. An algorithmic approach to the Lovász Local Lemma I. *Random Structures & Algorithms*, 2(4):343–365, 1991.
- [BEG⁺94] M. Blum, W. Evans, P. Gemmell, S. Kannan, and M. Naor. Checking the correctness of memories. *Algorithmica*, 12(2/3):225–244, 1994. Preliminary Version in: Proceedings of the 32nd IEEE Symposium on the Foundation of Computer Science 1991 (FOCS ’91).
- [Ber73] C. Berge. *Graphs and Hypergraphs*. North Holland, Amsterdam, 1973.
- [BESW93] J. Błazewicz, K. Ecker, G. Schmidt, and J. Węglarz. *Scheduling in computer and manufacturing systems*. Springer-Verlag, Berlin, 1993.
- [BF81] J. Beck and T. Fiala. Integer-making theorems. *Discrete Applied Mathematics*, 3:1–8, 1981.
- [BL90] P. Beame and M. Luby. Parallel search for maximal independence given minimal dependence. In *Proceedings of the 1st Annual ACM-SIAM Symposium on Discrete Algorithms 1990 (SODA ’90)*, pages 212–218, 1990.
- [BM84] M. Blum and S. Micali. How to generate cryptographically strong sequences of pseudo-random bits. *SIAM Journal on Computing*, 13:850–864, 1984.
- [BM98] P.A. Beling and N. Megiddo. Using fast matrix multiplication to find basic solutions. *Theoretical Computer Science*, 205(1–2):307–316, 1998.
- [Bol98] B. Bollobás. *Modern Graph Theory*. Springer, New York, 1998.
- [BR91] B. Berger and J. Rompel. Simulating $(\log^c n)$ -wise independence in \mathcal{NC} . *Journal of the ACM*, 38(4):1026–1046, 1991. Preliminary Version in: Proceedings of the IEEE Symposium on the Foundation of Computer Science (FOCS) 1989.
- [BRS94] B. Berger, J. Rompel, and P. Shor. Efficient \mathcal{NC} algorithms for set cover with applications to learning and geometry. *Computer System Science*, 49:454–477, 1994. Preliminary Version in: Proceedings of the 30th Annual IEEE Symposium on the Foundations of Computer Science 1989 (FOCS ’89), pp. 54–59.
- [BS83] J. Beck and J. Spencer. Balancing matrices with line shifts. *Combinatorica*, 3(3 and 4):299–304, 1983.
- [BS84] J. Beck and J. Spencer. Well distributed 2-colorings of integers relative to long arithmetic progressions. *Acta Arithmetica*, 43:287–298, 1984.
- [BS95] J. Beck and V. Sós. Discrepancy theory. *Handbook of Combinatorics, Vol. II*, pages 1405–1446, 1995.
- [BS04] A. Baltz and A. Srivastav. Fast approximation of minimum multicast congestion - implementation versus theory. *RAIRO Operations Research*, 38:319–344, 2004.
- [CFG⁺85] B. Chor, J. Freidmann, O. Goldreich, J. Hastad, S. Rudich, and R. Smolensky. The bit extraction problem and t -resilient functions. In *Proceedings of the 26th IEEE Annual Symposium on the Foundations of Computer Science 1985 (FOCS ’85)*, pages 396–407, 1985.
- [CG89] B. Chor and O. Goldreich. On the power of two-point sampling. *Journal of*

- Complexity*, 5:96–106, 1989.
- [Cha00] B. Chazelle. *The Discrepancy Method: Randomness and Complexity*. Cambridge University Press, 2000.
- [Che52] H. Chernoff. A measure of asymptotic efficiency for test of a hypothesis based on the sum of observation. *Annals of Mathematical Statistics*, 23:493–509, 1952.
- [Coo85] S. Cook. A taxonomy of problems with fast parallel algorithms. *Information and Control*, 64(1–3), 1985.
- [DF06] B. Doerr and M. Fouz. *Hereditary discrepancies in different numbers of colors II*. Preprint, 2006.
- [DGH06] B. Doerr, M. Gnewuch, and N. Hebbinghaus. Discrepancy of symmetric products of hypergraphs. *Electronic Journal of Combinatorics*, 13, 2006.
- [DHW04] B. Doerr, N. Hebbinghaus, and S. Werth. Improved bounds and schemes for the declustering problem. In Jirí Fiala, Václav Koubek, and Jan Kratochvíl, editors, *Proceedings of the 29th Symposium on Mathematical Foundations of Computer Science 2004 (MFCS '04)*, 2004. Lecture Notes in Computer Science 3153 (2004), pp. 760–771, Berlin–Heidelberg, Springer-Verlag.
- [Dia88] P. Diaconis. Group representations in probability and statistics. *IMS Lecture Notes, Monograph Series*, 11, 1988.
- [Die97] R. Diestel. *Graph Theory*. Springer, New York, 1997.
- [DK89] E. Dahlhaus and M. Karpiński. An efficient algorithm for the 3MIS problem. Technical Report TR-89-052, International Computer Science Institute, Berkeley, CA, 1989.
- [DKK92a] E. Dahlhaus, M. Karpinski, and P. Kelsen. An efficient parallel algorithm for computing a maximal independent set in a hypergraph of dimension 3. *Information Processing Letters*, 42(6):309–314, 1992. Preliminary version: Manuscript, Department of Computer Sciences, University of Texas, 1990.
- [DKK92b] E. Dahlhaus, M. Karpiński, and P. Kelsen. An efficient parallel algorithm for computing a maximal independent set in a hypergraph of dimension 3. *Information Processing Letters*, 42(6):309–313, 1992.
- [dVL81] W.F. de la Vega and C.S. Lueker. Bin packing can be solved within $1 + \varepsilon$ in linear time. *Combinatorica*, 1:349–355, 1981.
- [Doe00] B. Doerr. Linear and hereditary discrepancy. *Combinatorics, Probability and Computing*, 9:349–354, 2000.
- [Doe01] B. Doerr. Lattice approximation and linear discrepancy of totally unimodular matrices. In *Proceedings of the 12th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 119–125, 2001.
- [Doe02a] B. Doerr. Balanced coloring: Equally easy for all numbers of colors? In H. Alt and A. Ferreira, editors, *Proceedings of the 19th Annual Symposium on Theoretical Aspects of Computer Science (STACS '02)*, 2002. Lecture Notes in Computer Science 2285 (2002), pp. 112–120, Berlin–Heidelberg, Springer-Verlag.
- [Doe02b] B. Doerr. Discrepancy in different numbers of colors. *Discrete Mathematics*, 250:63–70, 2002.
- [Doe04] B. Doerr. The hereditary discrepancy is nearly independent of the number of colors. *Proceedings of the American Mathematical Society*, 132:1905–1912, 2004.
- [DS99] B. Doerr and A. Srivastav. Approximation of multi-color discrepancy. In D. Hochbaum, K. Jansen, J. D. P. Rolim, and A. Sinclair, editors, *Randomization, Approximation and Combinatorial Optimization (Proceedings of*

- APPROX-RANDOM 1999*), 1999. Lecture Notes in Computer Science 1671 (1999), pp. 39–50, Berlin–Heidelberg, Springer-Verlag.
- [DS01] B. Doerr and A. Srivastav. Recursive randomized coloring beats fair dice random colorings. In A. Ferreira and H. Reichel, editors, *Proceedings of the 18th Annual Symposium on Theoretical Aspects of Computer Science (STACS '01)*, 2001. Lecture Notes in Computer Science 2010 (2001), pp. 183–194, Berlin–Heidelberg, Springer-Verlag.
- [DS03] B. Doerr and A. Srivastav. Multicolour discrepancies. *Combinatorics, Probability and Computing*, 12:365–399, 2003.
- [DSW04] B. Doerr, A. Srivastav, and P. Wehr. Discrepancies of cartesian products of arithmetic progressions. *Electronic Journal of Combinatorics*, 11(1):12 pages, 2004.
- [EGL⁺98] G. Even, O. Goldreich, M. Luby, N. Nisan, and B. Veličković. Efficient approximation of product distributions. *Random Structures & Algorithms*, 13(1):1–16, 1998. Preliminary Version *Approximation of general independent distributions*. in: Proceedings of the 24th Annual ACM Symposium on Theory of Computing 1992 (STOC '92), pp. 10–16.
- [ES73] P. Erdős and J.L. Selfridge. On a combinatorial game. *Journal of Combinatorial Theory (A)*, 14:298–301, 1973.
- [Fel67] W. Feller. *An introduction to the theory of probability and its applications*. John Wiley and Sons, 1967.
- [FKN95] T. Feder, E. Kushilevitz, and M. Naor. Amortized communication complexity. *SIAM Journal on Computing*, 24(4):736–750, 1995. Preliminary Version in: Proceedings of the 32nd IEEE Symposium on the Foundation of Computer Science 1991 (FOCS '91), pp. 239–248.
- [GGJY76] M.R. Garey, R.L. Graham, D.S. Johnson, and A.C.-C. Yao. Resource constrained scheduling as generalized bin packing. *Journal of Combinatorial Theory (A)*, 21:257–298, 1976.
- [GKS04] R. Gandhi, S. Khuller, and A. Srinivasan. Approximation algorithms for partial covering problems. *Journal of Algorithms*, 53(1):55–84, 2004.
- [GM84] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28:270–299, 1984. Preliminary version in: Proceedings STOC '82, pp. 365–377.
- [Goo93a] M.T. Goodrich. Constructing arrangements optimally in parallel. *Discrete & Computational Geometry*, 9:371–385, 1993.
- [Goo93b] M.T. Goodrich. Geometric partitioning made easier, even in parallel. In *Proceedings of the 9th Annual ACM Symposium on Computational Geometry 1993*, pages 73–82, 1993.
- [Goo96] M.T. Goodrich. Fixed-dimensional parallel linear programming via relative ϵ -approximations. In *Proceedings of the 7th Annual ACM-SIAM Symposium on Discrete Algorithms 1996 (SODA '96)*, pages 132–141, 1996.
- [GS89a] M. Goldberg and T. Spencer. Constructing a maximal independent set in parallel. *SIAM Journal on Discrete Mathematics*, 2:322–328, 1989.
- [GS89b] M. Goldberg and T. Spencer. A new parallel algorithm for the maximal independent set problem. *SIAM Journal on Computing*, 18:419–427, 1989.
- [GW95] M.X. Goemans and D.P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM*, 42(6):1115–1145, 1995.
- [Han92] Y. Han. Parallel derandomization techniques. *Advances in Parallel Algorithms*, pages 368–395, 1992.

- [HILL97] J. Håstad, R. Impagliazzo, L. Levin, and M. Luby. Construction of a pseudo-random generator from any one-way function. Technical Report 91-068, ICSI, 1997.
- [HILL99] J. Håstad, R. Impagliazzo, L. Levin, and M. Luby. A pseudo-random generator from any one-way function. *SIAM Journal on Computing*, 28(4):1364–1396, 1999.
- [HMRAR98] M. Habib, C. McDiarmid, J. Ramirez-Alfonsin, and B. Reed. *Probabilistic methods for algorithmic discrete mathematics*. Springer Series Algorithms and Combinatorics, Vol. 16. Springer-Verlag, 1998.
- [Hoe63] W. Hoeffding. Probability inequalities for sums of bounded random variables. *American Statistical Association Journal*, 58:13–30, 1963.
- [IW97] R. Impagliazzo and A. Wigderson. $P = BPP$ unless e has sub-exponential circuits, derandomizing the XOR lemma. In *Proceedings of the 29th Annual ACM Symposium on the Theory of Computing 1997 (STOC '97)*, pages 220–229, 1997.
- [IZ90] R. Impagliazzo and D. Zuckerman. How to recycle random bits. In *Proceedings of the 31st Annual IEEE Symposium on the Foundations of Computer Science 1990 (FOCS '89)*, pages 248–253, 1990.
- [Jof74] A. Joffe. On a sequence of almost deterministic k -independent random variables. *Annals of Probability*, 2:161–162, 1974. Preliminary Version in: Proceedings of the AMS, 29:381–382, 1971.
- [Kel92] P. Kelsen. On the parallel complexity of computing a maximal independent set in a hypergraph. In *Proceedings of the 24th ACM Symposium on the Theory of Computing 1992 (STOC '92)*, pages 339–350, 1992.
- [KK97] D. Karger and D. Koller. (De)randomized construction of small sample spaces in \mathcal{NC} . *Computer System Sciences*, 55(3):402–413, 1997. Preliminary Version in: Proceedings of the 35th IEEE Symposium on the Foundations of Computer Science 1994 (FOCS '94), pp. 252–263.
- [KM94] D. Koller and N. Megiddo. Constructing small sample spaces satisfying given constraints. *SIAM Journal on Discrete Mathematics*, 7(2):260–274, 1994. Preliminary Version in: Proceedings of the 25th ACM Symposium on Theory of Computing 1993 (STOC '93), pp. 268–277.
- [KM97] H. Karloff and Y. Mansour. On construction of k -wise independent random variables. *Combinatorica*, 17(1):91–107, 1997. Preliminary Version in: Proceedings of the 26th ACM Symposium on the Theory of Computing 1994 (STOC '94), pp. 564–573.
- [KPS88] R. Karp, N. Pippenger, and M. Sipser. Expanders, randomness, or time versus space. *Journal of Computer and System Sciences*, 36:379–383, 1988. Preliminary Version in: Proceedings of the First Annual Conference on Structure in Complexity Theory (1986), pp. 325–329.
- [KR93] H. Karloff and P. Raghavan. Randomized algorithm and pseudorandom numbers. *Journal of the ACM*, 40(3):454–476, 1993.
- [KUW88] R.M. Karp, E. Upfal, and A. Wigderson. The complexity of parallel search. *Computer and System Sciences*, 36:225–253, 1988.
- [KW85] R.M. Karp and A. Wigderson. A fast parallel algorithm for the maximal independent set problem. *Journal of the ACM*, 32:762–773, 1985. Preliminary Version in: Proceedings of the 16th ACM Symposium on Theory of Computing 1984 (STOC '84), pp. 266–272.
- [LLSZ97] N. Linial, M. Luby, M. Saks, and D. Zuckerman. Efficient construction of a small hitting set for combinatorial rectangles in high dimension. *Combi-*

- natorica*, 17(2):215–234, 1997. Preliminary Version in: Proceedings of the 25th ACM Symposium on the Theory of Computing 1993 (STOC '93), pp. 258–267.
- [LN93] Michael Luby and Noam Nisan. A parallel approximation algorithm for positive linear programming. In *ACM Symposium on Theory of Computing*, pages 448–457, 1993.
- [LPS88] A. Lubotzky, R. Phillips, and P. Sarnak. Ramanujan graphs. *Combinatorica*, 8(3):261–277, 1988.
- [LS97] T. Luczak and E. Szymańska. A parallel randomized algorithm for finding a maximal independent set in a linear hypergraph. *Algorithms*, 25(2):311–320, 1997.
- [LSV86] L. Lovász, J. Spencer, and K. Vesztergombi. Discrepancies of set-systems and matrices. *European Journal of Combinatorics*, 7:151–160, 1986.
- [Lub86] M. Luby. A simple parallel algorithm for the maximal independent set problem. *SIAM Journal on Computing*, 15:1036–1053, 1986.
- [Lub96] M. Luby. Pseudorandomness and cryptographic applications. *Princeton Computer Science Notes*, 1996.
- [LV96] M. Luby and B. Velickovic. On deterministic approximation of DNF. *Algorithmica*, 16(4–5):415–433, 1996.
- [LW95] M. Luby and A. Wigderson. Pairwise independence and derandomization. Technical Report TR-95-035, International Computer Science Institute, UC Berkeley, July 1995.
- [LWV93] M. Luby, A. Wigderson, and B. Velickovic. Deterministic approximate counting of depth-2 circuits. In *Proceedings of the Second Israel Symposium on Theory of Computing and Systems 1993*, pages 18–24, 1993.
- [Mat99] J. Matoušek. *Geometric Discrepancy*. Springer-Verlag, Heidelberg, New York, 1999.
- [Mat00] J. Matoušek. On the discrepancy for cartesian products. *Journal of the London Mathematical Society*, 61(3):737–747, 2000.
- [MNN94] R. Motwani, J. Naor, and M. Naor. The probabilistic method yields deterministic parallel algorithms. *Computer and System Sciences*, 49:478–516, 1994. Preliminary Version in: Proceedings of the 30th Annual IEEE Symposium on Foundations of Computer Science, 1989, pp. 8–13.
- [MR95] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.
- [MR99] S. Mahajan and H. Ramesh. Derandomizing approximation algorithms based on semidefinite programming. *SIAM Journal on Computing*, 28(5):1641–1663, 1999. Preliminary version *Derandomizing semidefinite programming based approximation algorithms*. in Proceedings of the 36th Annual IEEE Symposium on Foundations of Computer Science (1995), 162–169.
- [MRS97] S. Mahajan, E.A. Ramos, and K.V. Subrahmanyam. Solving some discrepancy problems in nc. *Lecture Notes in Computer Science*, 1346:22–36, 1997.
- [MS77] F.J. MacWilliams and N.J.A. Sloane. *The theory of error correcting codes*. North Holland, Amsterdam, 1977.
- [MS96] J. Matoušek and J. Spencer. Discrepancy of arithmetic progressions. *Journal of the American Mathematical Society*, 9:195–204, 1996.
- [Mul96] K. Mulmuley. Randomized geometric algorithms and pseudo-random generators. *Algorithmica*, 16:450–463, 1996. Preliminary Version in: Proceedings of the 33rd Annual IEEE Symposium on the Foundations of Computer Science 1992 (FOCS '92), pp. 90–100.

- [Nie92] H. Niederreiter. *Random Number Generation and Quasi-Monte Carlo Methods*, volume 63 of *CBMS-NSF regional conference series in Applied Mathematics*. SIAM, Philadelphia, 1992.
- [Nis91] N. Nisan. Pseudorandom bits for constant depth circuits. *Combinatorica*, 1:63–70, 1991.
- [Nis92] N. Nisan. Pseudorandom generators for space-bounded computation. *Combinatorica*, 12:449–461, 1992.
- [NN93] J. Naor and M. Naor. Small bias probability spaces: Efficient constructions and applications. *SIAM Journal on Computing*, 22(4):838–856, 1993. Preliminary Version in: Proceedings of the 22nd Annual ACM Symposium on the Theory of Computing 1990 (STOC '90), pp. 213–223.
- [NRT87] A.P.-C. Ng, P. Raghavan, and C.D. Thompson. Experimental results for a linear program global router. *Computers and Artificial Intelligence*, 6(3):229–242, 1987.
- [NSS95] M. Naor, L.J. Schulman, and A. Srinivasan. Splitters and near optimal derandomization. In *Proceedings of the 36th Annual IEEE Symposium on the Foundations of Computer Science 1995 (FOCS '95)*, pages 182–191, 1995.
- [Oka58] M. Okamoto. Some inequalities relating to the partial sum of binomial probabilities. *Annals of the Institute of Statistical Mathematics*, 10:29–35, 1958.
- [Pap94] C.H. Papadimitriou. *Computational Complexity*. Addison-Wesley Publ. Company, Reading, Massachusetts, 1994.
- [Per90] R. Peralta. On the randomness complexity of algorithm. Technical Report TR90-1, University of Wisconsin, Milwaukee, WI, 1990.
- [Rag88] P. Raghavan. Probabilistic construction of deterministic algorithms: Approximating packing integer programs. *Computer System Sciences*, 37:130–143, 1988.
- [Rot64] K.F. Roth. Remark concerning integer sequences. *Acta Arithmetica*, 9:257–260, 1964.
- [RS83] H. Röck and G. Schmidt. Machine aggregation heuristics in shop scheduling. *Methods of Operations Research*, 45:303–314, 1983.
- [RT87] P. Raghavan and C.D. Thompson. Randomized rounding: a technique for provably good algorithms and algorithmic proofs. *Combinatorica*, 7:365–374, 1987.
- [Sak96] M. Saks. Randomization and derandomization in space-bounded computation. In *Eleventh Annual IEEE Conference on Computational Complexity*, pages 128–149, Philadelphia, Pennsylvania, May 1996.
- [San87] M. Santha. On using deterministic functions to reduce randomness in probabilistic algorithms. *Information and Computation*, 74:241–249, 1987.
- [Sch92] L. Schulman. Sample spaces uniform on neighborhoods. In *Proceedings of 24th Annual ACM Symposium on Theory of Computing 1992 (STOC '92)*, pages 17–25, 1992.
- [Sip88] M. Sipser. Expanders, randomness, or time versus space. *Computer and Systems Science*, 36:379–383, 1988.
- [Spe85] J. Spencer. Six standard deviation suffice. *Transactions of the American Mathematical Society*, 289:679–706, 1985.
- [Spe87] J. Spencer. *Ten lectures on the probabilistic method*. SIAM, Philadelphia, 1987.
- [Sri97] A. Srinivasan. Improving the discrepancy bound for sparse matrices: better approximation for sparse lattice approximation problems. In *Proceedings of the 7th ACM/SIAM Symposium on Discrete Algorithms 1997 (SODA*

- '97), pages 692–701, 1997.
- [Sri99] A. Srinivasan. Improved approximation guarantees for packing and covering integer programs. *SIAM Journal on Computing*, 29(2):648–670, 1999.
- [Sri01a] A. Srinivasan. New approaches to covering and packing problems. In *Proceedings of the 12th Annual ACM-SIAM Symposium on Discrete Algorithms 2001 (SODA '01)*, pages 567–576, 2001.
- [Sri01b] A. Srivastav. Derandomization in combinatorial optimization. In Rajasekaran, Pardalos, Reif, and Rolim, editors, *Handbook of Randomized Computing, Volume II*, pages 731 – 842. Kluwer Academic Publishers, 2001.
- [SS93] A. Srivastav and P. Stangier. On quadratic lattice approximations. In K. W. Ng, P. Raghavan, and N. V. Balasubramanian, editors, *Proceedings of the 4th International Symposium on Algorithms and Computation (ISAAC '93)*, Hong-Kong, 1993. Lecture Notes in Computer Science 762 (1993), pp. 176–184, Springer-Verlag.
- [SS96] A. Srivastav and P. Stangier. Algorithmic chernoff-hoeffding inequalities in integer programming. *Random Structures & Algorithms*, 8(1):27–58, 1996.
- [SS97a] A. Srivastav and P. Stangier. A parallel approximation algorithm for resource constrained scheduling and bin packing. In G. Bilardi, A. Ferreira, R. Lüling, and J. Rolim, editors, *Proceedings of the 4th International Symposium on Solving Irregularly Structured Problems in Parallel*, 1997. Lecture Notes in Computer Science (1997), Vol. 1253, pp. 147–159, Springer-Verlag.
- [SS97b] A. Srivastav and P. Stangier. Tight approximations for resource constrained scheduling and bin packing. *Discrete Applied Mathematics*, 79:223–245, 1997.
- [SS01] M. Sitharam and T. Straney. Derandomized learning of boolean functions. *International Journal of Foundations of Computer Science*, 12(4):491–516, 2001. Preliminary Version in: Lecture Notes in Computer Science 1316 (1997), pp. 100–115.
- [Szy98] E. Szymańska. *PhD thesis*. PhD thesis, Adam Mickiewicz University, Poznań, Poland, 1998.
- [Vaz86] U.V. Vazirani. *Randomness, adversaries and computation*. PhD thesis, University of California, Berkeley, CA, 1986.
- [VV84] U.V. Vazirani and V.V. Vazirani. Efficient and secure pseudo-random number generation. In *Proceedings of the 25th Annual IEEE Symposium on Foundations of Computer Science 1984 (FOCS '84)*, pages 458–463, 1984.
- [Wes96] D.B. West. *Introduction to Graph Theory*. Prentice-Hall, Inc. Simon & Schuster, 1996.
- [Yao82] A. Yao. Theory and applications of trapdoor functions. In *Proceedings of the 23th Annual IEEE Symposium on Foundations of Computer Science 1982 (FOCS '82)*, pages 80–91, 1982.

Index

- \mathcal{NC} algorithm, 1-6
- BIT-RR, 1-30
- CONDEXP, 1-7
- CONDPROB, 1-7
- DERAND-BIT, 1-30
- FOOL, 1-42
- LFMIS, 1-19
- PARALLELMIS, 1-20
- RED, 1-39

- automaton, 1-35

- basis crashing, 1-37
 - \mathcal{NC} algorithm, 1-38
- bin packing
 - algorithm, 1-46
 - problem, 1-46
- binary representation, 1-2

- Chernoff-Hoeffding bounds, 1-4
- coloring, 1-22
- conditional probability method, 1-7
- CRCW, 1-6
- CREW, 1-6

- degree, 1-3
- discrepancy
 - algorithm, 1-26
 - of coloring, 1-23
 - of hypergraph, 1-23

- ERCW, 1-6
- EREW, 1-6

- fooling
 - absolute, 1-36
 - algorithm, 1-42
 - expectation, 1-45
 - relative, 1-37

- graph, 1-3

- hypergraph, 1-3

- incidence matrix, 1-3
- independence
 - k -wise, 1-8
 - almost k -wise, 1-12

- inner product, 1-2

- lattice approximation
 - derandomized algorithm, 1-30
 - half
 - algorithm, 1-28
 - problem, 1-28
 - problem, 1-27
 - randomized algorithm, 1-30
 - relative error, 1-34

- maximal independent set, 1-19
- multivalued functions, 1-17

- neighbors, 1-3

- packing integer program, 1-49
- parallel packing
 - algorithm, 1-54
 - derandomization, 1-51
 - idea, 1-50
- PRAM, 1-6

- random variables
 - (ε, k) -independent, 1-12
 - ε -biased, 1-13
 - k -wise δ -dependent, 1-13
 - k -wise independent, 1-8

- support
 - of distribution, 1-36
 - of vector, 1-34

- transition probability, 1-40
- twovalued functions, 1-15

- vector balancing problem, 1-28

- word, 1-36