Small-bias probability spaces: efficient constructions and applications

Joseph Naor * *Moni Naor* †

Abstract

We show how to efficiently construct a small probability space on n binary random variables such that for every subset, its parity is either zero or one with "almost" equal probability. They are called ϵ-biased random variables. The number of random bits needed to generate the random variables is $O(\log n + \log \frac{1}{\epsilon})$. Thus, if ϵ is polynomially small, then the size of the sample space is also polynomial. ϵ-biased random variables can be used to construct "almost" k-wise independent random variables where ϵ is a function of k. Applications are shown to derandomization of algorithms, reducing the number of random bits required by certain randomized algorithms, exhaustive testing of combinatorial circuits, communication complexity and construction of hash functions.

1 Introduction

Randomness plays a significant role in computer science. However, it is often desirable to reduce the amount of randomness required. The purpose of this paper is to construct small probability spaces that approximate larger ones. Let $x_1, \ldots, x_n$ be $\{0,1\}$ Bernoulli random variables and let Ω be the probability space associated with them. If the random variables are independent, then Ω contains all 2^n possible assignments. Our goal is to construct a much smaller probability space that will behave similarly to Ω in certain respects. Such small probability spaces have proved to be very useful.

One of the main approaches taken by previous researchers to reduce the size of the sample space was to require only limited independence among the random variables (as opposed to full independence). Our approach is different; it is based on the equivalence of the following two conditions: (See [10] and [35]).

1. The random variables are independent and for all i, $\text{Prob}[x_i = 0] = \text{Prob}[x_i = 1]$.
2. For every subset $S \in \{1, \ldots, n\}$, it is equally likely that the parity of the subset (i.e., the number of "ones") is either zero or one.

We are going to relax the second condition and construct a probability distribution such that for every subset $S \in \{1, \ldots, n\}$, it is "almost" equiprobable that the parity of the subset is zero or one. To be more precise, we require that for every subset S,

$$\left| \text{Prob}[\sum_{i \in S} x_i = 0] - \text{Prob}[\sum_{i \in S} x_i = 1] \right| \leq \epsilon$$

This quantity was called by Vazirani [35] the *bias* of the subset S. The cardinality of the sample space we construct is $2^{O(\log \frac{1}{\epsilon} + \log n)}$. Hence, if ϵ is typically polynomially small, then the size of the sample space is also polynomial.

We also define random variables that are k-wise ϵ-biased. For them, only the bias of subsets that are smaller than k is guaranteed to be bounded by ϵ. We present a more efficient construction for such random variables: the logarithm of the cardinality of the sample space is $O(\log k + \log \log n + \log \frac{1}{\epsilon})$. In Section 5 we show how the k^{th} moment of the sum of k-wise ϵ-biased random variables can be bounded via the k^{th} moment of the binomial distribution on uniform independent Bernoulli variables. This is an important tool for analyzing the behavior of the sum.

*Computer Science Department, Stanford University, Stanford, CA 94305-2140. Supported by contract ONR N00014-88-K-0166 and by a grant of Stanford's Center for Integrated Systems.

†IBM Almaden Research Center, 650 Harry Road, San Jose, CA

We can use k-wise ϵ-biased random variables to construct k-wise δ-dependent random variables, i.e., variables such that the variation distance between the distribution of any subset of k variables and the uniform distribution (on k variables) is at most δ. Their construction is described in Section 4.

The ϵ-biased random variables are constructed in three stages. In the first stage, we construct a sample space $\mathcal{F}$ of random variables such that the bias of every subset S is bounded by some constant. Sampling from $\mathcal{F}$ requires $O(\log n)$ random bits. In the second stage, $\mathcal{F}$ is sampled l times (not necessarily independently), where l depends on ϵ. In the third stage, the ϵ-biased random variables are generated by picking a linear combination of the assignments sampled in the second stage. Constructing the probability space is described in Section 3.

Another interpretation of our result is via Fourier transforms. (See Section 2 for precise definitions). For the uniform distribution, all the coefficients of its Fourier transform are zero, except for the free coefficient. For an ϵ-biased distribution, the absolute value of each coefficient is at most $\frac{\epsilon}{2^n}$.

Derandomizing algorithms has attracted much attention in recent years. For the purpose of derandomization, our distribution can replace the uniform one in many cases, so as to allow an exhaustive search for a good point in a polynomial-size sample space. We exemplify this by providing a *polynomial* size sample space for the set balancing problem; this also yields an NC^1 algorithm. A previous approach to derandomizing the set balancing problem [8, 25] was to first construct an $n^{O(\log n)}$ sample space, and then conduct a binary search for a good point. As a result, their time bounds are worse. Another problem we address is finding a heavy codeword in a linear code. Using ϵ-biased random variables, we provide the first NC algorithm to the problem.

Another application of our probability distribution is for reducing the number of random bits required for several randomized algorithms. Karp and Pippenger [19] suggested that random bits can be viewed as a resource (just as time and space) which is best to use as little as possible. One motivation to consider randomness as a resource is practical. Random bits are hard to produce and devices that generate them, such as Geiger counters and Zenner diodes are slow. Another reason is from the complexity theoretic point of view: to provide a full scale of options between an algorithm that is completely deterministic, and a randomized algorithm that consumes many bits.

Examples of previous work in reducing the number of random bits in randomized algorithms are [1, 5, 11, 16, 19, 20, 27, 31, 33]. Karloff and Raghavan [20] for example, studied several randomized algorithms for selection and sorting and showed that they can be run successfully when only $O(\log n)$ random bits are available.

In Section 7 we describe how to reduce the number of random bits for two problems. The first one is matrix multiplication verification. Given three $n \times n$ matrices A, B and C, how can one verify that $A \cdot B = C$ without resorting to matrix multiplication. We show how to do that in $O(n^2)$ time using $O(\log n)$ random bits, thus improving on a previous algorithm of [14] that required $O(n)$ random bits. The second problem is verifying n equalities of the form $a^{x_i} = y_i \bmod p$ where p is a prime. We show how to do that using only $O(\log n)$ random bits and n multiplications, instead of $n \log p$ multiplications needed for testing each equality separately.

In Section 8 we show how to apply our construction to generate fault-diagnostic tests for combinatorial circuits. A well known problem in that area is to construct a small collections of assignments to inputs of a circuit such that for any k any inputs, all possible configurations appear. Using k-wise δ-dependent probability spaces, we can get the best explicit constructions. These are optimal up to the constant factor in the exponent.

Our techniques can be used to minimize the communication complexity of protocols for testing equality of two strings, while achieving a very low probability error. Similarly, the techniques can be applied to construct a small family of hash functions with the property that summing the hash function over different sets yields a different value with high probability. These two applications are discussed in Section 9.

Independent of our work, Peralta [28] has considered ϵ-bias probability spaces as well, and showed some applications to number theoretic algorithms. His construction is based on quadratic residues and Weil's Theorem.

2 Preliminaries and definitions

Let $\mathbf{x} = x_1, \ldots, x_n$ be $\{0,1\}$ random variables and D their joint probability distribution.

Definition 2.1. *The bias of a subset $S \subseteq \{1, \ldots, n\}$ for a distribution D, $bias_D(S)$, is defined to be*

$$\left| \mathrm{Prob}_D[\sum_{i \in S} x_i = 0] - \mathrm{Prob}_D[\sum_{i \in S} x_i = 1] \right|$$

If $S = \emptyset$, then $bias(\emptyset) = 0$.

Definition 2.2. *If the variables are drawn from* $\{-1,1\}$, *then the bias is equivalently defined as,*

$$\left|\text{Prob}_D[\prod_{i\in S} x_i = -1] - \text{Prob}_D[\prod_{i\in S} x_i = 1]\right|$$

Definition 2.3. *The variables* $x_1,\ldots,x_n$ *are* ϵ*-biased if for all* S, $bias_D(S) \leq \epsilon$. *They are said to be* k*-wise* ϵ*-biased if for all subsets* S *such that* $|S| \leq k$, $bias_D(S) \leq \epsilon$.

Let U denote the uniform distribution and $D(S)$ the distribution D restricted to the subset S. The variation distance between two distributions D_1 and D_2 defined over the same probability space Ω is

$$||D_1 - D_2|| = \sum_{\omega\in\Omega} |D_1(\omega) - D_2(\omega)|$$

Definition 2.4. *The variables* $x_1, x_2, \ldots, x_n$ *are defined to be* k*-wise* δ*-dependent if for all subsets* S *such that* $|S| \leq k$,

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

A similar definition was made by R. Ben-Nathan. [6].

The set $f : \{-1,1\}^n \to \Re$ of real functions on the n-dimensional cube forms a 2^n−dimensional real vector space.

The basis of this vector space is given by the following family of functions, the characters of Z_2^n. Define for each subset S of $\{1\ldots n\}$,

$$\chi_S(x_1,\ldots,x_n) = \begin{cases} +1 & \text{if } \sum_{i\in s} x_i = 0 \\ -1 & \text{if } \sum_{i\in s} x_i = 1 \end{cases}$$

The Fourier transform of a function f is its expansion as a linear combination of the χ_S's. Every function has a unique expression and the coefficients in this expression are called the Fourier coefficients, denoted by $\hat{f}(S)$ (for $S \subseteq \{1\ldots n\}$). Hence, $f = \sum_S \hat{f}(S)\chi_S$. For a probability distribution D, $D = \sum_S \hat{D}(S)\chi_S$.

The following theorem of Vazirani [35] motivated us to consider ϵ-biased probability spaces.

Theorem 2.5. *Let* $x_1,\ldots,x_n$ *be* $\{-1,1\}$ *random variables and let* D *be their joint probability distribution. Then,*

$$||D - U|| \leq \sum_{S\subseteq\{1\ldots n\}} bias(S)$$

Corollary 2.6. *If the random variables* $x_1,\ldots,x_n$ *are* ϵ*-biased with respect to a distribution* D, *then they are also* k*-wise* δ*-dependent, for* $\delta = 2^k \cdot \epsilon$.

3 Constructing the probability distribution

In this section we show how to construct a probability space Ω of $\{0,1\}$ random variables $\mathbf{x} = x_1,\ldots,x_n$ which are ϵ-biased. The cardinality of the sample space will be $2^{O(\log\frac{1}{\epsilon}+\log n)}$. The joint probability distribution of the variables is denoted by D. The construction consists of 3 stages:

1. A polynomial size family $\mathcal{F}$ of $\{0,1\}^n$ vectors is generated with the following property. Let r be a vector chosen from $\mathcal{F}$ uniformly in random. For all subsets $S \subseteq \{1,\ldots,n\}$,

   $$\text{Prob}[\chi_S(r) = 1] \geq \beta$$

   where β is some constant. Constructing such a family is discussed in Section 3.1.

2. The vectors $r_1,\ldots,r_l$ are sampled from $\mathcal{F}$ (not necessarily independently) such that for all subsets $S \subseteq \{1,\ldots,n\}$,

   $$\text{Prob}[\text{For all } i,\ 1 \leq i \leq l,\ \chi_S(r_i) = 0] \leq \epsilon$$

   The value of l will turn out to be $O(\log\frac{1}{\epsilon})$. Sampling the family $\mathcal{F}$ is discussed in Section 3.2.

3. The assignment to the random variables $x_1,\ldots x_n$ is a combination of the vectors sampled at the previous stage. Let $\vec{a} = (a_1,\ldots,a_l)$ be chosen from $\{0,1\}^l$. Then,

   $$\mathbf{x} = \sum_{i=1}^{l} a_i r_i$$

   In Section 3.3 we discuss how to choose $\vec{a}$ so that $\mathbf{x}$ is ϵ-biased.

3.1 Constructing the family $\mathcal{F}$

We will need a family $\mathcal{F}$ with the above properties for constructing ϵ-biased probability spaces and also for other applications as well. (See Section 7). For the latter applications, we extend the requirements from $\mathcal{F}$ to any ring: given a vector v of elements in the ring, a vector r such that $< v\cdot r > \neq 0$ is called a *distinguisher* with respect to v. The goal is to find a small collection of vectors (the family $\mathcal{F}$) such that for any vector v, if r is chosen in random, then $\text{Prob}[< v\cdot r > \neq 0] \geq \beta$. If the ring is GF[2], this requirement is exactly the one mentioned above, where v is the characteristic vector of the subset S. Henceforth, the symbol 1 will denote any non-zero element of the ring.

We present two methods for constructing the family $\mathcal{F}$. The first method (Section 3.1.1) can be applied

to any ring, whereas the second one (Section 3.1.2) is applicable only to GF[2]. Another advantage of the first method is that computing the value of a random variable $x_i \in \mathbf{x}$ can be done in $O(1)$ operations on words of length $\log n$. On the other hand, the second method provides a general context, i.e., *linear codes.*

Proposition 3.1. *Suppose that r is a vector chosen uniformly in random from $\{0,1\}^n$. Then for all vectors $v \in \{0,1\}^n$, $v \neq \vec{0}$, Prob$[< v \cdot r >= 1] \geq \frac{1}{2}$.*

Unfortunately, this method of generating a distinguisher requires n random bits. Hence, our aim is to show that a distinguisher can be generated with probability at least β using only $O(\log n)$ random bits. This will guarantee that the size of $\mathcal{F}$ is polynomial. Note that $\mathcal{F}$ must contain at least n vectors (potential distinguishers). Otherwise, the rank of the matrix whose columns are the distinguishers is less than n.

For our purposes, a collection $\mathcal{F}$ of vectors has to be constructed such that:

1. The size of $\mathcal{F}$ is "small".
2. It is "easy" to sample uniformly from $\mathcal{F}$.
3. Given any vector v, a constant fraction of the vectors in $\mathcal{F}$ are distinguishers with respect to v.

3.1.1 Constructing a small set of distinguishers A natural approach to the problem of reducing the number of random bits in a randomized algorithm is to show that limited independence of the random variables suffices to assure a high probability of success. (See e.g., [4, 22, 24]). However, in our case, it is not clear from the proof of Proposition 3.1 how many vectors r remain distinguishers with respect to the vector v when the entries of R are not completely independent. Moreover, an example can be constructed in which if the entries of r are chosen pairwise independently, no distinguisher will be generated.

Though limited independence is not sufficient for our purposes, we make use of it in two ways: one is that suggested in [9] that if we sample a universe looking for elements of some fixed subset, and if the expected number of elements we hit is greater than 1, then by making our choices only pairwise independent, we are not decreasing the chances of hitting an element of the subset by much. The other is that if the vector v has at most c non-zero entries, then the elements of r can be chosen c-wise independent, and with probability at least $\frac{1}{2}$ r is a distinguisher.

We now describe how the above-mentioned difficulties for generating distinguishers can be overcome. In what follows we will need n random variables such that:

1. Each random variable is uniformly distributed in $\{1, \ldots n\}$.
2. Every subset of the random variables of cardinality at most c is independent.

There are known methods of generating such random variables that use only $O(c \log n)$ random bits. ([24, 4, 9]).

We first assume that l, the precise number of non-zero elements in v, is known to be in the range $[k \ldots 2k]$. A two-step process is applied.

1. The vector v is replaced by a new vector $v' = (v'_1, \ldots v'_n)$ that contains only c (for some constant) non-zero elements. (Any non-zero element of v' is also non-zero in v).
2. Now, the elements of the vector r can be chosen c-wise independent, yet Proposition 3.1 still holds.

As we do not have direct access to the elements of v, we show instead how to emulate the above Step 1 with high probability. Let $u = (u_1, \ldots, u_n)$ and $w = (w_1, \ldots, w_n)$ be two random vectors such that:

1. The elements of u are chosen c-wise independent (c is a constant whose value will be specified later) where for all $1 \leq i \leq n$, Prob$[u_i = 0]$ = Prob$[u_i = 1] = \frac{1}{2}$.
2. The elements of w are chosen pairwise independent where for all $1 \leq i \leq n$, Prob$[w_i = 1] = \frac{2}{k}$.

We can assume w.l.o.g that $k|n$. To generate the vector w we generate n random variables $z_1, z_2, \ldots, z_n$ that are pairwise independent, and each is uniformly distributed in $\{1, \ldots, n\}$. We then set w_i to 1 if $z_i < \frac{2k}{n}$.

Let us now define the random vector $r = (r_1, \ldots, r_n)$ that will be used as a distinguisher. For all $1 \leq i \leq n$,

$$r_i = \begin{cases} 1 & \text{if } u_i = 1 \text{ and } w_i = 1 \\ 0 & \text{otherwise} \end{cases}$$

Lemma 3.2. *The above vector r is a distinguisher with probability at least $\frac{1}{4}$ for any vector v for which l, the number of non-zero elements, is in the range $[k, 2k]$.*

Proof: Let us define the vector v' (from Step 1): for all $1 \leq i \leq n$,

$$v'_i = \begin{cases} 1 & \text{if } v_i = 1 \text{ and } w_i = 1 \\ 0 & \text{otherwise} \end{cases}$$

It is clear that the lemma will follow if we show that the vector u is a distinguisher with respect to v' with

probability at least $\frac{1}{4}$. To do that, it suffices to prove that v' will contain at least one non-zero element of v, and at most c non-zero elements of v with probability $\frac{1}{2}$. As the elements of the vector u are c-wise independent, the proof of Proposition 3.1 still holds when the number of non-zero elements is less than c.

Generating the vector v' can be thought of as a binomial random variable where each non-zero entry of v decides with probability p (to be specified later) whether it remains non-zero in v'. The random choices are pairwise independent. Let h be a random variable that denotes the number of non-zero elements in v'. It is well known that $E[h] = pk$ and $Var[h] = p(1-p)n$. We choose the value of p such that $pk = 2$.
Claim: $\text{Prob}[0 < h \leq 7] \geq \frac{1}{2}$

We prove the claim by Chebyshev's inequality [Fe] that states that

$$\text{Prob}[|X - E[X]| \geq \lambda] \leq \frac{Var[X]}{\lambda^2}$$

where X is a random variable. It is enough to verify the claim in the two extreme cases when $l = k$ and $l = 2k$. Thus, substituting $\lambda = 3$, we get that

$$\text{Prob}[|h - pk| \geq 2] \leq \frac{pk(1-p)}{4} \leq \frac{1}{2}$$

$$\text{Prob}[|h - 2pk| \geq 3] \leq \frac{2pk(1-p)}{9} \leq \frac{4}{9}$$

Hence, we can choose $c = 2kp + 3 = 7$ and this is enough to insure success with probability at least $\frac{1}{2}$. □.

We conclude that if the approximate number of non-zero entries in v is known, then $O(\log n)$ random bits suffice to insure high probability of success. What can we do if this is not known? We follow the above algorithm and construct $\log n$ collections $\mathcal{F}_1, \mathcal{F}_2, \ldots \mathcal{F}_{\log n}$, where $\mathcal{F}_i$ is generated under the assumption that the number of non-zero entries in v is between 2^{i-1} and 2^i. Lemma 3.1 implies that at least $\frac{1}{4}$ of the members of at least one collection will be distinguishers.

The same random bits can be used to sample the $\log n$ collections, and we obtain a set of $\log n$ vectors $r^1, r^2 \ldots r^{\log n}$ such that at least one of them is a distinguisher with probability at least $\frac{1}{4}$. Instead of testing each vector separately, we can generate from them a single vector that is a distinguisher with probability at least $\frac{1}{8}$.

Let S be a subset of $\{1, \ldots \log n\}$ which is chosen uniformly at random and let r be defined by

$$r = \sum_{i \in S} r^i$$

Lemma 3.3. *The vector r is a distinguisher with respect to v with probability at least $\frac{1}{8}$.*

To summarize, we describe the algorithm to generate a random vector r. We assume that n is a power of 2.

1. Generate the following random variables:
   (a) $z_1, z_2, \ldots, z_n$: n random variables that are pairwise independent and uniformly distributed in $\{1, \ldots, n\}$.
   (b) $u = (u_1, \ldots, u_n)$ a vector whose entries are 7-wise independent and uniformly distributed in $\{0, 1\}$.
   (c) A random subset $S \subset \{1, \ldots, \log n\}$.
2. For each $1 \leq i \leq n$, compute $j_i = \log n - \max\{j | 2^j < z_i\}$.
3. For $1 \leq l \leq \log n$, compute $c_l = |S \bigcap \{1, \ldots, l\}| \cdot 1$. (This is scalar multiplication in the ring, e.g., in GF[2] it is $|S \bigcap \{1, \ldots, l\}| \bmod 2$).
4. For each $1 \leq i \leq n$, compute:

$$r_i = \begin{cases} c_{j_i} & \text{if } u_i = 1 \\ 0 & \text{otherwise} \end{cases}$$

Theorem 3.4. *The algorithm described above uses $O(\log n)$ random bits and generates a distinguisher with probability at least $\beta = \frac{1}{8}$. For all i, $1 \leq i \leq n$, the complexity of computing the value of the random variable x_i is $O(1)$ operations on words of size $O(\log n)$.*

3.1.2 A construction based on linear codes

Here we describe how to generate a family $\mathcal{F}$ of size $O(n)$ via linear codes due to J. Bruck (private communication). The construction works for GF[2]. Let C be a linear code. The weight of a codeword is defined to be the number of non-zero entries.

Proposition 3.5. *The minimum distance of a linear code is equal to the minimum weight of a codeword.*

Suppose we have a linear code $C[n, m, \beta]$, i.e. it maps $\{0,1\}^n$ words into $\{0,1\}^m$ codewords and its minimum distance is βm. Linear codes for which $m = O(n)$ and β is some constant exist and are constructible. (For example, Justesen codes [18, 26]). Let G be the generator matrix of this code. For any $\{0,1\}^n$ vector v, $G \cdot v$ contains at least βm non-zero entries. (By the above proposition). Hence, if we choose a column in G uniformly in random, $\text{Prob}[< v \cdot r >= 1] \geq \beta$. The family $\mathcal{F}$ is the set of columns in G. The relation with linear codes holds in the other direction as well. Given a family $\mathcal{F}$, consider its members as columns of

a generator matrix of a linear code. It follows from the proposition that the minimum distance of this code is $\beta|\mathcal{F}|$. Hence, the construction in Section 3.1.1 can be regarded as a linear code.

Given that good codes exist, what advantages does the first method have? The first method has the property that it works for more general cases where the function χ_S is defined on any group, not just addition modulo 2. This will be used in Section 7.1. Another advantage is in computing a single entry of the sampled vector r. This is very simple in the first method (Theorem 3.1) whereas all known methods for using (traditional linear codes) are more complicated and require exponentiation.

Using our techniques we can actually enhance the error-correction of a linear code without increasing the rate by much. This is investigated in a forthcoming paper. (With J. Bruck and R. Roth).

3.2 Sampling the family $\mathcal{F}$

The problem of obtaining the vectors $r_1, \ldots, r_l$ in Stage 2 with the desired property can be abstracted in the following way. Suppose that there is a universe, and we wish to find a member in a certain subset $\mathcal{S}$ of it. (In our case it is the set of vectors for which $\chi_S = 1$). Suppose also that we have a sampling algorithm that uses k random bits and has probability β of picking a member of the desired set. By sampling l times independently, a set of l elements is generated, such that with probability greater than $1 - \beta^{-l}$, at least one of them is a member of the desired set. A straightforward implementation would require kl random bits.

Several papers have addressed the question of achieving the probability error while requiring fewer random bits [2, 11, 16, 19, 31, 33]. We consider the method of [2] which is used by [11, 16]: For a graph $G = (V, E)$, consider any 1-1 correspondence $f : V \to \{0,1\}^l$, the different assignments to the random bits of the sampling algorithm. To generate the l samples, choose a random vertex of G and perform a random walk of length l. Each vertex in the random walk corresponds to a sample point. The number of random bits required is the sum of those needed for sampling one vertex, and those needed for performing the random walk.

Let λ_0 be the largest eigenvalue of G and $\bar{\lambda}$ be the eigenvalue of second largest value in G. Let α denote the percentage of vertices that are not members in $\mathcal{S}$. Cohen and Wigderson [12, Thm 4.5] show that if G is a d-regular graph such that

$$2\alpha \leq \left(\frac{\bar{\lambda}}{\lambda_0}\right)^2$$

then the probability that at least one of the l samples is a member of $\mathcal{S}$ is at least

$$1 - \left(\frac{\sqrt{2}|\bar{\lambda}|}{\lambda_0}\right)^l.$$

Constructions for regular graphs such that the second eigenvalue is bounded away from λ_0 are known [15, 17, 23]. For degree regular graphs of degree d, $\lambda_0 = d$ and the value of $\bar{\lambda}$ can be almost $2\sqrt{d}$. For a given expander G, let

$$\alpha_G = \frac{1}{2}\left(\frac{\bar{\lambda}}{\lambda_0}\right)^2$$

and let

$$l' = \frac{\log \frac{1}{\epsilon}}{\log\left(\frac{\sqrt{2}|\bar{\lambda}|}{\lambda_0}\right)}.$$

In our case $1 - \beta$ is too large to apply the method directly and we need some initial amplification. To do that, each vertex would now correspond to a set of assignments to the random bits, such that the probability that at least one element associated with a randomly chosen vertex is a member of $\mathcal{S}$, is at least $1 - \alpha_G$. For the purposes of this paper, this can be done by letting each vertex correspond to h independent samples where $h = \log_{1-\beta} \alpha_G$. Following Karp and Pippenger [19], a more efficient method (in terms of constant factors) can be devised by taking the neighborhoods of a path. To conclude, the number of random bits used is $\log |\mathcal{F}| + \ell' \cdot \log d$.

3.3 Combining the samples

We have to specify how to choose a linear combination of the vectors $r_1, \ldots, r_l$, given that with probability $1 - \epsilon$, for any subset S, Prob[for all i, $\chi_S(r_i) = 0] \leq \epsilon$]. The simplest way is to choose $\vec{a}$ is uniformly at random. (Notice that for a particular subset S, if there exists a vector r_j among the vectors $r_1, \ldots, r_k$ such that $\chi_S(r_j) = 1$, then the bias of that subset is zero).

Claim 3.6. *The random variables $x_1, \ldots, x_n$ generated by choosing $\vec{a}$ uniformly in random from $\{0,1\}^l$ are ϵ-biased.*

The number of random bits required is l. Although the size of l is $O(\log n)$, to optimize on the constants, we save random bits by choosing $\vec{a}$ to be ϵ_1-biased. Because of the initial amplification, the constants may be large.

Remark 3.7. *If for all subsets S the probability that at least one r_i is a distinguisher for S is at least $1 - \epsilon_2$ and $\vec{a}$ is chosen to be ϵ_1-biased, then $\mathbf{x}$ is ϵ-biased for $\epsilon = \epsilon_1 + \frac{\epsilon_2}{2} - \epsilon_1 \cdot \epsilon_2$.*

To conclude,

Theorem 3.8. *Generating n $\{0,1\}$ random variables that are ϵ-biased can be done using $O(\log n + \log \frac{1}{\epsilon})$ random bits. Thus, the size of the sample space is $2^{O(\log n + \log \frac{1}{\epsilon})}$. Given the random bits, computing the value of a random variable can be done in time poly-logarithmic in n.*

4 Generating k-wise δ-dependent random variables

We are insured by Corollary 2.1 that if the random variables $x_1, \ldots, x_n$ are ϵ-biased, then they are k-wise δ-dependent for $\delta = 2^k \cdot \epsilon$. However, there is a more efficient construction. This can be done by combining our methods with those of [4] for generating k-wise independent variables.

Suppose we want to generate $\{0,1\}$ uniform random variables $y_1, \ldots, y_n$ that are k-wise independent. [4] suggest that this can be done by taking n vectors $L_1, \ldots, L_n$ of length h such that the vectors are k-wise linearly independent over GF[2]. If the vectors $L_1, \ldots, L_n$ are columns of the parity check matrix of BCH codes, then h is $\frac{k}{2} \log n$. Let R be a vector chosen uniformly in random from $\{0,1\}^h$; for all i, $1 \leq i \leq n$, let $y_i = L_i \cdot R$. The number of random bits required for the construction is $k \log n$.

In order to improve on Corollary 2.1, instead of choosing R uniformly at random, suppose that the entries of R are ϵ-biased random variables.

Lemma 4.1. *Let $y_1, \ldots, y_n$ be random variables generated by the above method, where the entries of R are ϵ-biased random variables. Then, $y_1, \ldots, y_n$ are k-wise ϵ-biased.*

Proof: Let $S \subseteq \{1, \ldots, n\}$ be a subset of cardinality at most k. We bound $bias(S)$. For all $i \in S$, $y_i = L_i \cdot R$. Hence,

$$\sum_{i \in S} y_i = \sum_{i \in S} L_i \cdot R = R \cdot \sum_{i \in S} L_i = R \cdot M$$

For $i \in S$, the vectors L_i are linearly independent, and hence $M \neq 0$ and $bias(S) \leq \epsilon$. □

The improvement over corollary 2.1 in the cardinality of the sample space is that now we have decreased the number of ϵ-biased random variables from n to $k \log n$. Recall that random variables that are k-wise ϵ-biased, are also k-wise $2^k \cdot \epsilon$-dependent.

Lemma 4.2. *The logarithm of the cardinality of the sample space needed for constructing k-wise δ-dependent random variables is $O(k + \log\log n + \log \frac{1}{\delta})$.*

5 A moment inequality

Basic tools in probabilistic analysis are the moment inequalities [13] that bound the deviation of a random variable from its expected value. More specifically, let y be a random variable such that $E[y] = 0$, then,

$$\text{Prob}[|y| \geq \lambda] \leq \frac{E[|y|^k]}{\lambda^k}$$

Let $b_1, \ldots, b_n$ be random variables that have a binomial distribution, i.e., they are independent and take their values from $\{-1, 1\}$ uniformly. The k^{th} moment of their sum will be denoted by B_k, that is $B_k = E\left[|b_1 + \ldots + b_n|^k\right]$.

We formulate a moment inequality for the sum of $\{-1, 1\}$ random variables $(x_1, \ldots, x_m)$ that are k-wise ϵ-biased. We denote their sum by $S = \sum_{i=1}^{m} x_i$. This will be done by bounding the k^{th} moment of S via the k^{th} moment of the binomial distribution on m uniform independent Bernoulli variables, denoted by B_k. (We assume w.l.o.g. here that k is even). The k^{th} moment of S contains m^k terms, where each term contains at most k variables. More specifically,

$$E[S^k] = E\left[\sum_{i_1 i_2 \ldots i_k} x_{i_1} x_{i_2} \ldots x_{i_k}\right] = \sum_{i_1 i_2 \ldots i_k} E\left[x_{i_1} x_{i_2} \ldots x_{i_k}\right]$$

Each term in the above summation is of the form $T_i = x_{i_1}^{p_1} x_{i_2}^{p_2} \ldots x_{i_r}^{p_r}$, such that $\sum_{j=1}^{r} p_j = k$. If a term T_i contains a variable whose power is odd, then in B_k its expectation is 0. However, in our case, it follows from the definition of k-wise ϵ-biased random variables (Definitions 2.2 and 2.3), that the expected value of T_i can be at most ϵ. If all the powers in a term T_i are even, then the expected value in both cases is the same and equal to 1. Hence,

$$\text{Prob}[|S| \geq \lambda] \leq \frac{B_k + \epsilon \cdot m^k}{\lambda^k}$$

6 Derandomization

The probability distribution D constructed in Section 3 can be used for derandomizing algorithms. A randomized algorithm $\mathcal{A}$ has a probability space (Ω, P) associated with it, where Ω is the sample space and P is some probability measure. We call a point $w \in \Omega$ a *good* point for some input instance I, if $\mathcal{A}(I, w)$ computes the correct solution. A derandomization of an algorithm means searching the associated sample space Ω for a good point w with respect to a given input instance I. Given such a point w, the algorithm $\mathcal{A}(I, w)$

is now a deterministic algorithm and it is guaranteed to find the correct solution. The problem faced in searching the sample space is that it is generally exponential in size.

[22, 24, 4] suggested the following strategy: show that the probabilistic choices of certain randomized algorithm are only required to be k-wise independent. Hence, a sample space of size $O(n^k)$ suffices. This sample space can be exhaustively searched for a good point (even in parallel) when k is a constant. A similar strategy can be used with ϵ-biased random variables. First, show that a randomized algorithm has a non-zero probability of success when the probabilistic choices are ϵ-biased. Then, conduct a search of the sample space associated with the variables to find a good point.

This scheme can in principle be applied to all the randomized algorithms for which the limited independence approach was applied. However, we do not necessarily get better algorithms. An attractive feature of our scheme is that random variables that are $\log n$-wise δ-dependent, for δ which is polynomially small, can be constructed with a *polynomial* sample space. Intuitively, this means that we can achieve "almost" $\log n$-wise independence with a polynomial sample space (as opposed to $n^{O(\log n)}$).

The k-wise ϵ-biased random variables are especially useful when the proof that a randomized algorithm is successful involves any moment inequality. In that case, one should compute what is the appropriate ϵ such that the error incurred leaves the probability of success non-zero. We exemplify this by showing how a RNC algorithm for the *set balancing* problem can be converted into an NC algorithm.

The set balancing problem is defined as follows. A collection of subsets $\mathcal{S} = \{S_1, S_2, \cdots S_m\}$ defined over a base set $B = \{b_1, b_2, \cdots b_n\}$ is given such that the cardinality of each subset is δ. The output is a $\{-1, 1\}$ coloring of B into two sets. Let the 2-coloring of B be denoted by $\mathbf{x} = (x_1, x_2, \cdots x_n)$. The discrepancy of a subset S_i, $\Delta(S_i, \mathbf{x})$, with respect to $\mathbf{x}$, is defined as $\sum_{j \in S_i} x_j$. The discrepancy of the set family $\mathcal{S}$, $\Delta(\mathcal{S}, \mathbf{x})$, is defined to be the maximum discrepancy among the subsets. Spencer [34] devised a polynomial-time deterministic algorithm which guarantees a 2-coloring $\mathbf{x}$ such that $\Delta(\mathcal{S}, \mathbf{x}) = O(\sqrt{n \log n})$. This was improved to $O(\sqrt{\delta \log n})$ by Raghavan [29]. For parallel algorithms we cannot guarantee as small a discrepancy as in the sequential case. However, we can come arbitrarily close to the sequential bounds by computing a τ-good coloring.

Definition 6.1. *A 2-coloring $\mathbf{x}$ is τ-good for a set family $\mathcal{S}$ if, for $0 < \tau < \frac{1}{2}$, $\Delta(\mathcal{S}, \mathbf{x})$ is bounded by $\delta^{0.5+\tau}\sqrt{\log n}$.*

There is a simple RNC algorithm to find a τ-good coloring for any $\mathcal{S}$: pick a random $\mathbf{x}$ uniformly in random from $\{0,1\}^n$. It turns out that for any $\mathcal{S}$, the 2-coloring is τ-good with sufficiently high probability. (See for example [21]). This algorithm was derandomized by [8, 25] by proving that $\log n$-wise independence suffices, and then showing how to conduct a binary search in a sample space of size $n^{O(\log n)}$. (The method of conditional probabilities). We present a direct derandomization which can be implemented in NC^1.

Assume that B is colored in random; [8, 25] prove the following lemma.

Lemma 6.2. *Let $k = a \log n / \log \delta$ be even, where $a > \frac{1}{\tau}$, and let $\mathbf{x}$ be k-wise independent uniform $\{-1, 1\}$ random variables. Then, for any input set family $\mathcal{S}$, the probability that $\mathbf{x}$ is τ-good is non-zero.*

We sketch very briefly the proof idea. Let T_i denote the sum of the random variables in the subset S_i. The main tool used is the k^{th} moment inequality. (See Section 5). More specifically, [8, 25] show that for a given subset S_i,

$$\mathrm{Prob}\left[\left|T_i - \frac{\delta}{2}\right| > \delta^{0.5+\tau}\sqrt{\log n}\right] \leq \frac{E[T_i^k]}{(\delta^{0.5+\tau}\sqrt{\log n})^k} < \frac{1}{\delta^{\tau k}} < \frac{1}{n}$$

Summing over all subsets, we get that for any set family $\mathcal{S}$, the probability that a random k-wise coloring is τ-good is non-zero.

What happens when the random variables in $\mathbf{x}$ are k-wise ϵ-biased? We want to choose ϵ such that the probability that the discrepancy in a subset is large, is smaller than $\frac{1}{n}$. Following Section 5, ϵ and k must be chosen such that

$$\frac{1}{\delta^{\tau k}} + \frac{\epsilon \delta^k}{(\delta^{0.5+\tau}\sqrt{\log n})^k} < \frac{1}{n}$$

We choose $k = \frac{\log 2n}{\tau \log \delta}$, and the above inequality holds if

$$\epsilon < \frac{1}{2n^{1+\frac{1}{\tau}}}$$

Hence, the sample space will be at most of cardinality $O(n^{\frac{1}{\tau}})$ and if τ is a constant, an exhaustive search of the sample space can be conducted. Both the construction and the search can be done in NC^1.

For the relationship between the set balancing problem and the lattice approximation problem [29], the reader is referred to [25].

6.1 Finding heavy codewords

It is well known that every linear code contains a word of weight $\frac{n}{2}$ (number of "1"s), where n is the length of the codewords. To see that, let G be the generator matrix of the linear code. If x is chosen uniformly in random, then $E[\text{weight of } Gx] = \frac{n}{2}$.

If x is chosen from an ϵ-biased probability space, then $E[\text{weight of } Gx] = n(\frac{1-\epsilon}{2})$. If $\epsilon < \frac{1}{2n}$, then there must be a codeword x in the ϵ-biased probability space such that its weight is at least $\frac{n}{2}$. This places the problem in NC for the first time. It also exhibits that ϵ-biased random variables are also needed as opposed to just k-wise ϵ-biased random variables.

7 Reducing the number of random bits

In this section we present two algorithms for which the number of random bits required can be reduced from linear to logarithmic.

7.1 Matrix multiplication verification

Suppose that three $n \times n$ matrices A, B and C over an arbitrary ring are given; what is the complexity of verifying whether $A \cdot B = C$? Can this be done by avoiding matrix multiplication? This problem was first considered by Freivalds [14] who suggested a randomized algorithm of complexity $O(n^2)$, but it required n random bits. In this section we show how to implement it using only $O(\log n)$ random bits with no time penalty. Our results can readily be generalized to non-square matrices as well.

Let us first review Freivalds' algorithm [14]. Choose a random vector $\vec{r}$ such that each entry in r is picked uniformly at random to be zero or some fixed non-zero element of the ring. Test whether $r \cdot A \cdot B - r \cdot C = 0$. The complexity of applying this procedure is that of multiplying a matrix by a vector which is obviously $O(n^2)$.

Theorem 7.1. *Suppose that r is a vector chosen in the manner described above. Then with probability at least $\frac{1}{2}$, if $A \cdot B \neq C$, then also $r \cdot A \cdot B \neq r \cdot C$.*

Proof: Let v be a non-zero column vector of $A \cdot B - C$ and let j denote the number of its non-zero entries. The number of different vectors r such that $v \cdot r = 0$ is at most $2^{j-1} \cdot 2^{n-j} = 2^{n-1}$, whereas the number of distinct choices for r is 2^n. Hence, the probability that $v \cdot r = 0$ is at most $\frac{1}{2}$. □

It follows from the above proof that we can restrict ourselves to the following problem: given a vector v, check whether it is identically zero, where the only operations permitted on the vector are taking its inner product with another vector. Recall from Section 3.1 that a vector r that verifies that a particular vector v is non-zero is called a *distinguisher* (with respect to v). Hence, Reducing the number of random bits for the above problem is equivalent to generating a small collection of vectors $\mathcal{F}$, such that the inner product of a constant fraction of them with *any* vector is non-zero.

The family $\mathcal{F}$ that is constructed in Section 3.1 has this property: with probability at least β, a vector r sampled uniformly will be a distinguisher. The number of random bits needed for sampling $\mathcal{F}$ is $O(\log n)$ and the complexity of the algorithm remains $O(n^2)$.

Notice that if the matrices are over an arbitrary ring, then the construction presented in Section 3.1.2 cannot be used. We can only use the one presented in Section 3.1.1.

7.2 Simultaneous verification of exponentiation

Suppose that for some prime p and a we are given n pairs $(x_1, y_1), (x_2, y_2), \ldots (x_n, y_n)$ and we want to verify whether the equality $a^{x_i} = y_i \bmod p$ is true for all $1 \leq i \leq n$. A. Fiat and M. Naor have suggested the following randomized algorithm:

1. Pick a random vector $r = (r_1, \ldots, r_n)$ where each r_i is chosen uniformly and randomly from $\{0, 1\}$.
2. Compute $t = \sum_{i=1}^{n} r_i \cdot x_i \bmod (p-1)$ and $m = \prod_{i \in S} y_i^{r_i} \bmod p$.
3. Test whether $a^t p = m \bmod p$.

As in Freivalds' algorithm, if for any $1 \leq i \leq n$, $a^{x_i} \neq y_i$, then the above algorithm will detect it with probability at least $\frac{1}{2}$. The complexity of this algorithm is n multiplications, instead of $n \log p$ for checking each equality separately.

We can actually phrase the problem as that of finding a distinguisher for a non-zero vector in the integer ring modulo $p-1$. Let z_i be such that $a^{z_i} = y_i$. Then, a distinguisher to the vector w must be found, where $w_i = x_i - z_i \bmod p$. For that we can use the construction of section 3.1 in a similar way to Section 7.1.

Note that the expected size of each entry in Step 4 of the algorithm in Section 3.1.1 is $O(1)$, and therefore, the expected number of multiplications remains $O(n)$.

Suppose that a and N are integers such that $(N, a) = 1$. The above procedure can be applied to verify n equalities of the type $x_i^a = y_i \bmod N$. This may be used for instance to check n given RSA equations [30].

8 Vector sets for exhaustive testing

A problem that has received attention in the fault diagnostic literature is that of generating a small set of vectors $T \subset \{0,1\}^n$ such that for any k-subset of indices $S = \{i_1, i_2, \ldots i_k\}$, the projection of T on the indices S contains all possible 2^k configurations. See [32] for bibliography on the problem. Such a set of vectors is called (n,k)-universal. The motivation for this problem is that such a test set allows exhaustive testing of a circuit where each component relies on at most k inputs.

The best known bounds for constructing (n,k) universal sets are given in [32] and [3]. The connection between k-wise δ-dependent probability spaces and small (n,k)-universal sets is made in the next proposition.

Proposition 8.1. *If Ω is a k-wise δ-dependent probability space for $\delta \leq 2^{-k}$, then Ω is also a (k,n)-universal set.*

Proof: If for a k-subset $i_1, i_2, \ldots i_k$, there is a $\{0,1\}^k$ configuration which has probability 0 in Ω, then the distance from the uniform distribution of $x_{i_1}, x_{i_2}, \ldots x_{i_k}$ is at least $2^{-(k-1)} > \delta$. □

From Lemma 4.2 we can construct a probability space Ω which is k-wise 2^{-k}-dependent such that Ω is $\log n \cdot 2^{O(k)}$. This is better than the best explicit construction given in [32], and matches the lower bound given there up to constant factors. Alon [3] has shown an explicit construction of size $\log n \cdot 2^{O(k^2)}$ which is optimal for constant k. No such constructions were known for larger values of k. For k which is $\Theta(\log n)$, this is the first explicit construction of an (n,k)-universal set of polynomial size.

9 Communication complexity

Suppose player A has a string $x \in \{0,1\}^n$ and player B has a string $y \in \{0,1\}^n$, and they wish to decide whether $x = y$ while exchanging as few bits as possible. It is well known that this can be done by exchanging $O(\log n)$ bits and the probability of the protocol arriving at the correct result is at least a constant. We can show how to achieve probability of success which is any $\frac{1}{\text{poly}(n)}$, while maintaining the logarithmic communication complexity.

The algorithm will use the first two stages in constructing the small biased probability spaces. To achieve probability of error ϵ:

1. Player A chooses $O(\log n + \log \frac{1}{\epsilon})$ random bits that define vectors $r_1, r_2, \ldots, r_l$. She sends the random bits to player B and also sends $< r_1, x >, < r_2, x >, \ldots < r_l, x >$.
2. B computes $< r_1, y >, < r_2, y >, \ldots < r_n, y >$. If for any i, $< r_i, y > \neq < r_i, x >$ he announces it. Otherwise, he conclude that $x = y$.

If $x \neq y$, then with probability at least $1-\epsilon$, for at least one i, $< r_i, x \oplus y > \neq 0$ and hence $< r_i, x > \neq < r_i, y >$.

Once a distinguisher has been found, detecting an index of a bit on which players A and B differ is easy by exchanging $\log n$ bits.

Another application is for the problem of determining set equality. Suppose that $A, B \subset \{1, \ldots, n\}$ and we wish to decide whether $A = B$. We would like a single pass on the elements of A and B and the amount of memory should be $O(\log n)$. Here again a method that achieves constant probability error is known (see Blum and Kannan [7] for a description).

We will show that it is possible to achieve any ϵ error while using only $O(\log n + log \frac{1}{\epsilon})$ bits of memory and a linear number of operations. Let v_A and v_B denote the incidence vectors of A and B. We can think of the problem as deciding whether $v_A \oplus v_B = 0$. Again, we can use the first two stage of the construction of small bias probability spaces.

1. pick $O(\log n + \log \frac{1}{\epsilon})$ random bits that define $r_1, r_2, \ldots, r_l$. (They are not computed explicitly at this point.) Let $r_i(a)$ denote the ath coordinate of r_i.
2. Initialize bits $d_1, d_2, \ldots d_l$ to 0.
3. For each element $a \in A$ and for each $1 \leq i \leq l$ $d_i \leftarrow d_i \oplus r_i(a)$.
4. For each element $b \in B$ and for each $1 \leq i \leq l$ $d_i \leftarrow d_i \oplus r_i(b)$.
5. If all the d_i are 0 decide that $A = B$, otherwise $A \neq B$.

As before, if for at least one $i < r_i, v_A \oplus v_B > \neq 0$ we will detect it. The probability that this happens is at least $1-\epsilon$. For this application the first method of construction might be useful, since we are not interested in the full vector r_i, but at selected locations of it.

This can be looked upon as a family $\mathcal{H}$ of hash functions with the following property. Let $h \in \mathcal{H}$, where $h : \{1 \ldots n\} \rightarrow \{1 \ldots m\}$. The family $\mathcal{H}$ is accessible with $O(\log n + \log m)$ bits and for any subsets A and B of $\{1 \ldots n\}$, the probability that $\sum_{a \in A} h(a) = \sum_{b \in B} h(b) < \frac{1}{m^\beta}$ for β constant, where addition is bitwise XoR.

Acknowledgement

We would like to thank Shuki Bruck for his contribution to Section 3.1.2. We would also like to thank Noga

Alon, Yishay Mansour and Avi Wigderson for helpful discussions.

References

[1] M. Ajtai and A. Wigderson, Deterministic simulation of probabilistic constant depth computation, FOCS '85, pp. 11-19.

[2] M. Ajtai, J. Komlos and E. Szemeredi, Deterministic simulation in LOGSPACE, STOC '87, pp. 132-140.

[3] N. Alon, Explicit constructions of exponential sized families of k-independent sets, Discrete Math, 58, pp. 191-193 (1986).

[4] N. Alon, L. Babai and A. Itai, A fast and simple randomized parallel algorithm for the maximal independent set problem, J. of Alg., 7, pp. 567-583 (1986).

[5] E. Bach, Realistic analysis of some randomized algorithms, STOC '87, pp. 453-461.

[6] R. Ben-Nathan, M.Sc. Thesis, Hebrew University (1990).

[7] M. Blum and R. Kannan, Designing programs that check their work, STOC '89, pp. 86-97.

[8] B. Berger and J. Rompel, Simulating $(\log^c n)$-wise independence in NC, FOCS '89 pp. 2-7.

[9] B. Chor and O. Goldreich, On the power of two-point based sampling, J. of Complexity, 5, pp. 96-106 (1989).

[10] B. Chor, O. Goldreich, J. Hastad, J. Friedman, S. Rudich and R. Smolensky, The bit extraction problem or t-resilient functions, FOCS '85, pp. 396-407.

[11] A. Cohen and A. Wigderson, Dispersers, deterministic amplification and weak random sources, FOCS '89, pp. 14-19.

[12] A. Cohen and A. Wigderson, Multigraph Amplification, Survey (1989).

[13] W. Feller, An Introduction to probability theory and its applications, John Wiley, 1968.

[14] R. Freivalds, Fast probabilistic algorithms, Springer Verlag Lecture Notes in CS #74, Mathematical Foundations of CS, pp. 57-69 (1979).

[15] O. Gaber and Z. Galil, Explicit construction of linear size superconcentrators, JCSS, 22, p. 407 (1981).

[16] R. Impagliazzo and D. Zuckerman, Recycling random bits, FOCS '89, pp. 248-253.

[17] S. Jimbo and A. Marouka Expanders obtained from affine transformations, STOC '85.

[18] J. Justesen, A class of asymptotically good algebraic codes, IEEE trans. Infor. Theory, 18 (1972) 652-656.

[19] R. Karp and N. Pippenger, A time randomness tradeoff, AMS conference on probabilistic computation and complexity, Durham NC, (1983).

[20] H. Karloff and P. Raghavan, Randomized algorithms and pseudorandom numbers, STOC '88, pp. 310-321.

[21] H. Karloff and D. Shmoys, Efficient parallel algorithms for edge coloring problems, J. of Alg., Vol. 8 (1987), pp. 39-52.

[22] R. M. Karp and A. Wigderson, A fast parallel algorithm for the maximal independent set problem, JACM, 32 (1985), pp. 762-773.

[23] A. Lubotzky, R. Phillips and P. Sarnak, Explicit expanders and the Ramanujan conjecture, STOC '86.

[24] M. Luby, A simple parallel algorithm for the maximal independent set problem, SICOMP, 15, pp. 1036-1053 (1986).

[25] R. Motwani, J. Naor and M. Naor, The probabilistic method yields deterministic parallel algorithms, FOCS '89, pp. 8-13.

[26] F. J. MacWilliams and N. J. A. Sloane, The theory of error correcting codes, North Holland, Amsterdam, 1977.

[27] N. Nisan and A. Wigderson, Hardness vs. Randomness, FOCS '88, pp. 2-11.

[28] R. Peralta, On the randomness complexity of algorithms, University of Wisconsin, Milwaukee, CS Research Report TR 90-1.

[29] P. Raghavan, Probabilistic construction of deterministic algorithms: approximating packing integer programs, JCSS, 37 (1988), pp. 130-143.

[30] R. Rivest, A. Shamir and L. Adelman, CACM (1978).

[31] M. Santha, On using deterministic functions to reduce randomness in probabilistic algorithms, Information and Computation 74, pp. 241-249 (1987).

[32] G. Seroussi and N. Bshouti, Vector sets for exhaustive testing of logic circuits, IEEE Trans. on Info. Theory, vol. 34, pp. 513-522 (1988).

[33] M. Sipser, Expanders, randomness, or time versus space, JCSS, Vol. 36, pp. 379-383 (1988).

[34] J. Spencer, Ten lectures on the probabilistic method. SIAM (Philadelphia), 1987.

[35] V. Vazirani, Randomness, adversaries and computation, Ph.D. Thesis, University of California, Berkeley (1986).