Dispersers, Deterministic Amplification, and Weak Random Sources.

Aviad Cohen Avi Wigderson *

The Hebrew University

Abstract

We use a certain type of expanding bipartite graphs, called **disperser graphs**, to design procedures for picking highly correlated samples from a finite set, with the property that the probability of hitting any sufficiently large subset is high. These procedures require a relatively small number of random bits and are robust with respect to the quality of the random bits.

Using these sampling procedures to sample random inputs of polynomial time probabilistic algorithms, we can simulate the performance of some probabilistic algorithms with less random bits or with low quality random bits. We obtain the following results:

1. The error probability of an RP or BPP algorithm that operates with a constant error bound and requires n random bits, can be made exponentially small (i.e. 2^{-n}), with only $(3+\epsilon)n$ random bits, as opposed to standard amplification techniques that require $\Omega(n^2)$ random bits for the same task. This result is nearly optimal, since the information theoretic lower bound on the number of bits required for such an amplification is $2n$.
2. It is shown that the output of any random source whose Renyi entropy rate exceeds $\frac{1}{2}$ ($\frac{3}{4}$), can be used to simulate RP (BPP) algorithms. This is far from the information theoretic lower bound which is $m^{\mu-1}$, where m is the number of bits and $0 < \mu < 1$ is any constant. We show that the lower bound can be achieved for a specific class of random sources called oblivious bit fixing sources.

1 Introduction

1.1 The Problem of Reducing Randomness in Probabilistic Algorithms.

Randomness is an important computational resource. In some computational problems (i.e. routing in degree bounded networks, decision trees, communication complexity, finite automatons), randomized algorithms are provably more efficient than their deterministic counterparts [SW86, KPU88, Lov90, FRE81]. As far as the main computational model the Turing machine is concerned, the power of randomization is not known. However, there are many problems for which the known randomized algorithms perform better (in time or space) than the known deterministic ones.

*Partially supported by a grant for research in Electronics, Computers and Communication, administered by the Israel Academy of Sciences and Humanities.

Since randomness is a computational resource, it is natural to try to reduce its amount and in general to investigate the extent to which it can be traded with the deterministic resources. There are two natural ways in which one might set about reducing the amount of randomness available to a probabilistic algorithm:

1. Reduce the number of random bits.
2. Use random bits output by an imperfect random source (i.e. allow nonuniform distributions and dependence among the random bits).

The problem of reducing the number of random bits has been approached in several ways. Pseudorandom generators have been given, for AC^0 circuits [NW88], and LOGSPACE machines [BNS88, Nis90]). Derandomization techniques for eliminating randomness altogether have been devised for certain situations where strong dependence among the random bits is allowed [Lub85, Lub88, Ra88, MNN89, BR89, NN90]. Various methods have been developed to reduce the error probability of a probabilistic algorithms, with a relatively small extra cost of random bits [KPS85, CG86].

The problems of extracting fully random bits from the outputs of imperfect random sources and simulating probabilistic algorithms with such outputs have been studied for a variety of imperfect random sources [VN51, Blu84, SV84, VV85, Vaz86, CG85, CGH$^+$85, BOL85, KKL88, LLS87]. Bit extraction has been proved possible in some cases [VN51, Blu84, CGH$^+$85] yielding immediately simulations of probabilistic algorithms with these sources. In other cases bit extraction was proved to be impossible [SV84, CG85, CGH$^+$85, BOL85, KKL88, LLS87]. Simulations of probabilistic algorithms when bit extraction is impossible, have been given in some cases [VV85, Vaz86, CG85].

1.2 The Approach of This Paper.

The approach of this paper is to translate the simulation of probabilistic algorithms to the following statistical problem: Given a finite domain X estimate with high confidence the size of a sufficiently large subset $S \subset X$, using a small number of highly correlated samples from X.

To tackle the statistical problem we develop efficient procedures for sampling elements from a finite set, with the property that the probability of hitting any sufficiently large subset is high. These procedures require a relatively small number of random bits since the samples are highly correlated, and turn out to be robust with respect to the quality of the random bits used in the sampling.

The sampling procedures developed here rely on a certain type of expanding bipartite graphs which we call **disperser** graphs. The more expanding the graphs are, the better are the statistical properties of the induced sampling procedure. We give explicit constructions of families of disperser graphs with better expansion properties than achieved so far, using a method of [AKS87] of random walks on expander digraphs.

Since the above graphs are to be used by polynomial time algorithms, we emphasize the notion of an **efficiently constructible** family of graphs, that is a family of graphs that can be constructed by a polynomial time Turing machine. It should be stressed that not all the explicit definitions of graph families given in literature, are efficient. We give a precise definition of efficient constructibility, and make sure that all algorithms use efficiently constructible graphs.

To get optimal results in the sampling procedures mentioned above, one requires degree bounded digraphs for which λ_0 and $\bar{\lambda}$, the largest and second largest eigenvalues, are optimally separated i.e.

digraphs for which $\bar{\lambda} = O(\sqrt{\lambda_0})$. [1] The simplest solution would be to use the explicitly defined, degree bounded digraphs given by [LPS86], but these constructions are not known to be efficient. Therefore we use degree unbounded family digraphs which achieve the above eigenvalue separation. Various efficient constructions of such digraphs do exist.

Using sampling procedures induced by efficiently constructible families of disperser graphs, we obtain a uniform method of simulating certain polynomial time probabilistic algorithms, with a small number of random bits, or with random bits of low quality. This enables us to give new results for two problems that have been studied in the context of reducing the randomness of polynomial time probabilistic algorithms:

- **The deterministic amplification problem:** amplify the success probability of a probabilistic algorithm, adding as few random bits as possible.
- **Simulation of probabilistic algorithms with imperfect random sources:** maintain the performance of a probabilistic algorithm, when the supply of random bits comes from some imperfect source of randomness.

We will now describe previous work concerning these problems and our contributions.

1.3 Deterministic Amplification.

Let A be an RP (BPP) algorithm [2] which decides a language $\mathcal{L}$ with n random bits and an error bounded by $\frac{1}{2}$. The standard way of amplifying the success of A from $\frac{1}{2}$ to $1-2^{-k}$, is to run it k independent times and accept if at least one of the computations accepts. This requires kn random bits. In particular, $\Omega(n \log n)$ random bits are required to make the error probability polynomially small (i.e $\frac{1}{P(n)}$ for some polynomial P), and n^2 random bits are required to make the error probability exponentially small (i.e. 2^{-n}). The deterministic amplification problem is to achieve these error bounds with less random bits.

The first result on deterministic amplification was given by Karp, Pippenger and Sipser [KPS85]. They showed that the error probability of an RP algorithm can be made polynomially small, with no additional random bits. Chor and Goldreich [CG86], showed that the same amplification can be achieved for BPP algorithms, at the cost of doubling the amount of random bits. Bach [Bac87] showed that an exponentially small error can be attained for specific algorithms (e.g. primality testing) , by increasing the number of random bits by only a constant factor.

Using the sampling technique based on random walks on expander graphs we show that one can amplify a constant success probability of an RP or BPP algorithm, to $1-2^{-k}$, using $n+O(k)$ random bits. In particular, we show that for any $\epsilon > 0$:

1. If the error probability of a BPP algorithm is bounded by $\frac{1}{8}$, then it can be made polynomially small with no further cost in random bits.
2. The error probability of an RP algorithm can be made less than 2^{-n}, with $(3+\epsilon)n$ random bits.

[1] It can be shown that any family of digraphs satisfies: $\bar{\lambda} = \Omega\left(\sqrt{\lambda_0}\right)$. See theorem 2.4.

[2] If you are not familiar with the complexity classes RP and BPP see 2.3.2.

3. The error probability of a BPP algorithm can be made less than 2^{-n}, with $(6+\epsilon)n$ random bits. These results are nearly optimal, since the information theoretic lower bound is $2n$ random bits.

is $2n$ random bits.

Impagliazzo and Zuckerman [IZ], have obtained independently the result that the error probability of an RP (BPP) algorithm can be made exponentially small at a linear cost of random bits.

1.4 Imperfect Random Sources.

Von Neumann [VN51] initiated the study of imperfect random sources by showing that a linear number of perfect random bits can be extracted from independent tosses of a biased coin. Blum [Blu84] generalized the ideas of Von Neuman and showed that a linear amount of random bits can be extracted from the output of a Markov source. This implies immediately that RP and BPP algorithms can be simulated with the output of a Von Neumann or Markov source. [SV84, VV85, Vaz86, CG85] studied imperfect sources from the output of which, not a single random bit can be extracted. They showed however, that RP and BPP can be simulated with these sources, using sophisticated methods to sample the random inputs for the algorithms.

The imperfect sources studied in [SV84, VV85, Vaz86, CG85] have a constant lower bound on their entropy rate. However, the methods used by these authors do not work for other classes of imperfect sources with a constant bound on their entropy rate. One such class are the constant rate bit fixing sources of [CGH$^+$85, BOL85, LLS87]. Here the adversary can choose a linear subset of positions (the values of the rest are set by independent coin flips) and fix their values before, during, or after the coins are flipped. For this class of sources no simulation results are known. These facts raise the question whether a constant lower bound on the entropy rate or some other global information-theoretic quantity, is a sufficient condition for the simulation, and if so, whether there exists a uniform simulation algorithm, that applies to all random sources satisfying the lower bound.

Using the sampling procedure developed in this work, we show that for any $\nu > 1$, any random source with with Renyi ν-entropy rate [3] greater than $\frac{1}{2}$ ($\frac{3}{4}$) can simulate RP (BPP). This yields a "black box" simulation method for any source satisfying the above information theoretic condition regardless of the specific details of its probability distribution. In particular we obtain new simulation results for constant rate bit fixing sources: RP and BPP can be simulated with bit fixing sources, if the fraction of the bits fixed, does not exceed $\frac{1}{2} - \epsilon$ ($\frac{1}{4} - \epsilon$). It also gives a new simulation method for the sources of [SV84, VV85, Vaz86, CG85], but only for a subrange of parameters values. Recently, Zuckerman [Zuc], has improved our results, showing that BPP can be simulated with any source that has a constant lower bound on its Renyi entropy rate.

A simple counting argument shows that a simulation of an RP or BPP algorithm using m random bits, requires $m^{\Omega(1)}$ Renyi entropy. This a leaves a wide gap between the lower bound and the upper bounds stated above. We show that for the weakest bit fixing source the lower bound can be achieved: An $m^{\Omega(1)}$ entropy (i.e. $m - m^{\Omega(1)}$ positions are fixed) suffices to simulate BPP.

[3] See section 6.3 for the definition of the Renyi generalized entropies.

1.5 Organization of the Paper.

In section 2 we present basic definitions and tools. 2.1 deals with the spectrum of a digraph and its relation to the expansion properties. 2.2 gives notations concerning bipartite graphs that are used quite frequently in this paper. 2.3 recalls the basic notions of probabilistic algorithms relevant to this paper.

Section 3 deals with procedures for picking highly correlated samples (called sampling procedures) and their relation to certain expanding bipartite graphs called disperser graphs. 3.1 discusses the connection between weak randomness in probabilistic algorithms and sampling procedures. In 3.2 we define the concepts of sampling procedures and amplification properties of sampling procedures. Amplifying sampling procedures will prove useful in simulations of probabilistic algorithms. In 3.3 it is shown that bipartite graphs give rise to sampling procedures and that certain expanding bipartite graphs (disperser graphs) yield amplifying sampling procedures. 3.4, 3.5 and 3.6, discuss the property of efficient constructibility, that sampling procedures or their associated graphs should satisfy if they are to be used in polynomial time algorithms.

In section 4 we give constructions of disperser graphs with amplification properties that have not been achieved before. 4.1 shows by a probabilistic argument that highly expanding disperser graphs do exist. In 4.2 explicit definitions of highly expanding disperser graphs are given. In 4.3 it is shown how these explicit definitions can be turned into efficient constructions.

Section 5 gives an application of disperser graphs to the deterministic amplification problem. 5.1 describes the general approach. 5.2 and 5.3 give results on polynomial and exponential amplifications of RP and BPP.

Section 6 gives an application of disperser graphs to the problem of simulating probabilistic algorithms with imperfect random sources. 6 defines the problem. 6.2 discusses results on imperfect random sources obtained so far. 6.3 introduces a new model of a random source more general than those discussed before. The source is defined by a lower bound on its Renyi entropy rate. In 6.4 we give simulations of RP and BPP with this general random sources. We show that $\frac{1}{2}+\epsilon$ ($\frac{3}{4}+\epsilon$) suffices for the simulation of RP (BPP). This yields new simulation results for various random sources. in 6.5 we point out that any simulation using disperser graphs requires a certain lower bound on its Renyi entropy, but this lower bound is much smaller than the constant upper bound of 6.4. In 6.6 we match the lower bound for a particular random source, the oblivious bit fixing source.

2 Preliminaries.

2.1 Digraphs, Their Spectra and Expansion.

In this paper we work with strongly connected regular digraphs. Multiple edges and self loops are allowed. The adjacency matrix A of a digraph G is given by:

$$A_{ij} = \text{number of edges directed from vertex i to vertex j}$$

The spectrum of a digraph G on n vertices, is the set of eigenvalues (with multiplicities) of its adjacency matrix. G has n eigenvalues. We use $\lambda_0, \ldots, \lambda_{n-1}$ to denote the eigenvalues in a nonincreasing absolute value order.

The Perron-Frobenius theory (see any text on nonnegative matrices, e.x. [Gan59, Min88]) implies some useful facts about the spectra of strongly connected digraphs. One consequence of the theory is:

Fact 1 *A strongly connected digraph has a real positive eigenvalue r such that:*

1. *The multiplicity of r is 1.*
2. *There is a positive eigenvector corresponding to r. Any eigenvector associated with an eigenvalue different from r, cannot be nonnegative.*
3. *Every eigenvalue λ satisfies $|\lambda| \leq r$*

The eigenvalue r will henceforth be referred to as the **maximal eigenvalue** of A. Although the multiplicity of r is 1, G may several distinct eigenvalues of absolute value r. The number of these eigenvalues k, is called the **index of imprimitivity** of G. If $k = 1$ then G is called **primitive**. If the digraph is regular the Perron-Frobenius theory provides further information:

Fact 2 *Let G be a strongly connected d-regular digraph on N vertices, with index of imprimitivity k and adjacency matrix A.*

1. *The maximal eigenvalue is d. The k eigenvalues with maximal absolute value are $d\theta_i$ where $\{\theta_i\}_{i=1}^{k}$ are the roots of unity of order k. These eigenvalues have multiplicity 1. Under a certain numbering of the vertices the vectors:*

$$v_i = \frac{1}{\sqrt{N}}(\overbrace{\theta_i^0, \cdots, \theta_i^0}^{\frac{N}{k}}, \ldots, \overbrace{\theta_i^{k-1}, \cdots, \theta_i^{k-1}}^{\frac{N}{k}})$$

   are eigenvectors of eigenvalues $d\theta_i$

2. *Let W be the k-dimensional subspace spanned by the eigenvectors $\{v_i\}_{i=0}^{k}$. Then W and $W^{\perp}$ (the orthogonal complement of W) are invariant under the action of both A and A^T (viewed as linear operators).*

Fact 2 allows the following definition:

Definition 2.1 *Let G be a connected regular digraph with adjacency matrix A. $\bar{\lambda}(G)$ is the L_2 norm of the restriction of A to $W^{\perp}$.*

Remark 2.1 *It can be verified that if G is a graph, then $\bar{\lambda}(G)$ is equal to the second largest absolute value of the eigenvalues. Otherwise it is not less than this quantity.*

The importance of the quantity $\bar{\lambda}(G)$ lies in the fact that estimates for the expansion as well as other parameters of a digraph G can be given in terms of the separation between λ_0 and $\bar{\lambda}$. For example we will need the following theorems:

Theorem 2.2 *[FP87, AC88] Let G be a primitive d-regular digraph on N vertices, and Z a subset of vertices s.t. $|Z| = N\alpha$. For any sets of vertices S, T let E_{ST} denote the collection of of edges connecting S to T. Then:*

$$\begin{aligned} \left||E_{ZZ}| - dN\alpha^2\right| &\leq \bar{\lambda}N\alpha(1-\alpha) \\ \left||E_{ZZ^C}| - dN\alpha(1-\alpha)\right| &\leq \bar{\lambda}N\alpha(1-\alpha) \end{aligned}$$

Theorem 2.3 *[Pip85] Let G be a primitive d-regular digraph on N vertices with adjacency matrix A. Let $S \subset V(G)$ with $|S| = N\alpha$. Then the number of vertices, at least half of whose neighbours lie in S is bounded by:*

$$4N\alpha^2 + 4\left(\frac{\bar{\lambda}(A)}{\lambda_0(A)}\right)^2 N\alpha(1-\alpha)$$

In view of the above it is natural to introduce a parameter of a regular digraph, that quantifies the separation between λ_0 and $\bar{\lambda}$. Most authors use the ratio $\frac{\bar{\lambda}}{\lambda_0}$ but for our goals a more sensitive quantity is required:

Definition 2.2 *Let G be a strongly connected regular digraph. The* **separation exponent** *of G is defined by:*

$$\delta(G) = \frac{\log \bar{\lambda}}{\log \lambda_0}$$

G is said to be $\delta(G)$ **separable.** *A strongly connected d-regular digraph for which $\delta < 1$ is called separable. A family of digraphs $\{G_i\}$ is η-separable if $\delta(G_i) \leq \eta$ for each i.*

By theorems 2.2 and 2.3, the smaller δ is, the better the estimate of the expansion of the digraph. However δ cannot be made arbitrarily small. Alon and Boppana showed that for a constant degree family of d regular graphs:

$$\liminf_{i\to\infty} \bar{\lambda}(G_i) \geq 2(\sqrt{d-1})$$

This can be extended to the case of unbounded degree:

Theorem 2.4 *Let $\{G_i\}_{i=1}^{\infty}$ be a family of primitive d_i regular digraphs with adjacency matrices $\{A_i\}_{i=1}^{\infty}$ and $|G_i| = N_i$. Suppose:*

$$d_i = o(N_i)$$

then:

$$\frac{\bar{\lambda}(G_i)}{\sqrt{\lambda_0(G_i)}} \geq 1 - o(1)$$

Proof We will show that:

$$\frac{|\lambda_1(G_i)|}{\sqrt{\lambda_0(G_i)}} \geq 1 - o(1)$$

The theorem then follows since $|\lambda_1| \leq \bar{\lambda}$ for any primitive regular digraph.

As usual let $\lambda_0, \ldots, \lambda_{n-1}$ denote the eigenvalues of A_i in nonincreasing moduli order. Since A_i is similar to a triangular matrix and the trace is invariant under a similarity transformation:

$$\sum_{k=0}^{N_i-1} \lambda_k^2 = \sum_{k=0}^{N_i-1} (A_i^2)_{kk}$$

Using:

$$\begin{aligned} \lambda_0^2 &= d_i^2 \\ (A_i^2)_{kk} &= d_i \end{aligned}$$

one gets:

$$(N_i - 1)\lambda_1^2 \geq \sum_{k=0}^{N_i-1} \lambda_k^2 - \lambda_0^2 = N_i d_i - d_i^2$$

Therefore:

$$\lambda_1^2 \geq \left(\frac{N_i - d_i}{N_i - 1}\right) d_i$$

And:

$$\frac{|\lambda_1(G_i)|}{\sqrt{d_i}} \geq \sqrt{\left(\frac{N_i - d_i}{N_i - 1}\right)}$$

The statement of the theorem follows, by taking the limit $N_i \mapsto \infty$. ∎

2.2 Some Conventions Concerning Bipartite Graphs.

In this paper we use the following notations concerning a connected bipartite graph G. The two subsets of vertices which define the partition of G are denoted by $V_0(G)$ and $V_1(G)$ and their respective cardinalities are denoted by N and M. It is always assumed that $M \geq N$. Also, it is always assumed that vertices of $V_1(G)$ have the same degree denoted by $\bar{d}$.

The **double cover** of a digraph G, is a bipartite graph D such that $|V_0(D)| = |V_1(D)| = |V(G)|$ and $i \in V_1(D)$ is connected to $j \in V_0(D)$ with one edge for each edge (i, j) in G.

2.3 Probabilistic Algorithms.

This subsection reviews briefly, the basic notions of probabilistic algorithms, needed in the paper.

2.3.1 Probabilistic Turing Machines.

The computational model dealt with in the paper is the PTM (**probabilistic Turing machine**) [GILL77]. A PTM is a nondeterministic Turing machine with the property that there are precisely two choices for each step of the computation. The computation is thought to proceed in the following way: at each stage the machine tosses a coin and chooses one of the two possible computation steps, according to the outcome of the coin toss. It will be assumed without loss of generality, that the same number of coin tosses is required for all inputs of the same length.

When there are no storage limitations on the PTM, it may be assumed without loss of generality that the PTM starts by performing as many coin flips as it would require and recording the outcomes on a special input tape. [4] Then it proceeds deterministically. In such a case a PTM may be viewed alternatively as a deterministic Turing machine T with two inputs: the **actual input** and the **random input.** The actual input is the input whose membership in some language $\mathcal{L}$ is to be decided. The random input plays the role of the sequence of coin tosses. We shall use the notation $A[x, r]$ to denote a PTM with actual input x and random input r.

Given a PTM and an actual input x, the set of random inputs that produce the correct answer for x is called the **witness set** of x. Its complement is called the **nonwitness** set of x. The error probability for the actual input x, is just the probability given to the nonwitness set of x by the probability distribution on the random inputs.

2.3.2 The RP and BPP complexity classes.

Efficient computation on PTMs is captured by the RP and BPP complexity classes, representing fast probabilistic algorithms with a one sided and two sided error respectively:

[4] This would not be possible in a randomized LOGSPACE machine for example.

Definition 2.3 *1. The language $\mathcal{L}$ is in the complexity class RP if for some constant $0 < \delta < 1$ there exists a polynomial time PTM such that:*

(a) if $x \in \mathcal{L}$ then the fraction of the accepting computations exceeds δ.

(b) if $x \notin \mathcal{L}$ then all the computations reject.

2. The language $\mathcal{L}$ is in the complexity class BPP if for some constant $\delta \in (\frac{1}{2}, 1)$ there exists a polynomial time PTM such that:

(a) if $x \in \mathcal{L}$ then the fraction of the accepting computations exceeds δ.

(b) if $x \notin \mathcal{L}$ then the fraction of the rejecting computations exceeds δ.

2.3.3 Random sources.

The coin tosses of the PTM represent the randomness of the computation. It is convenient to think of them as being implemented by a **random source**. The notion of a random source provides a convenient conceptual framework within which, the randomness can be isolated from the computational details of the algorithm.

In information theory a random source is a device that outputs a sequence of symbols from a finite alphabet Σ (we always assume $\Sigma = 0, 1$), according to some probability distribution. In theoretical computer science, one would like a random source to model the more general situation, where one has some uncertainty or only a partial information about the probability distribution of the output. Such information is given as a set of conditions on the probability distribution and these determine a class of allowable distributions. Accordingly we adopt the following definition:

Definition 2.4 *A random source is sequence $\{\Pi_m\}_{m=1}^{\infty}$, where Π_m is a family of probability distributions on $\{0,1\}^m$. A* **strategy** *of the random source is a sequence $\{\pi_m\}_{m=1}^{\infty}$, where $\pi_m \in \Pi_m$ (i.e. a strategy is a sequence of allowable probability distribution - one distribution for each binary string length).*

Note that the definition generalizes the concept of a random source as it is usually formulated in information theory, in three respects:

1. More than one probability distribution is allowed for $\{0,1\}^m$.
2. No consistency conditions are required between probability distributions on $\{0,1\}^n$ and $\{0,1\}^m$ for $m \neq n$.
3. The random source is not necessarily "sequential": the value of the i-th output symbol may depend on all other symbols not just the first i-1.

Given a strategy of the random source, the random variable associated with the i-th output symbol, is called the i-th **random bit**. A source which has only one strategy consisting of independent and unbiased (uniformly distributed) random bits, is called a **perfect random source.** Otherwise it is an **imperfect** random source. Bits output by an (im)perfect random source are referred to as im(perfect) random bits.

3 Sampling Procedures and Disperser Graphs.

3.1 Probabilistic Algorithms and Correlated Sampling.

Let $A[x,r]$ be an RP (BPP) algorithm deciding the language $\mathcal{L}$ with a constant error bound $\alpha < \frac{1}{2}$. Our goal is to decrease the error probability of the algorithm below a prescribed bound, using imperfect random bits or as few as possible perfect random bits. This task can be viewed as a statistical problem: Fix an actual input x, and define a $0,1$ random variable Y on the probability space of random inputs:

$$Y = \begin{cases} 1 & \text{if } A[x,r] \text{ accepts} \\ 0 & \text{otherwise} \end{cases}$$

In other words Y is the characteristic function of the witness or nonwitness set of x, according to whether $x \in \mathcal{L}$ or not. The definition of BPP guarantees the existence of two numbers $0 < \nu < \eta < 1$ such that precisely one of the two following statistical hypothesises is true for Y (depending on whether $x \in \mathcal{L}$ or not):

1. $E[Y] \geq \eta$ iff $x \in \mathcal{L}$.
2. $E[Y] \leq \nu$ iff $x \notin \mathcal{L}$.

(for example take $\eta = 1 - \alpha$ and $\nu = \alpha$). Our problem is to decide with high probability whether $x \in \mathcal{L}$ and this is equivalent to deciding with high probability which of the statistical hypothesises about the random variable Y holds. To see how the second task might be achieved consider first the case in which the random bits are perfect and we do not have to minimize their number. Then one can distinguish between the two statistical hypothesises about Y with probability $2^{-\Omega(k)}$ as follows: sample k random inputs $r_1, \ldots, r_k$ independently, and compute the empirical mean $\frac{1}{k}\sum_{i=1}^{k} Y(r_i)$. If the empirical mean exceeds $\frac{\nu+\eta}{2}$ decide on the first hypothesis, else decide on the second. The probability bound is an immediate consequence of the Chernoff inequality:

Theorem 3.1 *[Che52, HR90] Let $\{X_i\}_{i=1}^{\infty}$ be i.i.d (independent and identically distributed) $0,1$ random variables with a mean μ. Then:*

$$\Pr\left[\frac{1}{k}\sum_{i=1}^{k} X_i - \mu \geq a\right] \leq \left\{\left(\frac{\alpha}{\beta}\right)^{\beta}\left(\frac{1-\alpha}{1-\beta}\right)^{1-\beta}\right\}^{k}$$

where $\alpha = \mu$ and $\beta = \mu + a$

Note that $k|r|$ perfect random bits are required and that k must be bounded by a polynomial in $|r|$ if the new procedure is to run in polynomial time.

Returning to our problem, if we sample k random inputs with less than $k|r|$ perfect random bits or with imperfect random bits, the samples are no longer independent and theorem 3.1 can no longer be invoked to guarantee the $2^{-\Omega(k)}$ error bound. What we need are procedures for picking highly dependent samples from a set of size $2^{|r|}$ for which a Chernoff-like inequality holds. This section is devoted to a general discussion of procedures for dependent samplings and their relation to certain expanding biaprtite graphs called disperser graphs. In section 4 we shall show how expanding graphs can used to generate dependent samples satisfying a Chernoff-like inequality (see theorem 4.5 and the discussion following it).

3.2 Sampling Procedures.

Procedures for sampling elements from finite sets where the samples are not necessarily independent are called **sampling procedures**:

Definition 3.1 *Let $\Psi = \{\psi_k\}_{k=1}^{\infty}$ be a family of finite sets with increasing cardinalities. A* **sampling procedure** *for Ψ is an algorithm which for each k samples a finite multiset of elements of ψ_k in the following way:*

1. *It gets an output of $f(|\psi_k|)$ random bits from a random source (f is an increasing computable function).*
2. *It computes deterministically a finite multiset of elements of ψ_k of size d_k, from the output of the random source.*

The sampling procedure which picks d_k independent samples from ψ_k is termed the **standard sampling procedure.** It requires $d_k \log|\psi_k|$ random bits (assume for simplicity that $|\psi_k|$ is a power of 2), and as has already been noted distinguishes between two statistical hypothesises of the form $E[Y] \leq \nu$ and $E[Y] \geq \eta$ ($\nu < \eta$) with a $2^{-\Omega(d_k)}$ error bound. Our goal is to design sampling procedures that achieve the same error bound with less random bits, and whose performance is as robust as possible with respect to the quality of the random bits.

The following **amplification** property will prove useful in designing such sampling procedures:

Definition 3.2 *Let γ, δ, t be numbers in the interval $(0,1)$. A sampling procedure for Ψ satisfies the (γ, δ, t) amplification property, if for any $W \subset \psi_k$ with $|W| \geq |\psi_k|\gamma$, the fraction of the multisets that intersect W with more than $d_k t$ elements, is at least δ.*

3.3 Disperser Graphs and Sampling Procedures.

Again let $\Psi = \{\psi_k\}_{k=1}^{\infty}$ be a family of finite sets with increasing cardinalities, and let $\{G_k\}$ be a sequence of bipartite graphs. Assume that:

1. $|V_0(G_k)| = |\psi_k|$ and $\phi : V_0(G_k) \mapsto \psi_k$ is a computable bijection.
2. $|V_1(G_k)| = 2^{m_k}$ for some positive integer m_k and $\varphi : \{0,1\}^{m_k} \mapsto V_1(G_k)$ is a computable bijection.

Definition 3.3 *The sampling procedure for Ψ induced by the family of bipartite graphs $\{G_k\}$ is defined by:*

1. *A vertex $x \in V_1(G_k)$ is sampled by obtaining a string of m_k random bits (from the output of a random source) and using the mapping φ.*
2. *A multiset of elements of ψ_k is obtained by computing the neighbours of x and applying to each of them the mapping ϕ*

Although it is not essential to associate bipartite graphs with sampling procedures, this point of view has its advantages. First, as will be seen, the amplification properties of procedures can be linked to certain expansion properties of the graphs. Thus various mathematical tools from combinatorics, group theory and number theory that have been used to construct expanding graphs

can hopefully be applied to the problem of constructing amplifying sampling procedures. Second, expanding graphs that crop up in other problems of computer science such as the simulation of shared memory, the simulation of space with time, and implicit $O(1)$ probe search [UW87, Sip86, FN89], turn out to be closely related to the graphs needed for amplifying sampling procedures.

What are then, the properties of a bipartite graph necessary to guarantee a good amplification in the induced procedure? They are given by the following definitions which generalize the approach of [Pip85], [Sip86] and [San]:

Definition 3.4 *Let G be a bipartite graph, and $0 < \alpha, \beta, < 1$ and $0 < t \leq 1$ be fixed numbers. G is an (α, β, t) (threshold) disperser, if it satisfies the following: if X is a subset of $V_0(G)$, of cardinality $N\alpha$, the cardinality of the set of vertices of $V_1(G)$ connected with at least $\bar{d}t$ of their edges to X, is at most $M\beta$. A family $\{G_n\}$ is a family of (α_n, β_n, t) dispersers if there exists n_0, such that for every $n \geq n_0$ G_n is an (α_n, β_n, t) disperser.*

Remark 3.2 *Although we shall not state it explicitly, it will always be assumed that t is a multiple of $\frac{1}{\bar{d}}$. In particular when an assertion about all $0 < t \leq 1$ is made, it is meant to hold only for such values.*

Motivated by the way dispersers are used in simulations of probabilistic algorithms (see 5.1), we shall refer to $(\alpha, \beta, 1)$ dispersers as (α, β) OR dispersers, while $(\alpha, \beta, \frac{1}{2})$ disperser will be called (α, β) MAJORITY dispersers.

There is a close relationship between the dispersion properties of a bipartite graph and the amplification properties of the induced sampling procedures. The formal statement of this relationship is deferred to section 5 (lemma 5.4), but in general the more dispersing the graph is, the better is the amplification of the induced sampling procedure.

3.4 Efficient Sampling Procedures.

The ultimate goal of this paper, is to employ sampling procedures (constructed from disperser graphs), in simulations of polynomial time probabilistic algorithms, having access only to weak randomness. These simulations are discussed in detail in 5.1 and have the following pattern:

1. A specific sampling procedure is used to pick a multiset of random inputs for the probabilistic polynomial time algorithm A.

2. A is run on the actual input, with each member of the multiset as a random input, and the decision to accept or reject is made on the basis of the results of all the runs.

If we are to employ a sampling procedure in simulations of the above form we must ensure that all the computations involved should be performed in a time polynomial in the input size. Since the simulation samples elements from the set of random inputs whose size is exponential in the input length, one concludes that it can use a sampling procedure if the latter is **efficient** in the following sense:

Definition 3.5 *A sampling procedure for Ψ is* **efficient** *if:*

1. $f(|\psi_k|) = O\left(polylog(|\psi_k|)\right)$.

2. $f \in DTIME\left(polylog(|\psi_k|)\right)$.

3. *The computation of the multiset is in* $DTIME(polylog(|\psi_k|))$. *In particular the size of the multiset is* $O(polylog(|\psi_k|))$.

3.5 Efficient Constructions of Bipartite Graphs.

In section 3.3 it was seen how bipartite graphs give rise to sampling procedures. However in section 3.4 it was maintained, that simulations of probabilistic algorithms required **efficient** sampling procedures. Thus the next natural question is: what properties should a family of bipartite graphs satisfy if it is to induce an **efficient** sampling procedure?
There are two basic requirements:
Density of cardinalities.
Our intention is to identify vertices of digraphs with random inputs of probabilistic algorithms, i.e. with elements of sets whose size is a power of 2. However, the cardinalities of the members of an explicitly constructible digraph family usually constitute a proper subset of the natural numbers, and are not necessarily powers of 2. Such a disparity can be handled as long as the gaps between the cardinalities of the graphs, and the sets of random inputs are not too large. Specifically, it will be required that there exists a natural number k, such that for every natural number N there is an index i satisfying:

$$N \leq |V_0(G_i)| \leq N(\log N)^k.$$

For the need for this specific density requirement, see the discussion concerning definition 5.3.
Polynomiality of relevant graph operations.
Since $V_0(G)$ is interpreted as the set of random inputs of a probabilistic algorithm A, $\log|V_0(G)|$ is polynomial in the input length of A, and it suffices to require that all sampling operations be performed in a time polynomial in $\log|V_0(G)|$. The sampling operations consist of obtaining $\lceil\log|V_1(G)|\rceil$ random bits from a random source to pick an element of $V_1(G)$, and computing the $\bar{d}$ neighbours of the picked vertex. Thus $\log|V_1(G)|$ should be polynomial in $\log|V_0(G)|$, each neighbour should be computed in a time polynomial in $\log|V_0(G)|$, and $\bar{d}$ should be polynomial in $\log|V_0(G)|$.

Construction of bipartite graph families, satisfying the above requirements, will be termed **efficient constructions**. Their properties are summarized in the following definition:

Definition 3.6 *Let* $G = \{G_i\}_{i=1}^{\infty}$ *be a family of bipartite graph. Then* G *is efficiently constructible if:*

1. *There exists a natural number* k, *such that for every natural number* N *there is an efficiently computable index* i *satisfying* $N \leq |V_0(G_i)| \leq N(\log N)^k$.

2. *Given an encoding of a vertex* v *of* $V_1(G_i)$, *its multiset of neighbours can be computed in* $DTIME(polylog(|V_0(G_i)|))$. *In particular* $\bar{d}$ *and* $\log|V_1(G_i)|$ *should be bounded by* $polylog(|V_0(G_i)|)$.

3.6 Efficient Constructions and Finite field Arithmetics.

In various instances the vertices of digraphs employed in algorithms, are elements of a finite field $GF(p^n)$. In such cases finite field arithmetic must be performed efficiently. To this end one can use the following theorem:

Theorem 3.3 *[Sho88] Let F be a finite field of characteristic p. Given an extension K of F of degree d, we can construct an irreducible polynomial over K of degree n, deterministically with*

$$O\left(p^{\frac{1}{2}+\epsilon}(nd)^{3+\epsilon} + (\log p)^2(nd)^{4+\epsilon}\right)$$

operations in F.

Consider the case where the digraph G is defined on $GF(p^n)$. Since in the applications we have in mind, the size of the digraphs is exponential in the input size of the algorithm, n is polynomial in the input size if p does not change with input length. An algorithm using G, typically has to perform arithmetic operations such as addition, subtraction multiplication, and division in a time polynomial in n. Addition and subtraction offer no problem while multiplication and division can be performed if an irreducible polynomial over $GF(p)$ of degree n can be found in polynomial time. By theorem 3.3, this can be done if one adheres to a fixed characteristic p or does not let the characteristic grow too fast (as a function of input size).

4 Efficient Construction of Dispersers

In this section we give constructions of disperser graphs with amplification properties that have not been achieved before. The constructions use a method of [AKS87] based on random walks on expander graphs, and dense Cayley digraphs with an almost optimal separation between their first and second eigenvalue.

Anticipating the use of dispersers in the amplification of success probabilities of probabilistic algorithms (see 5.1), we adopt the following point of view: if G is an (α, β, t) disperser, α and β are thought of as probabilities, and G is said to **amplify** a probability $1-\alpha$ to $1-\beta$. Since $N = |V_0(G)|$ is exponential in the input length in the intended applications, the following terminology is used:

- If $x \in [0,1]$ is a number, and $p(N)$ is a probability such that $|p - x| = O\left(\frac{1}{polylog(N)}\right)$ then $p(N)$ is said to be **polynomially close** to x. Similarly if $|p - x| = O\left(\frac{1}{N^c}\right)$ for some positive constant c, then $p(N)$ is said to be **exponentially close** to x.
- If $x \in [0,1]$ is a number, and $p(N)$ is a probability such that $|p - x| = \Omega\left(\frac{1}{polylog(N)}\right)$ then $p(N)$ is said to be **polynomially apart** from x.
- An amplification of a probability bounded by a constant, to a probability polynomially close to 1, is called a **polynomial** amplification.
- An amplification of a probability bounded by a constant, to a probability exponentially close to 1, is called an **exponential** amplification.

4.1 The Existence of Disperser Graphs.

The first issue that needs to be addressed, is the range of parameter values for which (α, β, t) disperser graphs do exist. For the applications of section 6 it is convenient to set $\alpha = N^{\theta-1}$ and $\beta = M^{\mu-1}$ and enquire about the values of the exponents θ and μ for which disperser graphs exist. The exponents θ, μ will be called the **amplification exponents**, and it will be seen that they are

closely related to entropy rates of imperfect random sources for which a simulation of RP and BPP is possible (theorem 6.1). Clearly the larger the degree of the graph, the better the parameters that can be achieved. The next theorem shows that the condition:

$$\bar{d} \approx \frac{(1-\mu)\log M}{(1-\theta)\log N}$$

suffices to ensure the existence of dispersers with parameters μ, θ. The proof uses a counting sieve argument similar to those given in [Sip86, San]:

Theorem 4.1 *Let* $0 < \theta, \mu < 1$. *An* $\left(N^{\theta-1}, M^{\mu-1}, t\right)$ *disperser exists if:*

$$\bar{d} > \frac{(1-\mu)\log M}{(1-\theta)\log Nt - H(t)} + \frac{N^{\theta}}{M^{\mu}}\left(\frac{1}{t - \frac{H(t)}{(1-\theta)\log N}}\right) + \frac{\log e}{(1-\theta)\log Nt - H(t)}\left(1 + \frac{N^{\theta}}{M^{\mu}}\right)$$

Where $H(t)$ *is the binary entropy function.*

Proof For each vertex of $V_1(G)$ pick its $\bar{d}$ neighbours at random from $V_0(G)$. For any $S \subset V_0(G)$ and $T \subset V_1(G)$ with $|S| = N^{\theta}$ and $|T| = M^{\mu}$, let $A_{S,T}$ denote the event that each vertex of T has at least $\bar{d}t$ neighbours in S. The event that G is not an $\left(N^{\theta-1}, M^{\mu-1}, t\right)$ disperser, is contained in the event $\bigcup_{S,T} A_{S,T}$ whose probability is bounded by:

$$\binom{M}{M^{\mu}}\binom{N}{N^{\theta}}\binom{\bar{d}}{\bar{d}t}^{M^{\mu}}\left(N^{\theta-1}\right)^{\bar{d}tM^{\mu}}$$

Hence the desired graph exists if the above quantity is less than 1. Let $H(x)$ be the binary entropy function. Since $\binom{n}{n\alpha} < 2^{nH(\alpha)}$ it suffices to require that:

$$2^{MH(M^{\mu-1})}2^{NH(N^{\theta-1})}2^{\bar{d}H(t)M^{\mu}}2^{(\theta-1)\bar{d}t\log NM^{\mu}} < 1$$

Therefore the disperser graph exists if:

$$\left[\frac{H(x)}{x}\right]_{x=M^{\mu-1}} + \frac{N^{\theta}}{M^{\mu}}\left[\frac{H(x)}{x}\right]_{x=N^{\theta-1}} < \bar{d}(t(1-\theta)\log N - H(t))$$

and since:

$$\frac{H(x)}{x} \leq \log\frac{1}{x} + \log e$$

it suffices to require that:

$$((1-\mu)\log M + \log e) + \frac{N^{\theta}}{M^{\mu}}((1-\theta)\log N + \log e) < \bar{d}(t(1-\theta)\log N - H(t))$$

And the conclusion of the theorem follows. ∎

Theorem 4.1 establishes the existence of graphs that could play an important role in various theoretical problems if they could be constructed efficiently. One possible use of the graphs of the next corollary is discussed after theorem 6.1. Others appear in [Sip86, FN89].

Corollary 4.2 *Let k be a positive integer and let α, t be fixed real numbers in the interval $(0,1)$. For every N and $M = N^m$ with $m = \lceil (\log N)^k \rceil$, there exist an $(\alpha, M^{\mu-1}, t)$ disperser, with $\mu = (\log M)^{-\frac{k}{k+1}}$, and a degree $\bar{d} = O\left((\log N)^k\right)$.*

Theorem 4.1 gives the sufficient upper bound $\approx \frac{(1-\mu)\log M}{(1-\theta)\log N}$ on $\bar{d}$ for the existence of disperser graphs with parameter values μ, θ. The next theorem shows that these bounds are virtually tight:

Theorem 4.3 *Let $0 < \theta, \mu < 1$. Let $\{N_i\}$ $\{M_i\}$ $\{\bar{d}_i\}$ be increasing sequences of natural numbers such that $\bar{d}_i = O\,(polylog N_i)$ and $M_i < N_i^{\frac{\bar{d}_i}{1-\mu}}$. Then for a sufficiently large i an $\left(N_i^{\theta-1}, M_i^{\mu-1}\right)$ OR disperser with degree $\bar{d}_i$ exists only if:*

$$\bar{d}_i > \left(\frac{1-\mu}{1-\theta+\frac{\log(\bar{d}_i+1)}{\log N_i}}\right)\frac{\log M_i}{\log N_i}$$

Proof Let G be an $\left(N_i^{\theta-1}, M_i^{\mu-1}\right)$ OR disperser with degree $\bar{d}_i$. We may view each vertex of $V_0(G)$ as an element of $[1..N_i]$, and each vertex of $V_1(G)$ as a $\bar{d}_i$ tuple, of elements from $[1..N_i]$ (i.e. the $\bar{d}_i$ tuple of vertices to which it is connected). Thus each vertex of $V_1(G)$ can be represented by a point in a $\bar{d}_i$ dimensional lattice of side length N_i. Define γ by the equality:

$$\frac{N_i^{\bar{d}_i}}{N_i^{\gamma\bar{d}_i}} = M_i^{1-\mu}$$

so that:

$$\bar{d}_i = \left(\frac{1-\mu}{1-\gamma}\right)\frac{\log M_i}{\log N_i} \tag{1}$$

Partition the lattice into cubes of side length N_i^γ. Since there are $M_i^{1-\mu}$ such cubes, one of them must contain a set S of at least M_i^μ elements of $V_1(G)$. S is connected in G to at most $\bar{d}_i\,(N_i^\gamma + 1)$ elements of $V_0(G)$. Since $\bar{d}_i < N_i^\gamma$ for a sufficiently large i, we have:

$$N_i^\theta \le (\bar{d}_i + 1)N_i^\gamma$$

Hence:

$$\gamma \ge \theta - \frac{\log(\bar{d}_i+1)}{\log N_i} \tag{2}$$

Combining 1 and 2 we get the statement of the theorem. ∎

4.2 Random Walks on Digraphs and Exponentially Amplifying Dispersers.

Here we give constructions of dispersers graphs that amplify constant probabilities to probabilities exponentially close to 1. These constructions are based on a method of [AKS87] which uses random walks on expander digraphs. We begin with a definition:

Definition 4.1 *Let G be a digraph. The bipartite graph W_G^l has the vertex sets:*

$$\begin{aligned} V_0(W_G^l) &= V(G) \\ V_1(W_G^l) &= \text{the set of all walks on } G \text{ of length } l-1. \end{aligned}$$

A walk is connected to all its vertices (one edge for each occurrence).

The main tool in the construction to follow, is generalization of a lemma of [AKS87]:

Lemma 4.4 *Let A be the adjacency matrix of a d-regular connected digraph on N vertices G, with index of imprimitivity k. Suppose S is a subset of the vertices with $|S| = N\alpha$ and let P_S be the matrix of the projection onto the coordinates in R^N corresponding to S, that is:*

$$(P_S)_{ij} = \begin{cases} \delta_{ij} & \text{if } v_i \in S \\ 0 & \text{otherwise} \end{cases}$$

then:

$$||P_S A||_2 \leq \sqrt{k\alpha\lambda_0^2 + \bar{\lambda}^2}$$

Proof

Recall the following facts (subsection 2.1): W, the subspace spanned by the eigenvectors corresponding to the eigenvalues of maximal modulus, is k-dimensional. The vectors:

$$v_i = \frac{1}{\sqrt{N}}(\overbrace{\theta_i^0, \cdots, \theta_i^0,}^{\frac{N}{k}}, \ldots, \overbrace{\theta_i^{k-1}, \cdots, \theta_i^{k-1},}^{\frac{N}{k}})$$

are eigenvectors corresponding to the eigenvalues $d\theta_i$, and constitute an orthonormal basis for W. Here θ_i is the i-th root of unity of order k. Both W and $W^\perp$ (the orthogonal complement of W), are invariant under the action of A and A^T.

Now let v be a vector in W of unit length. Then $v = \sum_{j=0}^{k-1} a_j v_j$ with $\sum_{j=0}^{k-1} |a_j|^2 = 1$. The action of the operator $P_S A$ on v annihilates $N - |S|$ coordinates, and produces $|S|$ coordinates with values:

$$\lambda_0 \sum_{j=0}^{k-1} \frac{a_j}{\sqrt{N}} \theta_j^l$$

where $0 \leq l \leq k-1$. Thus by the Cauchy-Schwartz inequality, the absolute value of each nonzero coordinate is bounded by $\lambda_0\sqrt{\frac{k}{N}}$. We conclude that:

$$sup_{v \in W} \frac{||P_S A v||}{||v||} \leq \lambda_0 \sqrt{\frac{|S|k}{N}}$$

where $||\ ||$ denote the L_2 norm. Furthermore, by definition, the norm of the restriction of A to $W^\perp$ is $\bar{\lambda}$, and the norm of P (being a projection operator) is 1, so that one has:

$$sup_{v \in W^\perp} \frac{||P_S A v||}{||v||} \leq \bar{\lambda}$$

One can now study the action of P_S on $\vec{a}$ - an arbitrary vector of R^N, by representing $\vec{a}$ as a sum:

$$\vec{a} = \vec{v} + \vec{w} \qquad \vec{v} \in W^\perp \qquad \vec{w} \in W$$

and looking at the action of P_S on each of the components. By the subadditivity of the norm, and the Cauchy-Schwartz inequality, one gets:

$$\begin{aligned} ||P_S A(\vec{a})|| &= ||(P_S A)\vec{v} + (P_S A)\vec{w}|| \\ &\leq ||(P_S A)\vec{v}|| + ||(P_S A)\vec{w}|| \\ &\leq \bar{\lambda}||\vec{v}|| + \sqrt{k\alpha}\lambda_0||\vec{w}|| \\ &\leq \sqrt{k\alpha\lambda_0^2 + \bar{\lambda}^2}\sqrt{||\vec{v}||^2 + ||\vec{w}||^2} \\ &\leq \sqrt{k\alpha\lambda_0^2 + \bar{\lambda}^2}||\vec{a}|| \end{aligned}$$

and the conclusion of the lemma follows. ∎

With lemma 4.4 one readily obtains the key theorem:

Theorem 4.5 *Let G be a d-regular strongly connected digraph on N vertices, with index of imprimitivity k, and let $0 < t \leq 1$. Then for any $l > 0$, W_G^l is an*

$$\left(\alpha, 2^{lH(t)}d\left(k\alpha + \frac{\bar{\lambda}^2}{d^2}\right)^{\frac{lt}{2}}, t\right)$$

disperser, where $H(x)$ is the binary entropy function.

Proof Note that:

$$|V_0\left(W_G^l\right)| = N \qquad |V_1\left(W_G^l\right)| = Nd^{l-1} \qquad \bar{d} = l$$

Let $S \subset V_0\left(W_G^l\right)$ be a set of vertices such that $|S| = N\alpha$ and let $S_1 \subset V_1\left(W_G^l\right)$ be the set of vertices with at least lt neighbours in S. We have to show that

$$|S_1| \leq Nd^{l-1}2^{lH(t)}d\left(k\alpha + \frac{\bar{\lambda}^2}{d^2}\right)^{\frac{lt}{2}}$$

Let A be the adjacency matrix of G, and $W \subseteq [1..l]$. Define the sequence of matrices $\{A_i\}_1^l$ by:

$$A_i^W = \begin{cases} P_S & i \in W \quad i = 1 \\ P_S A^T & i \in W \quad i \neq 1 \\ A^T & i \notin W \end{cases}$$

With this notation, the number of walks of length $l-1$ on G, that have vertices of S in the positions indexed by W is $||(\prod_i A_i^W)\vec{1}||_1$, where $\vec{1}$ denotes the vector all of whose coordinates are 1. Therefore:

$$\begin{aligned} |S_1| &\leq \sum_{|W|=lt} ||(\textstyle\prod_i A_i^W)\vec{1}||_1 \leq \sum_{|W|=lt} \sqrt{N}\left\|\textstyle\prod_i A_i^W\right\|_2 ||\vec{1}||_2 \\ &\leq \sum_{|W|=lt} \sqrt{N}\left(\textstyle\prod_i \left\|A_i^W\right\|_2\right)||\vec{1}||_2 \leq \binom{l}{lt}\sqrt{N}\left\|(P_S A^T)\right\|_2^{lt}\left\|A^T\right\|_2^{l(1-t)}\left\|\vec{1}\right\|_2 \end{aligned}$$

By lemma 4.4 $\|(P_S A^T)\| \leq \sqrt{k\alpha\lambda_0^2 + \bar{\lambda}^2}$. Since $\binom{l}{lt} \leq 2^{lH(t)}$, $\left\|A^T\right\|_2 = d$, and $\left\|\vec{1}\right\|_2 = \sqrt{N}$, the desired result follows. ∎

Theorem 4.5 has the following statistical interpretation: Let X be a domain with $|X| = |G|$. Let Y be the characteristic function of some subset $S \subset X$. Suppose that if the elements of X are picked at random, one of the following statistical hypothesises holds:

1. $E[Y] \leq \alpha$
2. $E[Y] \geq \beta$

Where $0 < \alpha < \beta < 1$. Suppose further that one can find two numbers $0 < \nu < \eta < 1$ that satisfy:

$$H(\nu) + \frac{\nu}{2}\log\left(k\alpha + \frac{\bar{\lambda}^2}{d^2}\right) < 0 \qquad H(1-\eta) + \frac{1-\eta}{2}\log\left(k(1-\beta) + \frac{\bar{\lambda}^2}{d^2}\right) < 0$$

Then theorem 4.5 says that we can use l highly correlated samples from X, to distinguish between the two statistical hypothesises about the random variable Y with probability $2^{-\Omega(l)}$. This is done as follows: Identify X with $V_0(W_G^l)$. Pick with uniform probability a vertex of $V_1(W_G^l)$. Compute its l neighbours and identify them with l elements $x_1, \ldots, x_l$ of X. Finally decide on the first hypothesis if $\frac{1}{l}\sum_{i=1}^{l} Y(x_i) \geq \frac{\nu+\eta}{2}$ and on the second hypothesis otherwise.

If l is taken to be $c \log N$ in theorem 4.5, one gets exponential amplification. We will formulate two versions each tailored for a different application. The first is used in section 5 (see theorem 5.9):

Corollary 4.6 *Let G be a primitive, d-regular, δ-separable digraph on N vertices. Assume:*

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

Then for any $c > 0$, $W_G^{c\log N}$ is an

$$(\alpha, N^e, t)$$

disperser, with $M = N^{1+c\log d}$ and:

$$e = c\left(H(t) + \frac{t}{2}\right) + (\delta - 1)ct\log d + \frac{\log d}{\log N}$$

In the second corollary we are interested in the amplification exponents (defined in 4.1). It is used in section 6 (see theorem 6.1 and corollary 6.2):

Corollary 4.7 *Let G be a primitive, d-regular, δ-separable digraph on N vertices. Assume:*

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

Then for any $c > 0$, $W_G^{c\log N}$ is an

$$\left(\alpha, M^{\mu-1}, t\right)$$

disperser with $M = N^{1+c\log d}$ and:

$$\mu = \frac{c\log d}{1 + c\log d}(t\delta + (1-t)) + \frac{c(H(t) + \frac{t}{2})}{1 + c\log d} + \frac{1}{c\log N} > t\delta + (1-t)$$

4.3 Efficient Constructions of Exponentially Amplifying Disperser Graphs.

Let $\{G_i\}_{i=1}^{\infty}$ be a δ_i-separable family of d_i-regular digraphs and Let l_i be a sequence of natural numbers. Theorem 4.5 provides **explicit** definitions of families of disperser graphs with a certain amplification, namely $H_i = W_{G_i}^{l_i}$. But we are interested in **efficient** constructibility. It is easy to see that the following two conditions suffice for the efficiency. First, l_i should be bounded by a polynomial in $\log|G_i|$. Second the family G_i should be **strongly constructible** in the following sense:

Definition 4.2 *A family $G = \{G_i\}$ of d_i-regular digraphs is strongly constructible if*

1. *G is "dense enough" - There exists a natural number k, such that for every natural number N there is an index i computable in $DTIME(polylog(N))$ such that $N \leq |G_i| \leq N(\log N)^k$.*
2. *There is a numbering of the neighbours of each vertex such that given as input a vertex v of G_i, and a number $1 \leq k \leq d_i$, the k-th neighbour of v can be computed in $DTIME(polylog(|G_i|))$.*

An example of an efficiently constructible family of separable graphs is the Gabber and Galil graphs [GG81] [5] defined as follows: For each $n = m^2$, G_n is a bipartite graph on $2n$ vertices. with the vertex sets $V_0 = Z_m \times Z_m$ and $V_1 = Z_m \times Z_m$. The edges between V_0 and V_1 are given by the permutations:

$$\sigma_0(x,y) = (x,y)$$
$$\sigma_1(x,y) = (x, x+y) mod\ m$$
$$\sigma_2(x,y) = (x, x+y+1) mod\ m$$
$$\sigma_3(x,y) = (x+y, y) mod\ m$$
$$\sigma_4(x,y) = (x+y+1, y) mod\ m$$

Thus theorem 4.5 implies the following:

Theorem 4.8 *Let $0 < t \leq 1$. For any $0 < \alpha < 2^{-\frac{2H(t)}{t}}$, and for any sufficiently large positive integer l, there exists an efficiently constructible family of:*

$$\left(\alpha, 2^{-\Omega(l)}, t\right)$$

dispersers, with $M = N^{1+O(\frac{l}{\log N})}$.

Proof By adding to the Gabber Galil graphs self loops we can make them primitive. [6] By taking a sufficiently large (but constant) power of the resulting graphs we can generate a primitive graph family G satisfying the condition:

$$\frac{\bar{\lambda}^2}{d^2} + \alpha < 2^{-\frac{2H(t)}{t}}$$

By theorem 4.5 the efficiently constructible family W_G^l satisfies the theorem. ∎

Although we can use theorem 4.8 to make the error probability of RP and BPP exponentially small at a linear cost of random bits, it does not give optimal results. To get such results we will need to optimize two parameters:

[5] Actually these graphs are shown to be expanders in [GG81], but for constant degree graph families, expansion and separability are equivalent [Tan84, AM85, Alo86a].

[6] Any irreducible nonnegative matrix with a positive trace is primitive.

1. Separation exponent. The smaller the separation exponent δ, the greater the amplification.

2. Degree of the digraph. The smallest probability that can be amplified is $1 - d^{2(\delta-1)}$, hence the smaller the degree of the digraph, the wider the range of probabilities that can be amplified.

¿From the above and theorem 2.4, it follows that $\frac{1}{2}$-separable digraphs with a constant degree are optimal. [LPS86] have given explicit definitions of $\frac{1}{2}$-separable constant degree families, using nontrivial group-theoretic notions, and with cardinalities defined in terms of primes but it is not clear that these construction are efficient. All the other constructions of constant degree separable families that have appeared in the literature (e.x. [Mar75, GG81, Buc86, AM85, JM85]) are far from optimal, yielding separation exponents quite larger than $\frac{1}{2}$.

However the difficulty of efficiently constructing $\frac{1}{2}$- separable digraphs can be obviated if the degree of the digraphs is not required to be bounded. We shall now describe an efficient construction of digraphs with an almost optimal separation exponent, and a degree polylogarithmic in the size of the digraph.

The basis for our construction is an explicit definition of highly separable digraphs given by Chung [Chu]. The construction of Chung is obtained as follows: Let F be a finite field say of characteristic 2, and f an irreducible polynomial of degree t. Let E be the extension field of F formed by adjoining a root w of f to F, and g a generator for E^* (the multiplicative group of invertible elements of E). We shall use the notation $Ind(x)$ to denote the discrete log of x with respect to g. Let S be the set of natural numbers in $[0..|E^*| - 1]$ given by:

$$S = \{Ind(w + i)\}_{i \in F}$$

We may now define G the Cayley digraph on $Z_{|E^*|}$ with respect to S. In [Chu] it is shown that the digraph G is primitive and satisfies:

$$\bar{\lambda} \leq (t - 1)\sqrt{\lambda_0}$$

We shall think of G as a digraph on E^* defined by the rule: there is a directed edge from x to y, if :

$$(Ind(x) - Ind(y)) \in S$$

that is, there is a directed edge from x to y if $(xy^{-1} - w) \in F$. Note that this definition is generator independent.

We now have:

Theorem 4.9 *For every $0 < \epsilon < \frac{1}{2}$,there exists a strongly constructible family G_i of d_i-regular, primitive, $\frac{1}{2} + \epsilon$ separable digraphs, with:*

$$d_i \leq (\log |G_i|)^{\frac{1}{\epsilon}}$$

Proof
We construct G_i as follows: Choose numbers k, m such that:

$$\begin{array}{rcl} i & < km & \leq i + O(\log i) \\ \epsilon k & < \log m & \leq (k + 3)\epsilon \end{array}$$

Let $F = GF(2^k)$, E an extension field of F of dimension $t = m$, and G_i the resulting Chung digraph. Note that since:

$$\lambda_0 = |F| = 2^k \qquad t = m \leq |F|^{\frac{k+3}{k}\epsilon}$$

one has:

$$\bar{\lambda}(G_i) < t\sqrt{\lambda_0} \leq \lambda_0^{\frac{1}{2}+\frac{k+3}{k}\epsilon}$$

Also the cardinality of G_i is $|E| - 1$, and the degree of G_i is $|F|$ so that:

$$deg(G_i) = |F| = 2^k < m^{\frac{1}{\epsilon}} < (km-1)^{\frac{1}{\epsilon}} = (\log(|G_i|)^{\frac{1}{\epsilon}}$$

Note that this construction is efficient in the sense of definition 4.2. The density condition is satisfied since the numbers k, m chosen above ensure that for every $N_i = 2^i$:

$$N_i < |G_i| \leq N_i(\log N_i)^k$$

for any $k > \frac{1}{\epsilon}$. Also all graph operations are performable in polynomial time, since one is doing finite field arithmetic with a constant characteristic (see section 3.6). ∎
We will use these strongly constructible digraphs to obtain optimal results in theorem 5.9 and corollary 6.2.

5 Optimal Deterministic Amplification of RP and BPP

In this section we use the disperser graphs constructed previously to address the the **deterministic amplification problem**: Given an RP(BPP) algorithm A, which requires n random bits and operates with a constant error bound, reduce the error probability (amplify the success probability) of the algorithm, adding as few random bits as possible.

There is a standard amplification method alluded to in section 3. Let A be an RP (BPP) algorithm which decides a language $\mathcal{L}$ with n random bits, and an error bound q ($\frac{1}{2} - q$) where q is a constant. Suppose A is run k independent times, and the actual input is accepted iff there is at least one accepting computation (RP case), or iff at least a majority of the computations accept (BPP case). The new algorithm decides $\mathcal{L}$, with an error bound q^k ($(\sqrt{1-4q^2})^k$). If k is bounded by a polynomial in n, the extra cost in deterministic computation is in $DTIME(poly(n))$. The number of random bits used is kn. For example, the error probability can be made **exponentially** small (i.e. $O(2^{-n})$), with $\Theta(n)$ runs, so that $\Theta(n^2)$ random bits are required. Similarly, $\Theta(n \log n)$ random bits are required to make the error probability **polynomially** small (i.e. $O(n^{-c})$) for some positive constant c). In the deterministic amplification problem, one strives to achieve the same amplification with significantly less random bits.

The results that have been achieved so far for the deterministic amplification problem are due to Karp Pippenger and Sipser and Chor and Goldreich:

Theorem 5.1 *[KPS85] Let A be an RP algorithm which decides $\mathcal{L}$ with n random bits and an error bound polynomially apart from 1. Let P be a polynomial. Then there exists an RP algorithm which decides L with n random bits and an error bound $\frac{1}{P(n)}$.*

Theorem 5.2 *[CG86] Let A be a BPP algorithm which decides $\mathcal{L}$ with n random bits and an error bound polynomially apart from $\frac{1}{2}$. Let P be a polynomial. Then there exists a BPP algorithm which decides L with $2n$ random bits and an error bound $\frac{1}{P(2n)}$.*

Using disperser graphs, we give the following results: Let A be an RP (BPP) algorithm using n random bits, and operating with error bound $\alpha(n)$. Let $0 < \epsilon < 1$ be a number, and P be a polynomial.

1. BPP: if $\alpha(n) < \frac{1}{8}$, then α can be made less than $\frac{1}{P(n)}$, with no increase in the amount of random bits.
2. RP: if $\alpha(n) < 1 - \frac{1}{P(n)}$, α can be made less than 2^{-n}, with $(3+\epsilon)n$ random bits.
3. BPP: if $\alpha(n) < \frac{1}{2} - \frac{1}{P(n)}$, α can be made less than 2^{-n}, with $(6+\epsilon)n$ random bits.
4. BPP: if $\alpha(n) < \frac{1}{8}$, α can be made less than 2^{-n}, with $(3+\epsilon)n$ random bits.

Impagliazzo and Zuckerman [IZ], have obtained independently the result that the error probability of an RP (BPP) algorithm can be made exponentially small at a linear cost of random bits.

5.1 Disperser Graphs and Deterministic Amplification.

Let A be an RP (BPP) algorithm which requires n random bits and decides a language $\mathcal{L}$ with an error probability $\alpha(n)$, where $\alpha(n)$ is assumed to be **polynomially apart** from 1 ($\frac{1}{2}$). Our goal is to achieve a large amplification at a small additional cost of random bits.

Our approach is to simulate A, using a sampling procedure which requires a "small" number of random bits but has strong amplification properties:

Definition 5.1 *Let Γ be an efficient sampling procedure for $\{0,1\}^n$, and let A be as above. The simulation of A induced by Γ is the probabilistic algorithm B operating in two phases:*

1. **Sampling phase.** *B uses Γ to pick multiset of binary strings of length n.*
2. **Deterministic phase.** *B runs A on the actual input, with each binary string obtained in the sampling phase as a random input, and accepts if at least one computation accepts (RP case), or if a majority of the computations accept (BPP case).*

Note that since A is a polynomial time algorithm and the sampling procedure Γ is efficient (see definition 3.5), B is a polynomial time probabilistic algorithm.

In order to provide results in the deterministic amplification problem, a simulation algorithm induced by a sampling procedure, should satisfy two properties: First, it should decide $\mathcal{L}$ with an error probability bounded by $\beta < \alpha$ (amplification). Second, the amount of random bits used by B, should be significantly smaller than that required by the standard amplification method.

Amplification can be achieved, if the sampling procedure satisfies the appropriate amplification properties:

Lemma 5.3 *Let A be an RP (BPP) algorithm which decides $\mathcal{L}$, and operates with n random bits, and an error bound $\alpha(n)$. Suppose B is a simulation of A, induced by a sampling procedure that satisfies the $(1-\alpha(n), 1-\beta(n), t)$ property, $t = 0$ ($t = \frac{1}{2}$). Then B is an RP (BPP) algorithm which decides $\mathcal{L}$ with an error bound $\beta(n)$.*

Proof We deal with the BPP case, the RP case being similar. Since B accepts iff a majority of the computations accept, B outputs the correct answer if a majority of the sampled random inputs of A are witnesses. The error probability of A is bounded by α so the size of the witness set is at least $2^n(1-\alpha)$. Since the sampling procedure satisfies the $(1-\alpha(n), 1-\beta(n), \frac{1}{2})$ property, definition 3.2 implies that the probability that a majority of the sampled random inputs are witnesses, is at least $1-\beta$ ∎

Thus one has to construct efficient, highly amplifying sampling procedures. Now let $G = \{G_n\}_{n=1}^{\infty}$ be an efficiently constructible family of bipartite graphs satisfying:

1. $|V_0(G_n)| = 2^n$ and there exists a bijection $\phi : V_0(G_n) \mapsto \{0,1\}^n$, such that $\phi \in DTIME\,(poly(n))$.
2. $|V_1(G_n)| = 2^{m_n}$ for some positive integer $m_n \geq n$ and there exists a bijection $\varphi : \{0,1\}^{m_n} \mapsto V_1(G_n)$, such that $\varphi \in DTIME\,(poly(n))$.

Since G_n induces a sampling procedure for $\{0,1\}^n$ (see definition 3.3), one can define:

Definition 5.2 *The simulation of A induced by G_n is the simulation of A induced by the sampling procedure corresponding to G_n.*

By definitions 3.3 and 5.1, the simulation of A induced by G_n has the following form:

1. It gets a string of $\log |V_1(G_n)|$ random bits.
2. It uses φ to map the random input to a vertex $v \in V_1(G_n)$.
3. It computes the multiset of neighbours of v (which are vertices of $V_0(G_n)$).
4. It uses ϕ to map the multiset of vertices of $V_0(G_n)$ to a multiset of binary strings of length n.
5. It runs A with each of the binary strings obtained above, as a random input.
6. It accepts iff there is at least one accepting computation (RP case), or iff a majority of the computations accept (BPP case).

Now the key observation is that disperser graphs yield amplifying sampling procedures:

Lemma 5.4 *The sampling procedure induced by a family of (α, β, t) disperser graphs, satisfies the $(1-\alpha, 1-\beta, 1-t)$ amplification property.*

Proof This an immediate consequence of definitions 3.2 and 3.4 ∎

Lemmas 5.3 and 5.4 imply the basic:

Theorem 5.5 *Let A be an RP (BPP) algorithm which decides $\mathcal{L}$, and operates with n random bits, and an error bound $\alpha(n)$. Suppose B is a simulation of A, induced by a family of (α, β, t) disperser graphs with $t = 1$ ($(t = \frac{1}{2})$. Then B is an RP (BPP) algorithm which decides $\mathcal{L}$ with an error probability bounded by $\beta(n)$ and requires $\log |V_1(G_n)|$ random bits.*

Thus to decrease the error probability of A from $\alpha(n)$ to $\beta(n)$, one should use simulation algorithms induced by an efficiently constructible family of $(\alpha(n), \beta(n))$ dispersers. What about the additional cost of random bits? The simulation algorithm, requires $\log |V_1(G_n)|$ random bits, while the standard amplification method requires $n\frac{\log\beta}{\log\alpha}$ random bits for the same amplification. Therefore, a simulation of A, will improve upon the standard amplification method if it is induced by $(\alpha(n), \beta(n))$ dispersers, that satisfy:

$$\log |V_1(G_n)| \ll n\frac{\log\beta}{\log\alpha}$$

This is the basis for our approach to the deterministic amplification problem.

Theorems 5.5 and 4.8 imply

Theorem 5.6 *Let A be an RP (BPP) algorithm probabilistic algorithm which uses n random and operate with a constant error bound $\alpha < 1$ ($\alpha < \frac{1}{16}$. Then one can decrease the error probability to 2^{-k}, with $n + O(k)$ random bits. In particular the error probability can be made exponentially small at a linear cost of random bits.*

In 5.3 we will optimize the exponential amplification.

Before we move on, there is a technicality to be dispensed with. In definition 5.2 it is assumed that $|V_0(G_n)|$ and $|V_0(G_n)|$ are powers of 2. Although the disperser we use do not meet this condition they still induce a sampling procedure for $\{0,1\}^n$ as follows:

Definition 5.3 *Let $G = \{G_m\}$ be an efficiently constructible family of bipartite graphs. The sampling procedure induced by G on $\{0,1\}^n$, samples a multiset of n-bit strings as follows:*

1. *It computes an index i that satisfies $2^n \leq |V_0(G_i)| \leq 2^n n^k$ for some fixed positive integer k. The fact that such an index can be found, is ensured by the definition of efficient constructibility. We will suppose that the vertices of $V_0(G_i)$ are indexed by the numbers $0, \ldots, |V_0(G_i)|-1$, and the vertices of $V_1(G_i)$ are indexed by the numbers $0, \ldots, |V_1(G_i)| - 1$.*

2. *The algorithm obtains a string of $\lceil \log |V_1(G_i)| \rceil$ random bits. Let x be the number encoded by the random string. The algorithm computes the vertex $v \in V_1(G_i)$ whose index is x mod $|V_1(G_i)|$.*

3. *The algorithm computes the multiset of neighbours of v, and converts each of them to an n-bit string, by taking the binary representation of its index mod 2^n.*

What was actually done here, is to construct from a family G_m of (α, β) dispersers, a new family of bipartite graph H_n such that $|V_0(H_n)| = 2^n$, and $|V_1(H_n)| = 2^{m(n)}$, so that H_n induces a sampling procedure on binary strings. Now if α is polynomially small, and β is exponentially small, then amplification properties of H_n remain essentially the same. α can change by a polynomial factor, but this can be taken into account in the initial polynomial amplification. β can increase by a factor of at most 2.

5.2 Polynomial Amplification of BPP.

In this section we give a result which is in one sense an improvement of theorem 5.2. It is shown that the error probability of a BPP algorithm can be made polynomially small, with no further cost in random bits. However the initial error should be bounded by a constant ($< \frac{1}{8}$).

The following lemma is needed:

Lemma 5.7 *Let G be a connected d regular primitive digraph on $N = 2^n$ vertices with adjacency matrix A such that:*

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

Then the double cover [7] *of G is an $\left(\alpha, 8\alpha^2, \frac{1}{2}\right)$ disperser.*

Proof This is a direct consequence of theorem 2.3 ∎

Theorem 5.8 *Let A be a BPP decision algorithm for a language $\mathcal{L}$, which uses n random bits, with error probability $\alpha < \frac{1}{8}$. Then for any polynomial P, there exists a BPP algorithm that decides $\mathcal{L}$, with n random bits, and error probability bounded by $\frac{1}{P(n)}$.*

Proof Our intention is to perform a sequence of amplifications which achieve the following sequence of bounds on the error probability:

$$\begin{aligned} \alpha_0 &= \tfrac{1}{8} \\ \alpha_{l+1} &= 8\,(\alpha_l)^2 \end{aligned}$$

Let $\{G_n\}$ a family of primitive, constant degree, η-separable digraphs with $\eta < 1$, and $|G_n| = 2^n$. Such digraphs can be obtained for example, from the Gabber-Galil expander graphs (see 4.3). By raising G_n to a sufficiently large (but fixed) power, one gets a digraph $\mathcal{G}$ with an adjacency matrix A such that $\frac{\bar{\lambda}^2}{\lambda_0^2} < \frac{1}{8}$.

Now define the following sequence of matrices $\{\Lambda_k\}_{l=0}^{i}$:

$$\begin{aligned} \Lambda_0 &= A \\ \Lambda_{l+1} &= \Lambda_l^2 \end{aligned}$$

Let H_l be the double cover of the digraph with adjacency matrix Λ_l. Since:

$$\frac{\bar{\lambda}(\Lambda_{l+1})}{\lambda_0(\Lambda_{l+1})} \leq \left(\frac{\bar{\lambda}(\Lambda_l)}{\lambda_0(\Lambda_l)}\right)^2$$

We have for each l:

$$\left(\frac{\bar{\lambda}(\Lambda_l)}{\lambda_0(\Lambda_l)}\right)^2 < \alpha_l$$

Therefore lemma 5.7 implies that H_l is an $\left(\alpha_l, \alpha_{l+1}, \frac{1}{2}\right)$ disperser.

Now we define the sequence of probabilistic algorithms $\left\{A^l\right\}_{l=0}^{i}$ as follows:

1. A_0 is the original algorithm.
2. A_l is the simulation of A_{l-1} induced by H_{l-1}.

[7] See 2.2 for a definition of double cover.

By theorem 5.5 A_l decides $\mathcal{L}$ with error probability bounded by α_l. Now since $\alpha_i = \frac{1}{8}(8\alpha)^{2^i}$ $l = \log\log n + c$ iterations make α_l polynomially small:

$$\alpha_l = \frac{1}{8\left(n^{\log(\frac{1}{8\alpha})}\right)^{2^c}}$$

It remains to verify that A_l is polynomial time. Observe that there are d^{2^k} recursive calls to the algorithm A_{k-1} in the algorithm A_k where d is the degree of $\mathcal{G}$. Hence there are at most $d^{2^{l+1}}$ recursive calls in A_l. For $l = \log\log n + c$ this number is bounded by a polynomial in n. Since the graphs $\{H_i\}_{i=1}^{l}$ are efficiently constructible, A_l is a polynomial time algorithm. ∎

5.3 Exponential Amplification.

The main result of this chapter is:

Theorem 5.9 *Let A be an RP (BPP) algorithm, which decides the language $\mathcal{L}$ with n random bits, and an error bound $\alpha(n) < 1 - \frac{1}{P(n)}$ $(\alpha(n) < \frac{1}{2} - \frac{1}{P(n)})$ where P is a polynomial. Let $\epsilon > 0$ be a constant.*

1. *RP: There exists an RP algorithm B, which decides $\mathcal{L}$, with an error probability bounded by 2^{-n} and uses $(3+\epsilon)n$ random bits.*
2. *BPP: There exists a BPP algorithm B, which decides $\mathcal{L}$, with an error probability bounded by 2^{-n}, and uses $(6+\epsilon)n$ random bits.*
3. *BPP: If $\alpha(n)$ is bounded by a constant smaller than $\frac{1}{8}$, there exists a BPP algorithm B, which decides $\mathcal{L}$ with an error probability bounded by 2^{-n}, and uses $(3+\epsilon)n$ random bits.*

Proof The algorithms are constructed in two stages. First we achieve a polynomial amplification using theorems 5.1, 5.2 and 5.8 respectively. In the first and third case there is no increase in the amount of random bits used. In the second case the amount of random bits is doubled. Next we use the dispersers constructed in section 4 to attain an exponential amplification. We shall give a detailed proof only of the second case the others being completely analogous.

Let $\eta < \epsilon$ be a positive constant whose value will be specified later, and let κ be any positive constant smaller than η. By theorem 5.2 there exists a BPP algorithm C which decides $\mathcal{L}$ with $2n$ random bits, and an error bound $(2n)^{-\frac{1}{\kappa}}$.

By theorem 4.9 there exists a strongly constructible family G_n of d_n-regular, primitive, $\frac{1}{2}+\eta$ separable digraphs, with:

$$d_n \leq (\log|G_n|)^{\frac{1}{\eta}}$$

Let $W_n = W_{G_n}^{c_n \log|G_n|}$ where c_n satisfies:

$$c_n \log d_n = 2 + \epsilon$$

By definition 5.3, and the strong constructibility of G_n, we may assume:

$$2^{2n} = |V_0(W_n)| \leq |G_n| \leq 2^{2n}(2n)^k$$

for some fixed natural k. By theorem 2.4 one has for a sufficiently large n:

$$\begin{array}{rcccl} \left(\frac{\bar{\lambda}(G_n)}{\lambda_0(G_n)}\right)^2 & \geq & (1-o(1))\left(\frac{1}{d_n}\right) & \geq & (1-o(1))\left((\log|G_n|)^{-\frac{1}{\eta}}\right) \\ & \geq & (1-o(1))(2n+k\log 2n)^{-\frac{1}{\eta}} & \geq & (2n)^{-\frac{1}{\kappa}} \end{array}$$

Hence by corollary 4.6, W_n is for a sufficiently large n a:

$$\left((2n)^{-\frac{1}{\kappa}}, 2^{2en}, \frac{1}{2}\right)$$

disperser, with $|V_1(W_n)| = 2^{(1+c_n \log d_n)2n}$ and:

$$e = c_n\left(H(\frac{1}{2})+\frac{1}{4}\right)+\left(\eta-\frac{1}{2}\right)\frac{1}{2}c_n\log d_n+\frac{\log d_n}{2n}$$

Now η is chosen so that:

$$\left(\eta-\frac{1}{2}\right)c_n\log d_n = \left(\eta-\frac{1}{2}\right)(2+\epsilon) < -1$$

Since $c_n \mapsto 0$, $2e < -1$ for a sufficiently large n, and W_n is (for a sufficiently large n) a:

$$\left((2n)^{-\frac{1}{\kappa}}, 2^{-n}, \frac{1}{2}\right)$$

disperser with $|V_1(W_n)| = 2^{(1+c_n \log d_n)2n}$.

Let B be the simulation induced on C by W_n. By theorem 5.5, B decides $\mathcal{L}$ with error bound 2^{-n}, and requires $\log|V_1(W_n)|$ random bits, i.e $(1+c_n\log d_n)2n$ or $(6+2\epsilon)n$ random bits. ∎

6 Random Sources

6.1 Simulating RP and BPP with Imperfect Random Sources.

Let A be an RP (BPP) algorithm, which decides a language $\mathcal{L}$. A is guaranteed to operate with a specified bound on its error probability only if its random bits are independent and unbiased. But what happens if one has at its disposal only low quality random bits (output by some imperfect random source)? Can one still perform the task with a small error probability?

Definition 6.1 *Let S be a random source. We say that RP (BPP) can be simulated by S, if for every $\mathcal{L} \in$ RP (BPP) there exists a probabilistic polynomial algorithm which decides $\mathcal{L}$ with an error probability that decreases to 0 with input length, when its random input comes from S.*

Note that it does not suffice to require only a constant error probability in the above definition, as we did in the definition of RP (BPP). This is due to the fact that one cannot decrease the error bound to $2^{-\Omega(k)}$ by sampling k random inputs from the source and running A on each of them, since the random inputs may be correlated.

6.2 Some Models of Imperfect Random Sources.

Various models of imperfect random sources have been investigated. The simplest are the Von Neumann [VN51] and Markov sources [Blu84]. A Von Neumann source outputs bits obtained by independent tosses of a biased coin and a Markov source outputs bits obtained from transitions of a Markov chain. As is shown in [VN51] and [Blu84], in both cases it is possible to extract a linear amount of perfect random bits from the outputs of the sources. This solves immediately the problem of simulating RP and BPP with such sources, since one can extract perfect random bits from the output of the random source and use them as a random input for the original algorithm.

However bit extraction is not always possible. Santha and Vazirani [SV84] introduced the the δ-semi random source (abbreviated δ-SR source) for any $\delta \in [0, \frac{1}{2}]$). The probability distributions that the source induces on $\{0,1\}^m$ consists of those that satisfy for each $1 \leq i \leq m$:

$$\delta \leq Pr\{X_i = 1 | X_1 = a_1, \ldots, X_{i-1} = a_{i-1}\} \leq 1 - \delta$$

Here X_i denotes the i-th random bit and $a_k \in \{0,1\}$. This source model was generalized by [CG85] who introduced the (l,b)-probability bounded source (abbreviated (l,b)-PRB source). Here l is a positive integer and $b \leq l$ is a positive real number. The probability distributions that the source induces on $\{0,1\}^m$ consists of those that have the following property: If the output of the source is divided into blocks of l bits each, then the probability of a block being assigned any value, given any assignment to the previous blocks, is bounded from above by 2^{-b}. In [SV84] and [CG85] it was shown that not a single perfect random bit can be extracted from the output of SR and PRB sources. However [VV85, Vaz86] and [CG85] showed that BPP could be simulated with sources of these type provided they have a constant lower bound on their entropy rate (i.e. provided δ and $\frac{b}{l}$ are bounded below by a positive constant).

A class of random sources for which no simulation results were known are the bit fixing sources. Here we think of the source as an adversary who may fix the values of up to $(1-\omega)m$ bits. The other bits are obtained by tossing independent unbiased coins. We distinguish four types of bit fixing sources:

1. The oblivious bit fixing source. The adversary picks a set of up to $(1-\omega)m$ bit locations and fixes the bits **before** the coins for the other positions are tossed. This model was investigated in [CGH$^+$85].

2. The nonoblivious bit fixing source. The adversary picks a set of up to $(1-\omega)m$ bit locations and sets the bits **after** the coin tosses for the other positions. This model was investigated in [BOL85, KKL88].

3. The adaptive bit fixing source. Here the bits $b_1, \ldots, b_m$ are ordered by their indices. Depending on the values of the bits $b_1, \ldots, b_{i-1}$ the adversary may toss a fair coin for bit b_i or fix its value provided less than $(1-\omega)m$ bits have hitherto been fixed. These sources were studied in [LLS87].

4. The strong bit fixing source. The adversary tosses a coin for each bit location, and then picks a set of up to $(1-\omega)m$ bit locations and sets them.

In [LLS87, KKL88] it was shown that not a single perfect bit can be extracted from the output of adaptive and nonoblivious sources. [CGH$^+$85] showed that if $\omega < \frac{1}{3}$ only one perfect bit can be

extracted from the output of an oblivious bit fixing source. In all these cases bit extraction cannot be used to simulate RP and BPP.

6.3 The Renyi Random Source.

In this section we introduce a new model of a random source, which includes all the models discussed in 6.2. Let P be a probability distribution on $\{0,1\}^m$. Recall that the entropy of P $I(P)$, and the entropy rate of P $R(P)$, are defined by:

$$I(P) = \sum_{i=0}^{2^m-1} p_i \log \frac{1}{p_i} \qquad R(P) = \frac{I(P)}{m}$$

All the random sources described in section 6.2 have parameters which place lower bound on their entropies (and rates). In the case of the Von Neumann source the parameter is the bias of the coin, and in the case of the Markov source the parameter is the minimal transition probability. δ-SR sources, have at least $\log \frac{1}{1-\delta}$ entropy rate and (l,b)-PRB sources has at least $\frac{b}{l}$ entropy rate. $\omega(m)$-bit fixing sources, have at least ω entropy rate.

On the other hand, the simulation technique of [VV85, Vaz86, CG85], used to simulate BPP with $\omega(n)$-SR, and (l,b)-PRB random sources, of entropy rate $\Omega(1)$, does not apply to all random sources which have a constant lower bound on the entropy rate. For example, it can be verified that the technique does not work in the case of constant rate bit fixing sources. The reason for this is that the simulation relies on the fact that if the output of the source is broken into contiguous blocks, each block can be regarded as being itself an output of the source. In particular each block has $\Omega(1)$ entropy rate. Clearly this property is not satisfied by bit fixing sources.

The above discussion leads naturally to the following questions:

- Is a constant lower bound on the entropy rate a sufficient condition for the simulation of RP and BPP?
- If not, is there another general information-theoretic quantity such that a constant lower bound on it, is a sufficient condition for the simulation?
- If so, does there exist one simulation which works for every source satisfying the lower bound?
- Can a $o(1)$ lower bound also suffice for the simulation?

In the rest of the subsection, the first two questions are discussed. The last two are dealt with, later on.

The first question is answered in the negative. A constant lower bound on the entropy rate is too general a condition: a source of rate ω can concentrate a probability of almost $1-\omega$ on a single string so the error probability cannot be decreased below $1-\omega$.

Turning to the second question we note that the inadequacy of the entropy rate is due to the fact that "large" probabilities are allowed for individual strings even if the rate is large. Therefore we look for an information-theoretic quantity, a constant lower bound on which, prohibits large probabilities for individual strings. Such quantities are supplied by the Renyi generalized entropies [Ren70]. [8] Let S be a source with a probability distribution $p = (p_0 \ldots p_{2^m-1})$ on $\{0,1\}^m$. Let

[8] We are grateful to Mauricio Karchmer and Noam Nisan for drawing our attention to the Renyi entropies.

$\nu \neq 1$ be a real number. The entropy of order ν, I_ν, and the rate R_ν, are defined by:

$$I_\nu = \frac{1}{1-\nu} \log \sum_{i=0}^{2^m-1} p_i^\nu \qquad R_\nu = \frac{I_\nu}{m}$$

It can be easily verified that:

$$I_\infty = \lim_{\nu \to \infty} I_\nu = \log \frac{1}{p_{max}} \qquad I_1 = \lim_{\nu \to 1} I_r = \sum_{i=0}^{2^m-1} p_i \log \frac{1}{p_i}$$

Note also, that $I_\nu = \log \frac{1}{||p||_{\nu-1}}$ if $\nu > 1$, and $I_\nu = \log ||\frac{1}{p}||_{1-\nu}$ if $\nu < 1$. [9] Therefore I_ν is a decreasing function of ν.

The quantity I_1 has already proved inadequate for our needs. Because of the nonincreasing monotonicity, I_ν for $\nu < 1$ is ruled out as well. On the other hand the quantities I_ν $\nu > 1$ turn out to be appropriate since:

1. I_ν actually measures the L_ν norm of the probability distribution vector. A high value for I_ν means that the probability distribution is "close" to the uniform distribution, in the L_ν norm, and this precludes "large" probabilities for individual strings.
2. The family of probability distributions with a constant lower bound on R_ν, is very general and includes $\omega(m)$-Von Newman, Markov, and SR random sources, with $\omega(m) = \Omega(1)$, as well as (l, b) PRB sources with $\frac{b}{l} = \Omega(1)$, and $\omega(m)$-bit fixing sources, with $\omega(m) = \Omega(1)$.

Therefore we choose to define:

Definition 6.2 *An $(\gamma(m), \nu)$-Renyi random source is the sequence $\{\Pi_m\}_{m=1}^\infty$, where Π_m is the family of probability distributions on $\{0,1\}^m$ which satisfy:*

$$R_\nu \geq \gamma(m)$$

6.4 Simulations of RP and BPP with a Renyi Source.

The connection between disperser graphs and the simulation of RP (BPP) with imperfect sources is established by:

Theorem 6.1 *Suppose there exists an efficiently constructible family $\{G_i\}$ of*

$$\left(\alpha(N_i), M_i^{\mu-1}, t\right)$$

for $t = 1$ ($t = \frac{1}{2}$). Assume $\alpha(N_i) = \Omega\left(\frac{1}{polylog(N_i)}\right)$. Then any RP (BPP) algorithm can be simulated by a $(\mu + \frac{1}{m^c}, \nu)$-Renyi source for any $\nu > 1$ and $0 < c < 1$.

Proof Let A be an RP (BPP) algorithm which decides a language $\mathcal{L}$, with n random bits, and a constant error probability.

First, by theorems 5.1 and 5.2, there exists an RP (BPP) algorithm B, which decides $\mathcal{L}$ with n ($2n$) random bits and error probability bounded by α.

[9] Recall that if X is a random variable, then $\|X\|_k = E\left[X^k\right]^{\frac{1}{k}}$, where E denotes the expectation operator.

Next, let C be the simulation algorithm for B, induced by the family $\{G_i\}$. By theorem 5.5, algorithm C decides $\mathcal{L}$ with m random bits and error probability bounded by $2^{(\mu-1)m}$.

Therefore, given an actual input for C, the cardinality of its nonwitness set N, does not exceed $2^{m\mu}$. Let χ_N be the characteristic vector of N, and $p = (p_0 \dots p_{2^m-1})$ the vector of probabilities induced by a random source on strings of length m. Now if algorithm C gets its random input from the random source, the error probability of the algorithm is just the probability given by the source to the nonwitness set N. By the Holder inequality this can bounded by:

$$\sum_{x \in N} p(x) = \langle p, \chi_N \rangle \leq ||p||_\nu ||\chi_N||_{\frac{\nu}{\nu-1}} = 2^{\frac{(\nu-1)m}{\nu}(\mu - R_\nu)}$$

So if $R_\nu - \mu \geq \frac{1}{m^c}$, the error probability tends to 0 with m. ∎

Corollary 4.2 establishes the existence of dispersers with $\mu = (\log M_i)^{-\eta}$ for any $0 < \eta < 1$, and $\bar{d} = O(polylog(N_i))$. Efficient constructions of such dispersers would imply by theorem 6.1, that BPP can be simulated with m bits satisfying $R_\nu > m^{-\eta}$ for any $0 < \eta < 1$ (i.e. $I_\nu > m^\xi$ for any $0 < \xi < 1$). We conjecture that such efficient constructions do exist, and hence that the simulations do exist. However in this work we have only managed to construct dispersers with $\mu > \frac{1}{2}$, and this leads to the following:

Corollary 6.2 *Let $\nu > 1$. RP can be simulated by a $(\frac{1}{2} + \epsilon, \nu)$-Renyi source, and BPP can be simulated by a $(\frac{3}{4} + \epsilon, \nu)$-Renyi source.*

Proof Let $\eta < \epsilon$. By theorem 4.9 there exists a strongly constructible family G_i of d_i-regular, primitive, $\frac{1}{2} + \eta$ separable digraphs, with:

$$d_i \leq (\log |G_i|)^{\frac{1}{\eta}}$$

Let $W_i = W_{G_i}^{c_i \log |G_i|}$. By theorem 2.4 one can assume that $\left(\frac{\bar{\lambda}}{\lambda_0}\right)^2 \geq \frac{1}{d_i}$, so that by corollary 4.7, W_i is an efficiently constructible

$$\left((\log N_i)^{-\frac{1}{\eta}}, N_i^{\mu-1}, t\right)$$

disperser, with $M_i = N_i^{1+c_i \log d_i}$ and:

$$\mu = \frac{c_i \log d_i}{1 + c_i \log d_i} \left(t \left(\frac{1}{2} + \eta \right) + (1 - t) \right) + \frac{c(H(t) + \frac{t}{2})}{1 + c_i \log d_i} + \frac{1}{c_i \log N_i}$$

Now if one sets $c_i \log d_i$ to be a large enough constant (or even $O(polylog(N_i))$, then since $c_i \mapsto 0$ μ can be made asymptotically as close to $\frac{1}{2} + \eta$ as desired. The corollary now follows from theorem 6.1 with $t = 1$ in the RP case, and $t = \frac{1}{2}$ in the BPP case. ∎

Corollary 6.2 implies immediately simulations of RP and BPP with bit fixing sources:

Corollary 6.3 *1. For any $\epsilon > 0$ any RP (BPP) algorithm can be simulated by an adaptive or nonoblivious bit fixing source if the fraction of the fixed bits does not exceed $\frac{1}{2} - \epsilon$ ($\frac{1}{4} - \epsilon$).*

2. For any $\epsilon > 0$ any RP (BPP) algorithm can be simulated by a strong bit fixing source if the fraction of the fixed bits does not exceed $H_2^{-1}\left(\frac{1}{2} - \epsilon\right)$ ($H_2^{-1}\left(\frac{1}{4} - \epsilon\right)$), where H_2^{-1} is the inverse of the binary entropy function, on the interval $[0, \frac{1}{2}]$.

6.5 A Lower Bound on the Renyi Entropy Necessary for the Simulation of RP and BPP.

The attacks of [VV85, CG85] on the imperfect random sources problem use simulations for probabilistic algorithms induced by sampling procedures. It is natural to enquire about the limitations of this approach. A simple counting argument shows that a certain amount of I_ν entropy is required for such simulation algorithms if one does not make any assumption on the structure of the witness set.

Theorem 6.4 *A simulation of an RP (BPP) algorithm A induced by an efficiently constructible family of bipartite graphs, which uses m random bits, requires* $m^{\Omega(1)}$ I_ν *entropy.*

Proof If the original RP (BPP) algorithm A, requires n random bits to decide a string x then the simulating algorithm B can use at most $O(poly(n))$ random bits, to pick a multiset of random inputs for A. The size of the multiset must be $O(poly(n))$. If G is used to sample random inputs for A, then one must have:

$$\begin{array}{ll} |V_0(G)| & = N = 2^n \\ |V_1(G)| & = M = 2^m = N^{O(polylog(N))} \\ deg(V_1(G)) & = \bar{d} = O(polylog(N)) \end{array}$$

Choose a subset of vertices $S \subset V_1(G)$ such that $|S| < \frac{N}{2\bar{d}}$. Then $\Gamma(S)$ the neighbour set of S satisfies $|\Gamma(S)| < \frac{N}{2}$. Since there are no conditions on the structure of the nonwitness set, only on its size, and this only has to be bounded by $\frac{N}{2}$, we may assume that $\Gamma(S)$ is contained in the nonwitness set of some actual input x of A. Hence S is contained in T, the nonwitness set of x in algorithm B. Hence:

$$|T| = \Omega\left(\frac{N}{polylog(N)}\right)$$

Since one must have $I_\nu > \log|T|$ in order to ensure that the source cannot give the nonwitness set probability 1, the source must have $\Omega(\log N)$ i.e. $m^{\Omega(1)}$ I_ν. ∎

6.6 Matching the Lower Bound for Oblivious Bit Fixing Sources

There is a wide gap between the lower bound of section 6.5 and the upper bounds of section 6.4. We shall show here that this gap can be eliminated in the case of oblivious bit fixing sources.

Theorem 6.5 *For any constant* $0 < \xi < 1$, *any BPP algorithm A, can be simulated by any oblivious bit fixing source whose entropy is bounded from below by* m^ξ.

Theorem 6.5 follows from the next two lemmas:

Lemma 6.6 *Suppose a BPP algorithm A, can be simulated by oblivious bit fixing sources of rate bounded by a constant* α *with error probability* ϵ. *Then for any constant* $\beta < \alpha$, *A can be simulated by oblivious bit fixing sources of rate bound* β *and error probability* $\left(\frac{\log\beta}{\log\alpha} + 1\right)\epsilon$.

Proof Choose a natural number k such that:

$$\alpha^k < \beta$$

Suppose A requires n random bits. The simulation algorithm for rate bound β B will be of the form:

1. B gets $m = n^{2k-1}$ bits from a source of a rate bound β.

2. Produces $2k-1$ strings of n bits each.

3. Runs A on each string (as random input) and takes a majority vote.

The n bit strings are produced by regarding the m bits as a $2k-1$ dimensional cube of side length n, and producing one string for each direction by xoring bits in hyperplanes perpendicular to that direction.

For an estimate of the error of B note that a bit produced by xoring a hyperplane is good if the hyperplane contains at least one bit not fixed by the source. Hence if $\alpha_1 \ldots \alpha_{2k-1}$ are the fractions of "good bits" in the strings corresponding to the $2k-1$ directions of the cube then we have:

$$\prod_{i=1}^{2k-1} \alpha \geq \beta$$

Hence at least k of the α_i are not less than α. The error probability of B is bounded by the probability that A will err on any of these k strings so it is bounded by $k\epsilon$. ∎

Lemma 6.7 *For a sufficiently small constant α we have: If BPP algorithm A, can be simulated by n bits of oblivious bit fixing sources of rate bound α with error probability ϵ, then for any $\frac{1}{2} < \delta < 1$ there is a constant k, such that A can be simulated by $2n$ bits of oblivious bit fixing sources of rate bound $\alpha\delta$, and error probability $k\epsilon$.*

Proof The simulation algorithm B for rate bound $\alpha\delta$ will be of the form:

1. B gets $m = 2n$ bits from a source of a rate bound $\alpha\delta$.

2. Produces $2k-1$ strings of n bits each (k will be specified later).

3. Runs A on each string (as random input) and takes a majority vote.

The $2k-1$ n bit strings are produced by treating the m bits as vertices of a digraph G which is the l-th power of a d-regular expander (d and l are odd), partitioning the edge set into d^l perfect matching, and from each matching producing a string of n bits by xoring the endpoints of the n edges of the matching.

Suppose Z is the set of vertices of G corresponding to the "good bits". By theorem 2.2 we have:

$$|ZZ^c| \geq |Z|(1-\alpha\delta)(d^l - |\bar{\lambda}|^l)$$

where $|\bar{\lambda}|$ is the eigenvalue of the original expander. Therefore the average number of edges of ZZ^c per matching is bounded below by

$$|Z|(1-\alpha\delta)(1-\frac{|\bar{\lambda}|^l}{d^l})$$

It follows that a majority of the matchings contain at least:

$$|Z|\left(2(1-\alpha\delta)(1-\frac{|\bar{\lambda}|^l}{d^l})-1\right)$$

edges of ZZ^c.

Since xoring the endvertices of an edge of ZZ^c produces a "good bit", and since $|Z| \geq 2n\alpha\delta$, a majority of the n bits strings contain at least a

$$2\alpha\delta\left(2(1-\alpha\delta)(1-\frac{|\bar{\lambda}|^l}{d^l})-1\right)$$

fraction of good bits. Choosing α small enough, and l large enough, the factors $(1-\alpha\delta)$ and $(1-\frac{|\bar{\lambda}|^l}{d^l})$ can made as close to 1 as desired, and hence the above bound can be made to exceed α.

Setting $2k-1=d^l$ it follows that at least k of the n bit strings produced by B contain at least $n\alpha$ good bits. The error probability of B is bounded by the probability that A errs on any of the k strings and hence by $k\epsilon$. ∎

Proof (of theorem 6.5): By corollary 6.3 and lemma 6.6 we may assume that A can be simulated by a source of rate bound α which is sufficiently small to meet the condition of lemma 6.7. Also as the proof of theorem 6.1 shows, one can assume that the error is exponentially small. Applying lemma 6.7 $c\log n$ times, one gets a probabilistic algorithm which requires $m=n^{1+c}$ random bits from a source of entropy rate bound $\alpha n^{c\log\delta}$, with error probability exponentially small. Hence m^ϵ entropy suffices where:

$$\epsilon=\frac{1+c\log 2\delta}{1+c}$$

and by choosing c and δ appropriately ϵ can be made arbitrarily small. ∎

7 Acknowledgements

We are grateful to Michael Ben Or, Nati Linial, Endre Szemeredi, Mauricio Karchmer, Noam Nisan, and Yosi Ben Asher, for helpful discussions. The second author recognizes the origin of this paper in a Berkeley Cafe conversation with Mike Sipser, where he first told me of his results in [Sip86].

References

[AC88] N. Alon and F. R. K. Chung. Explicit constructions of linear sized tolerant networks. *Discrete Mathematics*, 72:15–19, 1988.

[AKS87] M. Ajtai, J. Komlós, and E. Szemerédi. deterministic simulation in LOGSPACE. *STOC*, page 132, 1987.

[Alo86a] N. Alon. Eigenvalues and expanders. *Combinatorica*, 6(2):83, 1986.

[Alo86b] N. Alon. Eigenvalues, geometric expanders, sorting in rounds and ramsey theory. *combinatorica*, 6:207–219, 1986.

[AM85] N. Alon and V. Milman. λ_1, isoperimetric inequalities for graphs and superconcentrators. *Jour of Combinatorial Theory*, B 38:73, 1985.

[BNS88] L. Babai, N. Nisan, and M. Szegedy. Multiparty protocols and logspace hard pseudorandom sequences. In *STOC*, 1988.

[Bac87] E. Bach. Realistic analysis of some randomized algorithms. In *STOC*, pages 453–461, 1987.

[Blu84] M. Blum. Independent unbiased coin flips from a correlated source: a finite state markov chain. In *FOCS*, pages 425–433, 1984.

[BOL85] M. Ben-Or and N. Linial. Collective coin flipping robust voting schemes and minimal banzhaf values. *FOCS*, pages 408–416, 1985.

[BR89] B. Berger and J. Rompel. simulating ($\log^c n$)-wise independence in NC. *FOCS* 1989 2-7.

[Buc86] M. W. Buck. Expanders and diffusers. *Siam J Alg Disc Meth*, 7(2):282, 1986.

[Che52] H. Chernoff. A measure of symptotic efficiency for tests of a hypothesis based on the sum of observations. *Annals of Mathematical Statistics*, **23**, 493-507.

[CG85] B. Chor and O. Goldreich. Unbiased bits from sources of weak randomness and probabilistic communication complexity. In *FOCS*, pages 429–442, 1985.

[CG86] B. Chor and O. Goldriech. On the power of two points based sampling. 1986.

[CGH+85] B. Chor, O. Goldreich, J. Hastad, J. Friedman, S. Rudich, and R. Smolenski. The bit extraction problem or t-resilient functions. In *FOCS*, pages 396–407, 1985.

[Chu] F. R. K. Chung. Diameters and eigenvalues. Manuscript.

[FN89] A. Fiat and M. Naor. Implicit O(1) probe search. *STOC* 1989 336-344.

[FRE81] R. Freivalds. Probabilistic two way machines. *International Symposium on Mathematical Foundations of Computer Science* Lecture notes in Computer Science, vol 118, Springer Berlin (1981) 33-45.

[FP87] J. Friedman and N. Pippenger. Expanding graphs contain all small trees. *Combinatorica*, 7(1):71–76, 1987.

[Gan59] F. R. Gantmacher. The theory of matrices, volume 2. *Chelsea Publishing Company*, 1959.

[GG81] O. Gaber and Z. Galil. Explicit construction of linear sized superconcentrators. *J Comput System Sci*, 22:407, 1981.

[GILL77] J. GILL. *Computational Complexity of Probabilistic Turing Machines. SIAM J. Comput.*, 6 (1977) 675-695.

[HR90] T. Hagerup, and C. Rüb, A Guided tour of Chernoff bounds. *Inf. Proc. Lett*, **33**, 305-308.

[IZ] R. Impagliazzo and D. Zuckerman. How to recycle random bits. FOCS 89.

[JM85] S. Jimbo and A. Maruoka. Expanders obtained from affine transformations. In *STOC*, page 88, 1985.

[KKL88] J. Kahn, G. Kalai, and N. Linial. The influence of variables on boolean functions. In *FOCS*, pages 68–80, 1988.

[KPS85] R. Karp, N. Pippenger, and M. Sipser. A time randomness tradeoff. In *AMS Conference on Probabilistic Computational Complexity Durham New Hampshire*, 1985.

[KPU88] D. Krizanc, D. Peleg, and E. Upfal. A time-randomness tradeoff for oblivious routing. In *STOC*, pages 93–102, 1988.

[LLS87] D. Lichtenstein, N. Linial, and M. Saks. Imperfect random sources and discrete controlled processes. In *STOC*, pages 169–177, 1987.

[Lov90] L. Lovász. Communication complexity: a survey, in path flowers and VLSI. Springer Verlag, 1990.

[LPS86] A. Lubotzky, R. Phillips, and P. Sarnak. Explicit expanders and the ramanujan conjecture. In *STOC*, page 240, 1986.

[Lub85] M. Luby. A simple parallel algorithm for the maximal independent set problem. *STOC* 1985 page 1.

[Lub88] M. Luby. Removing randomness in paralell without a processor penalty. *FOCS* 1988 162-173.

[Mar75] M. A. Margulis. Explicit construction of concentators. *Prob Info Trans*, 9, 1975.

[Min88] H. Minc. Nonnegative matrices. John Wiley and Sons, 1988.

[MNN89] R. Motwani, J. Naor, and M. Naor. The probabilistic method yields deterministic parallel algorithms. *FOCS*, pages 8 – 13, 1989.

[NN90] J. Naor, and M. Naor. Small-biase probability spaces: efficient constructions and applications. *STOC* 1990 213-223.

[Nis90] N. Nisan. Pseudorandom generators for space bounded computation. In *STOC*, 1990.

[NW88] N. Nisan and A. Wigderson. Hardness vs. randomness. In *FOCS*, 1988.

[Pip85] N. Pippenger. On networks of noisy gates. In *FOCS*, page 33, 1985.

[Ra88] P. Raghavan. Probabilistic construction of deterministic algorithms: approximating packing integer programs. *JCSS* vol. 37 1988 130-143.

[Ren70] A. Renyi. Probability theory. North-Holland Publishing Company, 1970.

[San] M. Santha. On using deterministic functions to reduce randomness in probabilistic algorithms. Manuscript.

[SW86] M. Saks and A. Wigderson. Probabilistic boolean decision trees, and the complexity of evaluating game trees. In *FOCS*, pages 29–38, 1986.

[Sho88] V. Shoup. New algorithms for finding irreducible polynomials over finite fields. In *FOCS*, pages 283–290, 1988.

[Sip86] M. Sipser. Expanders randomness or time vs space. In *Structure in Complexity Theory*, page 325. Springer Verlag, 1986.

[SV84] M. Santha and U. V. Vazirani. Generating quasi-random sequences from slightly random sources. In *FOCS*, pages 434–440, 1984.

[Tan84] R. M. Tanner. Explicit construction of concentrators from generalized n-gons. *SIAM J. Algebraic Discrete Methods*, 5:287–293, 1984.

[UW87] E. Upfal and A. Wigderson. How to share memory in a distributed system. *J. of the ACM.*, pages 116–127, 1987.

[Vaz86] U. Vazirani. Randomness adversaries and computation. PhD thesis, University of California Berkeley, 1986.

[VN51] J. von Neumann. Various techniques used in connection with random digits. *Notes by G. E. Forsythe, National Bureau of Standards, Applied Math Series*, 1951 vol 12 36-38.

[VV85] U. V. Vazirani and V. V. Vazirani. Random polynomial time is equal to semi random polynomial time. In *FOCS*, pages 417–428, 1985.

[Zuc] D. Zuckerman. General weak random sources. Preprint.