Journal of Computer and System Sciences **58**, 148–173 (1999)
Article ID jcss.1997.1546, available online at http://www.idealibrary.com on IDEAL®

Extracting Randomness: A Survey and New Constructions*

Noam Nisan and Amnon Ta-Shma

Institute of Computer Science, Hebrew University, Jerusalem, Israel

Received September 17, 1996

Extractors are Boolean functions that allow, in some precise sense, extraction of randomness from somewhat random distributions, using only a small amount of truly random bits. Extractors, and the closely related ''dispersers,'' exhibit some of the most ''random-like'' properties of explicitly constructed combinatorial structures. In this paper we do two things. First, we survey extractors and dispersers: what they are, how they can be designed, and some of their applications. The work described in the survey is due to a long list of research papers by various authors—most notably by David Zuckerman. Then, we present a new tool for constructing explicit extractors and give two new constructions that greatly improve upon previous results. The new tool we devise, a ''merger,'' is a function that accepts d strings, one of which is uniformly distributed and outputs a single string that is guaranteed to be uniformly distributed. We show how to build good explicit mergers, and how mergers can be used to build better extractors. Using this, we present two new constructions. The first construction succeeds in extracting *all* of the randomness from *any* somewhat random source. This improves upon previous extractors that extract only *some* of the randomness from somewhat random sources with ''enough'' randomness. The amount of truly random bits used by this extractor, however, is not optimal. The second extractor we build extracts only some of the randomness and works only for sources with enough randomness, but uses a near-optimal amount of truly random bits. Extractors and dispersers have many applications in ''removing randomness'' in various settings and in making randomized constructions explicit. We survey some of these applications and note whenever our new constructions yield better results, e.g., plugging our new extractors into a previous construction we achieve the first explicit N-superconcentrators of linear size and polyloglog(N) depth.

CONTENTS

1. INTRODUCTION

During the last two decades the use of randomization in the design of algorithms has become common place. There are many examples of randomized algorithms for various problems which are better than any known deterministic algorithm for the problem. The randomized algorithms may be faster, more space-efficient, use less communication, allow parallelization, or may be simply simpler than the deterministic counterparts. We refer the reader, e.g., to [MR95] for a textbook on randomized algorithms.

Despite the widespread use of randomized algorithms, in almost all cases it is not at all clear whether randomization is really necessary. As far as we know, it may be possible to convert *any* randomized algorithm to a deterministic one without paying any penalty in time, space, or other resources. In many cases, in fact, we do know how to convert a randomized algorithm to a deterministic one—"de-randomizing" the algorithm. This notion of "de-randomization" has also become quite common. By now, a standard technique for designing a deterministic algorithm is to first design a randomized one (which in many cases is easier to do) and then de-randomize it.

This state of affairs in algorithmic design is parallel in the design of various combinatorial objects (such as graphs or hypergraphs with certain properties). The "probabilistic method" is many times used to nonconstructively prove the existence of these sought-after objects. Again, in many cases

* This paper is a combination of the paper "On Extracting Randomness from Weak Random Sources" [Ta-96] and the paper "Refining Randomness: Why and How" [Nis96]. This work was supported by BSF Grant 92-0043 and by a Wolfeson award administered by the Israeli Academy of Sciences.

0022-0000/99 $30.00

it is known how to "de-randomize" these probabilistic proofs and achieve an explicit construction. We refer the reader to [AS92a] for a survey of the probabilistic method.

De-randomization techniques. It is possible to roughly categorize the techniques used for de-randomization according to their generality. On one extreme are techniques which relate very strongly to the problem and algorithm at hand. These usually rely on a sophisticated understanding of the structure of the problem and are not applicable to different ones. On the other extreme are completely general de-randomization results—e.g., converting *any* polynomial time-randomized algorithm to a deterministic one. This may be done with a sufficiently good pseudo-random generator, but with our current understanding of computational complexity such results *always* rely on unproved assumptions.

In the middle range of generality lie various techniques that apply to certain "types" of randomized algorithms. Algorithms that use their randomness in a "similar" way be de-randomized using similar techniques. Two main strategies for de-randomization are commonly employed. The first is the construction of a "small sample space" for the algorithm. Instead of choosing a truly random string, we fix a small set of strings—the "small sample space"—and then take a string from the sample space, instead of choosing it completely at random. This requires, of course, a proof that this sample space is good enough as a replacement for a truly random string. The second strategy is to adaptively "construct" a replacement for the random string—e.g., by gradually improving some conditional probability. In many cases both strategies are combined, and a replacement string is constructed in the small sample space.

It is probably fair to say that there are only two or three basic types of tools which are commonly used in the construction of small sample spaces for de-randomizations:

1. pairwise (and k-wise) independence and hashing,
2. small bias spaces,
3. expanders,
4. extractors and dispersers.

We refer the reader, again, to [AS92a, MR95] for further information as well as for references.

Dispersers and extractors. In this paper we deal with the fourth general type of tool: a family of graphs called *dispersers and extractors*. These graphs have certain strong "random-like" properties—hence they can be used in many cases where "random-like" properties are needed.

Think of a bipartite graph with $N = 2^n$ vertices on the left-hand side, $M = 2^m$ vertices on the right-hand side, and degree $D = 2^d$. Our functions will take a name of a vertex on the left-hand side and an index of an edge and will return the name of the vertex on the right-hand side: $G: (n) \times (d) \to (m)$, where (l) denotes $\{0, 1\}^l$.

For $A \subseteq \{0, 1\}^n$ we denote $\Gamma(A) = \{z = G(x, y): x \in A, y \in \{0, 1\}^d\}$. Similarly, for a distribution X on $\{0, 1\}^n$, $\Gamma(X)$ denotes the distribution of $G(x, y)$ induced by choosing x according to X, and y uniformly in $\{0, 1\}^d$.

DEFINITION 1.1. A function $G: (n) \times (d) \to (m)$ is called a (k, ε)-disperser if for any $A \subseteq \{0, 1\}^n$, $|A| \geqslant K = 2^k$, we have that $|\Gamma(A)| \geqslant (1 - \varepsilon) M$.

DEFINITION 1.2. A function $G: (n) \times (d) \to (m)$ is called a (k, ε)-extractor if for any distribution X on $\{0, 1\}^n$ "with k bits of randomness," we have that $\Gamma(X)$ is ε-close to uniform.

For the time being think of the term "X has k bits of randomness" as having the meaning that X is uniformly distributed over a set of size $K = 2^k$. We will later (Section 2) precisely define this notion.

It is easy to see that any (k, ε)-extractor is also a (k, ε)-disperser (consider the uniform distribution on A).

We can view extractors as taking an n bit string with some k randomness, investing d truly random bits and extracting m quasi-random bits. The quality of a disperser/extractor is therefore measured by the number of truly random bits d (we would like it to be small), the required amount of randomness in the somewhat random source k (again, we would like it to be small) and the number of quasi-random bits that are extracted m (we would like it to be as close as possible to k).

Survey of previous results. Dispersers were first defined (with somewhat different parameters) by Sipser [Sip88], while extractors were defined by Nisan and Zuckerman [NZ93]. The roots of the research on extractors lie mostly in the work on "somewhat random sources" done in the late 1980's, by Vazirani, Santha, and Vazirani, Vazirani, and Vazirani, Chor and Goldreich, and others [SV86, Vaz87a, Vaz86, Vaz87b, VV85, CG88]. The direct development of the constructions and applications of extractors and dispersers came, first, in papers written by Zuckerman in the early 1990s [Zuc90, Zuc91] and then in a sequence of papers by various authors [NZ93, WZ93, SZ94, SSZ95, Zuc93, Zuc].

The first explicit construction of extractors came in [NZ93] and relied on techniques developed in [Zuc90, Zuc91]. This construction had $d = \text{polylog}(n)$ for $k \geqslant n/\text{polylog}(n)$. An efficient extractor working for small k's, $k = \Theta(\log(n))$, was obtained by [GW94, SZ94] using tiny families of hash functions [NN93, AGHP92]. This was used in [SZ94] to improve upon the [NZ93] extractor and to get it work for any $k > \sqrt{n}$. In [SSZ95] a disperser with $d = O(\log n)$ was obtained for any $k = n^{\Omega(1)}$. In [Zuc], $d = O(\log n)$ was obtained for $k = \Omega(n)$.

In Table 1 we list the currently known best explicitly constructible extractors and dispersers for various parameters.

TABLE 1

Required crude randomness	No. of truly random bits	No. of output bits	Reference
$k=\Omega(\log(n))$	$d=k$	$m=(1+\Omega(1))\cdot k$	$\varepsilon=2^{-\Omega(k)}$ [GW94, SZ94]
$k=\Omega(n)$	$d=O\left(\log(n)+\log\left(\frac{1}{\varepsilon}\right)\right)$	$m=\Omega(k)$	[Zuc]
$k=\Omega(n^{1/2+\gamma})$	$d=O\left(\log^2 n\cdot\log\left(\frac{1}{\varepsilon}\right)\right)$	$m=n^{\delta}$, $\delta\leqslant\gamma$	[SZ94]
$k=\Omega(n^{\gamma})$	$d=O(\log n)$	$m=n^{\delta}$, $\delta<\gamma$	Disperser, $\varepsilon=\frac{1}{2}$ [SSZ95]

New results. We devise a new tool, a "merger," and show how to build good explicit mergers and how mergers can be used to build better extractors. We then give two new explicit constructions. The first extractor we build works for *any* source whatever its min-entropy is, and it extracts *all* the randomness in the given source.

THEOREM 1. *For every constant $\gamma<1$, $\varepsilon\geqslant 2^{-n^{\gamma}}$, and every $k=k(n)$ there is an explicit (k,ε) extractor $E\colon (n)\times(\mathrm{polylog}(n)\cdot\log(1/\varepsilon))\mapsto(k)$.*

The extractor is the first to work for *any* source, no matter how much randomness it contains, and also the first to extract *all* of the randomness. Its only drawback is that the amount of truly random bits used is polynomial in what is optimal. The second extractor we present works only for some of the sources, extracts only some of the randomness, but uses a near-optimal number of truly random bits (Table 2).

THEOREM 2. *For every constant c and $\gamma>0$ there is some constant $\delta>0$ and an explicit $(n^{\gamma},1/n)$ extractor $E\colon(n)\times(O(\log(n)\log^{(c)}n))\mapsto(\Omega(n^{\delta}))$, where $\log^{(c)}n=\underbrace{\log\log\cdots\log}_{c}n$.*

Applications. There are many examples where extractors and dispersers are used for the purposes of de-randomization—of algorithms or in explicit constructions. Our new constructions improve some of them. In the following we state these results. Formal definitions and proofs are given in Section 6.

The first application is constructing explicit a-expanding graphs, obtained by plugging our first extractor into the [WZ93] construction.

COROLLARY 1.1. *For any N and $1\leqslant a\leqslant N$ there is an explicitly constructible a-expanding graph with N vertices, and maximum degree $O((N/a)\,2^{\mathrm{polyloglog}(N)})$.*[1]

This corollary has applications on sorting [Pip87a, WZ93] and selecting [AKSS89, WZ93] in k rounds.

COROLLARY 1.2. *There are explicit algorithms for sorting in k rounds using $O(n^{1+1/k}\cdot 2^{\mathrm{polyloglog}(n)})$ comparisons and for selecting in k rounds using $O(n^{1+1/(2^k-1)}\cdot 2^{\mathrm{polyloglog}(n)})$ comparisons.*

COROLLARY 1.3. *There are explicit algorithms to find all relations except $O(a\cdot n\log(n))$ among n elements, in one round and using $O((n^2/a)\,2^{\mathrm{polyloglog}(n)})$ comparisons.*

Another corollary is for the construction of explicit small-depth superconcentrators. It is again obtained by plugging our first extractor into a previous construction by [WZ93].

COROLLARY 1.4. *For every N there is an efficiently constructible depth-2 superconcentrator over N vertices with size $O(N\cdot 2^{\mathrm{polyloglog}(N)})$.*[2]

Wigderson and Zuckerman [WZ93] prove that a direct corollary of this is:

COROLLARY 1.5. *For any N there is an explicitly constructible superconcentrator over N vertices with linear size and $\mathrm{polyloglog}(N)$ depth.*[3]

We can also prove a deterministic version of the hardness of approximating the iterated log of MaxClique. See [Zuc93] for more details.

Finally, using our second extractor we get:

COROLLARY 1.6. *For any $\delta>0$ and constant $k>0$, BPP can be simulated in time $n^{O(\underbrace{\log\log\cdots\log}_{k}n)}$ using a weak random source X with minentropy at least n^{δ}.*[4]

Organization of the paper. In Section 2 we give formal definitions. We also provide the basics, go over some preliminaries, and discuss simple lower and upper bounds. In

[1] The obvious lower bound is N/a. The previous upper bound [WZ93, SZ94] was $O(N/a)\,2^{\log(N)^{1/2+o(1)}}$.

[2] This improves the current upper bound of $O(N\cdot 2^{\log(N)^{1/2+o(1)}})$ achieved using the [WZ93] technique and the [SZ94] extractor.

[3] This improves the current upper bound of $O(\log(N)^{1/2+o(1)})$.

[4] Previous result [SZ94] was for $\delta>1/2$ and required $n^{O(\log(n))}$ time.

TABLE 2

Required crude randomness	No. of truly random bits	No. of output bits			
Any k	$d = poly\left(\log(n) + \log\left(\frac{1}{\varepsilon}\right)\right)$	$m = k$			
$k = \Omega(n^{\gamma})$	$d = O(\log(n) \cdot \underbrace{\log\log\cdots\log}_{c} n)$	$m = \Omega(n^{\delta})$,	$\delta < \gamma$	$\varepsilon = 1/n$	Any constant c

Section 3 we give a survey of some of the previous constructions of extractors and dispersers and explain some of the main ideas behind them. In Section 4 we present our new tool, the "merger," and construct our first explicit extractor, to be followed in Section 5 by our second explicit extractor. Finally, in Section 6, we survey what applications the existence of good dispersers/extractors have.

2. BASICS

2.1. Preliminaries

Let us first define some notions which will be used throughout this paper.

Probability distributions. We will constantly be discussing probability distributions, the distances between them, and the randomness hidden in them. In this subsection we give the basic definitions used to allow such a discussion.

A probability distribution X over a (finite) space Λ simply assigns to each $a \in \Lambda$ a positive real $X(a) > 0$, with the property that $\sum_{a \in \Lambda} X(a) = 1$. For a subset $S \subseteq \Lambda$ we denote $X(S) = \sum_{a \in S} X(a)$. The uniform distribution U on Λ is defined as $u(a) = 1/|\Lambda|$ for all $a \in \Lambda$.

When presenting our new constructions, we would like to be precise and distinguish between distributions and random variables. We denote random variables by capital letters, and their corresponding distributions by a barred variable, e.g. $\bar{X}$ is the distribution corresponding to X. All conditional expressions, e.g. $(A \mid B)$, denote distributions. U_k denotes the uniform distribution over k bits. In all other sections we usually identify a random variable with its probability distribution (we make the distinction only where necessary) and use capitals $X, Z, \ldots$ to denote such random variables and probability distributions. We use small letters $a, x, z, \ldots$ to denote elements in the probability space. Unless stated otherwise x is distributed according to X, z according to Z, etc.

We denote by $A \circ B$ the concatenation of two random variables A and B. A variable that appears twice (or more) in the same expression has the same value in all occurrences, i.e. $A \circ A$ denotes a random variable with values $a \circ a$. On the contrary, $A \times B$ denotes taking A and B independently; thus $A \times A$ denotes a random variable with values $a_1 \circ a_2$, where a_1 and a_2 are completely independent.

Finally, given a random variable $X = X_1 \circ \cdots \circ X_n$ we denote $X_i \circ \cdots \circ X_j$ by $X_{[i, j]}$, and the same applies for instances $x_{[i, j]}$.

Statistical distance.

DEFINITION 2.1. Let X, Y be two distributions over the same space Λ. The *statistical distance* between them is given by $d(X, Y) = \frac{1}{2}|X - Y|_1 = \frac{1}{2}\sum_{a \in \Gamma} |X(a) - Y(a)| = \max_{S \subseteq \Lambda} |X(S) - Y(S)|$.

It is easy to verify that the statistical distance is indeed a metric. We say X is ε-close to Y if $d(X, Y) \leqslant \varepsilon$. We say X is ε quasi-random if it is ε-close to uniform.

Min-entropy. We will need to measure the "amount of randomness" that a given distribution X has in it. The Shannon entropy of X certainly springs to mind, but it turns out that we will require a different notion.

DEFINITION 2.2. The min-entropy of a distribution X is $H_{\infty}(X) = \min_a(-\log_2(X(a)))$.

I.e., if $H_{\infty}(X) \geqslant k$ then for any x, $\text{Prob}(X = x) \leqslant 2^{-k}$. It is easy to verify that $H_{\infty}(X)$ is always bounded from above by the Shannon entropy $H(X)$, $H_{\infty}(X) \leqslant H(X)$. Equality holds whenever X is uniform over a subset $S \subseteq \Lambda$, in which case $H_{\infty}(X) = H(X) = \log_2 |S|$. It is useful to think of a distribution X with $H_{\infty}(X) = k$ as a generalization of being uniform over a set of size 2^k.

2.2. Extractors and Dispersers

Extractors and dispersers are very similar to each other, yet, in the literature, dispersers have been usually defined as graphs [Sip88, SSZ95], while extractors as functions [NZ93, SZ94, Zuc]. We will define both extractors and dispersers both as functions and as graphs, taking the view that it is the same combinatorial object, viewed in two different, useful, ways.

Graph definitions. Extractors and dispersers are certain types of bipartite (multi-)graphs. Throughout this paper the left-hand side of the graph will have $N = 2^n$ vertices and the right-hand side of the graph $M = 2^m$ vertices. Vertices will be numbered by integers which we identify with their binary representation. Thus the left-hand side of the graph is always $[N] = \{1 \cdots N\} = \{0, 1\}^n$, and the right-hand side is

always $[M] = \{1 \cdots M\} = \{0,1\}^m$. The graphs will usually be highly unbalanced $n > m$. Furthermore, all vertices on the left-hand side will have the same degree, $D = 2^d$, which is usually very small $d \ll m$.

Given a graph $G = ([N], [M], E)$; for each vertex $a \in [N]$, we denote its neighbor set by $\Gamma(a) = \{z \in [M] \mid (a, z) \in E\}$. For a subset $A \subseteq [N]$ we denote its neighbor set by $\Gamma(A) = \bigcup_{a \in A} \Gamma(a)$. For a distribution X on $[N]$, we denote by $\Gamma(X)$ the probability distribution induced on $[M]$ by first choosing $x \in [N]$ according to X and then choosing uniformly a random neighbor $z \in \Gamma(x)$.

DEFINITION 2.3. A (multi-)graph $G = ([N], [M], E)$ is a (k, ε)-disperser if for any $A \subseteq [N]$, $|A| \geqslant K = 2^k$, $|\Gamma(A)| \geqslant (1-\varepsilon) M$.

DEFINITION 2.4. A (multi-)graph $G = ([N], [M], E)$ is a (k, ε)-extractor if for any distribution X on $[N]$, with $H_\infty(X) \geqslant k$, we have that $\Gamma(X)$ is ε-close to uniform on $[M]$.

Remark 2.1. This definition is slightly different from the one in [NZ93, SZ94], who require that the random choice of the edge originating at x is also almost independent from $\Gamma(X)$. The difference is minor, though, and we prefer this definition.

It is easy to see that any (k, ε)-extractor is also a (k, ε)-disperser (consider the uniform distribution on A).

The name "disperser" is natural from this point of view; the graph disperses any set of size K, to almost all the right-hand side.

Function definitions. The definitions below are equivalent to the previous definitions, but take the functional view. Recall that the left-hand side of our graphs is $\{1 \cdots N\} = [N] = \{0,1\}^n$, the right-hand side $\{1 \cdots M\} = [M] = \{0,1\}^m$. The left degree is $D = 2^d$, so the edges originating at each vertex on the left-hand side will be labeled by $\{1 \cdots D\} = [D] = \{0,1\}^d$. For ease of notation we will denote $\{0,1\}^k$ by (k). Our functions will take the name of a vertex on the left-hand side and an index of an edge and will return the name of the vertex on the right-hand side: $G\colon (n) \times (d) \to (m)$. For $A \subseteq \{0,1\}^n$. We will denote $\Gamma(A) = \{z = G(x, y)\colon x \in A,\ y \in \{0,1\}^d\}$. Similarly, for a distribution X on $\{0,1\}^n$, $\Gamma(X)$ will denote the distribution of $g(x, y)$ induced by choosing x according to X, and y uniformly in $\{0,1\}^d$.

DEFINITION 2.5. A function $G\colon (n) \times (d) \to (m)$ is called a (k, ε)-disperser if for any $A \subseteq \{0,1\}^n$, $|A| \geqslant K = 2^k$, we have that $|\Gamma(A)| \geqslant (1-\varepsilon) M$.

DEFINITION 2.6. A function $G\colon (n) \times (d) \to (m)$ is called a (k, ε)-extractor if for any distribution X on $\{0,1\}^n$, $H_\infty(X) \geqslant k$, we have that $\Gamma(X)$ is ε-close to uniform.

The name "extractor" is natural from this point of view: the function extracts m-bits of randomness from the n-bits given to it, as long as these n bits have k min-entropy. In order to do this it uses d truly random bits as well.

Explicitness. We will be constructing explicit extractors and dispersers. Let us define what an explicit construction is. Let us recall that (in theoretical computer science) defining a combinatorial object as explicitly constructed only makes sense for a *family* of objects. In our case, we will be taking n as the parameter of the family, have $N = 2^n$, and view $M = 2^m$, $K = 2^k$, $D = 2^d$, ε as depending on it.

DEFINITION 2.7. A family of functions $G = \{G_n\colon (n) \times (d) \to (m)\}$ $(d = d(n), m = m(n))$, is called explicit if there is a deterministic Turing machine that for every $x \in \{0,1\}^n$ and $y \in \{0,1\}^d$, outputs $G_n(x, y)$ in polynomial time (in the length of the input x, y).

2.3. Behavior of Parameters

In this section we discuss the basic relations between the different parameters of extractors and dispersers. Let us recall first the five different parameters we are interested in, and their intuitive meanings, both in the graph view and in the function view:

- $N = 2^n$. N is the number of vertices in the left-hand side of G; n is the number of bits in the string x from which randomness should be extracted.
- $M = 2^m$. M is the number of vertices on the right-hand side of G; m is the length of the near-random output z.
- $D = 2^d$. D is the left-degree of G; d is the length of the truly random input y of G.
- $K = 2^k$. K is the size of the set which should dispersed; k is the min-entropy of x.
- ε. The error parameter.

We view n as a parameter, m as being a second parameter anywhere in the range $1 \leqslant m \leqslant n$, and then we try to have k and ε as small as possible. Our main goal for each choice of these parameters is to get d to be as small as possible.

Nonexplicit existence. Rather standard applications of the probabilistic method show that for all $n \geqslant m \geqslant 1$, $\varepsilon > 0$, there exist extractors with $k = m$ and $D = O(n/\varepsilon^2)$ and dispersers with $D = O(n/\varepsilon)$ (i.e., in both cases $d = O(\log n + \log \varepsilon^{-1})$). This calculation was done, for certain dispersers, in [Sip88].

Lower bounds. A trivial lower bound, for any n, m and $\varepsilon < 1/2$, is $d \geqslant m - k - 1$. We will mostly consider the case where $k \geqslant m$ so this does not help us. In [NZ93] a lower bound of $d \geqslant \min(m, \Omega(\log(n-k) + \log \varepsilon^{-1}))$ was proved for all $n, m, k \leqslant n-1$, $\varepsilon < 1/2$.

Expanders versus dispersers and extractors. Extractors and dispersers resemble expanders in that the requirement imposed in both cases is that each subset of vertices have

enough neighbors. A major difference is the strongly unbalanced sizes of the different sides of the graph in dispersers and extractors. A more important difference is that we can obtain much stronger parameters with extractors and dispersers. Such strong parameters cannot be obtained by using the eigenvalue methods used in all expander constructions. On the other hand, expanders with constant degree exist, while we already saw an $\Omega(n)$ lower bound on the degree of dispersers.

3. SURVEY OF PREVIOUS CONSTRUCTIONS

In this section we survey the main ingredients used in previous constructions and indicate how they are put together. We describe the constructions precisely, but we provide only sketches of proofs for their validity.

All constructions for extractors and dispersers build, as a starting point, an extractor with very weak parameters, but which can be obtained easily. This is described in Section 3.1. In Section 3.2 we describe how extractors can be composed when given a source with some special properties—a "block-wise" source. In Section 3.3, we show how such a composition may be applied to arbitrary sources with high min-entropy. Finally, Section 3.4 sketches the different ways in which these elements can be combined to obtain some constructions of extractors and dispersers.

3.1. The Mother of All Extractors

A useful way to think about extractors is to consider the edges of the extractor as hashing the vertices on the left to vertices on the right. The requirement is that large enough sets are well hashed. Viewed this way we can construct an extractor from a family of hash functions as follows.

A construction based on hashing. Let H be a family of function $h\colon [N] \to [L]$. The extractor defined by H is given by $G(x, h) = h(x) \circ h$. Thus $D = |H|$, and $M = DL$.

Now we define the property that we require from the family of hash functions in order for the graph to be an extractor.

DEFINITION 3.1. $H = \{h\colon [N] \mapsto [L]\}$ is a family of hash functions with collision error δ, if for any $x_1 \neq x_2 \in [N]$, $\text{Prob}_{h \in H}[h(x_1) = h(x_2)] \leqslant (1+\delta)/L$.

The following lemma is a variant of the leftover hash lemma of Impagliazzo, Levin, and Luby [ILL89], stated in our terms.

LEMMA 3.1 [ILL89]. *Let H be a family of hash functions from $[N]$ to $[L]$ with collision error δ. Then, the extractor defined from H is an (k, ε)-extractor for $K = 2^k = O(L/\delta)$ and $\varepsilon = O(\sqrt{\delta})$.*

Proof (Sketch). We need to show that for every distribution X on $[N]$, with $H_\infty(X) \geqslant \log(L/\delta)$, the distribution of $h \circ h(x)$ is ε-close to uniform (where x is chosen according to X and h uniformly in H). It turns out that the notion of collision probability is a convenient tool for this proof. The collision probability of a distribution X is defined to be $\text{col}(X) = \sum_a X(a)^2$. With this definition the following three steps, which imply the lemma, can be shown.

1. If $H_\infty(X) \geqslant K$ then $\text{col}(X) \leqslant 1/K$.
2. If $\text{col}(X)$ is small and H has a small collision error, then the collision probability of $Z = (h \circ h(x))$ is close to the collision probability of the uniform distribution (on $H \times [L]$).
3. If the collision probability of Z is close to the collision probability of the uniform distribution (on the domain of Z) then Z is close to uniform. ∎

Universal hashing. The first family of hash function which can be used for extractors are Carter–Wegman universal hash functions.

DEFINITION 3.2. $H = \{h\colon [N] \to [L]\}$ is called a universal family of hash functions, if for any $x_1 \neq x_2 \in [N]$ and for any $w_1, w_2 \in [L]$, $\text{Prob}_{h \in H}(h(x_1) = w_1 \wedge h(x_2) = w_2) = 1/L^2$.

It is clear that a universal family of hash functions, has, in particular, 0 collision error. For all ranges of $1 \leqslant l \leqslant n$, there exist constructions of universal families H of hash functions of size $|H| = \text{poly}(N = 2^n)$.

Using these families and the construction presented above, we get, for every $n \leqslant m \leqslant 2n$ extractors with high degree $d = O(n)$ and with $k = m - d + O(\log \varepsilon^{-1})$ which is optimal for k (for this choice of n, m, d). This degree is much higher than we desire, and the output size is larger than we are usually interested in (usually we want $m < n$).

We did get something useful though; the min-entropy on the left-hand side is really extracted in full since the output randomness, m, is essentially the sum of the input min-entropy, k, and the extra randomness, d.

This construction is sufficient, as a starting point, for the composition of extractors described in Section 3.2. A better construction (which is easier to use) is implied by better families of hash functions, described next.

Tiny families of hash functions. Universal families of hash functions have 0-collision error, and, in fact, 0 collision error does imply that their size $|H|$ must be large $|H| \geqslant N$. On the other hand, our purposes do not quite require 0-collision error but only small collision error. This will allow us to reduce the size of the family of hash functions and, hence, to reduce the extractor degree. It turns out that "tiny families of hash functions" [SZ94, GW94] can be built using small-biased distributions.

LEMMA 3.2 [SZ94, GW94]. *For all $1 \leqslant L = 2^l \leqslant N = 2^n$ and $\varepsilon > 0$. There exist explicit families H of hash functions from $[N]$ to $[L]$, with ε collision error and with size $|H| = \text{poly}(n, \varepsilon, L)$.*

This translates to

COROLLARY 3.3. *For every* $M = 2^m \leqslant N = 2^n$ *and* $\varepsilon > 0$, *there exists an explicit* (k, ε)*-extractor with* $D = \text{poly}(n, \varepsilon, M)$ *and* $k = m - d + O(\log \varepsilon^{-1})$.

Notice that this extractor is almost optimal, except that D is also polynomial in M. Thus, for small M, i.e. with size $M \leqslant \text{poly}(n)$, we get an optimal extractor.

A useful way to view this is that we manage to multiply the number of truly random bits from d to $m = (1 + c)\, d$ $(c > 0)$ using a source with enough min-entropy to supply this increase.

3.2. An Extractor for Blockwise Sources

Another useful way to look at extractors is to consider them as multipliers of pure randomness (from d bits to m bits), as long as they are also given a source with enough min-entropy. They take a very short random string y and output a long near-random string z. Of course, the extra randomness in z really comes from the source, X. Viewed this way, it is natural to take the output of one extractor and feed it into another extractor, compounding their "randomness multiplying" powers.

Composing extractors.

DEFINITION 3.3. Let $G_1 : [N_1] \times [D_1] \to [M_1]$, and let $G_2 : [N_2] \times [D_2] \to [M_2 = D_1]$ be extractors. Define $G_1 \circ G_2 : [N_1] \times [N_2] \times [D_2] \to [M_1]$ to be $G_1(x_1, G_2(x_2, y))$.

Intuitively, it is clear that if X_1 has as much min-entropy in it as G_1 requires and X_2 as much min-entropy as G_2 requires then the output of $G_2 \circ G_1$ will indeed be close to random. A delicate issue to consider is the independence of the random variables X_1 and X_2; if they are indeed independent then this intuitive claim is true. If they are allowed to be correlated then the previous statement may not hold. The following key definition, first formulated in [CG88], turns out to suffice, instead of total independence.

DEFINITION 3.4. Let X_1, X_2 be (possibly correlated) random variables taking values, respectively, in $[N_1]$ and $[N_2]$. We say that (X_1, X_2) is a blockwise (k_1, k_2) source if

1. $H_\infty(X_1) \geqslant k_1$.
2. For every fixed values x_1 of X_1, $H_\infty(X_2 \mid X_1 = x_1) \geqslant k_2$. ("$X_2 \mid X_1 = x_1$" denotes the marginal distribution on X_2 conditioned on the event $X_1 = x_1$.)

LEMMA 3.4. *Let* $G_1 : [N_1] \times [D_1] \to [M_1]$ *be a* (k_1, ε_1)*-extractor, and let* $G_2 : [N_2] \times [D_2 = M_1] \to [M_2]$ *be a* (k_2, ε_2)*-extractor. Let* X_1, X_2 *be a blockwise* (k_1, k_2) *source, and let* y *be chosen in random. Then the output of* $G_1 \circ G_2(x_1, x_2, y)$ *is* $(\varepsilon_1 + \varepsilon_2)$*-close to uniform.*

Proof (Sketch). Denote $w = G_2(x_2, y)$ and consider this random variable W. Fix any value x_1. Conditioned on $X_1 = x_1$, the distribution of W is ε_2 close to uniform (since G_2 is an extractor and by condition 2 of being a blockwise source). However, this is true for every value x_1, thus the joint distribution of (X_1, W) is ε_2 close to the product distribution $X_1 \times \mathit{Uniform}$. Thus the distribution of $G_1(x_1, w)$ is ε_2-close to the distribution of $G_1(x_1, w')$, where w' is chosen uniformly in $[D_2 = M_1]$, independently from X_1. This last distribution is ε_1-close to uniform since G_1 is an extractor and by condition 1 of being a blockwise source. The lemma follows. ∎

This idea can, of course, be extended to allow composition of an arbitrary number of extractors. The truly random input, y_i, to each extractor is obtained from the output of the previous one, and they each get a separate "block" as the somewhat random input, x_i. We can also relax the requirement that for every prefix the next block has much min-entropy, to that for most prefixes the next block is close to a distribution with much min-entropy. Thus, the requirement that is needed for the output to be close to uniform is

DEFINITION 3.5 (Extending [CG88]). Let $B_1, ..., B_t$ be correlated random variables taking values, respectively, in $[N_1], ..., [N_t]$. We say that $(B_1, ..., B_t)$ is a blockwise $(k_1, ..., k_t)$ source to within ε, if for each $1 \leqslant i \leqslant t$ and for all but an ε fraction of the sequences of values $x_1 \cdots x_{i-1}$, we have that $(B_i \mid B_1 = x_1, ..., b_{i-1} = x_{i-1})$ is ε close to a distribution with at least k_i min-entropy.

Having as input a blockwise $(k_1, ..., k_t)$ source to within ε, we can combine t extractors $G_1 \cdots G_t$ (with $D_{i-1} = M_i$ for all $1 < i \leqslant t$, and G_i being a (k_i, ε_i)-extractor) to output m_1 near-random bits starting from d_t truly random ones. A useful way to view this is that $G_1 \circ \cdots \circ G_t$ multiplies the number of random bits by the product of the "randomness-multiplying" capabilities of each extractor. Of course, this multiplication require a blockwise source and not just any source with enough min-entropy.

3.3. Getting a Blockwise Source

By composing, as described above, the basic extractors constructed in Section 3.1, we can directly extract essentially all the randomness from a blockwise source using only a very small number of truly random bits. Our problem, though, is to extract randomness from an arbitrary distribution with enough min-entropy. One possible solution is to first convert a general distribution X into a blockwise source $B_1 \cdots B_t$, and then proceed using this composition.

The first step in getting a blockwise source from a general distribution is to get the first block $B = B_1$. It turns out that once we can do this, in an appropriate way, the other blocks can be obtained in essentially the same way. What we need is for B to have enough min-entropy, even though it is much shorter than the original string (so that enough min-entropy is left for further blocks).

In [NZ93] it is shown that choosing a subset of the bits of X in a pairwise independent way suffices for this.

Other ways of sampling a subset of the bits of X behave similarly.

Given an n bit string $x_1 \cdots x_n$ and a subset $S \subseteq \{1 \cdots n\}$, denote x_S to be the string obtained by concatenating the bits x_i for all $i \in S$ (in the natural order).

We take any distribution S on strings of length l taken from $\{1 \cdots n\}$ such that for every $1 \leqslant i < j \leqslant n$: $\Pr_{s \in S}[i \in s \text{ and } j \in s] = \Pr_{s \in S}[i \in s] \cdot \Pr_{s \in S}[j \in s]$. This may be achieved in a sample space of size $O(n^2)$, by choosing the l elements pairwise independently; thus the description of S requires only $O(\log n)$ bits.

Getting a block. Given a string $x_1 \cdots x_n$ and an index of a set S (of size l in a pairwise independent sample space), we output x_S.

LEMMA 3.5 [NZ93]. *For every distribution X and for almost all choices of S, the distribution of X_S is close to some distribution W with $H_\infty(W) \geqslant \tilde{\Omega}((l/n)\, H_\infty(X))$.*

Proof (Intuition only). Consider the entropy $H(X)$ of the source instead of the min-entropy. We associate with each bit i of X its conditional entropy $p_i = H(X_i \mid X_1 \cdots X_{i-1})$. A standard fact regarding conditional entropies implies that $\sum_i p_i = H(X)$. Since S is chosen in a pairwise independent way, w.h.p., $\sum_{i \in S} p_i$ is close to its expectation $l/n \cdot H(X)$. Again a standard calculation with conditional entropies will sow that $H(X_S) \geqslant \sum_{i \in S} p_i$, which provides the lower bound required.

The above argument can indeed be, quite easily, made formal and does show that for almost all choice of S, $H(X_S) \geqslant \Omega(l/n \cdot H(X))$. The problem is that we require a lower bound on $H_\infty(X_S)$, which we cannot obtain in a similar way since the "conditional min-entropies" of distributions do not behave as nicely as the conditional entropies.

The actual proof of this lemma proceeds by essentially carrying out the above argument separately for each possible value of x, and then combining all the x's back together. Unfortunately, this argument is quite cumbersome and delicate. ∎

We can use the above lemma to convert any random source into a blockwise source. Let us start with a general random source X with k min-entropy. We start by extracting, as in the previous lemma, a block B_1 of length $l_1 \ll m$. Notice that for any value b_1 of B_1, $H_\infty(X \mid B_1 = b_1) \geqslant H_\infty(X) - \log \Pr[B_1 = b_1]$. We expect $\Pr[B_1 = b_1]$ to be about 2^{-l_1}. Values of b_1 which take significantly smaller probability can be safely ignored since they happen so rarely. This means that even conditioned on (almost) any value of B_1, X has sufficient min-entropy. Hence, choosing another subset of entries S_2 (independently from the first S), we can extract one more block B_2, which has high min-entropy even conditioned on the history b_1. Actually, as long as $l_1 + \cdots + l_{i-1} \ll m$, we can go on and extract another block B_i, which has high min-entropy even conditioned on the history.

Thus we get

LEMMA 3.6 [NZ93]. *Let X be a distribution on $[N]$ with $H_\infty(X) \geqslant k$. Then for any choice of $l_1 \cdots l_t$, where $\sum l_i < k/2$, the above procedure extract a source $B = B_1 \circ \cdots \circ B_t$ which is close to a blockwise $(k_1, ..., k_t)$ source with $k_i = \tilde{\Omega}(kl_i/n)$.*

Notice that this lemma gives non-trivial parameters as long as $k \geqslant \Omega(\sqrt{n})$. Next we shortly mention a different idea which works even for smaller values of k.

An alternative method. This alternative method for effectively getting a blockwise source was introduced in [SSZ95] and will be further extended in Section 4. The intuition can again be best obtained by considering the entropy $H(X)$ instead of the min-entropy.

Let us again be given a source $X = (X_1 \cdots X_n)$ from which we aim to get a blockwise $(k_1 \cdots k_t)$ source. If we denote $e_i = H(X_1 \cdots X_i)$, then we have for all i, $0 \leqslant e_i - e_{i-1} \leqslant 1$. We also have $e_0 = 0$ and $e_n \geqslant k$. For any value $k_1 \leqslant k$, there thus always exists an index i_1 such that $k_1 \leqslant H(X_1 \cdots X_{i_1}) \leqslant k_1 + 1$—this is our first block, B_1. Now, the conditional entropy of X conditioned on B_1 is $H(X_{i_1+1} \cdots X_n \mid X_1 \cdots X_{i_1}) = H(X) - H(X_1 \cdots X_{i_1}) \geqslant k - k_1$. We can thus continue and find i_2 such that $k_2 \leqslant H(X_{i_1+1} \cdots X_{i_2} \mid X_1 \cdots X_{i_1}) \leqslant k_2 + 1$. This process can be continued as long as $\sum_j k_j < k$ and gives our blockwise source.

As before, this indeed gives blocks which have high entropy conditioned on the previous ones, but it cannot directly give a real blockwise source, which requires high min-entropy. Again, the solution is to mimic the above idea for each input value $x_1 \cdots x_n$ separately—which requires nontrivial details that we give in Section 4.

An important aspect omitted is how these $i_1, ..., i_t$ are explicitly found. This indeed cannot be done without knowing the source X. However, it turns out that for dispersers we can simply try all the choices and combine them together (i.e. take the union of edges that each choice implies). As long as t is relatively small, the total number of choices is manageable. Two more tricks can be used. In [SSZ95] a "universal" set of choices for $i_1, ..., i_t$ with smaller (for the interesting cases—polynomial) size is presented, thus achieving good dispersers (for certain ranges of parameters). In Section 4 we present a method by which these many choices for $i_1, ..., i_t$ can be "merged," thus achieving near-optimal constructions of extractors.

3.4. Some Constructions of Extractors

In Subsection 3.3 we saw how to convert any random source with high min-entropy into a blockwise source; in Subsection 3.3 we saw how to extract randomness from such sources, using a composition of basic extractors; and in Subsection 3.1 we saw how to build these basic extractors.

Let us see several ways by which these ingredients can be put together in order to get good extractors.

We first discuss the basic strategy used in [Zuc90, Zuc91, NZ93, SZ94] and present it essentially in the form found in [SZ94].

Choose $t = O(\log n)$ blocks each of size $O(k/\log n)$ as described in Subsection 3.3. This gives a (near) blockwise source. We now compose t extractors each which multiplies the number of random bits by a constant factor—as presented in Subsection 3.1. We start with $O(\log n)$ bits, which after going through the composed t extractors get multiplied all the way to the min-entropy present in the final block, $\tilde{O}(k/n \cdot k/\log n)$—which is our output($m = \tilde{\Omega}(k^2/n)$).

The total number d of truly random bits used by this construction is the sum of the following; (A) $O(\log n)$ bits for the random input of the first extractor; (B) $O(\log n)$ bits for each of the t blocks—used to choose the pairwise independent set used to get the block. All together this gives $d = O(\log^2 n)$. As remarked in Subsection 3.3, this can work as long as $k = H_\infty(X) \geqslant \tilde{\Omega}(\sqrt{n})$.

For $k = H_\infty(X) = \Omega(n)$, we can choose the sizes of the t blocks more carefully, making sure that the final block has length $\Omega(n)$, and thus we can extract $m = \Omega(n)$ bits.

Reducing d. We now show a simple idea from [SZ94], showing that by composing two of the extractors just designed we can reduce d even further. Let us first choose a block of length $k/2$ and a second block of length $n' = 2^{O(\sqrt{\log n})}$. Choose an extractor for the second block as described above—taking into account that now the length is n' (instead of n). Thus, using $O(\log^2 n') = O(\log n)$ bits we can extract many more random bits. A simple calculation will show that the number of bits extracted is more than the $O(\log^2 n)$ bits required for extraction of randomness from the first block. The total number of bits required is twice $O(\log n)$ bits for getting the two blocks, and $O(\log n)$ bits for random input to the second block.

The error parameter ε of this construction depends on the smaller block and $1/\text{poly}(n')$, which is larger than $1/\text{poly}(n)$. By carefully controlling the composition of many extractors of differing lengths, [SZ94, Zuc] show that $d = O(\log n + \log \varepsilon^{-1})$ suffices for any ε as long as $k = \Omega(n)$ (and even slightly less).

A different way of decreasing d is given in the construction of our second extractor in Section 5.

Increasing m. A simple repetition can increase the value of m. After using d_1 bits to extract m_1 bits from a source X with $H_\infty(X) \geqslant k$, we can use d_2 fresh random bits to extract m_2 more bits from X, as long as we use an extractor that can extract random bits from sources with min-entropy $k - m_1$. These m_2 bits are independent from the first m_1 bits since, even conditioned on any particular value of the first m_1 bits, X still has min-entropy of at least $k - m_1$, and thus the extractor produces random bits. This idea is implicit in [WZ93].

The above idea uses $d_1 + d_2$ bits to extract $m_1 + m_2$ bits and can of course be repeated until $k - \sum m_i$ is too small. Repeating the previous construction $O(1)$ times allows extracting $m = k(1-\eta)$ random bits using only $d = O(\log n)$ truly random ones for any constant $\eta > 0$ and $k = \Omega(n)$ [Zuc].

Decreasing k. This is done using the "alternative method" sketched in Section 3.3, and will be presented in Section 4.

4. THE FIRST CONSTRUCTION: AN EXTRACTOR FOR ANY MIN-ENTROPY!

In this section we present our first new extractor. The extractor works for any min-entropy, and extracts almost all of the min-entropy from the given source. We start with some preliminaries, and we continue with an informal construction followed by rigorous constructions and proofs.

4.1. Preliminaries

First, we formally state the results presented in the previous section. We first state "the mother of all extractors" presented in Section 3.1.

LEMMA 4.1 [SZ94]. *There is some constant $c > 1$ s.t. for any $k = \Omega(\log(n))$ there is an explicit $(2k, 2^{-k/5})$ extractor A_k: $(n) \times (k) \mapsto (ck)$. We denote this constant c by c_{tiny}.*

Next, we state a very useful lemma that we already saw during the proof of Lemma 3.4.

LEMMA 4.2 [NZ93]. *Let X and Y be two correlated random variables. Let $\bar{B}$ be a distribution, and call an x "bad" if $(Y \mid X = x)$ is not ε close to $\bar{B}$. If $\text{Prob}_{x \in \bar{X}}(x \text{ is bad}) \leqslant \eta$ then $\overline{X \circ Y}$ is $\varepsilon + \eta$ close to $\bar{X} \times \bar{B}$.*

We restate the blockwise extractor.

LEMMA 4.3 [CG88, NZ93]. *Let $X = X_1 \circ X_2 \circ \cdots \circ X_t$ be a $(k_1, ..., k_t)$ blockwise source to within ε where $k_t = \Omega(\log(n))$ and $k_{i-1} = c_{\text{tiny}} k_i$. Then there is an explicit block extractor $BE(X, U)$, using k_t truly random bits and extracting $\Omega(\sum_{i=1}^{t} k_i)$ ε quasi-random bits with $O(2^{-\Omega(k_t)} + t\varepsilon)$ error.*

Next, we state a result showing a way to convert an arbitrary source (with high min-entropy) into a blockwise source (Section 3.3).

LEMMA 4.4 [NZ93, SZ94]. *Let X be a random source over $\{0,1\}^n$. For any $l > 0$ and $\delta > 0$, there is an explicit function $B(x, y)$ which gets $x \in X$ and a short random string y, and returns l bits, s.t.: If $H_\infty(\bar{X}) \geqslant \delta n$, then $\overline{B(X, U)}$ is $(\delta l)^{\Omega(-1)}$ close to a distribution $\bar{W}$ with $H_\infty(\bar{W}) \geqslant \Omega(\delta \cdot l / \log(\delta^{-1})) \geqslant \Omega(\delta \cdot l / \log(n))$.*

Finally, all these lemmas were used by [SZ94] to get the following extractor.

LEMMA 4.5 [SZ94].[5] *Let $k(n) \geq n^{1+2+\gamma}$ for some constant $\gamma > 0$, then for any ε there is an explicit $(k(n), \varepsilon)$ extractor E: $(n) \times (O(\log^2 n \cdot \log(1/\varepsilon))) \mapsto (k^2(n)/n)$.*

4.2. An Informal Construction

Recall the "alternative method" for getting a blockwise source, presented in Section 3. The argument there shows that for any source $X = X_1 \cdots X_n$ and any $k < H(X)$, there is a splitting point $1 < i < n$ s.t. $k < H(X_1 \cdots X_i) < k+1$ and $H(X_{[i+1,n]} \mid X_{[1,i]}) = H(X) - H(X_{[1,i]})$.

Unfortunately, such a splitting point does not exist when we consider min-entropy. This can be demonstrated by considering the uniform distribution over the set $\{0 \circ \{0,1\}^k \circ 0^{n-k-1},\ 1 \circ 0^{n-k-1} \circ \{0,1\}^k\}$, where $k < n/2$. To get a splitting point with $H_\infty(X_{[1,i]}) \geq 1$ we have to take $i > n-k$, but then $H_\infty(X_{[i+1,n]} \mid X_{[1,i]})$ is 0.

Instead, for any source X with enough min-entropy, and most strings $x \in X$, there is some splitting point $1 \leq i \leq n$ that splits x into $x_1 \circ x_2$ s.t. both $\Pr(X_1 = x_1)$ and $\Pr(X_2 = x_2 \mid X_1 = x_1)$ are small. E.g., in the distribution given above strings of the form $0 \circ \{0,1\}^k \circ 0^{n-k-1}$ have their splitting point i in the range $[1, k]$ while strings of the form $1 \circ 0^{n-k-1} \circ \{0,1\}^k$ splits in the range $[n-k, n]$. Thus, instead of having one global splitting point, each string has its own "good" splitting point.

LEMMA 4.6. *Let $\bar{X}$ be a distribution over $\{0,1\}^n$ with $H_\infty(\bar{X}) \geq k_1 + k_2 + s$. Call an $x \in \bar{X}$ "good," if there is some i (dependent on x) s.t.*

- $\Pr(X_{[1,i]} = x_{[1,i]}) \leq 2^{-k_1}$ *and*
- $\Pr(X_{[i+1,n]} = x_{[i+1,n]} \mid X_{[1,i]} = x_{[1,i]}) \leq 2^{-k_2}$

Then $\Pr_{x \in \bar{X}}(x \text{ is not } good) \leq 2^{-s}$.

Proof. Let $x \in \bar{X}$. Let i be the first location splitting x into two blocks $x_{[1,i]} \circ x_{[i+1,n]}$ s.t.

$$\Pr(X_{[1,i]} = x_{[1,i]}) \leq 2^{-k_1}. \tag{1}$$

Since i is the first such location,

$$\Pr(X_{[1,i-1]} = x_{[1,i-1]}) \geq 2^{-k_1}. \tag{2}$$

Since $H_\infty(\bar{X}) \geq k_1 + k_2 + s$,

$$\Pr(X_{[1,n]} = x_{[1,n]}) \leq 2^{-(k_1+k_2+s)}. \tag{3}$$

[5] The parameters here are simplified. The real parameters appearing in [SZ94] are somewhat better.

Putting this together we get

$$\begin{aligned}
&\Pr(X_{[i+1,n]} = x_{[i+1,n]} \mid X_{[1,i]} = x_{[1,i]}) \\
&= \frac{\Pr(X_{[1,n]} = x_{[1,n]})}{\Pr(X_{[1,i]} = x_{[1,i]})} \\
&= \frac{\Pr(X_{[1,n]} = x_{[1,n]})}{\Pr(X_{[1,i-1]} = x_{[1,i-1]}) \cdot \Pr(X_i = x_i \mid X_{[1,i-1]} = x_{[1,i-1]})} \\
&\leq \frac{2^{-(k_1+k_2+s)}}{2^{-k_1} \cdot \Pr(X_i = x_i \mid X_{[1,i-1]} = x_{[1,i-1]})}.
\end{aligned}$$

Hence, for all strings $x \in X$ s.t. $\Pr(X_i = x_i \mid X_{[1,i-1]} = x_{[1,i-1]}) \geq 2^{-s}$, it holds that $\Pr(X_{[i+1,n]} = x_{[i+1,n]} \mid X_{[1,i]} = x_{[1,i]}) \leq 2^{-k_2}$, and x is good. In particular, $\Pr(x \text{ is not good}) \leq 2^{-s}$. ∎

A crucial point is that there are only n possible splitting points. If we want to split $x_{[1,n]}$ into t blocks, there are only n^t sets of splitting points, and most strings (all but $t \cdot 2^{-s}$) have a good splitting set. Therefore, we can split the universe $\{0,1\}^n$ to $n^t + 1$ classes, each class containing strings that are good for one particular splitting set, and one for all strings that do not have a good splitting set.

Suppose we are only given inputs x that belong to a specific class S. Then, by definition, X is a blockwise source with the partition S. Therefore, by the results of Section 3, we can extract randomness from it.

Of course, given x we do not know what is the right partition for it. However, since there are so few classes (n^t), we can try all of them. Let us denote by Z_i the output of the blockwise extractor over X assuming the ith possible partition S_i. Trying all n^t possible partitions gives us n^t outputs $Z_1, ..., Z_{n^t}$. Intuitively, one of the n^t output strings is random.

Let us define more precisely the type of source we get. We have $b = n^t$ distributions $Z_1, ..., Z_b$, and we know there is some selector function $Y = Y(x)$ (that assigns each good string to a class with a right splitting set), s.t. $(Z_i \mid Y = i) \approx U$. So let us define:

DEFINITION 4.1. $Z = Z_1 \circ \cdots \circ Z_b$ is a b-block (k, ε, η) somewhere random source, if each Z_i is a random variable over $\{0,1\}^k$, and there is a random variable Y over $[0 \cdots b]$ s.t.:

- for any $i \in [1 \cdots b]$: $d((Z_i \mid Y = i-), U_k) \leq \varepsilon$.
- $\text{Prob}(Y = 0) \leq \eta$.

We also say that Y is a (k, ε, η) selector for Z.

A b-block (k, ε, η)-somewhere random source Z can be viewed intuitively as a source composed of b strings of length k with a selector function that for all but an η fraction of the inputs finds a block that is ε quasi-random. The following lemma (proved in Appendix A) states that such a

source is $\varepsilon+\eta$ close to a "pure" $(k, 0, 0)$ somewhere random source and that it contains k min-entropy.

LEMMA 4.7. (1) *Any* (k, ε, η) *somewhere random source* Z *is* $\varepsilon+\eta$ *close to an* $(k, 0, 0)$*-somewhere random source* Z'. (2) *For any* $(k, 0, 0)$ *somewhere random source* Z, $H_\infty(\bar{Z}) \geqslant k$.

Notice the nice and simple structure somewhere random sources have. We will see (Section 4.6) that it is much easier to extract randomness from such sources. Let us call an extractor working only on somewhere random sources a "somewhere random merger."

DEFINITION 4.2. $M: (k)^b \times (d) \mapsto (m)$ is an ε-somewhere random merger, if for any b-block $(k, 0, 0)$ somewhere random source Z, the distribution of $E(z, y)$ when choosing $z \in \bar{Z}$ and $y \in U_d$, is ε close to U_m.

Note that by Lemma 4.7 it is indeed enough to define somewhere random mergers only for pure $(k, 0, 0)$ somewhere random sources.

Next we define

DEFINITION 4.3. We say $M = \{M_n\}$ is an explicit ε-somewhere random merger if there is a Turing machine that, given an input z, y to M_n, computes $M_n(z, y)$ in polynomial time (in the length of the inputs z, y).

We will see that it is not hard to build efficient somewhere random mergers. Building on that, our extractor does the following:

1. Try all $b = n^t$ partitions of x into $t = \Theta(\log(n))$ blocks.
2. For each partition set S_i, consider X as a block-wise source with the partition S_i and use the techniques of Section 3 to extract the randomness from it. Call the output Z_i.
3. $Z = Z_1 \circ \cdots \circ Z_b$ form a somewhere random source. Use a merger to merge the randomness in Z into a single almost uniform distribution.

In the coming sections we rigorously develop the above ideas. The formal presentation differs from the informal ideas above in two ways: first, the formal construction is done in polynomial time as opposed to time $n^{O(\log(n))}$ above; second, in the formal description we will give full formal proofs and, thus, we will have to specify all the details needed to implement it. To ease the reading, we advise the reader to keep this intuitive and informal construction in mind.

4.3. Composing Two Extractors

In the previous section we suggested an extractor that works by trying all partitions into $t = \Theta(\log(n))$ blocks and then merging all the results. In this section we build an extractor that works by trying all partitions into *two* blocks (i.e., $t = 2$). It turns out that once we build such an extractor, we can build an extractor that works for any t.

ALGORITHM 4.1. Suppose $E_1: (n) \times (d_1) \mapsto (m_1 = d_2)$ is a (k_1, ζ_1)-extractor, $E_2: (n) \times (d_2) \mapsto (m_2)$ is a (k_2, ζ_2)-extractor, and $M: (m_2)^n \times (\mu_1) \mapsto (m)$ is a ζ_3-somewhere random merger. Define the function $E_2 \overset{M}{\odot} E_1: (n) \times (d_1 + \mu_1) \mapsto (m)$ as follows: Given $a \in \{0, 1\}^n, r_1 \in \{0, 1\}^{d_1}, r_2 \in \{0, 1\}^{\mu_1}$,

1. Let $q_i = E_1(a_{[i, n]}, r_1)$ and $z_i = E_2(a_{[1, i-1]}, q_i)$ for $i = 1, \ldots, n$.
2. Let $E_2 \ominus E_1 = z_1 \circ \cdots \circ z_n$, and $E_2 \overset{M}{\odot} E_1 = M(E_2 \ominus E_1, r_2)$.

THEOREM 3. *Suppose* E_1, E_2, *and* M *are as above. Then for every safety parameter* $s > 0$, $E_1 \overset{M}{\odot} E_2$ *is a* $(k_1 + k_2 + s, \zeta_1 + \zeta_2 + \zeta_3 + 8n2^{-s/3})$*-extractor.*

Proof. Obviously, it is enough to show that $E_1 \ominus E_2$ is a $(m_2, \zeta_1 + \zeta_2, 8n2^{-s/3})$-somewhere random source. To prove this, assume $H_\infty(\bar{X}) \geqslant k_1 + k_2 + s$. Denote by Q_i and Z_i the random variables with values q_i and z_i, respectively. Also, let $\varepsilon_3 = 2^{-s/3}$, $\varepsilon_2 = 2\varepsilon_3$, and $\varepsilon_1 = 2\varepsilon_2$.

We define a selector for $Z = Z_1 \circ \cdots \circ Z_n = E_1 \ominus E_2$ in two phases: first we define a function f which is intuitively what the selector function should be; for each string x split it at the last place s.t. the remaining block is still "random" enough (notice the similarity to Lemma 4.6).

DEFINITION 4.4. Define $f(w)$ to be the last i s.t. $\text{Prob}(X_{[i, n]} = w_{[i, n]} \mid X_{[1, i-1]} = w_{[1, i-1]}) \leqslant (\varepsilon_2 - \varepsilon_3) \cdot 2^{-k_1}$.

Some of the splitting points are rare (e.g., if only a few strings split at some location i) and, therefore, may cause strange behavior (see the proof of Lemma 4.6 for an example). Next, we identify these rare (and bad) cases.

DEFINITION 4.5. Define w to be "bad" if $f(w) = i$ and

1. $\text{Prob}_{x \in X}(f(x) = i) \leqslant \varepsilon_1$, or
2. $\text{Prob}_{x \in X}(f(x) = i \mid x_{[1, i-1]} = w_{[1, i-1]}) \leqslant \varepsilon_2$, or
3. $\text{Prob}_{x \in X}(X_i = w_i \mid x_{[1, i-1]} = w_{[1, i-1]}) \leqslant \varepsilon_3$.

We denote by B the set of all bad w. We denote by B_i $(i = 1, 2, 3)$ the set of all bad w satisfying condition (i).

Finally, we get rid of the bad cases to get our selector function.

DEFINITION 4.6. Let Y be the random variable obtained by taking the input a and letting $Y = Y(a)$, where

$$Y(w) = \begin{cases} 0, & w \text{ is bad,} \\ f(w), & \text{otherwise.} \end{cases}$$

It holds that $\text{Prob}(w \text{ is bad}) \leqslant n(\varepsilon_1 + \varepsilon_2 + \varepsilon_3) \leqslant 8n \cdot 2^{-s/3}$ (the proof is easy; see Appendix C). We complete the proof by showing that $(Z_i \mid Y = i)$ is $(\zeta_1 + \zeta_2)$-close to uniform.

CLAIM 4.1. *If* $\text{Prob}(Y=i \mid X_{[1,i-1]}=w_{[1,i-1]})>0$ *then* $H_\infty(X_{[i,n]} \mid Y=i \text{ and } X_{[1,i-1]}=w_{[1,i-1]}) \geqslant k_1$.

Therefore, for any such $w_{[1,i-1]}$, $(Q_i \mid Y=i \text{ and } X_{[1,i-1]}=w_{[1,i-1]})$ is ζ_1-close to uniform (since E_1 is an extractor). Hence by Lemma 4.2, the distribution $(X_{[1,i-1]} \mid Y=i) \times (Q_i \mid Y=i \text{ and } X_{[1,i-1]}=w_{[1,i-1]})$ is ζ_1-close to the distribution $(X_{[1,i-1]} \mid Y=i) \times U$. But

CLAIM 4.2. $H_\infty(X_{[1,i-1]} \mid Y=i) \geqslant k_2$.

Therefore, using the extractor E_2 we get that $(Z_i \mid Y=i)$ is $\zeta_1+\zeta_2$-close to uniform. ▮

Now we prove Claims 4.1 and 4.2.

Proof of Claim 4.1. For any w s.t. $Y(w)=i$

$$\begin{aligned}
&\text{Prob}(X_{[i,n]}=w_{[i,n]} \mid X_{[1,i-1]}=w_{[1,i-1]},\ Y(x)=i)\\
&\quad\leqslant \frac{\text{Prob}(X_{[i,n]}=w_{[i,n]} \mid X_{[1,i-1]}=w_{[1,i-1]})}{\text{Prob}(Y(x)=i \mid X_{[1,i-1]}=w_{[1,i-1]})}\\
&\quad\leqslant \frac{(\varepsilon_2-\varepsilon_3)\cdot 2^{-k_1}}{\text{Prob}(Y(x)=i \mid X_{[1,i-1]}=w_{[1,i-1]})}\\
&\quad\leqslant \frac{(\varepsilon_2-\varepsilon_3)\cdot 2^{-k_1}}{\varepsilon_2-\varepsilon_3}=2^{-k_1}.
\end{aligned}$$

The first line is true since $\text{Prob}(A \mid B) \leqslant \text{Prob}(A)/\text{Prob}(B)$; the second line, since $f(w)=i$; and the third follows from Claim C.1. ▮

Proof of Claim 4.2. Take any $w_{[1,i-1]}$ that can be extended to some w with $Y(w)=i$:

$$\begin{aligned}
&\text{Prob}(X_{[1,i-1]}=w_{[1,i-1]})\\
&=\frac{\text{Prob}(X_{[1,n]}=w_{[1,n]})}{\text{Prob}(X_{[i,n]}=w_{[i,n]} \mid X_{[1,i-1]}=w_{[1,i-1]})}\\
&=\frac{\text{Prob}(X_{[1,n]}=w_{[1,n]})}{\text{Prob}(X_i=w_i \mid X_{[1,i-1]})\cdot \text{Prob}(X_{[i+1,n]}=w_{[i+1,n]} \mid X_{[1,i]})}.
\end{aligned}$$

However,

- $\text{Prob}(X_{[i+1,n]}=w_{[i+1,n]} \mid X_{[1,i]}) \geqslant (\varepsilon_2-\varepsilon_3)\,2^{-k_1}$
- $\text{Prob}(X_i=w_i \mid X_{[1,i-1]}=w_{[1,i-1]}) \geqslant \varepsilon_3$
- $\text{Prob}(X_{[1,n]}=w_{[1,n]}) \leqslant 2^{-(k_1+k_2+s)}$.

The first line is true because $f(w)=i$; the second, because $w \notin B_3$; and the third, because $H_\infty(X) \geqslant k_1+k_2+s$. Thus,

$$\text{Prob}(X_{[1,i-1]}=w_{[1,i-1]}) \leqslant \frac{2^{-k_2-s}}{\varepsilon_3\cdot(\varepsilon_2-\varepsilon_3)}. \tag{4}$$

Therefore,

$$\begin{aligned}
&\text{Prob}(X_{[1,i-1]}=w_{[1,i-1]} \mid Y(x)=i)\\
&\quad\leqslant \frac{\text{Prob}(X_{[1,i-1]}=w_{[1,i-1]})}{\text{Prob}(Y(x)=i)}\\
&\quad\leqslant \frac{2^{-k_2-s}}{\varepsilon_3\cdot(\varepsilon_2-\varepsilon_3)\cdot \text{Prob}(Y(x)=i)}\\
&\quad\leqslant \frac{2^{k_2-s}}{\varepsilon_3\cdot(\varepsilon_2-\varepsilon_3)\cdot(\varepsilon_1-\varepsilon_2-\varepsilon_3)}=2^{-k_2}.
\end{aligned}$$

The first line is true because $\text{Prob}(A \mid B) \leqslant \text{Prob}(A)/\text{Prob}(B)$. The second follows from Eq. (4). The third follows from Claim (C.2). ▮

4.4. Composing Many Extractors

Now we build an extractor that works by trying all input partitions into t blocks. We do that by defining the composition of many extractors.

DEFINITION 4.7. Suppose $E_i\colon (n)\times(d_i)\mapsto(d_{i+1}+s_{i+1})$ is a (k_i,ζ_i)-extractor for $i=1,\ldots,t$; $s_i\geqslant 0$; and $s_2=0$. Suppose $M_i\colon (d_{i+2}+s_{i+2})^n\times(\mu_i)\mapsto(d_{i+2})$ is a $\bar{\zeta}_i$-somewhere random merger for any $i=1\cdots t-1$. We define the function $E=E_t \overset{M_{t-1}}{\odot} E_{t-1} \overset{M_{t-2}}{\odot} \cdots E_2 \overset{M_1}{\odot} E_1$ by induction to equal $E_t \overset{M_{t-1}}{\odot} (E_{t-1} \overset{M_{t-2}}{\odot} \cdots E_2 \overset{M_1}{\odot} E_1)$.

THEOREM 4. *Suppose* E_i, M_i, E *are as above, then for any safety parameter* $s>0$, $E\colon (n)\times(d_1+\mu_1+\cdots+\mu_{y-1})\mapsto(d_{t+1})$ *is an* $(n, \sum_{i=1}^t k_i+(t-1)\,s, \sum_{i=1}^t \zeta_i+\sum_{i=1}^{t-1}\bar{\zeta}_i+(t-1)\,n2^{-s/3+3})$*-extractor. If* E_i, M_i *are explicit, then so is* E.

Proof. Correctness. By induction on t. For $t=2$ this follows from Theorem 3. For larger t's this is a straightforward combination of the induction hypothesis and Theorem 3.

Running time. We compute $E_t \overset{M_{t-1}}{\odot} E_{t-1} \overset{M_{t-2}}{\odot} \cdots E_2 \overset{M_1}{\odot} E_1$ using a dynamic programming procedure:

1. Input: $x\in\bar{X}$, $y\in\{0,1\}^{d_1}$ and $y_j\in\{0,1\}^{\mu_j}$ for $j=1,\ldots,t-1$.
2. We compute the matrix M, where

$$M[j,i]=(E_j \overset{M_{j-1}}{\odot} E_{j-1} \overset{M_{j-2}}{\odot} \cdots E_2 \overset{M_1}{\odot} E_1)(x_{[i,n]}, y\circ y_1\circ\cdots\circ y_{j-1})$$

for $1\leqslant i\leqslant n$ and $1\leqslant j\leqslant t$.

The entries of the first row of M, $M[1,i]$ can be filled by evaluating $E_1(x_{[i,n]}, y)$. Suppose we know how to fill the jth row of M. We show how to fill the $(j+1)$th row.

• Denote $q_l = M[j, l]$ for $l = i, ..., n$, and let $z_l = E_{j+1}(x_{[i, l-1]}, q_l)$.

• Set $M[j+1, i] = M_j(z_i \circ \cdots \circ z_n, y_j)$.

By the definition of composition $M[j, i]$ has the correct value, and clearly, the computation takes polynomial time in n. ▮

Remark 4.1. Notice that using left associativity (rather than right associativity) we could use some of the quasi-randomness we get for doing the merges. Thus, it may appear that left associativity is more efficient in terms of the number of truly random bits used. However, we know how to implement right associativity composition in polynomial time (using a dynamic programming procedure) and we do not know of such an algorithm for left associativity composition.

4.5. Good Mergers Imply Good Extractors

Let us continue following the informal construction given in Section 4.2. We want to try all input partitions into $t = \Theta(\log(n))$ blocks, and for each partition we want to use the blockwise extractor of Lemma 4.3. In our new terminology this amounts to taking $E = A_k \overset{M}{\odot} \cdots A_{b^2\bar{k}} \overset{M}{\odot} A_{b\bar{k}} \overset{M}{\odot} A_{\bar{k}}$, where A_m is the extractor of Lemma 4.1 and b is some constant, $1 < b < c_{\text{tiny}}$. This extractor extracts $\Omega(k)$ bits from sources having k min-entropy. The only missing component in this construction is the existence of explicit good somewhere random mergers. However, assuming the existence of good somewhere random mergers, we get good extractors.

LEMMA 4.8. *Suppose for any $\bar{k} \leqslant k \leqslant \bar{\bar{k}}$ there is an explicit ε somewhere random merger M_k: $(k)^n \times (d) \mapsto (\Delta \cdot k)$, where Δ is a constant and $1/c_{\text{tiny}} < \Delta < 1$. Then, for any $\bar{k} \leqslant k \leqslant \bar{\bar{k}}$ there is an explicit $(k, \text{poly}(n) \cdot \varepsilon)$ extractor E: $(n) \times (O(\bar{k} \cdot \log(1/\varepsilon) + \log(n) \cdot d)) \mapsto (\Omega(k))$ extractor.*

Proof. Let $b = c_{\text{tiny}} \cdot \Delta$. Clearly b is a constant and $1 < b < c_{\text{tiny}}$. Define $k_i = b^i \cdot \bar{k} \cdot \log(1/\varepsilon)$, and let t be the first integer s.t. $\sum_{i=1}^{t} 2k_i \leqslant k/2$.

Define $E = E_t \overset{M_{t-1}}{\odot} E_{t-1} \cdots \overset{M_1}{\odot} E_1$, where

• E_i: $(n) \times (k_i) \mapsto (c_{\text{tiny}} k_i)$ is the $(k_i, 2^{-k_i/5})$-extractor A_{k_i} from Lemma 4.1.

• M_i: $(c_{\text{tiny}} \cdot k_{i+1})^n \times (d) \mapsto (k_{i+2} = b \cdot k_{i+1} = \Delta \cdot c_{\text{tiny}} \cdot k_{i+1})$ is an ε-somewhere random merger given in the hypothesis of the lemma.

Now we use Theorem 4 with $d_i = k_i$ and $s_i = (c_{\text{tiny}} - b)\, k_{i-1}$. To see that the above extractors and mergers can be indeed composed, notice that $d_i + s_i = bk_{i-1} + (c_{\text{tiny}} - b)\, k_{i-1} = c_{\text{tiny}} \cdot k_{i-1}$. Therefore, by Theorem 4, E is an extractor.

Now choose the safety parameter s to be $s = k/2t$, and let us check the parameters we get are as stated. Note that $t = O(\log(n))$ and that $\sum_{i=1}^{t} k_i + (t-1)\, s < k$. Also note that $k_{t+1} = \Omega(\sum_{i=1}^{t} k_i) = \Omega(k)$. It is easy to check that the error is as stated. Finally, since A_i, M_i are explicit, so is E. ▮

Just to demonstrate the above, assume for every k there is an explicit ε somewhere random merger M: $(m)^n \times (d) \mapsto (\Delta \cdot m)$. Then notice that by Lemma 4.8 we get an extractor for any min-entropy that extracts almost all of the min-entropy. Formally, assuming the above, we get for every m and explicit (m, ε) extractor E: $(n) \times (d \cdot \text{polylog}(n) \cdot \log(1/\varepsilon)) \mapsto (\Omega(m))$. Thus, our next objective is designing good somewhere random mergers.

4.6. Explicit Mergers

Now we turn to the construction of somewhere random mergers. We start with a somewhere random merger for a 2-block somewhere random source. By definition, a (k, ε) extractor E: $(2k) \times (d) \mapsto (m)$ extracts randomness from any source X over $\{0, 1\}^{2k}$ with $H_\infty(\bar{X}) \geqslant k$. In particular, by Lemma A, it extracts randomness from any 2-block $(k, 0, 0)$ somewhere random source.

COROLLARY 4.9. *Any (k, ε) extractor E: $(2k) \times (d) \mapsto (m)$ is also an ε somewhere random merger E: $(k)^2 \times (d) \mapsto (m)$.*

A b-block somewhere random merger. Now we show how to use 2-block mergers to build b-block mergers. Given a b-block somewhere random source, we merge the blocks in pairs in a treewise fashion, resulting in a single block. We show that after each level of merges we still have a somewhere random source, and thus, the resulting single block is necessarily quasi-random.

ALGORITHM 4.2. Let M: $(k)^2 \times (d(k)) \mapsto (k - m(k))$ be a merger. We build a merger M_l: $(k)^{2^l} \times (l \cdot d(k)) \mapsto (k - l \cdot m(k))$, by induction on l:

Input: $x^l = x_1^l \circ \cdots \circ x_{2^l}^l$, where each $x_i^l \in \{0, 1\}^k$.
Output: Let $d = d_1 \circ \cdots \circ d_l$, where d_j is chosen from $\{0, 1\}^{d(k)}$. If $l = 0$ output x^l, otherwise:

1. Let $x_i^{l-1} = M(x_{2i-1}^l \circ x_{2i}^l, d_l)$ for $i = 1, ..., 2^{l-1}$.
2. Let the output be $M_{l-1}(x_1^{l-1} \circ \cdots \circ x_{2^{l-1}}^{l-1}, d_1 \circ \cdots \circ d_{l-1})$.

THEOREM 5. *Assume for every k there is an explicit $\varepsilon(k)$ somewhere random merger M: $(k)^2 \times (d(k)) \mapsto (k - m(k))$ for some monotone functions t, k, and ε^{-1}. Let M_l: $(k)^{2^l} \times (l \cdot d(k)) \mapsto (k - l \cdot m(k))$ be as above. Then M_l is an $l \cdot \varepsilon(m - l \cdot k(m)))$ somewhere random merger.*

Proof. For $j = l, ..., 0$ denote by Z^j the random variable whose value is $x^j = x_1^j \circ \cdots \circ x_{2^j}^j$, where the input x is chosen according to $\bar{X}$ and d is uniform. Notice that $\overline{Z^l}$ is the distribution $\bar{X}$ and $\overline{Z^0}$ is the distribution of the output.

The theorem clearly follows from the following claim.

CLAIM 4.3. *Denote* $k_j = k - (l-j)\,m(k)$. *If* X *is a* $(k, 0, 0)$ *somewhere random source, then for any* $1 \leqslant i \leqslant 2^j$, $d(Z_i^j \mid Y \in [2^{l-j}(i-1)+1, 2^{l-j}i]), U_{k_j}) \leqslant (l-j)\cdot\varepsilon(k)$. ▮

Proof. The proof is by downward induction on j. The basis $j = l$ simply says that for any i, $d((X_i \mid Y = i), U_k) = 0$, which is exactly the hypothesis. Suppose it is true for j, we prove it for $j-1$. By the induction hypothesis,

- $d((Z_{2i-1}^j \mid Y \in [2^{l-j}(2i-2)+1,\ 2^{l-j}(2i-1)]),\ U_{k_j}) \leqslant (l-j)\,\varepsilon(k_j)$
- $d((Z_{2i}^j \mid Y \in [2^{l-j}(2i-1)+1,\ 2^{l-j}2i]),\ U_{k_j}) \leqslant (l-j)\cdot\varepsilon(k_j)$.

In Appendix B we prove

LEMMA 4.10. *Let* A, B, *and* Y *be any random variables. Suppose that* $d((A \mid Y \in S_1), U_k) \leqslant \varepsilon$ *and* $d((B \mid Y \in S_2), U_k) \leqslant \varepsilon$ *for some disjoint sets* S_1 *and* S_2. *Then* $(A \circ B \mid Y \in S_1 \cup S_2)$ *is* ε*-close to some* $\bar{X}$ *with* $H_\infty(\bar{X}) \geqslant k$.

Therefore,

- $(Z_{2i-1}^j \circ Z_{2i}^j \mid Y \in [2^{l-j+1}(i-1)+1, 2^{l-j+1}i])$ is $(l-j)\cdot\varepsilon(k_j)$ close to some $\bar{W}$ with $H_\infty(\bar{W}) \geqslant k_j$.
- Since $Z_i^{j-1} = M(Z_{2i-1}^j \circ Z_{2i}^j, d_j)$, it follows that $(Z_i^{j-1} \mid Y \in [2^{l-j+1}(i-1)+1,\ 2^{l-j+1}i])$ is $(l-j)\cdot\varepsilon(m)$ close to $M(x, d_j)$, where $x \in \bar{X}$ and $H_\infty(\bar{X}) \geqslant k_j$. Therefore, it is $(l-j)\cdot\varepsilon(k_j) + \varepsilon(k_j)$ close to uniform, as required. ▮

Remark 4.2. Notice that we use the same random string d_j for all merges occurring in the jth layer and that this is possible because in a somewhere random source we do not care about dependencies between different blocks. Also notice that the error is additive in the depth of the tree of merges (i.e. in l), rather than in the size of the tree (2^l).

4.7. Putting It Together

Recall that by Lemma 4.8 having $M\colon (k)^n \times (d) \mapsto (\Delta k)$ mergers, where Δ is some constant, implies the existence of good extractors. Theorem 5 states that it is enough to find good 2-block somewhere random mergers that do not lose much of the min-entropy. We also saw (Corollary 4.9) that it is enough to find a (k, ε) extractor $E\colon (2k) \times (d) \mapsto (m)$ with m very close to k. To be more specific, we need $m \geqslant k - O(k/\log(n))$. By [NZ93, SZ94, Zuc] we know to fund such extractors with $m = \Omega(k)$. Using a simple idea due to Wigderson and Zuckerman [WZ93] we can get m much closer to k.

More bits using the same extractor. Suppose we have an extractor E that extracts randomness from any source having at least k min-entropy. How much randomness can we extract from sources having K min-entropy when $K \gg k$?

The following algorithm is implicit in [WZ93]: use the same extractor E many times over the same string x, each time with a fresh truly random string r_i, until you get $K-k$ output bits. The idea is that as long as $|E(x, r_1) \circ \cdots \circ E(x, r_t)|$ is less then $K-k$, with high probability $(X \mid E(x, r_1) \circ \cdots \circ E(x, r_t))$ still contains k min-entropy, and therefore we can use the extractor E to further extract randomness from it. Thus, we have the following two lemmas, that are proven in detail in Appendix D.

LEMMA 4.11. *Suppose that for some* k *there is an explicit* (k, ε) *extractor* $E_k\colon (n) \times (d) \mapsto (m)$. *Then, for any* $K \geqslant k$ *and any safety parameter* $s > 0$ *there is an explicit* $(K, t(\varepsilon + 2^{-s}))$ *extractor* $E\colon (n) \times (td) \mapsto \min\{tm, K-k-s\}$.

LEMMA 4.12. *Suppose that for any* $k \geqslant \bar{k}$ *there is an explicit* $(k, \varepsilon(n))$ *extractor* $E_k\colon (n) \times (d(n)) \times (k/f(n))$. *Then, for any* k, *there is an explicit* $(k, f(n)\log(n)(\varepsilon + 2^{-d(n)}))$ *extractor* $E\colon (n) \times (O(f(n)\log(n)\,d(n))) \mapsto (k-\bar{k})$.

As a side corollary we can strengthen Lemma 4.12 to get an extractor that extracts the whole entropy of the source rather than just a constant fraction if it. This is done by applying Lemma 4.12, or more specifically, by using the same extractor $O(\log(n))$ times. Thus, combining Lemma 4.8 and Lemma 4.12 we get

COROLLARY 4.13. *Suppose* $\bar{k} = \bar{k}(n)$ *is a function s.t. for every* $k \geqslant \bar{k}(n)$ *there is an explicit* ε *somewhere random merger* $M\colon (k)^n \times (d) \mapsto (\Delta \cdot k)$, *where* $1/c_{\text{tiny}} < \Delta < 1$. *Then for any* k *there is an explicit* $(k, \text{poly}(n)\cdot\varepsilon)$ *extractor* $E\colon (n) \times (O(\bar{k}\cdot\log(n)\cdot\log(1/\varepsilon) + \log^2(n)\cdot d)) \mapsto (k)$.

Mergers that do not lose much min-entropy. The [SZ94] extractor of Lemma 4.5 works for any source with $H_\infty(\bar{X}) \geqslant n^{1/2+\gamma}$. Thus, using Lemma 4.11 by repeatedly using the [SZ94] extractor, we can extract at least $n/2 - n^{1/2+\gamma}$ quasi-random bits from a source having $H_\infty(\bar{X}) \geqslant n/2$. Thus, we have a 2-merger that does not lose much randomness in the merging process. Applying Theorem 5 we get a good n-merger. Thus we have

LEMMA 4.14. *Let* $b > 1$ *be a constant and suppose* $f = f(k) = f(k(n))$ *is a function s.t.* $f(k) \leqslant \sqrt[3]{k}$ *and for every* $k \geqslant k_0(n) : f(k) \geqslant b\cdot\log(n)$. *Then for every* $k \geqslant k_0$ *there is an explicit* $\log(n)\cdot\text{poly}(k)\cdot\varepsilon$ *somewhere random merger* $E\colon (n) \times (\log(n)\cdot\text{polylog}(k)\cdot f^2(k)\cdot\log(1/\varepsilon)) \mapsto k - k/b$.

Proof.
- By Lemma 4.5 there is an explicit $(k/f(k), \varepsilon)$ extractor $E\colon (k) \times (O(\log^2 k\cdot\log(1/\varepsilon))) \mapsto (k/f^2(k))$.
- By Lemma 4.11 there is an explicit $(k, \text{poly}(k)\cdot\varepsilon)$ extractor $E\colon (2k) \times (O(f^2(k)\cdot\log^2 k\cdot\log(1/\varepsilon)) \mapsto (k - k/f(k))$.
- By Theorem 5 there is an explicit $\log(n)\cdot\text{poly}(k)\cdot\varepsilon$ somewhere random merger $M\colon (k)^n \times (O(\log(n)\cdot\text{polylog}(k)\cdot f^2(k)\cdot\log(1/\varepsilon)) \mapsto (k - \log(n)\cdot k/f(k))$. Since $k/f(k) \leqslant k/b\log(n)$ for any $k \geqslant k_0$, we have that $\log(n)(k/f(k)) \leqslant k/b$. ▮

COROLLARY 4.15. *For every* $k \geqslant 2^{\sqrt{\log(n)}}$, *there is a* $\text{polylog}(n) \cdot \varepsilon$ *somewhere random merger* M_k: $(k)^n \times (\text{polylog}(n) \cdot \log(1/\varepsilon)) \mapsto (\Omega(k))$.

Proof. Take $f(k) = \log^c k$ for some constant $c > 2$. For any constant b, $k \geqslant 2^{\sqrt{\log(n)}}$ and n large enough, $\log^c m \geqslant b \cdot \log(n)$, and the corollary follows Lemma 4.14. ▮

Notice that Theorem 5 and Corollary 4.15 take advantage of the simple structure of somewhere random sources, giving us an explicit somewhere random merger that works even for sources with very small min-entropy to which the [SZ94] extractor of Lemma 4.5 does not apply.

Extractors that work for high min-entropy. Corollary 4.15 asserts the existence of good mergers for $k \geqslant 2^{\sqrt{\log(n)}}$, and therefore, plugging this into Corollary 4.13 we get

COROLLARY 4.16. *For every* k *there is an* $(k, \text{poly}(n) \cdot \varepsilon)$ *extractor* B_k: $(k) \times (O(2^{\sqrt{\log(n)}} \cdot \text{polylog}(n) \cdot \log(1/\varepsilon))) \mapsto (k)$.

The extractor B in Corollary 4.16 uses $O(2^{\sqrt{\log(n)}} \cdot \text{polylog}(n) \cdot \log(1/\varepsilon))$ truly random bits to extract all the randomness in the given source. Although $2^{\sqrt{\log(n)}}$ is quite a large amount of truly random bits, we can use the [SZ94] extractor to extract $n^{1/3}$ bits from $n^{2/3}$ min-entropy and then use these $n^{1/3} \gg O(2^{\sqrt{\log(n)}} \cdot \text{polylog}(n) \cdot \log(1/\varepsilon))$ bits to further extract all the remaining min-entropy. More precisely, if B is the extractor in Corollary 4.16, E_{sz} the extractor from Lemma 4.5 and M is the merger from Corollary 4.15, then $E = B \overset{M}{\odot} E_{sz}$ extracts $\Omega(k)$ bits from sources having $k \geqslant n^{2/3}$ min-entropy, using only $\text{polylog}(n)$ truly random bits! That is, we get the following lemma.

LEMMA 4.17. *Let* $\varepsilon \geqslant 2^{-n^\gamma}$ *for some constant* $\gamma < 1$. *There is some constant* $\beta < 1$ *s.t. for every* $k \geqslant n^\beta$ *there is an explicit* $(k, \text{poly}(n) \cdot \varepsilon)$ *extractor* E: $(n) \times (\text{polylog}(n) \cdot \log(1/\varepsilon)) \mapsto (\Omega(k))$.

Proof. Choose $\delta = (1-\gamma)/2$ and $\beta = 1 - \delta/2$. Let the extractor E be $E = B_K \overset{M}{\odot} E_{sz}$, where

- E_{sz} is the (n^β, ε) extractor E: $(n) \times (O(\log^2 n \cdot \log(1/\varepsilon))) \mapsto (n^{2\beta - 1})$ of Lemma 4.5.
- B_k is the extractor from Corollary 4.16.
- M is the merger from Corollary 4.15.

Since $n^{2\beta-1} = n^\delta \cdot n^\gamma = \Omega(2^{\sqrt{\log(n)}} \cdot \log(1/\varepsilon))$, $E = B_k \overset{M}{\odot} E_{sz}$ is well defined. By Theorem 3, for every k, E is an explicit $(k + n^\beta + n^\gamma, \text{poly}(n) \cdot \varepsilon)$ extractor E: $(n) \times (\text{polylog}(n) \cdot \log(1/\varepsilon)) \mapsto (\Omega(k))$. In particular if $H_\infty(X) = \Omega(n^\beta)$ we extract $\Omega(H_\infty(X))$ as required. ▮

The final result. Now that we know to extract a lot of randomness from sources having $\Omega(n^\beta)$ min-entropy with only $\text{polylog}(n)$ truly random bits, by Lemmas 4.12 and Theorem 5 we have good somewhere random mergers for *every* k. Thus by Corollary 4.13 we have good extractors for every k.

THEOREM 6. *For every constant* $\gamma < 1$, $\varepsilon \geqslant 2^{-n^\gamma}$, *and every* $k = k(n)$ *there is an explicit* (k, ε) *extractor* E: $(n) \times (\text{polylog}(n) \cdot \log(1/\varepsilon)) \mapsto (k)$.

Proof.

- By Lemma 4.12, Lemma 4.17 implies an explicit $(n, \text{poly}(n) \cdot \varepsilon)$ extractor E: $(2n) \times (\text{polylog}(n) \cdot \log(1/\varepsilon)) \mapsto (n - n^\beta)$.
- There is some constant c (that depends only on γ) s.t. for every $\log^c n \leqslant k \leqslant n$, $\log(n) \cdot k^\beta \leqslant k/\bar{c}$, where $\bar{c}$ is some constant s.t. $1/c_{\text{tiny}} < 1 - 1/\bar{c} < 1$ (e.g., $\bar{c} = 2c_{\text{tiny}}/c_{\text{tiny}} - 1$). Therefore, by Theorem 5 for every k there is an explicit $\text{poly}(n) \cdot \varepsilon$ somewhere random merger M: $(k)^n \times (\text{polylog}(n) \cdot \log(1/\varepsilon)) \mapsto (m - m/\bar{c})$.
- By Corollary 4.13, this implies an explicit $(k, \text{poly}(n) \cdot \varepsilon)$ extractor E: $(n) \times (\text{polylog}(n) \cdot \log(1/\varepsilon)) \mapsto (k)$ for any k. Plugging $\varepsilon' = \varepsilon/\text{poly}(n)$ gives the theorem. ▮

5. THE SECOND CONSTRUCTION: AN EXTRACTOR USING LESS TRULY RANDOM BITS

In this section we build an extractor that uses less truly random bits. The extractor works only for sources having at least $n^{\Omega(1)}$ min-entropy, and extracts only some small fraction of the min-entropy present in the original source. Yet, this extractor improves upon the previous construction of [SZ94] in two ways: first it uses much less truly random bits (almost linear), and second, it works for sources having less than $n^{1/2}$ min-entropy.

THEOREM 7. *For every constant* c *and* $\gamma > 0$ *there is some constant* $\delta > 0$ *and an* $(n^\gamma, 1/n)$ *extractor* E: $(n) \times (O(\log(n) \log^{(c)} n)) \mapsto (\Omega(n^\delta))$, *where*

$$\log^{(c)} n = \underbrace{\log \log \cdots \log}_{c} n.$$

The extractor uses two main building blocks: The first shows how to reduce the number of truly random bits needed for sources having $n^{1/2}$ min-entropy. The second shows how to use extractors for $n^{1/d}$ min-entropy to achieve extractors for $n^{1/d+1}$ min-entropy.

LEMMA 5.1. *Let* $f(n)$ *be an arbitrary function. Assume* $\forall \gamma > 0$, $\exists \delta > 0$ *s.t. there is a* $(k = n^\gamma, n^{-\Omega(1)})$ *extractor* E: $(n) \times (\log(n) \cdot f(n)) \mapsto (m = \Omega(n^\delta))$. *Then* $\forall \gamma' > 0$, $\exists \delta' > 0$ *s.t. there is a* $(k = n^{1/2 + \gamma'}, \varepsilon = n^{-\Omega(1)})$ *extractor* E': $(n) \times (O(\log(n) \cdot \log(f(n)))) \mapsto (\Omega(n^{\delta'}))$.

LEMMA 5.2. *Assume*

- $\forall \gamma' > 0$, $\exists \delta' > 0$ *s.t. there exists a* $(k = n^{\gamma'}, n^{-\Omega(1)})$ *extractor* E: $(n) \times (O(\log(n) \cdot 2^{f(n)})) \mapsto (\Omega(n^{\delta'}))$.
- $\forall \gamma'' > 0$, $\exists \delta'' > 0$ *s.t. there exists a* $(k = n^{1/d+\gamma''}, n^{-\Omega(1)})$ *extractor* F: $(n) \times (O(\log(n) \cdot f(n))) \mapsto (\Omega(n^{\delta''}))$.

Then $\forall \gamma > 0$, $\exists \delta > 0$ *s.t. there exists a* $(k = n^{1/(d+1)+\gamma}, n^{-\Omega(1)})$ *extractor* E': $(n) \times (O(\log(n) \cdot f(n))) \mapsto (\Omega(n^{\delta}))$.

Using these two lemmas we can prove Theorem 7.

Proof of Theorem 7. We prove the equivalent claim.

CLAIM. *For every constant* c *and* $\gamma > 0$ *there is some constant* $\delta > 0$ *and an* $(n^{\gamma}, 1/n)$ *extractor* E: $(n) \times (O(\log(n) \cdot (\log^{(c)} n)^a)) \mapsto (\Omega(n^{\delta}))$, *where* a *is some fixed constant.*

Proof. By induction on c. For $c = 1$ this follows from Theorem 6. Assume for c. Denote $f(n) = \log^{(c+1)} n$. The induction hypothesis states that $\forall \gamma > 0$, $\exists \delta > 0$ s.t. there is a $(k = n^{\gamma}, n^{-\Omega(1)})$ extractor E: $(n) \times (\log(n) \cdot 2^{O(f(n))}) \mapsto (\Omega(n^{\delta}))$.

By Lemma 5.1, $\forall \gamma' > 0$, $\exists \delta' > 0$ s.t. there is an $(n^{1/2+\gamma'}, n^{-\Omega(1)})$ extractor E': $(n) \times (O(\log(n) \cdot f(n))) \mapsto (\Omega(n^{\delta'}))$.

But now all the requirements of Lemma 5.2 are met, and therefore, using Lemma 5.2 repeatedly c times, we get the desired extractor. ∎

Therefore, we have proved Theorem 7. ∎

5.1. A Better Extractor for Sources Having $n^{1/2+\gamma}$ Min-entropy

In this section we prove Lemma 5.1. We show that combining the extractor of Theorem 6 with the [NZ93] block extractor, we can extract randomness from sources having $n^{1/2+\gamma}$ min-entropy using less random bits. The idea behind the construction is the following: since the given source X has $H_{\infty}(\bar{X}) \geqslant n^{1/2+\gamma'}$, we can use the [NZ93] block extraction to extract $t = O(\log(f(n)))$ blocks that together form a blockwise source with each block containing some $n^{\Omega(1)}$ min-entropy. Then, by investing $O(\log(n))$ bits, we can extract some $\log(n) \times 2^{\Omega(t)} = \log(n) \cdot f(n)$ random bits. Finally, we can use these bits in the extractor, given by the hypothesis of the lemma to extract $n^{\Omega(1)}$ quasi-random bits.

Proof of Lemma 5.1. Consider the following algorithm.

ALGORITHM 5.1. Fix $t = O(\log(f(n)))$, $l = n^{1/2}$. Choose $y_1, ..., y_t \in \{0,1\}^{O(\log(n))}$ and $y \in \{0,1\}^{O(\log(n))}$. Given $x \in X$,

1. Extract t blocks $b_1 = B(x, y_1), ..., b_t = B(x, y_t)$, where B is the block extraction operator of Lemma 4.4.
2. Compute $z = BE(b_2 \cdots b_t, y)$, where BE is the function extracting randomness from blockwise sources, of Lemma 4.3.
3. Finally, let the output be $E(b_1, z)$, where E is the extractor given in the hypothesis.

To prove correctness, notice that

CLAIM. *Fix* $y_1, ..., y_t$ *arbitrarily. The probability that there exists an* $1 \leqslant i \leqslant t$, *s.t.* $H_{\infty}(X \mid B_1 = b_1, ..., B_i = b_i) \leqslant n^{1/2+\gamma/2}$ *is less than* ε.

Proof. The total number of bits in $b_1 \cdots b_i$ is at most $t \cdot l \ll n^{1/2} \cdot \log(n)$. Denote

$$\text{Bad} = \{b_1 \circ \cdots \circ b_i \mid \Pr(b_1 \cdots b_i) \leqslant 2^{-n^{1/2+\gamma/2}}\}.$$

Then, $\Pr(\text{Bad}) \leqslant |\text{Bad}| \cdot 2^{-n^{1/2+\gamma/2}} \ll n^{-\Omega(1)}$. Also, for any $b_1 \cdots b_i \notin \text{Bad}$ and any x,

$$\begin{aligned}
&\Pr(X = x \mid B_1 = b_1, ..., B_i = b_i) \\
&\quad = \frac{\Pr(X = x)}{\Pr(B_1 = b_1, ..., B_i = b_i)} \\
&\quad \leqslant \frac{2^{-n^{1/2+\gamma}}}{2^{-n^{1/2+\gamma/2}}} \\
&\quad \ll 2^{-n^{1/2+\gamma/2}},
\end{aligned}$$

and therefore, for most prefixes $H_{\infty}(X \mid B_1 = b_1, ..., B_i = b_i) \geqslant n^{1/2+\gamma/2}$ as required. ∎

Therefore, using the block extractor of Lemma 4.4 we get

CLAIM. $B = B_1 \circ \cdots \circ B_t$ *is an* $(n^{\gamma/3}, ..., n^{\gamma/3})$ *blockwise source to within* $n^{-\Omega(1)}$.

Proof. For almost every prefix $b_1, ..., b_{i-1}$, $H_{\infty}(X \mid B_1 = b_1, ..., B_i = b_i) \geqslant n^{1/2+\gamma/2}$. Therefore, by Lemma 4.4, for almost every prefix $b_1, ..., b_{i-1}$ $(B_i \mid B_1 = b_1, ..., B_{i-1} = b_{i-1})$ is close to a distribution with at least $n^{\gamma/3}$ min-entropy. ∎

Therefore,

CLAIM. $\overline{B_1 \times Z}$ *is* $n^{-\Omega(1)}$ *close to* $\overline{B_1} \times U$.

Proof. For most prefixes b_1 $(B_2 \times \cdots \times B_t \mid B_1 = b_1)$ is a blockwise source. Therefore, by Lemma 4.3, $(Z \mid B_1 = b_1)$ is close to uniform. Hence, the claim follows by Lemma 4.2. ∎

Notice that $\bar{Z}$ is distributed over $\log(n) \cdot 2^{\Omega(d)} = O(\log(n) \cdot f(n))$ bits; therefore applying the extractor E, we get $\Omega(n^{\delta'})$ quasi-random bits. ∎

5.2. An Extractor for n^{γ} Min-entropy

Here we prove Lemma 5.2. We show how given an extractor for sources with $n^{1/d+\gamma}$ min-entropy, we can build an extractor for sources with $n^{1/(d+1)+\gamma}$ min-entropy. The idea is to extract t blocks. If all the blocks took their "fair share" of randomness, then they form a blockwise source, and we can treat them as before. If they do not form a blockwise source, then it must be the case that one of the blocks "stole all the randomness" present in the source. But then

this block is much more condensed, and we can extract randomness from it.

ALGORITHM 5.2. Fix $t = O(f(n))$ and $l = n^{1-1/(d+1)}$.

Choose $r_1, ..., r_t \in \{0,1\}^{O(\log(n))}$, $r', r'' \in \{0,1\}^{O(\log(n)\cdot f(n))}$ and $r'' \in \{0,1\}^{O(\log(n))}$.

Compute:

- $b_i = B(x, r_i)$ for $i = 1, ..., t$, where B is the block extraction operator of Lemma 4.4.
- $b'_i = F(b_i, r')$ for $i = 1, ..., t-1$, where F is given in the hypothesis of the lemma.
- $b'_t = E(b_1, BE(b_2, ..., b_t, r''))$, where BE is the function extracting randomness from blockwise sources of Lemma 4.3.

Let the output be $F(b'_1 \circ \cdots \circ b'_t, r''')$.

Proof of Lemma 5.2. Let us denote $B = (B_1, ..., B_t)$ and $B' = (B'_1, ..., B'_t)$. We will soon prove that

CLAIM 5.1. *B' is a $(t, k = n^{\Omega(1)}, n^{-\Omega(1)})$ somewhere random source.*

Hence, by Lemma 4.7, B' is $n^{-\Omega(1)}$-close to some $\overline{B''}$ that is distributed over $N = kt$ variables and has $H_\infty(\overline{B''}) \geqslant k$. Thus, $\overline{B''}$ is very "condensed," i.e. $H_\infty(\overline{B''}) \geqslant k \gg N^{2/3}$. Thus, by the hypothesis, $F(B', r''')$ is $n^{-\Omega(1)}$-close to the uniform distribution. ∎

Proof of Claim 5.1. First we define the selector function. Given $b \in B$ we look for a prefix i s.t. the min-entropy of $(X \mid B_{[1,i]} = b_{[1,i]})$ has dropped significantly. If such an i exists, and assume i_0 is the first such i, then it means that the i_0 block "stole" a lot of randomness, and we let $f(b) = i_0$. If no such i exists, we expect B to be a blockwise source, and we let $f(b) = t$.

Now we have to quantify what "dropped significantly" means. Initially, $H_\infty(\bar{X}) \geqslant n^{1/(d+1)+\gamma}$. Let us denote $\mu_0 = n^{1/(d+1)+\gamma}$. If no block stole randomness, at the end we expect $H_\infty(X \mid B_{[1,t]} = b_{[1,t]})$ to be at least $\mu_0/2$. So let us define, for $i = 1, ..., t-1$, $\mu_i = \mu_0 - (i/2t)\,\mu_0$, and for $b \in B$ let the selector function $f(b)$ be

$$f(b) = \begin{cases} i & \text{if } H_\infty(X \mid B_{[1,i]} = b_{[1,i]}) < \mu_i, \\ & \text{and this first happens at } i; \\ t & \text{otherwise.} \end{cases}$$

Now we correct the selector function to avoid some rare bad cases.

DEFINITION 5.1. *b is bad if either of the following two cases happen:*

- $f(b) = i \in [1 \cdots t-1]$ and $\text{Prob}(f = i \mid B_{[1,i-1]} = b_{[1,i-1]}) \leqslant \varepsilon$,
- $f(b) = t$ and there is some $1 \leqslant i \leqslant t$ s.t. $\text{Prob}(f = t \mid B_{[1,i-1]} = b_{[1,i-1]}) \leqslant \varepsilon_i$,

where $\varepsilon_t = \varepsilon = n^{-\Theta(1)}$ and $\varepsilon_{i-1} = 4\varepsilon_i$ for $i = 2, ..., t$.

Define

$$Y(b) = \begin{cases} 0, & b \text{ is bad,} \\ f(b), & \text{otherwise.} \end{cases}$$

We are going to prove several lemmas. First,

LEMMA 5.3. $\Pr(Y = 0) \leqslant t\varepsilon + \Sigma_i \varepsilon_i$.

Second, we will show that

CLAIM 5.2. *For any $1 \leqslant i \leqslant t-1$, $H_\infty(B_i \mid Y = i) \geqslant n^{1/(d+1)+\gamma/2}$.*

Therefore,

$$H_\infty(B_i \mid Y = i) \geqslant n^{(d/(d+1))(1/d)+(d/(d+1))((d+1)/d)(\gamma/2)} = l^{(1/d)+((d+1)/d)(\gamma/2)},$$

and by assumption, from such sources F extracts randomness. Hence, $(B'_i \mid Y = i)$ is $n^{-\Omega(1)}$-close to uniform.

Finally, we will show that:

CLAIM 5.3. *$(B \mid Y = t)$ is an $n^{\gamma/2}$ blockwise source to within $n^{-\Omega(1)}$.*

Therefore with the conditioning that $Y = t$,

$$\overline{B_1 \circ BE(B_1 \circ \cdots \circ B_t, r'')}$$

is $n^{-\Omega(1)}$ close to $\overline{B_1} \times U_{\log(n)\cdot 2^{\Omega(t)}}$. Hence, using the extractor E, $(B'_t \mid Y = t)$ is $n^{-\Omega(t)}$ close to uniform.

Putting it together, B' is a $(t, n^{\Omega(1)}, n^{-\Omega(1)})$ somewhere random source. ∎

So now we have to prove the above three claims. The proof does not involve any new idea and is a straightforward check that, indeed, all the necessary things hold. We first need a technical claim which we prove in Appendix E.

CLAIM 5.4. *For any $0 < i < t$*

1. *For any $b_{[1,i-1]}$ that can be extended to some b with $Y(b) = i$,*

$$\Pr(Y = i \mid B_{[1,i-1]} = b_{[1,i-1]}) = \Pr(f = i \mid B_{[1,i-1]} = b_{[1,i-1]}) \geqslant \varepsilon.$$

2. *For any $b_{[1,i-1]}$ that can be extended to some b with $y(b) = t$,*

$$\Pr(Y = t \mid B_{[1,i-1]} = b_{[1,i-1]}) \geqslant \varepsilon_{i-1} - \sum_{j=i}^{t} \varepsilon_j \geqslant \varepsilon.$$

We prove Claim 5.3 in Appendix E. Let us now prove Claim 5.2.

Proof of Claim 5.2. Let $1 \leqslant i \leqslant t-1$. Fix any prefix $b_{[1 \cdots i-1]}$ that can be extended to some b_0 with $Y(b_0)=i$. Take any b with that prefix and $Y(b)=i$.

Since $Y(b)=i$,

$$H_\infty(X \mid B_{[1,i]}=b_{[1,i]}) < \mu_i.$$

Therefore, there is some x_0 s.t.

$$\text{Prob}(X=x_0 \mid B_{[1,i]}=b_{[1,i]}) \geqslant 2^{-\mu_i}.$$

Since $Y(b) > i-1$,

$$H_\infty(X \mid B_{[1,i-1]}=b_{[1,i-1]}) \leqslant \mu_{i-1}.$$

Therefore,

$$\begin{aligned} 2^{-\mu_{i-1}} &\geqslant \text{Prob}(X=x_0 \mid B_{[1,i-1]}=b_{[1,i-1]}) \\ &\geqslant \Pr(X=x_0 \mid B_i=b_i \text{ and } B_{[1,i-1]}=b_{[1,i-1]}) \\ &\quad \times \Pr(B_i=b_i \mid B_{[1,i-1]}=b_{[1,i-1]}) \\ &\geqslant 2^{-\mu_i} \cdot \Pr(B_i=b_i \mid B_{[1,i-1]}=b_{[1,i-1]}). \end{aligned}$$

Therefore for any b with the prefix $b_{[1,i-1]}$ and $Y(b)=i$,

$$\Pr(B_i=b_i \mid B_{[1,i-1]}=b_{[1,i-1]}) \leqslant 2^{\mu_i-\mu_{i-1}} = 2^{-\mu_0/2t}.$$

Also, by Claim 5.4, $\text{Prob}(Y=i \mid B_{[1,i-1]}=b_{[1,i-1]}) \geqslant \varepsilon$. Therefore,

$$\begin{aligned} &\Pr(B_i=b_i \mid B_{[1,i-1]}=b_{[1,i-1]} \text{ and } Y=i) \\ &\quad \leqslant \frac{\text{Prob}(B_i=b_i \mid B_{[1,i-1]}=b_{[1,i-1]})}{\text{Prob}(Y=i \mid B_{[1,i-1]}=b_{[1,i-1]})} \leqslant \frac{1}{\varepsilon} \cdot 2^{-\mu_0/2t}. \end{aligned}$$

Since this holds for any prefix $b_{[1,i-1]}$, $H_\infty(B_i \mid Y=i) \geqslant \mu_0/2t - O(\log(n)) \geqslant n^{1/(d+1)+\gamma/2}$ as required. ▮

Finally, let us prove Claim 5.3.

Proof of Claim 5.3. Fix an $i \in [1 \cdots t]$. We need to show that for any prefix $b_{[1,i-1]}$ $(B_i \mid Y=t \text{ and } B_{[1,i-1]}=b_{[1,i-1]})$ is $n^{-\Omega(1)}$-close to a distribution $\bar{W}$ with $H_\infty(\bar{W}) \geqslant n^{\gamma/2}$.

Fix any $b_{[1,i-1]}$ that can be extended to some b_0 with $Y(b_0)=t$. Since $Y(b_0)=t$, no block so far "stole" too much entropy; i.e., if we denote $\bar{Z}=(X \mid B_{[1,i-1]}=b_{[1,i-1]} \text{ and } Y=t)$, then

$$H_\infty(X \mid B_{[1,i-1]}=b_{[1,i-1]}) \geqslant \mu_{i-1};$$

i.e., for any x

$$\Pr(X=x \mid B_{[1,i-1]}=b_{[1,i-1]}) \leqslant 2^{-\mu_{i-1}}.$$

Also, by Claim 5.4, $\text{Prob}(Y=t \mid B_{[1,i-1]}=b_{[1,i-1]}) \geqslant \varepsilon$. Therefore,

$$\begin{aligned} &\text{Prob}(X=x \mid B_{[1,i-1]}=b_{[1,i-1]} \text{ and } Y=t) \\ &\quad \leqslant \frac{\text{Prob}(X=x \mid B_{[1,i-1]}=b_{[1,i-1]})}{\text{Prob}(Y=t \mid B_{[1,i-1]}=b_{[1,i-1]})} \leqslant \frac{1}{\varepsilon} \cdot 2^{-\mu_{i-1}}. \end{aligned}$$

Hence, $H_\infty(\bar{Z}) \geqslant \mu_{i-1} - O(\log(n)) = \Omega(\mu_{i-1})$, which formally states that, after the first $i-1$ blocks of b, there is still a lot of min-entropy in X.

By Lemma 4.4, $(B_i \mid B_{[1,i-1]}=b_{[1,i-1]} \text{ and } Y=t)$ is $O(l^{-\Omega(1)}) = n^{-\Omega(1)}$ close to a distribution $\bar{W}$, with $H_\infty(\bar{W}) = (l/n)\,\Omega(\mu_{i-1}/\log(n)) \geqslant \Omega(n^{1-1/(d+1)} \cdot n^{1/(d+1)+\gamma}/n \text{ polylog}(n)) \geqslant \Omega(n^{\gamma/2})$. ▮

6. A SURVEY OF THE APPLICATIONS

In this section we survey the main applications of extractors and dispersers. For each application we mention the best result achieved and whether our new constructions improve the previous best result.

We will mostly use the graph view of extractors and dispersers and will continue working with the parameters used so far, $N=2^n$, $M=2^m$, $K=2^k$, $D=2^d$, and ε. In many cases the parameters of the extractor or disperser used will be derived directly from the application in mind. In these cases we will "cleverly" define the application already with the appropriate names for parameters. All theorems thus implicitly start with "let G be any extractor with parameters N, M, K, D, ε, then ...."

6.1. Simulating BPP Using Defective Random Sources

While randomized algorithms seem useful, the following question must be addressed before they can actually be used: How do computers get random bits? Currently some pseudo-random generator is used, usually without any proof that it "works." In many cases, these pseudo-random bits are not good enough as a replacement for the truly random bits needed by the algorithm.

An alternative solution is to rely on some physical source which produces some "real" randomness. A popular example of such sources are Zener diodes, which (can be made to) produce quantum mechanical noise—thus are random since their output should be impossible to predict physically.

The problem is that, although these sources contain a substantial amount of randomness, they do not output truly unbiased and independent random bits—thus they cannot be used directly in randomized algorithms.

A natural idea is to, deterministically, convert this source into truly random bits. For certain types of sources this can indeed be done. E.g., [Blu86] shows how it can be done if the source is a (known) Markov chain. For more general sources it can be shown that this cannot be done [SV86]. Instead, we may use the somewhat random source to indirectly simulate a given randomized algorithm.

The kind of simulation we have in mind is a black-box simulation; one that does not require us to understand the randomized algorithm in question. Let us define this "universal" type of simulation explicitly for randomized algorithms which have one-sided error (RP-type algorithms.)

A black-box simulation of RP. We want to run a randomize algorithm which has one-sided error and requires m random bits; we are given a random source which has distribution X on $[N]$, from which we can get a single element $x \in [N]$. From x we generate, deterministically, D different strings $z_1 \cdots z_D \in [M]$ $(M = 2^m)$ and then run the original algorithm on each z_i. We accept if the original algorithm accepts for at least one of the z_i's. The simulation is polynomial if D is polynomial (in n.)

DEFINITION 6.1. The above procedure is a black-bow simulation of RP using source X with error α if for every set $W \subseteq [M]$, $|W| \geqslant M/2$, $\Pr_x[\exists i: z_i \in W] \geqslant 1 - \alpha$.

If the original *RP* algorithm accepts then for at least half the possible $z \in [M]$ it accepts—call this set W. In this case the simulation, using X, accepts with probability at least $1 - \alpha$. If, on the other hand, the original algorithm rejects, then it rejects for all $z \in [M]$ (since the error is one-sided), and thus, the simulation will reject. The analogous procedure for simulating *BPP*, does not accept whenever one of the z_i's causes the original algorithm to accept but rather accepts according to some other condition which depends on the answers obtained for all z_i's—usually their majority vote.

Our usual goal would be to obtain this simulation without knowing the exact distribution of the somewhat random source X. Our simulation should only rely on the fact that X has some given property, which is the model of "somewhat randomness." Clearly, physicists should try to produce physical source where this property is as strong as possible, while computer scientists should aim to rely on a property which is as week as possible.

Several models of "somewhat random" sources were considered in the literature. Santha and Vazirani [SV86], Vazirani [Vaz87a, Vaz86, Vaz87b], and Vazirani and Vazirani [VV85] studied a class of sources, called "slightly random sources," and showed that *BPP* can be simulated, given such a source. Chor and Goldreich [CG88] generalized this model and showed that *BPP* can be simulated even using the more general source. Many authors studied other restricted classes of random sources (e.g., [CW89, CGH+85, LLS89]). Finally, Zuckerman [Zuc90] suggested a general model generalizing all the previous models. His model was simply all random sources having high min-entropy. Later [Zuc91] he showed that *BPP* can be simulated given such a source (with enough min-entropy).

In a very basic sense, Zuckerman's model is the most general source we can think off.

LEMMA 6.1. *If a black box RP simulation requiring m bits can be obtained, with error α, using source X, then X is $O(\alpha)$-close to some distribution X' with $H_\infty(X') \geqslant m - \log D - 1$.*

Proof (Sketch). Consider the set S of all the elements x which get a probability higher than $2^m/(2D)$. This total probability cannot be greater than α since otherwise we can have W be the set of all elements z which are never produced by an element of S, a set which has size $|W| \geqslant M - |S| \cdot D \geqslant M/2$. The distribution X' is obtained by zeroing the probability of elements in S and correcting the total probability to 1. ∎

Using extractors a simulation good for all sources with enough min-entropy is easily obtained.

THEOREM. *RP (and BPP) with m bits can be simulated in time D using any random source X with $H_\infty(X) \geqslant k$. (The error is $\exp(k - H_\infty(X))$.)*

Proof for RP. Get x from the distribution, and set $z_i = G(x, i)$ for $i = 1 \cdots D$, where G is a disperser with $\varepsilon < 1/2$. Let us calculate the probability that this simulation fails, i.e. that all $z_i \notin W$. Let us look at the set S of these "bad" x's—it cannot be larger than K, since the neighbor set of any set of size K is larger than $M/2$ and thus intersects W. The total probability assigned by X to this set of "bad" x's can be at most $|S| \cdot 2^{-H_\infty(X)}$. ∎

A similar argument for BPP uses an extractor.

With current constructions, we get a polynomial time (in n) simulation (i.e., $D = \text{poly}(n)$) for *RP* as long as $H_\infty(X) \geqslant n^\delta$ for any $\delta > 0$ [SSZ95]; a polynomial time simulation for *BPP* as long as $H_\infty(X) = \Omega(n)$ (or even slightly less) [Zuc93]; and using our new results presented in Section 5, a slightly quasi-polynomial ($n^{\log^{(c)} n}$ for any constant c) time simulation for *BPP* for any $H_\infty(X)$:

COROLLARY 6.2. *For any $\delta > \underbrace{0}_{k}$ and constant $k > 0$, BPP can be simulated in time $n^{O(\log \log \cdots \log n)}$ using a weak random source X with minentropy at least n^δ.*[6]

6.2. Deterministic Amplification

It turns out that extractors and dispersers can be used to decrease the probability of error of randomized algorithms

[6] Previous result [SZ94] was for $\delta > 1/2$ and required $n^{O(\log(n))}$ time.

in several settings. We describe some of these settings in this subsection.

6.2.1. *Basic Amplification*

Our goal now is to convert a *BPP* (or *RP*) algorithm that uses m random bits and has error, say 1/4, into one that errs with probability at most 2^{-t}. A trivial solution is to run the original algorithm $O(t)$ times using independent random bits and take the majority vote—this requires $O(tm)$ random bits. We want to achieve this using as few random bits as possible.

This problem, known as the "deterministic amplification" problem, was extensively studied [CG89, IZ89, CW89]. Using expanders, this can be done using only $n+O(t)$ random bits [AKS87, IZ89, CW89]. Sipser [Sip88] defined dispersers as a tool which implies stronger *RP* amplification. Using extractors, we obtain *BPP* amplification.

THEOREM. *If L is accepted by a BPP algorithm using m random bits and error* 1/4, *then L is also accepted by a BPP algorithm using n random bits and having error K/N.*

Proof. Use an extractor G with $\varepsilon < 1/4$.

THE ALGORITHM. Choose randomly $x \in [N]$. For any $z \in \Gamma(x)$ run the machine accepting L with z as the random string and decide according to the majority of the results.

Correctness. Denote $W \subseteq [M]$ the set of witnesses z leading to the wrong answer by the machine accepting L. Thus $|W| \leqslant M/4$. Let $B \subseteq [N]$ be the set of "bad" x's—those with most of their neighbors in W. The simulation arrives at the wrong result iff a "bad" x is chosen. Since G is an extractor, $|B| < K$, since otherwise a uniform distribution on B would, on one hand have min-entropy k, but on the other hand $\Gamma(B)$ is far from uniform, since it gives weight of at least 1/2 to a set W of size at most 1/4. ∎

It can easily be seen that if we want to amplify an *RP* algorithm we can use dispersers instead of extractors.

Using current constructions we can use only $n = (1+\alpha)(m+t)$ bits to get error 2^{-t} for any fixed $\alpha > 0$ [Zuc93, Zuc]. Using the construction of Section 4, we can use only $m+t$ random bits, but the running time becomes quasi-polynomial.

6.2.2. *Oblivious Sampling*

The above simulation may be generalized to give what is called an oblivious sampler. These are sampling procedures that give a good estimate for the expected value of a real-valued function on some finite domain.

DEFINITION 6.2 [BR94]. An oblivious (δ, ε)-sampler is a deterministic function that for each $x \in [N]$ produces a set $\Gamma(x) = \{z_1 \cdots z_D\} \subseteq [M]$ such that for every function $f: [M] \to [0, 1]$ (where $[0, 1]$ is the real interval between 0 and 1),

$$\Pr_x \left[\left| \frac{\sum_{i \in [D]} f(z_i)}{D} - \frac{\sum_{z \in [M]} f(z)}{M} \right| \geqslant \varepsilon \right] \leqslant \delta.$$

For given ε, δ, and m, our wish is to have D be polynomial and n be as small as possible. Oblivious samplers were constructed in [BR94] who used them for interactive proof systems. The best known results to date, which use extractors, appears in [Zuc].

THEOREM. *There exist explicitly constructible (δ, ε) oblivious samplers, where $\delta = 2K/N$.*

Using the best constructions of extractors [Zuc], we get oblivious samplers with $n = (1+\alpha)(m + \log \delta^{-1})$ and $D = \text{poly}(m, \varepsilon^{-1}, \log \delta^{-1})$ (for any $\alpha > 0$).

Proof. Take an extractor G and use the $\Gamma(x)$ from the extractor. Denote the expected value of f by $e = (\sum_{z \in [M]} f(z))/M$. The proof follows by considering the set $S_<$ of x's such that $(\sum_{z \in \Gamma(x)} f(z))/D < e - \varepsilon$, and separately $S_>$, the set of x's such that $(\sum_{z \in \Gamma(x)} f(z))/D > e + \varepsilon$. These sets must each be of size less than K, since otherwise the uniform distribution on one of them has high enough min-entropy and this gives near (to within ε) uniform distribution on the z's. This cannot be true since an ε-close to uniform distribution on the z's gives an ε-close estimate of e, in contradiction to each element in $S_<$ (resp, $S_>$), erring on e by at least ε. ∎

6.2.3. *Approximating Clique*

Zuckerman [Zuc93] showed how deterministic amplification obtained by extractors implies that

THEOREM [Zuc93]. *Approximating* $\log(\text{Clique}(G))$ *to within a constant factor is $\tilde{N}P$-hard.*

In fact Zuckerman showed that for any constant j approximating the jth iterated log of clique to within any constant is $\tilde{N}P$-hard. For the precise statement see [Zuc93, SZ94]. Our next construction can derandomize Zuckerman's result.

The proof itself builds on the known hardness results for approximating clique [AS92b, ALM+92, FGL+91] and is best understood when viewing it as deterministic amplification for *PCP* system.

6.2.4. *Time versus Space*

Sipser [Sip88] defined dispersers in order to obtain the following theorem (which became a theorem only with the recent constructions of [SSZ95]). Again, the dispersers are needed for deterministic amplification. The [SSZ95]

dispersers are good enough, and our new constructions do not improve this result.

THEOREM. *If $P \neq RP$ then there is some $0 < \alpha$ s.t. for any (nice) function $t = t(n)$, $\text{Time}(t) \subseteq \text{ioSpace}(t^{1-\alpha})$, where $\text{ioSpace}(s)$ is the class of languages solvable by algorithms that for infinitely many inputs use at most space s.*

This is an unexpected connection between the question of the power of randomness and the (seemingly unrelated) question of time versus space.

6.3. Explicit Graphs with Random Properties

In this section we present some explicit constructions of certain kinds of graphs. In these cases, the "random-like" properties of extractors and dispersers suffice as a replacement for using random graphs and, thus, convert a nonconstructive proof to a construction.

6.3.1. *Super Concentrators*

DEFINITION 6.3 [Pip77]. Let $H = (V, E)$ be a directed graph with a specified subset $I \subset V$ of nodes called input nodes and a disjoint subset, $O \subseteq V$, called output nodes. Assume $|I| = |O| = N$. H is called a superconcentrator if for any sets $W \subseteq I$, $Z \subseteq O$ of size K each there are at least K vertex-disjoint paths from W to Z.

The parameters of interest are, first, the *size* which is the number of edges in H, and second, the *depth* which is the length of the longest directed path in it.

Gabber and Galil [GG81] constructed, using expanders, the first explicit linear size superconcentrators. The graph they construct has $O(\log(N))$ depth. For depth 2, nonconstructive proofs show that superconcentrators of depth 2 and size $O(N \cdot \log^2(N))$ exists [Pip82]. Meshulam [Mes84] showed an explicit depth 2, superconcentrator of size $O(N^{1+1/2})$; [WZ93] give a construction based on extractors.

THEOREM (Following [WZ93]). *There is an explicit superconcentrator of depth 2 and size $N \cdot O(\sum_{k=1}^{n} D_k 2^k / M_k)$, where D_k, M_k are the parameters for dispersers with $K = 2^k$.*

The work done in Section 4 implies depth 2 superconcentrators of size $N \cdot 2^{\text{polylog}(N)}$.

Proof. Meshulam [Mes84] showed that H is a superconcentrator of depth 2, iff for any $1 \leqslant K' \leqslant N$ and any two sets $W \subseteq I$, $Z \subseteq O$ of size K' each there are at least K' common neighbors. We will build a depth-2 graph with this property.

Build the graph as follows:

- I and O each have $N = 2^n$ vertices.
- The other vertices are partitioned into disjoint sets C_k for $1 \leqslant k \leqslant n$; $|C_k| = 4K$.
- For each k, we partition C_k into $4K/M_k$ equal-sized sets each of size M_k and put a disperser between I and each set, as well as between O and each set. (We take a disperser with $\varepsilon = 1/4$.)

Now take any two sets $W \subseteq I$ and $Z \subseteq O$ of size K' each. Let $K = 2^k$ be the largest power of 2 which is less than or equal to K'. Consider only edges connecting W and Z to C_k. In each one of the $4K/M_k$ different subsets of C_k, we have that $\Gamma(W), \Gamma(Z) \geqslant 3M_k/4$ (since we put a disperser there). It follows that in each of these subsets, $|\Gamma(W) \cap \Gamma(Z)| \geqslant M_k/2$. Taking all $4K/M_k$ subsets together, the intersection size is at least $2K \geqslant K'$. ∎

Using the extractor of Section 4,

COROLLARY 6.3. *For every N there is an efficiently constructible depth-2 superconcentrator over N vertices with size $O(N \cdot 2^{\text{polyloglog}(N)})$.*[7]

Wigderson and Zuckerman showed that such a construction would also imply an explicit linear size super concentrator of small depth, with the results of Section 4, of depth polyloglog(N).

COROLLARY 6.4. *For any N there is an explicitly constructible superconcentrator over N vertices, with linear size and* polyloglog(N) *depth.*[8]

6.3.2. *Highly Expanding Graphs*

We now consider expanders with very strong expansion properties.

DEFINITION 6.4. A graph H on N vertices is called a K-expander if any two sets W, Z of vertices, $|W| = |Z| = K$, have a common neighbor.

Since any two sets of size K have a common neighbor, a set of size K must have at least $N - K$ neighbors and, therefore, the degree of the graph is at least $(N - K)/K$. We would like to explicitly build K-expanding graphs with degree as close as possible to N/K. The eigenvalue methods for constructing expanders give such expanders for $K \geqslant \sqrt{N}$, but they do not give anything for smaller values of K; [WZ93] show how dispersers can be used to construct expanders with small values of K.

THEOREM (Following [WZ93]). *There are constructible K-expanding graph with N vertices and maximum degree $O(ND^2/M)$.*

Using the extractors of Section 4 we get maximum degree of $(N/K)\exp(\text{polyloglog } N)$ for any value of K. Let us remark that the construction takes time polynomial in N. It

[7] This improves the current upper bound of $O(N \cdot 2^{\log(N)^{1/2+o(1)}})$, achieved using the [WZ93] technique and the [SZ94] extractor.

[8] This improves the current upper bound of $O(\log(N)^{1/2+o(1)})$.

does not allow computing whether an edge exists between two given vertices in time polynomial in n.

Proof. We use a disperser G with $\varepsilon = 1/5$. Let $B \subseteq [M]$ be the set of z's of degree greater than $2DN/M$. Notice that $|B| \leqslant M/2$, since the total number of edges in the disperser is DN. Now we connect a vertex $x_1 \in [N]$ to a vertex $x_2 \in [N]$ if they share a neighbor not in B, i.e. if there exist $y_1, y_2 \in [D]$ such that $G(x_1, y_1) = G(x_2, y_2) \notin B$.

Now for every set $W \subseteq [N]$, $|W| \geqslant K$ we have, in the disperser, $\Gamma(W) \geqslant 4M/5$, and thus $|\Gamma(W) \cap \Gamma(Z)| \geqslant 3M/5 \geqslant |B|$. Thus, in the expander we have just built, W and Z will share a neighbor. The number of edges is bounded from above by $D \cdot 2DN/M$. ▮

Using the extractor of Section 4,

COROLLARY 6.5. *For any N and $1 \leqslant a \leqslant N$ there is an explicitly constructible a-expanding graph with N vertices and maximum degree $O((N/a)\, 2^{\text{polyloglog}(N)})$.*[9]

Pippenger [Pip87b] showed that good explicit highly expanding graphs yield good algorithms for "sorting in rounds" and for "selecting in rounds."

6.4. Pseudo-Random Generators

Extractors can also be used to construct pseudo-random generators which fool certain classes of algorithms. The generator can use any good extractor for high min-entropies, and our new constructions do not improve its operation. The following theorem of [NZ93] improves on previous results of [AKS87].

THEOREM [NZ93]. *There exits a pseudo-random generator which converts $O(S)$ truly random bits into* poly(S) *bits which look random to any algorithm which runs in space S. The generator runs in $O(S)$ space and* poly(S) *time.*

Proof (Sketch). We will build a generator that for any t converts $n + td$ bits into tm bits which look random (to within ε) to space $S = n - k - O(\log \varepsilon^{-1})$ algorithms. Current extractors G with $k, m = \Omega(n)$ imply for all $t \geqslant S$ conversion of $O(t)$ bits to $\Omega(t \cdot S/\log S)$ bits. To get any poly(S) factor gain in the number of bits simply iterate (i.e. compose the pseudo-random generators).

THE PSEUDO-RANDOM GENERATOR.

Input: $x, y_1 \cdots y_t$.
Output: $G(x, y_1), ..., G(x, y_t)$.

for each $i = 0 \cdots t$, consider the probability distribution A_i on the internal state of the algorithm after im random bits $z_1 \cdots z_i$ have been read by the algorithm versus the distribution A'_i in the case that pseudo-random bits $G(x, y_1) \cdots G(x, y_i)$ have been read. The proof that this generator indeed looks random to space S algorithms proceeds by induction on i of the following claim.

CLAIM. *The distribution A_i is close to the distribution A'_i.*

Proof. Fix a typical internal state a of the algorithm. We will show that, conditioned on a being the state reached after $(i-1)\, m$ bits were read, A_i and A'_i are close. The claim will follow by averaging over all a's and noticing that the probabilities of getting to a, in A_{i-1} versus A'_{i-1}, are close by the induction hypothesis.

Let us see what is the conditional distribution on X, conditioned upon a being the state reached after $(i-1)m$ bits were read. The space bound S on the algorithm implies that there are at most 2^S different possible values of a; thus we would expect that the probability of getting to a is about 2^{-S} (otherwise a can be ignored). Looking at the conditional probabilities we have that $H_\infty(X \mid a) \geqslant n - S \geqslant k$. Therefore, the distribution of $g(x, y_i)$ is nearly uniform under this conditioning, since G is an extractor. The distribution A_i and A'_i under this conditioning, are completely determined by z and $G(x, y_i)$, respectively and thus are close to each other. ▮

APPENDIX

A. A Somewhere Random Source Has Large Min-Entropy

LEMMA A.1 *If $X = X_1 \circ \cdots \circ X_b$ is a (k, ε, η) somewhere random source, then X is η-close to a $(k, \varepsilon, 0)$ somewhere random source X'.*

Proof of Lemma A.1. Let Y be a (k, ε, η) selector for X. Denote $p = \text{Prob}(Y = 0) \leqslant \eta$. Define the distribution $\bar{D}$ by:

$$\bar{D}(i, x) = \begin{cases} 0 & \text{if } i = 0, \\ \dfrac{\text{Prob}((Y, X) = (i, x))}{1 - p}, & \text{otherwise.} \end{cases}$$

It is easy to see that $\bar{D}$ is a distribution. Define the random variable $Y' \circ X'$ as the result of choosing (i, x) uniformly from $\bar{D}$, i.e. $\overline{Y' \circ X'} = \bar{D}$. It is clear that $d(\bar{X}, \bar{X}) \leqslant d(\overline{Y \circ X}, \overline{Y' \circ X'}) = p \leqslant \eta$.

Now we want to show that Y' is a $(k, \varepsilon, 0)$ selector for X'. It is clear that $\text{Prob}(Y' = 0) = 0$. It is not hard to see that for any $i > 0$ we have $\text{Prob}(X' = x \mid Y' = i) = \text{Prob}(X = x \mid Y = i)$. Therefore, since we know that $(X_i \mid Y = i)$ is ε-close to U_k, we also know that $(X'_i \mid Y' = i)$ is ε close to U_k, thus completing the proof. ▮

LEMMA A.2. *Let $X = X_1 \circ \cdots \circ X_b$ be a $(k, \varepsilon, 0)$-somewhere random source, then X is ε close to a $(k, 0, 0)$-somewhere random source Z.*

Proof of Lemma A.2. Let Y be a $(k, \varepsilon, 0)$ selector for X. Fix some $i \in [1 \cdots b]$. We know that $d((x_i \mid Y = i), U_k) \leqslant \varepsilon$. Define a distribution $Z^{(i)}$ by

[9] The obvious lower bound is N/a. The previous upper bound [WZ93, SZ94] was $O(N/a)$, $2^{\log(N)^{1/2 + O(1)}}$).

$$Z^{(i)}(x) = \begin{cases} \frac{1}{2^k} \cdot \text{Prob}(X = x \mid X_i = x_i \text{ and } i), \\ \qquad \text{if} \quad \text{Prob}(X_i = x_i \text{ and } Y = i) > 0, \\ \frac{1}{2^k} \cdot 1, \qquad \text{if} \quad \text{Prob}(X_i = x_i \text{ and } Y = i) = 0 \\ \qquad \text{and for every } j \neq i\text{: } x_j = 0^k, \\ 0, \qquad \text{otherwise.} \end{cases}$$

It is easy to check that $Z^{(i)}$ is indeed a distribution and that $Z_i^{(i)} = U_k$. Define $Y \circ Z$ to be the random variable obtained by choosing i according to Y; then choosing z according to $Z^{(i)}$, i.e. for all $i > 0$, $(Z \mid Y = i) = Z^{(i)}$. Also, denote $X^{(i)} = (X \mid Y = i)$. Then

$$\text{Prob}(Z_i = z_i \mid Y = i) = Z_i^{(i)}(z_i) = 2^{-k}.$$

We will soon prove that

CLAIM A.1. $d(X^{(i)}, Z^{(i)}) \leqslant \varepsilon$.

Thus

$$\begin{aligned} d(\bar{X}, \bar{Z}) &\leqslant (\overline{Y \circ X}, \overline{Y \circ Z}) \\ &= \sum_{i>0} \text{Pr}(Y = i) \cdot d((X \mid Y = i), (Z \mid Y = i)) \\ &= \sum_{i>0} \text{Pr}(Y = i) \cdot d(X^{(i)}, Z^{(i)}) \leqslant \varepsilon. \end{aligned}$$

Hence Z satisfies the requirements of the lemma. ▮

Proof of Claim A.1. We need to show that for any $A \subseteq \Lambda_X$, $|X^{(i)}(A)) - Z^{(i)}(A)| \leqslant \varepsilon$. It is sufficient to show this for the set A containing all $x \in \Lambda_X$ s.t. $X^{(i)}(x) > Z^{(i)}(x)$. This can be easily seen, using the fact that for any $x \in A$; $\text{Pr}(Z = x \mid Z_i = a_i \text{ and } Y = i) = \text{Pr}(Z = x \mid Y = i)/\text{Pr}(Z_i = a_i \mid Y = i) = \text{Pr}(X = x \mid X_i = a_i \text{ and } Y = i)$. ▮

LEMMA A.3. *Let* $X = X_1 \circ \cdots \circ X_b$ *be a* $(k, 0, 0)$ *somewhere random source*; *then* $H_\infty(\bar{X}) \geqslant k$.

Proof. Suppose Y is a $(k, 0, 0)$ selector for X:

$$\begin{aligned} \text{Prob}(X = x) &= \sum_{i \in [1 \cdots b]} \text{Prob}(Y = i) \cdot \text{Prob}(X_i = x_i \mid Y = i) \\ &\leqslant \sum_{i \in [1 \cdots b]} \text{Prob}(Y = i) \cdot 2^{-k} = 2^{-k}. \quad \blacksquare \end{aligned}$$

Combining Lemmas A.1, A.2, and A.3, we get Lemma 4.7. ▮

B. A Lemma for *b*-Block Mergers

Proof of Lemma 4.10. We define random variables Y', $A' \circ B'$ as follows:

- Choose $Y' = i \in S_1 \cup S_2$ with $\text{Pr}(Y' = i) = \text{Pr}(Y = i \mid Y \in S_1 \cup S_2)$.
- Choose $a' \circ b' \in (A \circ B \mid Y = i)$.

It is easy to prove that:

CLAIM. $\text{Pr}(A' = a' \mid Y' = i) = \text{Pr}(A = a' \mid Y = i)$ *and* $\text{Pr}(B' = b' \mid Y' = i) = \text{Pr}(B = b' \mid Y = i)$.

Define

$$Z' = \begin{cases} 1, & \text{if } Y' \in S_1, \\ 2, & \text{otherwise, i.e. } Y' \in S_2. \end{cases}$$

It is not hard to see that

CLAIM B.1. $(A' \mid Z' = 1) = (A \mid Y \in S_1)$ *and* $(B' \mid Z' = 2) = (B \mid Y \in S_2)$.

Hence, Z' is a $(k, \varepsilon, 0)$ selector for $A' \circ B'$. Therefore by Lemma 4.7, $\overline{A' \circ B'}$ is ε-close to some $\bar{X}$ with $H_\infty(\bar{X}) \geqslant k$. However, it is not hard to see that

CLAIM. $\overline{A' \circ B'} = (A \circ B \mid Y \in S_1 \cup S_2)$.

Thus, $(A \circ B \mid Y \in S_1 \cup S_2) = \overline{A' \circ B'}$ is ε-close to some $\bar{X}$ with $H_\infty(\bar{X}) \geqslant k$. ▮

C. Lemmas for Composing Two Extractors

In this section we prove some easy technical lemmas used in Section 4.3.

CLAIM C.1. *For any* i *and any* $w_{[1, i-1]}$, *if* $\text{Prob}_{x \in X}(Y(x) = i \mid x_{[1, i-1]} = w_{[1, i-1]}) > 0)$ *then* $\text{Prob}_{x \in X}(Y(x) = i \mid x_{[1, i-1]} = w_{[1, i-1]}) \geqslant \varepsilon_2 - \varepsilon_3$.

Proof. Since $w_{[1, i-1]}$ can be extended to some w with $Y(w) = i \neq 0$, by Definition 4.5,

$$\text{Prob}(f(x) = i) \geqslant \varepsilon_1,$$

and

$$\text{Prob}(f(x) = i \mid x_{[1, i-1]} = w_{[1, i-1]}) \geqslant \varepsilon_2.$$

However, this implies that for *any* extension w' of $w_{[1, i-1]}$ with $f(w') = i$, it holds that $w' \notin B_1 \cup B_2$. Hence,

$$\begin{aligned} &\text{Prob}(Y(x) = i \mid x_{[1, i-1]} = w_{[1, i-1]}) \\ &\quad = \text{Prob}(f(x) = i \mid x_{[1, i-1]} = w_{1, i-1}) \\ &\qquad - \text{Prob}(f(x) = i \text{ and } x \in B \mid x_{[1, i-1]} = w_{[1, i-1]}) \\ &\quad = \text{Prob}(f(x) = i \mid x_{[1, i-1]} = w_{[1, i-1]}) \\ &\qquad - \text{Prob}(f(x) = i \text{ and } x \in B_3 \mid x_{[1, i-1]} = w_{[1, i-1]}) \\ &\quad \geqslant \varepsilon_2 - \varepsilon_3. \end{aligned}$$

The last inequality uses Claim C.3. ▮

CLAIM C.2. *For any i, if* $\text{Prob}_{x\in X}(Y(x)=i)>0$, *then* $\text{Prob}_{x\in X}(Y(x)=i)\geqslant\varepsilon_1-\varepsilon_2-\varepsilon_3$.

Proof. Since there is some w' s.t. $y(w')=i\neq 0$, by Definition 4.5,

$$\text{Prob}(f(x)=i)\geqslant\varepsilon_1.$$

This implies that for *any* w' with $f(w')=i$, we know that $w'\notin B_1$. Hence

$$\begin{aligned}&\text{Prob}(Y(x)=i)\\&\quad=\text{Prob}(f(x)=i)-\text{Prob}(f(x)=i\text{ and }x\in B)\\&\quad\geqslant\text{Prob}(f(x)=i)-\text{Prob}(f(x)=i\text{ and }x\in B_2)\\&\qquad-\text{Prob}(f(x)=i\text{ and }x\in B_3)\\&\quad\geqslant\varepsilon_1-\varepsilon_2-\varepsilon_3.\end{aligned}$$

The last inequality uses Claim C.3. ∎

CLAIM C.3. 1. *For any i:* $\text{Prob}(f(x)=i\text{ and }x\in B_2)\leqslant\varepsilon_2$.

2. *For any i and $w_{[1,i-1]}$:* $\text{Prob}(f(x)=i\text{ and }x\in B_3\mid x_{[1,i-1]}=w_{[1,i-1]})\leqslant\varepsilon_3$.

3. *For any i:* $\text{Prob}(f(x)=i\text{ and }x\in B_3)\leqslant\varepsilon_3$.

4. $\text{Prob}(x\in B_i)\leqslant n\varepsilon_i$ *for* $i=1,2,3$.

Proof. (1) If for some $w_{[1,i-1]}$ $\text{Prob}(f(x)=i\text{ and }x\in B_2\mid x_{[1,i-1]}=w_{[1,i-1]})>0$ then there is an extension w of $w_{[1,i-1]}$ s.t. $f(w)=i$ and $w\in B_2$, and therefore, $\text{Prob}(f(x)=i\mid X_{[1,i-1]}=w_{[1,i-1]})\leqslant\varepsilon_2$. Thus, for all $w_{[1,i-1]}$, $\text{Prob}(f(x)=i\text{ and }x\in B_2\mid x_{[1,i-1]}=w_{[1,i-1]})\leqslant\varepsilon_2$. Therefore, $\text{Prob}(f(x)=i\text{ and }x\in B_2)=\sum_{w_{[1,i-1]}}\text{prob}(x_{[1,i-1]}=w_{[1,i-1]})\cdot\text{Prob}(f(x)=i\text{ and }x\in B_2\mid x_{[1,i-1]}=w_{[1,i-1]})\leqslant\sum_{w_{[1,i-1]}}\text{Prob}(x_{[1,i-1]}=w_{[1,i-1]})\cdot\varepsilon_2\leqslant\varepsilon_2$.

(2) If for some $w_{[1,i-1]}$ $\text{Prob}(f(x)=i\text{ and }x\in B_3\mid x_{[1,i-1]}=w_{[1,i-1]})>0$ then there is an extension w of $w_{[1,i-1]}$ s.t. $f(w)=i$ and $w\in B_3$, and therefore, $\text{Prob}(x_i=w_i\mid x_{[1,i-1]}=w_{[1,i-1]})\leqslant\varepsilon_3$. In particular, $\text{Prob}(x\in B_3\mid x_{[1,i-1]}=w_{[1,i-1]})\leqslant\text{Prob}(x_i=w_i\mid x_{[1,i-1]}=w_{[1,i-1]})\leqslant\varepsilon_3$. Thus, for all $w_{[1,i-1]}$, $\text{Prob}(f(x)=i\text{ and }x\in B_3\mid x_{[1,i-1]}=w_{[1,i-1]})\leqslant\varepsilon_3$.

(3) $\text{Prob}(f(x)=i\text{ and }x\in B_3)\leqslant\sum_{w_{[1,i-1]}}\text{Prob}(x_{[1,i-1]}=w_{[1,i-1]})\cdot\text{Prob}(f(x)=i\text{ and }x\in B_3\mid x_{[1,i-1]}=w_{[1,i-1]})\leqslant\sum_{w_{[1,i-1]}}\text{Prob}(x_{[1,i-1]}=w_{[1,i-1]})\cdot\varepsilon_3\leqslant\varepsilon_3$.

(4) The case $i=2$ follows (1) since, $\text{Prob}(x\in B_2)\leqslant\sum_{i=1}^{n}\text{Prob}(x\in B_2\text{ and }f(x)=i)\leqslant n\varepsilon_2$. Similarly for $i=3$. As for $i=1$, if there is an x with $f(x)=i$ and $x\in B_1$ then $\text{Prob}(f(x)=i)\leqslant\varepsilon_1$. Thus, $\text{Prob}(x\in B_1\text{ and }f(x)=i)\leqslant\varepsilon_1$, and $\text{Prob}(x\in B_1)\leqslant\sum_{i=1}^{n}\text{Prob}(x\in B_1\text{ and }f(x)=i)\leqslant n\varepsilon_1$. ∎

D. More Bits Using the Same Extractor

In this section we prove Lemmas 4.11 and 4.12.

Proof of Lemma 4.11. Denote by A_i the random variable with value $E(X,R_i)$. Denote by $A_{[1,i]}=A_1\circ\cdots\circ A_i$ the random variable whose value is $E(X,R_1)\circ\cdots\circ E(X,R_i)$, and let $l_i=|A_{[1,i]}|$.

DEFINITION D.1. We say that $a_{[1,i]}$ is "s-tiny" if $\text{Prob}(A_{[1,i]}=a_{[1,i]})\leqslant 2^{-l_i-s}$.

CLAIM. *For any* $1\leqslant i\leqslant t$, $\text{Prob}(a_{[1,i]}\text{ is }s-\text{tiny})\leqslant 2^{-s}$.

Proof. $A_{[1,i]}$ can have at most 2^{l_i} possible values, and each tiny value has probability at most 2^{l_i-s}. ∎

CLAIM. *For any prefix $a_{[1,i]}$ that is not s-tiny,* $H_\infty(X\mid A_{[1,i]}=a_{[1,i]})\geqslant K-l_i-s$.

Proof. For any x,

$$\begin{aligned}\text{Prob}(X=x\mid A_{[1,i]}=a_{[1,i]})&\leqslant\frac{\text{Prob}(X=x)}{\text{Prob}(A_{[1,i]}=a_{[1,i]})}\\&\leqslant\frac{2^{-K}}{2^{-l_i-s}}=2^{-K+l_i+s}.\end{aligned}$$ ∎

CLAIM. *If $l_{i-1}\leqslant K-k-s$, then $\overline{A_{[1,i]}}$ is $i(2^{-s}+\varepsilon)$ quasi random.*

Proof. By induction on i. For $i=1$ this follows from the properties of E. Assume for i, and let us prove for $i+1$.

Since $l_i\leqslant K-k-s$, then for any prefix $a_{[1,i]}$ that is not s-tiny, $H_\infty(X\mid A_{[1,i]}=a_{[1,i]})\geqslant K-l_i-s\geqslant k$. Therefore, for any nontiny prefix $_{[1,i]}$, $(A_{i+1}\mid A_{[1,i]}=a_{[1,i]})$ is ε quasi-random. Therefore by Lemma 4.2, $\overline{A_{[1,i+1]}}$ is $2^{-s}+\varepsilon$ close to the distribution $\overline{A_{[1,i]}}\times U$, and by induction $\overline{A_{[1,i+1]}}$ is $(i+1)(2^{-s}+\varepsilon)$ quasi-random. ∎

Therefore, if we take t s.t. $l_t\leqslant K-k$, we invest td random bits, and we get tm bits that are $t(2^{-s}+\varepsilon)$ quasi-random, as required. ∎

Proof of Lemma 4.12. Define $E(x,r_1\circ\cdots\circ r_t)=E_{k_1}(x,r_1)\circ\cdots\circ E_{k_t}(x,r_t)$, where $s=d(n)$, $l_0=0$, $k_i=k-l_{i-1}-s$, and $l_i=l_{i-1}+k_i/f(n)$. Denote by A_i the random variable $E_{k_i}(X,R_i)$, and let $A_{[1,i]}=A_1\circ\cdots\circ A_i$. Intuitively, $l_i=|A_{[1,i]}|$, and k_i is the amount of min-entropy left in $(X\mid A_{[1,i]}=a_{[1,i]})$ with the safety parameter $s=d(n)$.

CLAIM. *If $k_i\geqslant\bar{k}$ then $\overline{A_{[1,i]}}$ is $i(2^{-s}+\varepsilon)$ quasi-random.*

Proof. By induction on i. For $i=1$ this follows from the properties of E. Assume for i, and let us prove for $i+1$.

For any prefix $a_{[1,i]}$ that is not s-tiny, $H_\infty(X\mid A_{[1,i]}=a_{[1,i]})\geqslant k-l_i-s=k_{i+1}\geqslant\bar{k}$. Therefore, for any nontiny prefix $a_{[1,i]}$, $(A_{i+1}\mid A_{[1,i]}=a_{[1,i]})$ is ε quasi-random. Therefore, by Lemma 4.2, $\overline{A_{[1,i+1]}}$ is $2^{-s}+\varepsilon$ close to the

distribution $\overline{A_{[1,i]}} \times U$, and by induction $\overline{A_{[1,i+1]}}$ is $(i+1)(2^{-s}+\varepsilon)$ quasi-random. ▮

How big do we need t to be? Let us denote $q_i = k - l_i$; i.e., q_i is the number of bits still missing. Notice that $q_i = k - l_i = k - (l_{i-1} + k_i/f(n)) = q_{i-1} - k_i/f(n) = q_{i-1} - q_{i-1} - d(n)/f(n)$. Therefore, if $(q_{i-1}/2 \geqslant d(n)$, then $q_i \leqslant (1 - 1/2f(n))\, q_{i-1}$. Thus, after $O(f(n)\log(n))$ steps, either $q_{i-1} \leqslant 2d(n)$, or else $k_i \leqslant \bar{k}$. In the first case, $q_{i-1} \leqslant 2d(n)$, and we can fill all the $2d(n)$ missing bits with a truly random string. In the second case, $k_i \leqslant \bar{k}$, i.e., $q_{i-1} \leqslant \bar{k} + s$, so if we add $s = d(n)$ truly random bits, there are only $\bar{k}$ missing bits as required.

Therefore it is sufficient to take $t = O(f(n)\log(n))$, and let the final extractor be $E(x, r) \circ y$ where y is of length $2d(n)$ and is truly random. ▮

E. Lemmas for the Second Extractor

In this section we prove some easy technical lemmas used in Section 5. Let us start with the proof of Claim 5.4.

Proof of Claim 5.4. Proof of (1). Since $b_{[1,i-1]}$ can be extended to some b with $Y(b) = i$, any extension b' of $b_{[1,i-1]}$ with $f(b') = i$ is not bad. Therefore,

$$\begin{aligned}&\Pr(Y = i \mid B_{[1,i-1]} = b_{[1,i-1]})\\ &\quad = \Pr(f = i \mid B_{[1,i-1]} = b_{[1,i-1]}).\end{aligned}$$

Also, since b is not bad,

$$\Pr(f = i \mid B_{[1,i-1]} = b_{[1,i-1]}) > \varepsilon$$

and this completes the proof of (1).

Proof of (2).

$$\begin{aligned}&\Pr(Y = t \mid B_{[1,i-1]} = b_{[1,i-1]})\\ &\quad \geqslant \Pr(f = t \mid B_{[1,i-1]} = b_{[1,i-1]})\\ &\qquad - \Pr(f = t \text{ and } Y = 0 \mid B_{[1,i-1]} = b_{[1,i-1]})\\ &\quad \geqslant \varepsilon_{i-1} - \sum_{j=i}^{d} \varepsilon_j.\end{aligned}$$

The last inequality is from Claim E.1. ▮

Now we state our last lemma, from which Claim 5.3 also easily follows. First we give a definition.

Definition E.1. For b s.t. $f(b) = t$ and $Y(b) = 0$ define $YF(b)$ to be the first $i \in [1, t]$ s.t. $\text{Prob}(f = 1 \mid B_{[1,i-1]} = b_{[1,i-1]}) \leqslant \varepsilon_i$, i.e., $YF(b)$ indicates the reason why b is bad.

Claim E.1. 1. *For any* $1 \leqslant i \leqslant t-1$ *and any* $b_{[1,i-1]}$,

$$\Pr_b(f = i \wedge Y = 0 \mid B_{[1,i-1]} = b_{[1,i-1]}) \leqslant \varepsilon.$$

2. *For any* $b_{[1,i-1]}$ *that can be extended to* b *with* $Y(b) = t$,

$$\Pr_b(f = t \wedge YF = j \mid B_{[1,i-1]} = b_{[1,i-1]}) \leqslant \varepsilon_j.$$

3. *For any* $b_{[1,i-1]}$ *that can be extended to some* b *with* $Y(b) = t$.

$$\Pr(f = t \text{ and } Y = 0 \mid B_{[1,i-1]} = b_{[1,i-1]}) \leqslant \sum_{j=i}^{t} \varepsilon_j.$$

Proof of Claim E.1. Proof of (1). Given $b_{[1,i-1]}$, $f = i \wedge Y = 0$ implies that $\Pr_b(f = i \mid B_{[1,i-1]} = b_{[1,i-1]}) \leqslant \varepsilon$, which proves what we require.

Proof of (2). First, it is clear that $\Pr_b(f = t \wedge YF = j \mid B_{[1,i-1]} = b_{[1,j-1]}) \leqslant \varepsilon_j$. Now,

$$\begin{aligned}&\Pr_b(f = t \wedge YF = j \mid B_{[1,i-1]} = b_{[1,i-1]})\\ &\quad = \sum_{b_{[i,j-1]}} \Pr(B_{[i,j-1]} = b_{[i,j-1]} \mid B_{[1,i-1]} = b_{[1,i-1]})\\ &\qquad \times \Pr_b(f = t \wedge YF = j \mid B_{[1,j-1]} = b_{[1,j-1]})\\ &\quad \leqslant \sum_{b_{[i,j]}} \Pr(B_{[i,j-1]} = b_{[i,j-1]} \mid B_{[1,i-1]} = b_{[1,i-1]}) \cdot \varepsilon_j\\ &\quad \leqslant \varepsilon_j.\end{aligned}$$

Proof of (3). Since $b_{[1,i-1]}$ can be extended to some b with $Y(b) = t$, it must hold that $YF(b) \geqslant i$. Therefore,

$$\begin{aligned}&\Pr(f = t \text{ and } Y = 0 \mid B_{[1,i-1]} = b_{[1,i-1]})\\ &\quad \leqslant \sum_{j=1}^{t} \Pr(f = t \text{ and } YF = j \mid B_{[1,i-1]} = b_{[1,i-1]})\\ &\quad \leqslant \sum_{j=i}^{t} \varepsilon_j.\end{aligned}$$

The last inequality is by (2). ▮

ACKNOWLEDGMENTS

We thank David Zuckerman and Avi Wigderson for many helpful discussions. We thank Oded Goldreich for many helpful comments.

REFERENCES

[AGHP92] N. Alon, O. Goldreich, J. Hastad, and Peralta, Simple constructions of almost k-wise independent random variables, *Random Struct. and Algorithms* **3** (1992).

[AKS87] M. Ajtai, J. Komlos, and E. Szemeredi, Deterministic simulation in LOGSPACE, *in* "ACM Symposium on Theory of Computing (STOC), 1987."

[AKSS89] M. Ajtai, J. Komlos, W. Steiger, and E. Szemeredi, Almost sorting in one round, *Adv. Comput. Res.* **5** (1989), 117–125.

[ALM+92] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy, Proof verification and hardness of approximation problems, *in* "Proceedings, 33rd Annual IEEE Symposium on the Foundations of Computer Science IEEE, 1992," pp. 14–23.

[AS92a] N. Alon and J. H. Spencer, "The Probabilistic Method," Wiley, New York, 1992.

[AS92b] S. Arora and S. Safra, Probabilistic checking of proofs; a new characterization of NP, *in* "Proceedings, 33rd Annual IEEE Symposium on the Foundations of Computer Science, 1992," pp. 2–13.

[Blu86] M. Blum, Independent unbiased coin flips from a correlated biased source: A finite Markov chain, *Combinatorica* **6**(2) (1986), 97–108.

[BR94] Bellare and Rompel, Randomness-efficient oblivious sampling, *in* "IEEE Symposium on Foundations of Computer Science (FOCS), 1994."

[CG88] B. Chor and O. Goldreich, Unbiased bits from sources of weak randomness and probabilistic communication complexity, *SIAM J. Comput.* **17**(2) (1988), 230–261.

[CG89] B. Chor and O. Goldreich, On the power of two-point based sampling, *J. Complexity* **5** (1989).

[CGH+85] B. Chor, O. Goldreich, J. Hastad, J. Friedman, S. Rudich, and R. Smolensky, The bit extraction problem and *t*-resilient functions, *in* "Proceedings, 26th Annual IEEE Symposium on the Foundation of Computer Science, 1985," pp. 396–407.

[CW89] A. Cohen and A. Wigderson, Dispersers, deterministic amplification, and weak random sources, *in* "Proceedings, 30th Annual IEEE Symposium on the Foundation of Computer Science, 1989," pp. 14–19.

[FGL+91] U. Feige, S. Goldwasser, L. Lovasz, S. Safra, and M. Szegedy, Approximating clique is almost NP-complete, *in* "Proceedings, 32nd Annual IEEE Symposium on the Foundation of Computer Science, IEEE, 1991," pp. 2–12.

[CG81] Gabber and Galil, Explicit constructions of linear-sized superconcentrators, *J. Comput. and System Sci.* **22** (1981).

[GW94] O. Goldreich and A. Wigderson, Tiny families of functions with random properties: A quality-size trade-off for hashing, *in* "Proceedings, 26th Annual ACM Symposium on the Theory of Computing, ACM, 1994," pp. 574–583.

[ILL89] R. Impagliazzo, L. Levin, and M. Luby, Pseudo-random generation from one-way functions, *in* "Proceedings, 21st Annual ACM Symposium on the Theory of Computing, ACM, 1989," pp. 12–24.

[IZ89] R. Impagliazzo and D. Zuckerman, How to recycle random bits, *in* "Proceedings, 30th Annual IEEE Symposium on the Foundation of Computer Science, IEEE, 1989," pp. 248–253.

[LLS89] Liechtenstein, Linial, and Saks, Some extremal problems arising from discrete control processes, *Combinatorica* **9** (1989).

[Mes84] R. Meshulam, A geometric construction of a superconcentrator of depth 2, *Theor. Comput. Sci.* **32** (1984), 215–219.

[MR95] Rajeev Motwani and Prabhakar Raghavan, "Randomized Algorithms," Cambridge Univ. Press, Cambridge, 1995.

[Nis96] N. Nisan, Refining randomness: Why and how, *in* "Annual Conference on Structure in Complexity Theory, 1996."

[NN93] Naor and Naor, Small-bias probability spaces: Efficient constructions and applications, *SIAM J. Comput.* **22** (1993).

[NZ93] N. Nisan and D. Zuckerman, More deterministic simulation in logspace, *in* "Proceedings, 25st Annual ACM Symposium on the Theory of Computing, ACM, 1993," pp. 235–244.

[Pip77] N. Pippenger, Superconcentrators, *SIAM J. Comput.* (1977).

[Pip82] N. Pippenger, Superconcentrators of depth 2, *J. Comput. System Sci.* **24** (1982).

[Pip87a] N. Pippenger, Sorting and selecting in rounds, *SIAM J. Comput.* **16** (1987), 1032–1038.

[Pip87b] N. Pippenger, Sorting and selecting in rounds, *SIAM J. Comput.* **16** (1987), 1032–1038.

[Sip88] Sipser, Expanders, randomness, or time versus space, *J. Comput. and System Sci.* **36** (1988).

[SSZ95] M. Saks, A. Srinivasan, and S. Zhou, Explicit dispersers with polylog degree, *in* "Proceedings, 26th Annual ACM Symposium on the Theory of Computing, ACM, 1995."

[SV86] M. Santha and U. Vazirani, Generating quasi-random sequences from slightly random sources, *J. Comput. System Sci.* **33** (1986), 75–87.

[SZ94] A. Srinivasan and D. Zuckerman, Computing with very weak random sources, *in* "Proceedings, 35th Annual IEEE Symposium on the Foundations of Computer Science, IEEE, 1994."

[Ta-96] A. Ta-Shma, On extracting randomness from weak random sources, *in* "ACM Symposium on Theory of Computing (STOC), 1996."

[Vaz86] U. Vazirani, "Randomness, Adversaries and Computation," Ph.D. thesis, University of California, Berkeley, 1986.

[Vaz87a] U. Vazirani, Efficiency considerations in using semi-random sources, *in* "Proceedings, 19th Annual ACM Symposium on the Theory of Computing, ACM, 1987," pp. 160–168.

[Vaz87b] U. Vazirani, Strong communication complexity or generating quasi-random sequences from two communicating semi-random sources, *Combinatorica* **7**(4) (1987), 375–392.

[VV85] U. Vazirani and V. Vazirani, Random polynomial time is equal to slightly-random polynomial time, *in* "Proceedings, 26th Annual IEEE Symposium on the Foundations of Computer Science, IEEE, 1985," pp. 417–428.

[WZ93] A. Wigderson and D. Zuckerman, Expanders that beat the eigenvalue bound: Explicit construction and applications, *in* "Proceedings, 25th Annual ACM Symposium on the Theory of Computing, ACM, 1993," pp. 245–251.

[Zuc] D. Zuckerman, Randomness-optimal sampling, extractors, and constructive leader election, private communication.

[Zuc90] D. Zuckerman, General weak random sources, *in* "Proceedings, 31st Annual IEEE Symposium on the Foundation of Computer Science, 1990," pp. 534–543.

[Zuc91] D. Zuckerman, Simulating BPP using a general weak random source, *in* "Proceedings, 32nd Annual IEEE Symposium on the Foundations of Computer Science, IEEE, 1991," pp. 79–89.

[Zuc93] D. Zuckerman, NP-complete problems have a version that is hard to approximate, *in* "Proceedings, 8th Structures in Complexity Theory, IEEE, 1993," pp. 305–312.