Explicit OR-Dispersers with Polylogarithmic Degree*

Michael Saks† Aravind Srinivasan‡ Shiyu Zhou§

Abstract

An (N, M, T)-OR-disperser is a bipartite multigraph $G = (V, W, E)$ with $|V| = N$, and $|W| = M$, having the following expansion property: any subset of V having at least T vertices has a neighbor set of size at least $M/2$. For any pair of constants $\xi, \lambda, 1 \geq \xi > \lambda \geq 0$, any sufficiently large N, and for any $T \geq 2^{(\log N)^{\xi}}$, $M \leq 2^{(\log N)^{\lambda}}$, we give an explicit elementary construction of an (N, M, T)-OR-disperser such that the out-degree of any vertex in V is at most polylogarithmic in N. Using this with known applications of OR-dispersers yields several results. First, our construction implies that the complexity class Strong-RP defined by Sipser, equals RP. Second, for any fixed $\eta > 0$, we give the first polynomial-time simulation of RP algorithms using the output of any "η-minimally random" source. For any integral $R > 0$, such a source accepts a single request for an R-bit string and generates the string according to a distribution that assigns probability at most $2^{-R^{\eta}}$ to any string. It is minimally random in the sense that any weaker source is insufficient to do a black-box polynomial-time simulation of RP algorithms.

Key Words and Phrases. Derandomization, Expander Graphs, Explicit Constructions, Hashing Lemmas, Hardness of Approximation, Imperfect Sources of Randomness, Measures of Information, Pseudo-random Generators, Randomized Computation, Time-Space Tradeoffs.

AMS Subject Classifications. 60C05, 68Q15, 94A17

---

*The first and third authors were supported in part by NSF grant CCR-9215293. The second author was supported in parts by grant 93-6-6 of the Alfred P. Sloan Foundation to the Institute for Advanced Study, by ESPRIT Basic Research Action Programme of the EC under contract No. 7141 (project ALCOM II), and by National University of Singapore Academic Research Fund Grant RP960620. All three authors were also supported in part by DIMACS (Center for Discrete Mathematics & Theoretical Computer Science), through NSF grant NSF-STC91-19999 and by the New Jersey Commission on Science and Technology. A preliminary version of this paper appeared in the proceedings of the 27th ACM Symposium on Theory of Computing, 1995.

†Department of Mathematics, Rutgers University, New Brunswick, NJ 08854, USA. E-mail: `saks@math.rutgers.edu`.

‡Department of Information Systems and Computer Science, National University of Singapore, Singapore 119260. Parts of this work were done: (i) at the School of Mathematics, Institute for Advanced Study, Princeton, NJ 08540, USA, (ii) at DIMACS, (iii) while visiting the Department of Applied Mathematics and Computer Science, Weizmann Institute of Science, Rehovot 76100, Israel, (iv) while visiting the Department of Computer Science, University of Warwick, Coventry CV4 7AL, UK, and (v) at the National University of Singapore. E-mail: `aravind@iscs.nus.edu.sg`.

§Bell Laboratories, 700 Mountain Avenue, Murray Hill, New Jersey 07974, USA. E-mail: `shiyu@research.bell-labs.com`. The work was mainly done at the Department of Computer Science, Rutgers University, New Brunswick, NJ 08854, USA.

1 Introduction

A disperser is a type of expander which was first introduced by Sipser in [Sip88]. Cohen and Wigderson [CW89] classified dispersers into two types: OR-dispersers and MAJORITY-dispersers. A bipartite multigraph $G = (V, W, E)$ with $|V| = N$ and $|W| = M$ is called an (N, M, T)-OR-disperser if any subset of V having at least T vertices has a neighbor set in W of size at least $M/2$; G is called an (N, M, T)-MAJORITY-disperser if it has the (essentially stronger) property that for any subset Y of W of size $M/3$, there are at most T vertices $v \in V$ such that a majority of v's neighbors are in Y. The degree of $v \in V$ is defined to be the number of edges incident on v (since G is a multigraph, this may be bigger than the number of distinct neighbors of v). The *degree* of a disperser is defined to be the maximum degree of any vertex in V. (Note the *difference* from the usual definition wherein "degree" denotes the maximum degree of any vertex in $V \cup W$.) In this paper we investigate the problem of efficiently constructing OR-dispersers with small degree. For convenience, we will use the term "disperser" instead of "OR-disperser" unless otherwise specified.

It was proved in [San87] by a probabilistic argument that there exist (N, M, T)-dispersers for $N > T = M$ with degree at most $\log_2 N + 2$. The problem of giving an efficient algorithmic construction for dispersers with these or similar parameters, has remained open. (By *efficient*, we mean that for each vertex $v \in V$, its neighbor set can be generated in time polynomial in $\log(M+N)$ and in the degree of v.) It has been shown that such an efficient construction would have a variety of applications in the theory of computation. The requirement for such a construction in many of the applications is that it works for constants $1 \geq \xi > \lambda \geq 0$ and for sufficiently large N, with $T \geq 2^{(\log N)^\xi}$, $M \leq 2^{(\log N)^\lambda}$, and with the degree being at most polylogarithmic in N.

As we will see, the problem of finding an efficient disperser construction is closely related to the problem of making randomized algorithms robust with respect to imperfections in the random source. In fact, most of the previous disperser constructions are implicit in the work of simulating randomized algorithms using weak sources. The best previously known constructions are due to Zuckerman [Zuc91, Zuc96], who achieved degree polylogarithmic in N if $N = T^{O(1)}$, and Srinivasan and Zuckerman [SZ94], whose construction works for $N = 2^{polylog(T)}$ but requires degree $(\log N)^{O(\log\log N)}$. In recent work that was inspired in part by a preliminary version of the present paper, Ta-Shma [TaS96] improved the degree of the construction in [SZ94] to $(\log N)^{O(\log^{(k)} \log N)}$ for any fixed integer k, where $\log^{(k)}$ denotes the logarithm to the base 2 iterated k times (i.e., $\log^{(1)} x = \log_2 x$, $\log^{(2)} x = \log_2 \log_2 x$, etc.).

In this paper, we give an improved construction of an (N, M, T)-disperser.

Main Theorem: $\forall \xi, \lambda, 1 \geq \xi > \lambda \geq 0$, *$\exists N_0(\xi, \lambda)$ such that if $N \geq N_0(\xi, \lambda)$, then for any $2^{(\log N)^\xi} \leq T \leq N$ and $M \leq 2^{(\log N)^\lambda}$, there is an efficient construction of an explicit (N, M, T)-disperser with degree polylogarithmic in N.*

Remark. Our constructions can be easily generalized as follows: for any given constant $c > 0$ and for all sufficiently large N, we can construct (N, M, T)-dispersers $G = (V, W, E)$ with the above parameters, wherein any subset of V having at least T vertices has a neighbor set in W of size at least $M(1 - (\log N)^{-c})$.

It is worth noting that although the constructions of [Zuc91], [SZ94] and [TaS96] do not give polylogarithmic degree, they do give MAJORITY-dispersers. Unfortunately, we could not strengthen our construction to give a MAJORITY-disperser. Nevertheless, the construction of a good OR-disperser is itself an interesting combinatorial problem and, as we will see, has significant applications to both complexity theory and randomized algorithm design.

In the remainder of this section, we describe three main applications of our new disperser. First we show that the complexity class Strong-RP is equal to RP. In effect, what this says is that there is an extremely efficient amplification scheme for randomized polynomial time algorithms. Next, we use our disperser construction along with the work of [Zuc91, Zuc93] in extending the results of [Zuc91, Zuc93, SZ94] on the hardness of approximating the clique function. Then we show how our improved disperser family can be applied to designing randomized algorithms that achieve maximal robustness with respect to imperfections in the random source. Next, our disperser family also gives improvements on the expander construction and the consequent applications given in [WZ93]; however, further improvements here have been obtained by [TaS96]. We briefly sketch how these lead to improved solutions for a problem in data structures: implicit $O(1)$ probe search [FN93, Zuc91]. These consequences were each observed by previous researchers, and provided much of the motivation for the search for good dispersers. We also refer the reader to a comprehensive survey paper by Nisan [Nis96].

1.1 The Equivalence of RP and Strong-RP

Definition 1.1 *Random polynomial time (RP) is the set of languages $L \subseteq \{0,1\}^*$ such that there is a deterministic polynomial-time Turing machine $M_L(\cdot,\cdot)$ for which*

$$\begin{aligned} x \in L &\rightarrow Pr[M_L(x,y) \text{ accepts}] > 1/2, \text{ and} \\ x \notin L &\rightarrow Pr[M_L(x,y) \text{ accepts}] = 0, \end{aligned}$$

where the probabilities are for a y picked uniformly at random from $\{0,1\}^m$ where $m = p(|x|)$ for some polynomial function $p = p(M_L)$.

We call $W_L^x = \{y \in \{0,1\}^m | M_L(x,y) \text{ accepts}\}$ the *witness* set of M_L on input x, where m is the number of random bits used by M_L on inputs of length $|x|$.

Sipser [Sip88] defined the complexity class *strong random polynomial time* (Strong-RP) to be the class of languages L for which there is an RP machine $M_L(\cdot,\cdot)$ and a real number $0 < \eta < 1$ such that on input a string $x \in L$ of (any) length n, M_L uses $q(n)$ random bits for some polynomial $q(\cdot)$ and recognizes x with error probability at most $2^{-q(n)+q(n)^\eta}$. (If $x \notin L$, then the error probability is zero, as in Definition 1.1.) He asked whether RP = Strong-RP and showed that the existence of explicit constructions for sufficiently good dispersers would imply this equivalence. The construction we give here is sufficient for Sipser's purposes and so we obtain

Theorem 1.1 *RP = Strong-RP.*

Proof: The argument in this proof is essentially the same as in Sipser's original paper [Sip88]. We show the proof here for later reference. It is immediate that Strong-RP $\subseteq$ RP; we now show that RP $\subseteq$ Strong-RP. Suppose we have an RP machine M_L that needs m random bits on an input $x \in L$ of length n, for some m polynomial in n. We wish to simulate this by a machine that satisfies the conditions of Strong-RP.

Let $0 < \eta < 1$ be given. Let $\ell = m^{2/\eta} = n^{O(1)}$. By the Main Theorem with $\xi = \eta$ and $\lambda = \eta/2$, there is an explicit $(2^\ell, 2^m, 2^{\ell^\eta})$-disperser $G(\{0,1\}^\ell, \{0,1\}^m, E)$, and moreover, for any given $z \in \{0,1\}^\ell$, we can compute the neighborhood of z in $\{0,1\}^m$ in time $\ell^{O(1)} = n^{O(1)}$.

Our simulator on input x first samples a random $z \in \{0,1\}^\ell$, then computes the set of neighbors $Y \subseteq \{0,1\}^m$ of z. Finally, it runs $M_L(x,y)$ for each $y \in Y$ and accepts if and only if any of the runs accept. To see that it accepts with the required probability, let B be the set of strings in $\{0,1\}^\ell$

that do not have a neighbor in the witness set W_L^x (which are the strings on which the simulation fails to find a witness). Since the witness set W_L^x has more than half of the nodes in $\{0,1\}^m$, the definition of the disperser implies that B has size at most 2^{ℓ^η}. So the probability of failure is at most $2^{\ell^\eta - \ell}$. $\square$

Sipser's introduction of Strong-RP was motivated by:

Theorem 1.2 ([Sip88]) *If $P \neq$ Strong-RP then there is a positive ϵ such that for any time bound $t(n)$, and for any language $L \in TIME(t(n))$, there is a machine that accepts L and, for infinitely many inputs, requires space at most $t(n)^{1-\epsilon}$.*

Sipser wanted to replace the hypothesis "P $\neq$ Strong-RP" by the hypothesis "P $\neq$ RP"; Theorem 1.1 says that this can indeed be done.

1.2 The Hardness of Approximating the Clique Function

We now discuss the hardness of approximating $\log\log\omega(G)$, where $\omega(G)$ is the maximum size of a clique in an input graph G. Let $\tilde{P}$ denote quasi-polynomial time, $\cup_{c>0}\ DTIME(2^{(\log n)^c})$. In [Zuc93], it was shown that if $N\tilde{P} \neq \tilde{P}$, then approximating $\log\omega(G)$ to within any constant factor is not in $\tilde{P}$; see [Zuc91, Zuc93, SZ94] for further results. We build on the techniques of [Zuc91, Zuc93] to prove Theorem 1.3, which extends the work of [Zuc91, Zuc93, SZ94].

Let quasipoly(x) be short-hand for $2^{(\log x)^{O(1)}}$. We start with a lemma that is implicit from [Zuc91, Zuc93]:

Lemma 1.1 ([Zuc91, Zuc93]) *Suppose that for every integer n, there is an efficient construction of an $(N = N(n), n^{\Omega(1)}, T = T(n))$-disperser with degree $d = d(n)$. For some pair of functions $g_1, g_2 : [1,\infty) \to \Re^+$ such that $g_1(y) \leq g_2(y)$ for all $y \in [1,\infty)$ and for any input graph G, suppose a number $h(G) \in [g_1(\omega(G)), g_2(\omega(G))]$ can be computed in $\tilde{P}$. Then if $g_1(N) > g_2(T)$, we have $NP \subseteq DTIME(quasipoly(N + 2^d))$.*

Theorem 1.3 *If $N\tilde{P} \not\subseteq \tilde{P}$, then approximating $\log\log\omega(G)$ to within any constant factor is not in $\tilde{P}$. In other words, if we can compute, for some fixed $t > 1$, a number in the range*

$$[2^{(\log\omega(G))^{1/t}}, 2^{(\log\omega(G))^t}]$$

in $\tilde{P}$, then $N\tilde{P} \subseteq \tilde{P}$.

Proof: Choose constants $\alpha > \beta > 1$ such that $\alpha/\beta > t^2$. The main theorem implies that an $(N = 2^{(\log n)^\alpha}, n^{\Omega(1)}, T = 2^{(\log n)^\beta})$-disperser with degree $\log^{O(1)} n$ is efficiently constructible. Thus, by taking $g_1(y) = 2^{(\log y)^{1/t}}$ and $g_2(y) = 2^{(\log y)^t}$, we invoke Lemma 1.1 to conclude that NP and hence $N\tilde{P}$, by a simple padding argument, is contained in $\tilde{P}$. $\square$

This result improves, e.g., on a theorem from the full version of [SZ94] which showed that if $N\tilde{P} \not\subseteq DTIME(2^{(\log n)^{O(\log\log n)}})$, then approximating $\log\log\omega(G)$ to within any constant factor is not in $\tilde{P}$.

1.3 Computing With Weak Random Sources

In practice, randomized algorithms get their "random" bits by using pseudo-random number generators. Empirically, this often seems to be sufficient. However, there are reports of algorithms giving quite different results under different pseudo-random generators (see e.g., [FLW92] for such reports on Monte-Carlo simulations, and [Hsu93, HRD94] for the deviant performance of some RNC algorithms for graph problems). An alternative approach is to use the output of some physical source of randomness, such as a Zener diode, or the last digits of a real-time clock. For such a source, it is plausible to assume that the string of bits output by the source are selected from some unknown distribution which, while not necessarily uniform, is "somewhat random". This leads to the question: to what extent is it possible to design randomized algorithms that are robust with respect to deviations of the random source from true randomness?

In general, one can define the following notion of an abstract source. An *n-bit source* is a probability distribution on $\{0,1\}^n$. A *source* is a sequence $S = S_1, S_2, \ldots$ where S_n is an n-bit source. We allow an algorithm to make a *single request* for R bits from the source for some R, and the source produces a string that is distributed according to S_R.

A natural idea to compute with imperfect random sources is to "convert" any source from some family of faulty sources into a distribution that is (nearly) random, which can then be used by randomized algorithms. That is, for a class of faulty sources, we would like to construct an easily computable function (family) f such that for any faulty source in the class, if x is a string selected according to the source then the distribution induced on $f(x)$ is (close to) uniform. For example, von Neumann [Neu63] presented a technique to convert independent but biased coin-flips into independent and unbiased ones; Blum [Blu86] showed how to convert the bits output by an unknown Markov chain into a sequence of perfectly random bits. However, for more general faulty sources, it can be shown (see e.g. [SV86]) that to construct such a direct conversion f is impossible.

On the other hand, the following broader idea of conversion (introduced in [VV85, Vaz86]) works for simulating randomized algorithms: Suppose the randomized computation C we wish to simulate needs m random bits. The simulation first requests $R = R(m)$ bits from the faulty source. Then it maps this sequence of $R(m)$ bits to a *set* of $t(m)$ strings each of m bits, called *test strings*, where $t(\cdot)$ is some function that depends on the simulation. The construction of the set of test strings does not depend on the computation being simulated or its input, but only on the number m and the sequence of bits output from the semi-random source. It then performs the computation C $t(m)$ separate times, one for each test string s, using s as the random string in the computation. If the algorithm being simulated is an RP algorithm then the simulation accepts if and only if any of the runs of C accept, while if the algorithm being simulated is a BPP algorithm, the simulation accepts if and only if the majority of the runs of C accept. Simulating RP is of course no harder than simulating BPP.

It is easy to see that the running time of the simulation is $g(m) + t(m)TIME(C)$, where $g(m)$ is the time needed to generate the set of test strings, $t(m)$ is the size of the set of test strings and $TIME(C)$ is the time to perform the computation C. We say that the simulation is *efficient* if both $g(m)$ and $t(m)$ are bounded by polynomials of m. Note that in the case that we are simulating an RP or BPP algorithm, this is equivalent to saying that the simulation runs in polynomial time.

The high-level structure of this simulation is common to all subsequent work in this area. We refer to such a simulation as a "black-box" simulation. Applying this framework, efficient simulations of RP and BPP under various models of weak sources have been extensively studied. For example, Santha and Vazirani [SV86] studied the class of weak sources called "slightly random sources", which was further examined in [VV85, Vaz86, Vaz87a, Vaz87b]. A more general model

"PRB-sources" is considered later by Chor and Goldreich [CG88]. In [Zuc90, Zuc91], Zuckerman introduced the model of δ-sources which generalizes all the previous models.

Let D be a probability distribution on a set X. The *min-entropy* of D is defined to be the maximum real number d such that $D(x) \leq 2^{-d}$ for all $x \in X$, where $D(x)$ is the probability that D assigns to x. For a function $\delta(\cdot)$ mapping the positive integers to $[0,1]$, a source $S = S_1, S_2, \ldots$ is said to be a *δ-source* if each S_R is a distribution of min-entropy at least $\delta(R) \cdot R$. The function δ is called the *entropy rate* of the source. In other words, for any number R of random bits requested, a δ-source outputs an R-bit string such that no string has probability more than $2^{-\delta(R)R}$ of being output. One example of an n-bit δ-source is the uniform distribution on a subset of $\{0,1\}^n$ of size at least $2^{\delta(n)n}$. Any source is a 0-source, and a 1-source is the pure random source. In general, the smaller that δ gets, the weaker (stochastically) the source can be.

For the case where δ is a fixed positive constant, Zuckerman [Zuc91, Zuc96] showed how to simulate any BPP algorithm efficiently with δ-sources. What about sources whose entropy rate decreases with R: how weak can the source get and still be usable for efficient simulations? In [CW89], it was observed that, for information-theoretic reasons, if $\mathcal{S}$ is any class of sources for which there is an efficient black-box simulation of RP or BPP (that works correctly with high probability) using any source $S \in \mathcal{S}$, then every source S must be close to a δ-source with $\delta(R) \geq R^{\eta-1}$ for some fixed $\eta \in (0,1]$. Intuitively, if a source S is good enough to be used in an efficient simulation, then it must be sufficiently random so that to get m bits of entropy we only have to look at a polynomial in m number of bits from the source. This establishes a lower bound on the "amount of randomness" of the class of sources for which an efficient simulation is possible. For a given $\eta \in (0,1]$, we refer to the class of δ-sources with $\delta(R) = R^{\eta-1}$ as *η-minimally random* sources. The question is: for each $\eta > 0$, is there a polynomial-time simulation of BPP, or even RP, that works for all η-minimally random sources?

The previously best known simulation for RP with η-minimally random sources is due to [SZ94], which takes time $m^{O(\log m)}$, where m is the number of random bits needed by the original RP algorithm.

In [San87, Sip88, CW89] it was observed that the black-box simulation described above can be defined by a sequence $G_m = (V_m, W_m, E_m)$ of bipartite multigraphs, one for each m, where $V_m = \{0,1\}^{R(m)}$ and $W_m = \{0,1\}^m$. If the computation to be simulated needs m bits, then we use $R(m)$ bits from the faulty source to specify a vertex in V_m, and take as our test set the neighbors of the selected vertex. They further observed that to get an efficient simulation for RP (resp. BPP) that works for all δ-sources for a given function $\delta(\cdot)$, it suffices that the graphs G_m are good enough dispersers (resp. MAJORITY-dispersers).

The disperser construction we shall present is good enough to be used to do a polynomial-time black-box simulation of RP algorithms with η-minimally random sources for any fixed positive η:

Theorem 1.4 *For any fixed $0 < \eta \leq 1$, there is a polynomial time black-box simulation of RP using a δ-source with $\delta(R) = R^{\eta-1}$ and with error probability $2^{-n^{\Omega(1)}}$, where n is the length of the input to the RP algorithm.*

Proof: Suppose an RP machine M_L needs m random bits on an input x of length n, for some m polynomial in n. Let $R = m^{3/\eta} = n^{O(1)}$, and let distribution D be any δ-source on $\{0,1\}^R$. Setting $\xi = \eta/2$ and $\lambda = \eta/3$ in the Main Theorem, we get a $(2^R, 2^m, 2^{R^{\eta/2}})$-disperser and use the same simulation as in the proof of Theorem 1.1. Again the probability of failure is the probability that $z \in B$. Since D has min-entropy R^η, we conclude that the probability that $z \in B$ is at most $2^{-R^\eta + R^{\eta/2}}$. $\square$

Subsequent to our work, Ta-Shma in [TaS96], obtained a "nearly optimal" black box simulation of BPP that runs in time $m^{O(\log^{(k)} m)}$ for any fixed k, where $\log^{(k)} m$ is the k-th iterated logarithm. Recently, Andre'ev et al [ACRT97] introduced a more general type of simulation, which allows for a polynomial-time simulation of BPP using η-minimally random sources. This result is obtained by combining the work of [ACR96] and our disperser construction. (It should be noted that, unlike previous BPP simulations, their work does not provide a new MAJORITY-disperser.)

1.4 Improved Expander Constructions and Implicit $O(1)$ Probe Search

Recall that our main construction produces (N, M, T)-dispersers, where $T \geq 2^{(\log N)^\xi}$ and $M \leq 2^{(\log N)^\lambda}$ for any given pair of constants ξ, λ $(1 \geq \xi > \lambda \geq 0)$, and N is sufficiently large. Apart from intrinsic interest, it is also useful in some applications to have M ("almost") as high as T. It is possible to use our main construction with a bootstrapping approach from [WZ93] to explicitly construct, for any positive constants $\xi \leq 1$ and (small) $c_1 > 0$ and for all sufficiently large N $(N \geq N_0'(\xi, c_1))$, $(N, 2^{(\log N)^\xi}, 2^{(\log N)^\xi})$-dispersers of degree $2^{(\log N)^{c_1}}$. However, subsequent work of [TaS96] has presented further improved constructions of such dispersers, with degree $\exp(\text{poly}(\log\log N))$.

We now sketch an application of such dispersers to the problem of implicit $O(1)$ probe search: the reader is referred to [FN93, Zuc91] for all other definitions and background.

Define a (c, m, n, t)-rainbow to be a coloring of all t-tuples (without repetitions) of elements in $\{1, \ldots, m\}$ with c colors, so that for any $S \subset \{1, \ldots, m\}$, $|S| = n$, all c colors appear in the t-tuples over S. For $m \leq \exp(\text{poly}(n))$, implicit $O(1)$ search has a very close relationship to the explicit construction of $(n, m, n, O(1))$-rainbows [FN93]. The basic issue here is to efficiently construct $(n, m, n, O(1))$-rainbows for as high a value of $m = m(n)$ (with $m \leq \exp(\text{poly}(n))$) as possible. Bounds of $m = \text{poly}(n)$ and $m = n^{O(\sqrt{\log\log n}/\log\log\log n)}$ were achieved in [FN93] and [Zuc91] respectively. We just mention that the above-sketched families of (N, M, M)-dispersers can be used to efficiently construct $(n, m, n, O(1))$-rainbows for $m = 2^{(\log n)^c}$, for any fixed $c > 0$ and for all sufficiently large $n \geq f(c)$. The proof easily follows from the results of [FN93, Zuc91], and the interested reader is referred to these papers.

1.5 Simulations of Randomized Parallel Computation

As explained above, our disperser construction is efficient in the sense that given vertex $v \in V$, we can compute all of v's neighbors in time polynomial in $\log N$, i.e., in time polynomial in the length of the natural encoding of v. In fact, as we will show, we can compute v's neighbors in NC, i.e., in parallel using $poly(\log N)$ processors in $polylog(\log N)$ time.

This fact was used in [ACRT97] to get, for every fixed $\gamma > 0$, a simulation of BPNC in polylogarithmic time using a poly(n) number of processors, given access to any weak random source of min-entropy n^γ. (We originally consided the question of whether our construction is in NC in response to a question from these authors.)

Another application will appear in the full version of [ACR97], and is briefly as follows. There is a well-known line of research in complexity theory on "hardness versus randomness": showing that if certain tasks are computationally hard for some deterministic model of computation, then a certain probabilistic complexity class has a reasonably efficient deterministic simulation (see, e.g., [IW97]). Our NC construction is one of the ingredients of Theorem 1.5, which will appear in the full version of [ACR97]. Theorem 1.5 gives a "hardness versus randomness" tradeoff for parallel computation.

We need some notation to present Theorem 1.5. The term $H : k(n) \to n$ denotes any Boolean operator $H = \{H_1, H_2, ..H_n, \ldots\}$ such that for any $n > 0$, $H_n : \{0,1\}^{k(n)} \to \{0,1\}^n$. The term F_n^H represents the characteristic function of the output set of H_n. Let $L(f)$ denote the circuit complexity of a finite function f and, given any positive integer dp, the term $L_{dp}(f)$ denotes the minimum size of circuits of depth dp that can compute f.

Theorem 1.5 *Suppose there exists an NC operator $H : k(n) \to n$ with $k(n) = (1+\Theta(1)) \log n$ and such that, for any constant $d \geq 1$, the characteristic functions F_n^H of its output sets satisfy*

$$L_{\log^d n}(F_n^H) \geq \Omega(2^{k(n)} n^{c_0}),$$

for some constant $c_0 > 0$. Then, NC = BPNC.

A better "hardness versus randomness" result for sequential computation is shown in [IW97]; however, it does not seem to apply to parallel computation.

The rest of this paper contains six sections. In Section 2 we present various preliminary definitions and facts. The motivation and overview of the disperser construction is given in Section 3. We present the construction in Section 4 and its correctness proof is shown in Sections 5, 6 and 7. In Section 8 we give a summary of the parameters that appear in the construction and the proof.

2 Preliminaries

This section contains a large number of preliminary definitions concerning bipartite graphs and random sources. The key result of this section is Lemma 2.2, which provides a sufficient condition for a bipartite graph to be a disperser.

2.1 Bipartite Graphs

We consider bipartite multigraphs $G = (V, W, E)$. For simplicity, we will say "graphs" instead of "multigraphs". We use $deg(G)$ to denote the degree of G, which is defined to be the maximum degree of any vertex in V.

Suppose that $V_1, V_2, \ldots, V_k$ are disjoint sets and for each i between 1 and $k-1$, $G_i = (V_i, V_{i+1}, E_i)$ is a bipartite graph. The *composition* of $G_1, G_2, \ldots, G_{k-1}$, denoted $G_1 \circ G_2 \circ \cdots \circ G_{k-1}$, is defined to be the bipartite graph $G = (V_1, V_k, E)$ where $(v_1, v_k) \in E$ if and only if there exists a sequence $v_2, \ldots, v_{k-1}$ of vertices with $v_i \in V_i$ for $1 < i < k$ and $(v_i, v_{i+1}) \in E_i$ for $1 \leq i < k$. Observe:

Proposition 2.1 *If $G = G_1 \circ \cdots \circ G_{k-1}$, then $deg(G) \leq deg(G_1) \times \cdots \times deg(G_{k-1})$.*

2.2 Blocks, Segmentations and Bit-string Trees

For integers $i \leq j$, the *block* $[i, j]$ is defined to be the sequence of integers $(i, i+1, \ldots, j)$.

We will often be dealing with sequences $A = A_1, A_2, \ldots, A_k$ where the A_i are positive integers, sets or bit strings. If $[i, j]$ is a block contained in $[1, k]$ then we use the notation $A_{[i,j]}$ to denote the sum $A_i + A_{i+1} + \ldots + A_j$ in the case that the A_i are integers, the union $A_i \cup A_{i+1} \cup \ldots \cup A_j$ in the case that the A_i are sets and the concatenation $A_i A_{i+1} \ldots A_j$ in the case that the A_i are bit strings.

An (n, s)-*composition* is a sequence $l = (l_1, l_2, \ldots, l_s)$ of positive integers summing to n, an (n, s)-*tower* is a sequence $q = (q_0, q_1, \ldots, q_{s-1}, q_s)$ of integers with $0 = q_0 < q_1 < q_2 < \ldots < q_{s-1} < q_s = n$, and an (n,s)-*segmentation* is a sequence $\pi = (B_1, B_2, \ldots, B_s)$ of disjoint blocks whose union is $[1,n]$ such that for each $i < j$ every element of B_i is less than every element of B_j.

There is an obvious set of one-to-one correspondences between (n,s)-compositions, (n,s)-towers and (n,s)-segmentations as follows: the composition l corresponds to the tower q with $q_i = l_1 + l_2 + \cdots + l_i$ and to the segmentation π such that $B_i = [q_{i-1}+1, q_i]$ (thus $|B_i| = l_i$). For example, for $n = 6$ and $s = 3$, the (n,s)-segmentation $(\{1,2,3\},\{4,5\},\{6\})$ corresponds to the (n,s)-composition $l = (3,2,1)$ and the (n,s)-tower $q = (0,3,5,6)$.

Let π be an (n,s)-segmentation, l the corresponding (n,s)-composition and q the corresponding (n,s)-tower. The *bit-string tree* T^π associated to π is the rooted tree with $s+1$ layers of nodes labeled by bit-strings defined as follows:

1. the root of the tree is at depth 0 and is labeled by the empty string;
2. there are 2^{q_i} nodes at depth i labeled by the bit-strings of length q_i;
3. if a node at depth $i < s$ is labeled by y, then it has $2^{l_{i+1}}$ children at depth $i+1$ such that for each bit-string z of length l_{i+1}, the node has exactly one child labeled by yz. We denote the edge from y to yz by $(z|y)$.

2.3 Probability Distributions, Converters and Carriers

We will be considering probability distributions D defined on a finite set X. We denote by $\mathrm{supp}(D)$ the support of D, i.e., the set of elements of X to which D assigns non-zero probability. It is often convenient to think of D as a (row) vector indexed by the set X. Thus, a family of distributions $\mathcal{D}$ on X can be viewed as a collection of vectors in R^X. For $x \in X$, we use $D(x)$ to denote the probability that D assigns to x, and for $S \subseteq X$, we define $D(S) = \sum_{x\in S} D(x)$.

We denote by U_X the uniform distribution on the set X, and use U_l instead of $U_{\{0,1\}^l}$ for simplicity. If D is a distribution on X and E is a distribution on Y, then $D \times E$ denotes the product distribution on the set $X \times Y$ given by $(D \times E)(x,y) = D(x)E(y)$.

Definition 2.1 *Let V and W be finite sets. A* converter *Λ from V to W is a matrix with rows indexed by V and columns indexed by W, such that all entries are nonnegative and all row sums are 1. For $v \in V$ and $w \in W$, $\Lambda(v,w)$ denotes the entry in row v and column w.*

The usual term for a converter is "stochastic matrix"; we choose the term converter because it provides a mnemonic for what we want to do with it. Each row $\Lambda(v,\cdot)$ of a converter can be viewed as a probability distribution over W. Given any probability distribution D on V, and a converter Λ from V to W, we define the distribution $D\Lambda$ on W as follows: select $v \in V$ according to D, and then select $w \in W$ according to $\Lambda(v,\cdot)$. Thus Λ "converts" any distribution on V to a distribution on W. If we view distributions as (row) vectors, then the transformed distribution of D through Λ is exactly the vector-matrix product $D\Lambda$.

Any function $f : V \to W$ induces a converter Λ^f from V to W in a natural way: for $v \in V$ and $w \in W$, $\Lambda^f(v,w)$ is 1 if $f(v) = w$ and is 0 otherwise. If D is any distribution on V, then $f(D)$ is defined to be the distribution on W such that for $w \in W$, $f(D)(w) = \sum_{v\in V: f(v)=w} D(v)$. That is, $f(D) = D\Lambda^f$. An easy induction shows the following:

Proposition 2.2 *Let $V_1, V_2, \ldots, V_k$ be sets and for each i between 1 and $k-1$, let f_i be a function mapping V_i to V_{i+1}. Suppose $f : V_1 \to V_k$ is defined as $f = f_{k-1} \circ \ldots \circ f_2 \circ f_1$, where $\circ$ denotes the composition of functions. Then for any distribution D on V_1, $f(D)$ is the distribution on V_k such that $f(D) = D\Lambda^{f_1}\Lambda^{f_2} \ldots \Lambda^{f_{k-1}}$.*

The *support* of a converter Λ is defined to be the bipartite graph (V, W, E) such that $(v, w) \in E$ if and only if $\Lambda(v, w) \neq 0$. If G is a bipartite graph on V and W that contains the support of Λ, then G is said to be a *carrier* for Λ. Given a carrier G of the converter Λ, it is often convenient to visualize Λ as an assignment of nonnegative weights to the edges of G, where the weight on edge (v, w) is the probability assigned to vertex w by the distribution $\Lambda(v, \cdot)$. Thus the sum of weights on edges leaving v is exactly 1.

We next recall a standard measure of distance between distributions.

Definition 2.2 *1. The* variational distance *between two distributions D_1 and D_2 on the same set X is defined to be $\|D_1 - D_2\| = \max_{Y \subseteq X} |D_1(Y) - D_2(Y)| = \frac{1}{2}\sum_{x \in X} |D_1(x) - D_2(x)|$.*

2. D_1 is said to be ϵ-near *to D_2 if $\|D_1 - D_2\| \leq \epsilon$.*

3. The distribution D on X is said to be quasi-random to within ϵ *if D is ϵ-near to the uniform distribution on X.*

Some useful, well-known, and easily proved facts about variational distance are summarized below.

Proposition 2.3 *1. If D_1 and D_2 are distributions on the same set X, then $\|D_1 - D_2\|$ is at most $\min\{D_1(X_1), D_2(X_2)\}$, where X_1 (resp. X_2) is the set of elements $x \in X$ such that $D_2(x) < D_1(x)$ (resp. $D_1(x) < D_2(x)$).*

2. If D_1, D_2, D_3 are distributions on the same set X, then $\|D_1 - D_3\| \leq \|D_1 - D_2\| + \|D_2 - D_3\|$.

3. If D_1 and D_2 are distributions on V and Λ is a converter from V to W, then $\|D_1\Lambda - D_2\Lambda\| \leq \|D_1 - D_2\|$. In particular, for any function $f : V \to W$, $\|f(D_1) - f(D_2)\| \leq \|D_1 - D_2\|$.

The *distance* between a distribution D and a family $\mathcal{D}$ of distributions on the same set is defined to be the minimum of $\|D - D'\|$ over all $D' \in \mathcal{D}$. D is said to be *ϵ-near to the family $\mathcal{D}$* if the distance from D to $\mathcal{D}$ is at most ϵ.

We adopt the convention that the conditional probability given some impossible event is 0.

2.4 Bit Sources

An *n-bit source* is a distribution on $\{0,1\}^n$. We typically use the symbol $\tilde{X}$ to denote a random n-bit string selected according to some n-bit source D. If y is a bit string of length $j \leq n$, we write $D(y)$ as the probability that $\tilde{X}_{[1,j]} = y$. If z is another bit string of length k with $j + k \leq n$, we write $D(z|y)$ as the conditional probability that $\tilde{X}_{[1,j+k]} = yz$ given $\tilde{X}_{[1,j]} = y$.

For $q \leq n$, we define the *q-bit truncation* of the n-bit source D to be the q-bit source $D^{(q)}$ such that for $y \in \{0,1\}^q, D^{(q)}(y) = D(y)$.

Lemma 2.1 *If D is an n-bit source quasi-random to within ϵ and $q \leq n$, then $D^{(q)}$ is quasi-random to within ϵ.*

Proof: Define $f : \{0,1\}^n \to \{0,1\}^q$ to be the q-bit truncation function, i.e., for $x \in \{0,1\}^n$, $f(x)$ is equal to the first q bits of x. Then $D^{(q)} = f(D)$ and $f(U_n) = U_q$. The lemma follows from Proposition 2.3(3). □

Let $x \in \{0,1\}^n$ and $1 \leq i \leq n$. The *information in bit i of x relative to D* is defined to be $I_i(x) = I_i^D(x) = -\log_2 D(x_i|x_{[1,i-1]})$. (If $D(x_i|x_{[1,i-1]}) = 0$, we take $I_i(x) = \infty$.) Intuitively, $I_i(x)$ represents the amount of information (relative to the distribution) that the i-th bit of x provides if the bits are revealed one by one. We say that position i indexes a *good bit* in x if $I_i(x) \geq 1$. In the case that D is the uniform distribution over n-bit strings, every string has n good bits.

As an immediate consequence of the definition we get:

Proposition 2.4 *Let D be a distribution on $\{0,1\}^n$ and $x, y \in \{0,1\}^n$.*

1. *If $x_1 x_2 \ldots x_i = y_1 y_2 \ldots y_i$ then i indexes a good bit of x if and only if i indexes a good bit of y.*
2. *If $x_1 x_2 \ldots x_{i-1} = y_1 y_2 \ldots y_{i-1}$ and $x_i \neq y_i$ then i indexes a good bit of at least one of x and y.*

The *total information* of x relative to D is defined to be the sum of the $I_i(x)$ over $1 \leq i \leq n$, which is equal to $-\log_2 D(x)$. Thus, the min-entropy of D is the minimum total information of any string relative to D. More generally, if $B = [i,j]$ is any block of $[1,n]$ then the *information in block B of x relative to D* is given by $I_B(x) = I_B^D(x) = -\log_2 D(x_{[i,j]}|x_{[1,i-1]})$. Clearly, $I_B(x) = I_i(x) + I_{i+1}(x) + \ldots + I_j(x)$. In particular, $I_{[1,n]}(x)$ is just the total information of x as defined above.

The *good bit indicator* of x with respect to D, denoted $\chi^D(x)$, is the n-bit sequence whose i-th bit is 1 if i indexes a good bit and is 0 otherwise. As we will see, this notion is very useful in analyzing the distribution of the information of a string in a source. The definition of $\chi^D(x)$ can be extended to strings x of length $m \leq n$ by defining $\chi^D(x)$ to be the first m bits of $\chi^D(y)$ where y is any string of length n that has x as a prefix; it is easy to see that this is independent of the choice of y and is hence well-defined.

For a bit sequence β, we use $w(\beta)$ to denote the *weight* of β, i.e., the number of 1's in the sequence.

Proposition 2.5 *Let D be an n-bit source, let $m \leq n$ and $\beta \in \{0,1\}^m$. The number of $x \in \{0,1\}^m$ such that $\chi^D(x) = \beta$ is at most $2^{w(\beta)}$.*

Proof: We prove the result by induction on m; the basis $m = 0$ is trivial. For the induction step, suppose $m \geq 1$ and β is a string of length m and write $\beta = \beta' b$ where β' has length $m-1$. Let S be the set of strings y of length m such that $\chi^D(y) = \beta$ and let S' be the set of strings y' of length $m-1$ such that $\chi^D(y') = \beta'$. By induction, S' has size at most $2^{w(\beta')}$. Any string $y \in S$ has the form $y = y'a$ where $y' \in S'$. In the case $b = 0$, the second part of proposition 2.4 implies that for each $y' \in S'$, there is at most one $a \in \{0,1\}$ such that $y'a \in S$; so $|S| \leq |S'| \leq 2^{w(\beta')} = 2^{w(\beta)}$. On the other hand, in the case $b = 1$, we have trivially $|S| \leq 2|S'| \leq 2 \cdot 2^{w(\beta')} = 2^{w(\beta)}$. □

Remark: The original definition of good bit in [NZ96], was that i indexes a good bit in x if $I_i(x) > 1$ or if $I_i(x) = 1$ and $x_i = 0$. This is slightly more restrictive than ours. Under this definition, the conclusion of the second part of proposition 2.4 can be strengthened to the assertion that i indexes a good bit of exactly one of x and y. The conclusion of proposition 2.5 also is stronger with their definition: for each $\beta \in \{0,1\}^n$ there is at most one x such that $\chi^D(x) = \beta$. We modified their definition because we do not need this stronger conclusion, and our definition will be needed for Lemma 6.1.

An n-bit source D is said to be *δ-smooth* if $w(\chi^D(x)) \geq \delta n$ for every $x \in \mathrm{supp}(D)$, that is, every string in the support of D has at least δn good bits. Clearly, any δ-smooth source is a δ-source. Intuitively, if a source is smooth, then the information of every string in the source is well dispersed.

Let α be an n-bit sequence. A segmentation π of $[1,n]$ is said to be *t-equitable with respect to α* if for each block B in π we have $w(\alpha_B) \geq t$. We observe:

Proposition 2.6 *For an n-bit source D and an n-bit string x, if π is a segmentation of $[1,n]$ that is t-equitable with respect to $\chi^D(x)$, then for each block B in π, $I_B(x) \geq t$.*

The *block-wise min-entropy of D with respect to segmentation* π is the minimum of $I_{B_j}(x)$ over all strings x and blocks B_j of π. We say that D is a *block-wise (π, b)-source* if the block-wise min-entropy of D with respect to π is at least b. When trying to determine whether D is a block-wise (π, b)-source it is useful to represent D by an edge-labeling of the bit-string tree T^π in which edge $(z|y)$ is labeled by $D(z|y)$. We call this representation the *π-tree representation* of D. In the case that π is the segmentation into n blocks of size 1, we call this the *standard tree representation* of D. We note the following obvious facts:

Proposition 2.7 *Let D be an n-bit source. Suppose π is an (n,s)-segmentation and $l = (l_1, \ldots, l_s)$ the corresponding (n,s)-composition. Then in the π-tree representation of D:*

1. *for every internal node y at depth i, the sum of $D(z|y)$ over all $z \in \{0,1\}^{l_{i+1}}$ is equal to 1;*
2. *for any leaf $x \in \{0,1\}^n$, the probability $D(x)$ is just the product of the edge labels on the path to x;*
3. *D is a block-wise (π, b)-source if and only if every edge label is bounded above by 2^{-b}.*

2.5 Carriers for Families of Distributions

In this section we present the sufficient condition for a bipartite graph to be a disperser that we will use throughout the paper.

Let V and W be sets, $\mathcal{D}_1$ be a family of distributions on V, and $\mathcal{D}_2$ be a family of distributions on W. A bipartite graph $G = (V, W, E)$ is a *$(\mathcal{D}_1, \mathcal{D}_2, \epsilon)$-carrier* if for each $D_1 \in \mathcal{D}_1$ there is a converter Λ, carried by G, such that $D_1\Lambda$ is ϵ-near to some distribution in $\mathcal{D}_2$. A $(\mathcal{D}_1, \mathcal{D}_2, \epsilon)$-carrier is said to be *strong* if there is a converter Λ carried by G such that, for every distribution $D_1 \in \mathcal{D}_1$, the transformed distribution $D_1\Lambda$ is ϵ-near to some distribution $D_2 \in \mathcal{D}_2$.

In the case that the class $\mathcal{D}_2$ is a singleton set of distribution D_2, we may use the notation $(\mathcal{D}_1, D_2, \epsilon)$-carrier.

Remark: The condition that G is a strong $(\mathcal{D}_1, \mathcal{D}_2, \epsilon)$-carrier is stronger than the condition that it is a $(\mathcal{D}_1, \mathcal{D}_2, \epsilon)$-carrier, since the latter condition does not require that there is a *single* converter Λ carried by G that "works" for all $D_1 \in \mathcal{D}_1$.

We denote by $\mathcal{T}(X, d)$ the set of distributions on X of min-entropy at least d. The next lemma shows that to construct a good disperser, it suffices to construct an appropriate carrier.

Lemma 2.2 *Let N, M, T be positive integers and let V, W be sets with $|V| = N$ and $|W| = M$. If $G = (V, W, E)$ is a $(\mathcal{T}(V, \log T), U_W, \epsilon)$-carrier, then every subset of V with cardinality T has at least $M(1-\epsilon)$ neighbors in G; thus, in particular, if $\epsilon = 1/2$, then G is an (N, M, T)-disperser.*

Proof: Assume that G is a $(\mathcal{T}(V, \log T), U_W, \epsilon)$-carrier. Let X be an arbitrary subset of V of size T, and let Y be the neighbor set of X. We need to show that $|Y| \geq M(1-\epsilon)$. Let D be the distribution that is uniform on X and 0 on $V - X$. Then $D \in \mathcal{T}(V, \log T)$, and so by the assumption on G, there is a converter Λ supported by G that converts D into a distribution $Q = D\Lambda$ that is ϵ-near to U_W. Thus, $|Q(Y) - U_W(Y)| \leq \epsilon$. Since Λ is supported by G, Q assigns positive probability only to vertices that are in the neighborhood of X, so $Q(Y) = 1$. The uniform distribution assigns probability $|Y|/M$ to Y. Thus $|Q(Y) - U_W(Y)| = 1 - |Y|/M \leq \epsilon$, i.e., $|Y| \geq M(1-\epsilon)$. □

Next we discuss the composition of converters and carriers. A straightforward induction shows the following fact.

Proposition 2.8 *Let $V_1, V_2, \ldots, V_k$ be sets and for each i between 1 and $k-1$, let Λ_i be a converter from V_i to V_{i+1}. Then the matrix product $\Lambda_1\Lambda_2\cdots\Lambda_{k-1}$ is a converter from V_1 to V_k. Furthermore, if G_i is a bipartite graph that carries Λ_i, then the composition $G_1 \circ G_2 \circ \cdots \circ G_{k-1}$ carries $\Lambda_1\Lambda_2\cdots\Lambda_{k-1}$.*

Lemma 2.3 *Let $V_1, \ldots, V_k$ be finite sets and for each i, $1 \le i \le k$, let $\mathcal{D}_i$ be a family of distributions on V_i. If for each i, $1 \le i \le k-1$, G_i on V_i, V_{i+1} is a $(\mathcal{D}_i, \mathcal{D}_{i+1}, \epsilon_i)$-carrier, then $G = G_1 \circ \cdots \circ G_{k-1}$ is a $(\mathcal{D}_1, \mathcal{D}_k, \epsilon)$-carrier where $\epsilon = \epsilon_1 + \cdots + \epsilon_{k-1}$.*

Proof: The proof is by induction on k. The case where $k = 2$ is trivial. Assume it holds for $k-1$ and we show for k.

Let $G' = G_1 \circ \cdots \circ G_{k-2}$. Then $G = G' \circ G_{k-1}$ and we know G' on V_1, V_{k-1} is a $(\mathcal{D}_1, \mathcal{D}_{k-1}, \epsilon')$-carrier for $\epsilon' = \epsilon_1 + \cdots + \epsilon_{k-2}$ by induction hypothesis.

Fix any distribution $D_1 \in \mathcal{D}_1$. We know there is a converter Λ_1 carried by G' and a distribution $D_{k-1} \in \mathcal{D}_{k-1}$ such that $\|D_1\Lambda_1 - D_{k-1}\| \le \epsilon'$ by the carrier property of G', and also there is a converter Λ_2 carried by G_{k-1} and a distribution $D_k \in \mathcal{D}_k$ such that $\|D_{k-1}\Lambda_2 - D_k\| \le \epsilon_{k-1}$ by the carrier property of G_{k-1}. Define $\Lambda = \Lambda_1\Lambda_2$. Then

$$\begin{aligned}
\|D_1\Lambda - D_k\| &\le \|(D_1\Lambda_1)\Lambda_2 - D_{k-1}\Lambda_2\| + \|D_{k-1}\Lambda_2 - D_k\| \\
&\le \|D_1\Lambda_1 - D_{k-1}\| + \|D_{k-1}\Lambda_2 - D_k\| \\
&\le \epsilon' + \epsilon_{k-1} = \epsilon,
\end{aligned}$$

where the second inequality follows from Proposition 2.3(3). Now Proposition 2.8 completes the proof. □

3 Motivating the Disperser Construction

We present our disperser construction in Section 4. While the construction is itself elementary, it is not at all clear from the description why it is a disperser. A detailed proof of this is given following the description of the construction, but the technical details of the proof hide the key intuitions. In this section, we discuss the idea behind our construction. The discussion here is not rigorous, and is intended only to aid the reader in understanding what follows.

As Lemma 2.2 stated, to obtain an explicit construction of a sufficiently good disperser, it suffices to construct a carrier that converts distributions of small min-entropy to the ones that are nearly uniform. For this discussion, let us assume $V = \{0,1\}^n, n = \log N$ and $W = \{0,1\}^m, m = \log M$, and that N, M, T are related as in the Main Theorem. Building on the ideas developed in [NZ96] and [SZ94], our construction $G_D = (V, W, E)$ of an (N, M, T)-disperser with degree polylogarithmic in N is obtained by composing two bipartite graphs. Let $Z = \{0,1\}^{sn}$ where $s = a \log n$ for some constant a. The first bipartite graph is between V and Z and gives a carrier G_A that converts any distribution on V of min-entropy $\log T$ to a distribution on Z that is close to a block-wise source with s equal-sized blocks and with block-wise min-entropy nearly $\log T$. The second is between Z and W and gives a carrier G_C that converts any block-wise source on Z as above to a nearly uniform distribution on W. Both of these constructions have degree polylogarithmic in N. G_D is defined to be $G_A \circ G_C$, and so it has degree polylogarithmic in N as well and it gives a desired carrier by Lemma 2.3.

If X and Y are arbitrary sets, a function f from X to Y can be viewed as a bipartite graph from X to Y with edge set $\{(x, f(x))|x \in X\}$. In our construction, the edge set of the carrier G_A is described by a family of functions $\mathcal{F}_A$, each mapping V to Z, and consists of the union of the graphs obtained from each function. Carrier G_C is similarly specified by a family $\mathcal{F}_C$ of functions.

The family $\mathcal{F}_C$ of functions mapping Z to W that specifies carrier G_C is obtained by adjusting parameters in the "block-wise extractor" as presented in [SZ94], which is in turn an improvement on a similar construction in [NZ96]. Generally speaking, a block-wise extractor is a function taking two input strings z and y such that if z comes from a block-wise source and y from a pure random source, then the distribution induced on the output of the function is nearly uniform. By modifying the construction in [SZ94], we obtain a block-wise extractor C such that on input a string z coming from any block-wise source on Z with s equal-sized blocks and with block-wise min-entropy nearly $\log T$, and a purely random string y of length $O(\log n)$, the distribution induced on $C(z, y)$ is close to uniform. We use each such string y to index a function f_y mapping Z to W defined by $f_y(z) = C(z, y)$. The family $\mathcal{F}_C$ is defined to be the set of all f_y's, and therefore the size of $\mathcal{F}_C$ is polynomial in n (polylogarithmic in N). The above fact about C now can be restated in terms of $\mathcal{F}_C$ as follows: for a string z selected according to a block-wise source on Z as above, if we uniformly choose a function $f_y \in \mathcal{F}_C$ at random, then the distribution of $f_y(z)$ on W is nearly uniform. This is exactly what is needed for G_C.

The main novelty of our construction is the construction of carrier G_A. Intuitively, to convert an arbitrary distribution D into a block-wise source we want to "chop up" the sequence of bits from D into a sequence of blocks so that each block has a sufficient amount of information relative to D [Zuc90, Zuc91, SZ94]. As in [NZ96], the good bit indicator is an appropriate way to measure the dispersal of information within any string from the source. As suggested by Proposition 2.6, ideally what we would like is to find an (n, s)-segmentation π such that for any string x from D, π is t-equitable with respect to $\chi^D(x)$ for some sufficiently large t.

There are a few obvious difficulties in finding such a π. First, since we don't know what the source D is, we do not know how the information of any string is distributed. Second, it may be the case that for a particular string x from a particular source D, all the information of x is concentrated in a very few bits so that the weight of $\chi^D(x)$ is too small compared to the total information of x (or to the min-entropy of D). The second problem is easily handled: it turns out that any source D is close to a smooth source D' whose min-entropy is close to that of D. So we work with D'. The solution to the first problem is this: we look for a small family of segmentations (not just one single segmentation) of the bit positions such that if a string x has enough weight in $\chi^{D'}(x)$, then at least one of the segmentations in the family is t-equitable with respect to $\chi^{D'}(x)$ for some sufficiently large t.

Abstractly, here is the combinatorial problem we want to solve: for fixed $\eta' < \eta$ and sufficiently large n, we need an explicit polynomial-sized family of segmentations Π of $[1, n]$ into $s = a \log n$ blocks for some fixed a, such that for any n-bit sequence α of weight at least n^η, there is a segmentation in Π that is $n^{\eta'}$-equitable with respect to α. (Closely related segmentation issues were studied in [Zuc90, Zuc91]. The main focus of [Zuc90, Zuc91] was on the case where η' and η are $1 - \Theta(1/\log n)$, and the solutions arose from certain new results on paths in expander graphs.)

Each segmentation $\pi \in \Pi$ into s blocks defines a function f_π mapping $V = \{0,1\}^n$ to $Z = \{0,1\}^{sn}$ as follows: $f_\pi(x)$ is obtained by splitting x into the s segments $x^1, x^2, \ldots, x^s$ corresponding to the segmentation, and then padding each x^i by $n - |x^i|$ 0's, so that each block is of length n. The family $\mathcal{F}_A$ of functions that specifies carrier G_A is the set of f_π over $\pi \in \Pi$.

4 The Disperser Construction

We are now ready to give a full description of our disperser construction. The proof of its correctness will be presented in the following sections. Note that for reference, a summary of parameters that appear in the construction and the proof appears at the end of the paper. All logarithms are to the base 2, unless otherwise specified.

Fix ξ and λ with $1 \geq \xi > \lambda \geq 0$. We wish to construct an (N, M, T)-disperser for sufficiently large N, any $T \geq 2^{(\log N)^\xi}$, and any $M \leq 2^{(\log N)^\lambda}$. We assume that $N = 2^n$ and $M = 2^m$ for integers n, m, and take $V = \{0,1\}^n$ and $W = \{0,1\}^m$. This assumption is without loss of generality since in the case that N or M is not an integer power of 2, we can take $n = \lceil \log N \rceil$, and take $m = \lceil \log M \rceil$ if $M > \frac{3}{2} 2^{\lfloor \log M \rfloor}$ and $m = \lfloor \log M \rfloor$ otherwise, and construct a $(\mathcal{T}(V, \log T), U_W, 3/4)$-carrier. Our disperser will be denoted $G_D = (V, W, E)$.

We define G_D to be the composition of two bipartite graphs G_A and G_C, defined respectively in Sections 4.1 and 4.2. Let $a = \frac{\lambda}{\log(9/8)}$ and let $s = \lceil a \log n \rceil$, $r = sn$. Define $Z = \{0,1\}^r$. The first bipartite graph G_A is between V and Z. The second, denoted G_C, is between Z and W.

4.1 Constructing $G_A(V, Z, E_A)$

Given an (n, s)-segmentation $\pi = (B_1, \ldots, B_s)$ and $x \in \{0,1\}^n$, define $f_\pi(x)$ to be the sn-bit string obtained as follows: $f_\pi(x)$ consists of the concatenation of s n-bit strings, where the i-th string consists of the concatenation of x_{B_i} and $0^{n-|B_i|}$.

For integers n, k, d with $n \geq k, d \geq 4$, we will define a family $\Pi(n, k, d)$ of (n, k)-segmentations. The edge set E_A of G_A is then defined by $\{(x, f_\pi(x)) | \pi \in \Pi(n, s, d)\}$, and thus $deg(G_A)$ is $|\Pi(n, s, d)|$. It will be convenient for the reader to think of d as $\Theta(1)$, and k as $\Theta(\log n)$.

To describe $\Pi(n, k, d)$, we first define the *balanced d-ary tree T on $[1, n]$* to be the labeled rooted tree with n leaves constructed in the following way. Let $H = \lceil \log_d n \rceil$ and let T' be the rooted d-ary tree of depth H. Let T be the smallest subtree of T' containing the root and the leftmost n leaves of T'. Thus T has $\lceil \log_d n \rceil + 1$ layers of nodes and for each node v at depth $i \geq 0$, the subtree rooted at v has at most n/d^{i-1} leaves. The nodes of T are labeled as follows: the leaves are labeled left-to-right by $1, 2, \ldots, n$ and each internal node v is labeled by the interval formed by the labels of the leaves in the subtree of v.

For a node v in the tree, we denote by $L(v)$ the largest leaf label in the subtree of v. If $u_1, u_2, \ldots, u_{k-1}$ is any sequence of vertices in T such that $L(u_i)$ is strictly increasing and $L(u_{k-1}) < n$, we associate it with the (n, k)-segmentation whose corresponding (n, k)-tower is the sequence $(0, L(u_1), L(u_2), \ldots, L(u_{k-1}), n)$. We specify below a collection $\Gamma(n, k, d)$ of such sequences of vertices; $\Pi(n, k, d)$ is then defined to be the set of all (n, k)-segmentations arising from these sequences.

For each leaf w of T, let $A(w)$ be the set of vertices v such that (1) $L(v) < w$ and (2) the parent of v belongs to the unique path joining w to the root. That is, $A(w)$ is the set of vertices v of T such that v is a left sibling of some vertex on the path joining w to the root. We notice that the vertices in $A(w)$ all have distinct $L(\cdot)$ values and they are all less than n. Order the vertices in $A(w)$ according to their $L(\cdot)$ values in increasing order. We take $\Gamma(n, k, d)$ to be the union over all leaves w of the set of $(k-1)$-element subsequences of the ordered $A(w)$. Notice that the size of any $A(w)$ is at most $(d-1)H$ and the number of subsequences of $A(w)$ is at most $2^{H(d-1)} \leq n^{d-1}$ (since $d \geq 4$). So the size of $\Gamma(n, k, d)$ (and hence the size of $\Pi(n, k, d)$) is at most n^d.

4.2 Constructing $G_C(Z, W, E_C)$

For some q that is $O(\log n)$, we will define a function C from $Z \times \{0,1\}^q = \{0,1\}^r \times \{0,1\}^q$ to $W = \{0,1\}^m$. We then define the edge set E_C of G_C as $\{(z,w) | \exists y \in \{0,1\}^q$ such that $C(z,y) = w\}$, and thus $deg(G_C)$ will be 2^q which is polynomial in n. The function C is a straightforward modification of the block-wise extractor of [SZ94]. To describe the construction, we need the following Improved Leftover Hash Lemma, which is presented as Corollary 3.4 in the full version of [SZ94]:

Lemma 4.1 *Let $t \leq n$ be nonnegative integers, D be an n-bit source of min-entropy t, $k > 0$, and $\epsilon \geq 2^{1-k}$. There is an explicit construction of an efficiently computable family F of functions mapping $\{0,1\}^n$ to $\{0,1\}^{t-2k}$, such that the distribution of $(f, f(x))$ is quasi-random within ϵ (on the set $F \times \{0,1\}^{t-2k}$), where f is chosen uniformly at random from F, and x is sampled from D. Moreover, a random element from F can be specified using $4(t-k) + O(\log n)$ random bits.*

Here "efficiently computable" means computable in time polynomial in n and in $\log \epsilon^{-1}$. (Furthermore, as shown in the full version of [SZ94], we can in fact take "efficient computability" to be NC computation: given any $x \in \{0,1\}^n$ and $4(t-k) + O(\log n)$ bits that index an arbitrary member f of F, we can compute $(f, f(x))$ on an EREW PRAM, using $n^{O(1)}$ processors and in $\log^{O(1)} n$ time. This will be useful for the parallelization of our disperser construction, as is shown at the end of this section.

A calculation shows that $\log |F| \leq 4(t-k) + 6 \log n$ in the above lemma. Furthermore, F, as constructed above, will be of the form $\{0,1\}^\ell$ for some $\ell \leq 4(t-k) + 6 \log n$. Thus, for convenience, we may take $F = \{0,1\}^{\ell'}$ for any desired $\ell' \geq \ell$, without losing the above pseudorandom property of F.

The following reformulation is more convenient for our purposes.

Corollary 4.1 *There is an explicit function*

$$\tau : \{0,1\}^n \times \{0,1\}^{4t + \lceil 6 \log n \rceil} \to \{0,1\}^{4\lceil \frac{9}{8} t \rceil + \lceil 6 \log n \rceil}$$

such that for any n-bit source D of min-entropy t ($t \geq 11$), the induced distribution $\tau(D \times U_{4t + \lceil 6 \log n \rceil})$ on $\{0,1\}^{4\lceil \frac{9}{8} t \rceil + \lceil 6 \log n \rceil}$ is quasi-random to within $2^{1-t/8}$. Furthermore, τ is computable in time polynomial in n.

Proof: We first set $k = t/8$ and $\epsilon = 2^{1-k}$ in Lemma 4.1. As mentioned above, we may now take $|F| = \{0,1\}^{\ell'}$, for any desired $\ell' \geq 4(t-k) + 6 \log n$; we choose $\ell' = 4t + \lceil 6 \log n \rceil$. We view each $y \in \{0,1\}^{4t + \lceil 6 \log n \rceil}$ as indexing a function $h_y \in F$, and define $\tau(x,y)$ to be the first $4\lceil \frac{9}{8} t \rceil + \lceil 6 \log n \rceil$ bits of the concatenation of y and $h_y(x)$. Since the length of this concatenation is $4t + \lceil 6 \log n \rceil + \lfloor 3t/4 \rfloor$ which is at least $4\lceil \frac{9}{8} t \rceil + \lceil 6 \log n \rceil$ for $t \geq 11$, we can see that τ is well-defined. $\square$

Let s be as above. Let $t_s = 32$ and $t_{k-1} = \lceil \frac{9}{8} t_k \rceil$ for $0 < k \leq s$. Set $p_k = 4t_k + \lceil 6 \log n \rceil$ for $0 \leq k \leq s$. Now, using the function τ, we define a family of functions

$$C_k : (\{0,1\}^n)^k \times \{0,1\}^{p_k} \to (\{0,1\}^n)^{k-1} \times \{0,1\}^{p_{k-1}}$$

for $1 \leq k \leq s$ as follows: for $z_1, \ldots, z_k \in \{0,1\}^n, y_k \in \{0,1\}^{p_k}$,

$$C_k(z_1 \ldots z_{k-1} z_k y_k) = z_1 \ldots z_{k-1} y_{k-1}, \text{ where } y_{k-1} = \tau(z_k, y_k).$$

Let $C^* = C_1 \circ C_2 \circ \ldots \circ C_s$ and let $q = p_s$ (thus $q = O(\log n)$). Finally, for each $z \in Z$ and $y \in \{0,1\}^q$, $C(z, y)$ is defined to be the first m bits of $C^*(z, y)$, where in C^* we view z as a series of s blocks of n bits each. To see that function C is well-defined, we notice that the length of $C^*(z, y)$ is $p_0 = 4t_0 + \lceil 6 \log n \rceil$. This is bigger than m since by definition $t_{k-1} = \lceil \frac{9}{8} t_k \rceil$ for $0 < k \leq s$ and thus $t_0 \geq (\frac{9}{8})^s t_s \geq n^\lambda t_s \geq m$.

Thus far we have seen the constructions of G_A and G_C and hence that of our final disperser $G_D = G_A \circ G_C$, and the efficiency of G_D is easily seen. Starting from Section 5, we will prove the following lemma, which clearly implies the Main Theorem:

Main Lemma: *$\forall \xi, \lambda, 1 \geq \xi > \lambda \geq 0$, $\exists n_0(\xi, \lambda)$ such that if $n \geq n_0(\xi, \lambda)$, then for any T, m such that $\log T \geq n^\xi$ and $m \leq n^\lambda$, $G_D(V, W, E)$ is a $(2^n, 2^m, T)$-disperser.*

Remark: As mentioned in Section 1.5, our construction can be parallelized. We briefly sketch how this is done. It suffices to show that with $poly(n)$ processors and $poly(\log n)$ time it is possible to compute: (a) given any $x \in V$, all its neighbors in $G_A(V, Z, E_A)$, and (b) given any $z \in Z$, all its neighbors in $G_C(Z, W, E_C)$. These two results clearly imply an efficient parallel (NC) construction of our final OR-disperser.

It is apparent from Section 4.1 that task (a) above is straightforward. As for (b), we use the fact that the function family F is constructible in NC (see the remark following Lemma 4.1), to first observe that the function τ of Corollary 4.1 is computable in NC (i.e., using poly(n) processors and poly$(\log n)$ time). Since $s = O(\log n)$, this implies that the quantity C^* defined in Section 4.2 can be computed in NC; in turn, this shows an NC construction of all the neighbors in $G_C(Z, W, E_C)$ of any given $z \in Z$.

5 Proof of the Main Lemma

Let $1 \geq \xi > \lambda \geq 0$ be arbitrary but fixed, and $n_0(\xi, \lambda)$ be a sufficiently large constant. (n_0 will be specified as the proof proceeds.) Let $n \geq n_0$, and T and m be any integers such that $\log T \geq n^\xi$ and $m \leq n^\lambda$.

As defined in Section 4, let $a = \frac{\lambda}{\log(9/8)}$, $s = \lceil a \log n \rceil$ and $r = sn$. We take $V = \{0,1\}^n, W = \{0,1\}^m$ and $Z = \{0,1\}^r$. We want to show that $G_D(V, W, E) = G_A(V, Z, E_A) \circ G_C(Z, W, E_C)$ as constructed in Section 4 is a $(2^n, 2^m, T)$-disperser.

For the proof, we define the parameters η_i for $0 \leq i \leq 3$ by $\eta_i = \xi(1 - i/3) + \lambda(i/3)$. Note that $\xi = \eta_0 > \eta_1 > \eta_2 > \eta_3 = \lambda$. Define $\delta_i = \delta_i(n) = n^{\eta_i - 1}$.

Let π^* denote the (sn, s)-segmentation of $[1, sn]$ into s equal-sized blocks each of length n. We denote by $\mathcal{B}(Z, \pi^*, b)$ the set of all block-wise (π^*, b)-sources on Z. In the next two sections, we will prove the following two lemmas:

Lemma 5.1 *$G_A = (V, Z, E_A)$ is a $(\mathcal{T}(V, \delta_0 n), \mathcal{B}(Z, \pi^*, \delta_2 n), \epsilon = 1/4)$-carrier.*

Lemma 5.2 *$G_C = (Z, W, E_C)$ is a strong $(\mathcal{B}(Z, \pi^*, \delta_2 n), U_W, \epsilon = 1/4)$-carrier.*

Remark: Both lemmas can easily be strengthened to $\epsilon = n^{-c}$ for any constant c, provided that n_0 is chosen large enough depending on c.

These two lemmas together with Lemma 2.3 imply that G_D is a $(\mathcal{T}(V, \log T), U_W, 1/2)$-carrier. The Main Lemma follows from Lemma 2.2. In what follows, we shall first examine the properties of G_A and prove Lemma 5.1 in Section 6. Then in Section 7 we show the proof of Lemma 5.2 for G_C.

6 Converting Weak Sources to Block-wise Sources

Our goal is to show that $G_A = (V, Z, E_A)$ is a $(\mathcal{T}(V, \delta_0 n), \mathcal{B}(Z, \pi^*, \delta_2 n), \epsilon)$-carrier. To do this, we fix an arbitrary δ_0-source D on $\{0,1\}^n$, and show that there is a converter Λ carried by G_A, such that $D\Lambda$ is ϵ-near to a block-wise $(\pi^*, \delta_2 n)$-source on $\{0,1\}^r$. It is important to emphasize the order of quantifiers here: we do not need one converter that works for all δ_0-sources, but can choose a different converter for each δ_0-source D.

Recall that for each vertex $x \in V$, its incident edges in G_A correspond to the (n, s)-segmentations in $\Pi(n, s, d)$. The converter we construct for D will be of a particularly simple form. For each $x \in V$ we will choose a segmentation $\pi_x \in \Pi(n, s, d)$. We then consider the edge set

$$g = \{(x, f_{\pi_x}(x)) | x \in \{0,1\}^n\} \subseteq E_A.$$

Observe that g is a function mapping V to Z, it thus induces a converter Λ^g from V to Z. We take as our converter the matrix Λ^g.

We will now choose the segmentations π_x, and then show that for the resulting converter Λ^g, $D\Lambda^g$ is ϵ-near to a block-wise $(\pi^*, \delta_2 n)$-source.

6.1 Smoothing the Distribution

Let D be the fixed δ_0-source. It turns out that the analysis in the later sections will be simplified if we know that D is a smooth source. The next lemma, basically from [NZ96], says that any source is close to a smooth source with almost the same min-entropy.

Lemma 6.1 *There is a constant $c_0 \in (0,1)$ such that if D is a δ-source on $\{0,1\}^n$ with $1/n \leq \delta \leq 1/2$, then there is an n-bit source D' that is $2^{-c_0 \delta n}$-near to D and is $\psi(\delta)$-smooth, where the function $\psi : (0,1) \to (0,1)$ is defined as $\psi(\delta) = c_0 \delta / \log \delta^{-1}$.*

Proof: Let κ denote the probability with respect to D that x has fewer than $\psi(\delta)n + 1$ good bits. We first give an upper bound on κ.

By Proposition 2.5, for any string $\beta \in \{0,1\}^n$, the number of strings $x \in \{0,1\}^n$ such that $\chi^D(x) = \beta$ is at most $2^{w(\beta)}$. Therefore, the number of strings $x \in \{0,1\}^n$ with exactly i good bits is at most $\binom{n}{i} 2^i$. Thus for any $\rho \in (0, \frac{1}{2}]$, the total number of n-bit strings x with $w(\chi^D(x)) \leq \rho n$ is at most

$$\sum_{i=0}^{\lfloor \rho n \rfloor} \binom{n}{i} 2^i \leq \rho n \binom{n}{\rho n} 2^{\rho n} \leq \rho n \left(\frac{2en}{\rho n}\right)^{\rho n} \leq \rho n \left(\frac{2e}{\rho}\right)^{\rho n}.$$

To get an upper bound on κ we let $\rho = \psi(\delta) + \frac{1}{n} \leq \frac{1}{2}$. Since every string from D has probability at most $2^{-\delta n}$, we have $\kappa \leq 2^{-\delta n} \rho n (\frac{2e}{\rho})^{\rho n}$. Elementary estimates show that if $\rho = c_0 \delta / \log \delta^{-1}$ where c_0 is sufficiently small, then $\kappa \leq 2^{-c_0 \delta n}$.

Next we define the distribution D'. Consider the standard tree representation of D. We obtain D' by modifying the probability labels of this tree. We call a leaf in the tree *bad* if the n-bit string labeling the leaf has fewer than $\psi(\delta)n + 1$ good bits; we call an internal node in the tree *bad* if all the leaves residing in the subtree of the node are bad. We call a vertex *good* otherwise. Denote by B the set of all bad vertices v such that the parent of v is good. (By definition, along any path from the root to a bad leaf, there is a unique vertex on the path belonging to B.) For each $v \in B$ we change all the edge labels in the subtree of v to $\frac{1}{2}$. The resulting distribution is D'.

It is clear that the variational distance between D and D' is at most κ since we only modify the probabilities of strings that have fewer than $\psi(\delta)n+1$ good bits. It remains to show that D' is $\psi(\delta)$-smooth. For this it suffices to show that for each vertex v in B, all the leaves in the subtree of v become good after modification.

Let us fix an arbitrary v in B and let w be the parent of v. Denote by t_1 the number of good bits in the prefix corresponding to the path from the root to w and denote by t_2 the length of the path from v down to a leaf in the subtree of v. After modification, since the first t_1 good bits are not modified and all the t_2 bits after v become good, the total number of good bits on any path from the root to a leaf passing v becomes at least $t_1 + t_2$. Since w is a good node, we know that there is a path passing through w that ends at a leaf having at least $\psi(\delta)n+1$ good bits. It is clear then $t_1 + t_2 \geq \psi(\delta)n$. □

By this lemma, there is a $\psi(\delta_0)$-smooth source D' that is $\epsilon/2$-near to D for sufficiently large n. In what follows we will show that for an appropriate g, $D'\Lambda^g$ is $\epsilon/2$-near to a block-wise $(\pi^*, \delta_2 n)$-source. By Proposition 2.3 (2) and (3), $D\Lambda^g$ will then be seen to be ϵ-near to a block-wise $(\pi^*, \delta_2 n)$-source.

Remark: The above lemma holds for our modified definition of good bit but not for the original definition of good bit of [NZ96], and was our motivation for modifying the definition.

6.2 Extracting Blocks Using Segmentations

In this section we specify the converter Λ^g by choosing for each $x \in \{0,1\}^n$ a segmentation π_x from $\Pi(n,s,d)$, the family of (n,s)-segmentations constructed in Section 4.1. This family $\Pi(n,s,d)$ was chosen to satisfy two properties. The first, which was noted earlier, is that its size is at most n^d. The second is as follows (recall the definition of t-equitable from Section 2.4):

Lemma 6.2 *Let n,d,s,t be positive integers with $n \geq d,s,t$ and let $H = \lceil \log_d n \rceil$, i.e., H is the least integer such that $d^H \geq n$. Let $\varphi \in (0,1)$. If α is an n-bit sequence with $w(\alpha) \geq \frac{2t}{\varphi^H(1-\varphi)^s}$, then there is a segmentation $\pi \in \Pi(n,s,d)$ that is t-equitable with respect to α.*

Remark. The values of the parameters above that will be most relevant to our applications are: $d = \Theta(1)$, $s = \Theta(\log n)$, and $t = n^{\Theta(1)}$.

Proof of Lemma 6.2: We define an algorithm **Segment** which takes as input an n-bit string α and d,s,t,φ as in the lemma, and computes an (n,k)-tower $Q = (Q_0 = 0, Q_1, \ldots, Q_{k-1}, Q_k = n)$ with $k \leq s$ whose corresponding (n,k)-segmentation is contained in $\Pi(n,k,d)$. The lemma follows immediately from:

Claim: If the hypotheses of the lemma are satisfied, then the segmentation π corresponding to the output Q of **Segment** is an (n,s)-segmentation that is t-equitable with respect to α.

The algorithm **Segment** is given formally below. For notational simplicity, we denote $w(\alpha_{[i,j]})$ by $w([i,j])$ and call it the *weight in the block* $[i,j]$.

The algorithm first constructs the balanced d-ary tree T on $[1,n]$ and then traverses a subset of the nodes of T in a top-down and left-to-right fashion. The integers $Q_1, Q_2, \ldots$ are generated sequentially during the traversal; the parameter q represents the length of the sequence generated so far, i.e., $Q_1, \ldots, Q_q$ have been selected. We initialize $q = 0$ and $Q_0 = 0$.

At any point of time, the algorithm looks at a particular node v_h in T at depth h, where h is an integer parameter initialized to be 0. Recall that $L(v)$ denotes the largest leaf label in the subtree of v. The interval $[Q_q + 1, L(v_h)]$ is called the *active range* from which all further Q_i will be selected. The algorithm examines left-to-right the children of v_h, denoted $C_1(v_h), C_2(v_h), \ldots, C_{d(v_h)}(v_h)$, where $d(v)$ represents the number of children of node v. (By definition, $d(v) \le d$ for all v of T.) Each iteration of the loop in the algorithm corresponds to the examination of such a node.

If the i-th child of v_h is being examined, the algorithm takes $L(C_i(v_h))$ as a possible candidate for Q_{q+1}. One of three actions is taken depending on the weight in the block $[Q_q + 1, L(C_i(v_h))]$. If the weight in the block is "too small", i.e., less than t, then $L(C_i(v_h))$ is rejected. If the weight in the block is at least t, but not "too large", i.e., less than $\varphi w([Q_q + 1, L(v_h)])$, then Q_{q+1} is chosen to be $L(C_i(v_h))$. Finally if the weight in the block is too large, i.e., at least $\varphi w([Q_q + 1, L(v_h)])$, then the active range is reduced by setting $v_{h+1} = C_i(v_h)$. Consequently the algorithm moves one level down in the tree and h is increased by 1. The loop repeats until either $i > d(v_h)$ or $q \ge s$ or $h \ge h_{max}$ (where $h_{max} = \lceil \log_d \frac{n}{t} \rceil + 1$).

Algorithm **Segment**

Input: $\alpha \in \{0,1\}^n$, integers d, s, t, and $\varphi \in (0,1)$

Construct the balanced d-ary tree T on $[1, n]$;
$h_{max} \leftarrow \lceil \log_d \frac{n}{t} \rceil + 1$.
$q \leftarrow 0; Q_0 \leftarrow 0; h \leftarrow 0; v_0 \leftarrow \mathrm{root}(T); i \leftarrow 1;$
Loop
    if $w([Q_q + 1, L(C_i(v_h))]) < t$
        then $\{i \leftarrow i + 1;\}$
    else if $t \le w([Q_q + 1, L(C_i(v_h))]) < \varphi w([Q_q + 1, L(v_h)])$
        then $\{Q_{q+1} \leftarrow L(C_i(v_h)); q \leftarrow q + 1; i \leftarrow i + 1;\}$
    else if $w([Q_q + 1, L(C_i(v_h))]) \ge \varphi w([Q_q + 1, L(v_h)])$
        then $\{v_{h+1} \leftarrow C_i(v_h); h \leftarrow h + 1; i \leftarrow 1;\}$
Until $(i > d(v_h))$ **or** $(q \ge s)$ **or** $(h \ge h_{max})$;
$k = q$;

Output: $Q = (0, Q_1, Q_2, \ldots, Q_{k-1}, n)$

The algorithm always terminates since at each iteration at least one of the i, q, h increases and therefore the loop condition will eventually be violated. It is easy to see from the description of the algorithm that at termination, the sequence of vertices $v_0, v_1, \ldots, v_h$ form a path from the root and that all the Q_j have been chosen as $L(u)$ for some u that is a left sibling of some node v_i on the path. Comparing this fact with the construction of $\Pi(n, k, d)$, we can see that the (n, k)-segmentation π corresponding to the output (n, k)-tower Q is contained in $\Pi(n, k, d)$.

We show that at the end of the algorithm, two conditions hold: $w([Q_{i-1} + 1, Q_i]) \ge t$ for $1 \le i \le k$, and $k = s$. These two conditions would imply that π is an (n, s)-segmentation and is t-equitable with respect to α, which is sufficient for the proof of the claim and the lemma. By generating Q_s (rather than stopping at Q_{s-1}), we do not have to treat the last block of π separately;

since the last block of π, $[Q_{k-1}+1, n]$ contains $[Q_{k-1}+1, Q_k]$, the first condition for $i = k$ guarantees that the last block has enough weight.

The first condition is obvious by Steps 7 and 8 in the algorithm. To prove $k = s$, we first claim that the following invariants hold at the end of each iteration:

$$\begin{aligned} w([Q_q+1, L(C_{i-1}(v_h))]) &< t, &(1)\\ w([Q_q+1, L(v_h)]) &\geq \varphi^h(1-\varphi)^q w(\alpha), &(2)\\ w([Q_q+1, L(v_h)]) &< t + n/d^{h-1}. &(3)\end{aligned}$$

(When $i = 1$, we define $L(C_0(v))$ to be the largest leaf label that is smaller than the labels of the leaves in the subtree rooted at v.) We emphasize that in the above invariants, we refer to the values of the parameters i, q and h at the end of the iteration.

We show the claim by induction on the number of iterations. These invariants hold initially before entering the loop. During an iteration, if Step 6 is executed then there is clearly no effect on the invariants (2) and (3). Invariant (1) also holds because of the precondition in Step 5. If Step 8 is executed then $w([Q_q+1, L(C_{i-1}(v_h))])$ is 0 at the end and so invariant (1) holds trivially. After this step q increases by 1, $w([Q_q+1, L(v_h)])$ decreases and h is unchanged. Thus invariant (3) holds by induction. The precondition in Step 7 guarantees that $w([Q_q+1, L(v_h)])$ is at least $1-\varphi$ times what it was and so invariant (2) also holds. Let us now consider the case that Step 10 is executed. Then the left hand side of invariant (1) is $w([Q_q+1, L(C_0(v_h))])$, which is exactly the $w([Q_q+1, L(C_{i-1}(v_h))])$ at the end of the previous iteration, and is thus less than t by induction. Since $w([Q_q+1, L(v_h)])$ is at least φ times what it was and h increases by 1, invariant (2) is also maintained. After this step, $w([Q_q+1, L(v_h)])$ is at most the sum of $w([Q_q+1, L(C_0(v_h))])$ (which is less than t by invariant (1)) and the size of the interval labeling v_h (which is at most n/d^{h-1}). So invariant (3) holds as well.

Now suppose on the contrary that the algorithm terminates with $k < s$. Then at termination we must have either $i > d(v_h)$ or $h \geq h_{max}$.

If $i > d(v_h)$, then either Step 6 or Step 8 is executed at the last iteration with $i = d(v_h)$. We note that $L(C_{d(v)}(v)) = L(v)$ for any v. If Step 6 is executed, then by the precondition of this step, $w[Q_q+1, L(v_h)] < t$. But this together with invariant (2) violates the Lemma's hypothesis that $w(\alpha) \geq 2t/(\varphi^H(1-\varphi)^s)$. The precondition of Step 8 can never be satisfied in this case since we assumed $\varphi < 1$.

If $h \geq h_{max}$, then invariants (2) and (3) together imply that $\varphi^h(1-\varphi)^q w(\alpha) < 2t$, which again violates the hypothesis about $w(\alpha)$. The proof of Lemma 6.2 is complete. □

Corollary 6.1 *Let ξ, λ, s and η_i, δ_i for $i = 0, 1$ be as defined at the beginning of Section 5. There exists an integer $d = d(\xi, \lambda)$ such that for all sufficiently large n, if D' is a $\psi(\delta_0)$-smooth n-bit source, then for each string $x \in supp(D')$ there is a segmentation $\pi \in \Pi(n, s, d)$ that is $\delta_1 n$-equitable with respect to $\chi^{D'}(x)$.*

Proof: Fix any $x \in \mathrm{supp}(D')$ and let $\alpha = \chi^{D'}(x)$. Since D' is $\psi(\delta_0)$-smooth, $w(\alpha) \geq \psi(\delta_0)n$. Let $t = \delta_1 n$. We will show that constants d and φ can be chosen to satisfy $\psi(\delta_0)n \geq 2t/(\varphi^H(1-\varphi)^s)$ for sufficiently large n so that the hypotheses in Lemma 6.2 hold. This will be sufficient for the proof of the corollary.

Since $s < 1 + a\log n$ and $H < 1 + \log n/\log d$, it suffices to have

$$\frac{c_0 n^{\eta_0 - 1}}{(1-\eta_0)\log n} n \geq \frac{2n^{\eta_1 - 1} n}{\varphi^{1+\log n/\log d}(1-\varphi)^{1+a\log n}},$$

which is equivalent to

$$
\begin{aligned}
(\eta_0-\eta_1)\log n \geq \quad & 1+\log c_0^{-1}+\log(1-\eta_0)+\log\varphi^{-1}+\log(1-\varphi)^{-1} \\
& +\log\log n+(\log\varphi^{-1}/\log d+a\log(1-\varphi)^{-1})\log n.
\end{aligned}
$$

Choosing φ sufficiently close to 0 (depending only on a, η_0, η_1 which in turn depend only on ξ and λ) and then choosing d sufficiently large (depending on φ) makes the coefficient of $\log n$ on the right less than $(\eta_0-\eta_1)$. Then the inequality holds for all sufficiently large n. For example, we can set $\varphi = 1-2^{(\eta_1-\eta_0)/3a}$ and $d = 2^{3\log\varphi^{-1}/(\eta_0-\eta_1)}$ so that the coefficient of $\log n$ on the right is $2(\eta_0-\eta_1)/3$. □

We now define π_x for $x \in \mathrm{supp}(D')$ to be the segmentation given by the corollary. For $x \notin \mathrm{supp}(D')$, define π_x to be an arbitrary segmentation of $\Pi(n,s,d)$. Then, $g: V \to Z$ defined by $g(x) = f_{\pi_x}(x)$ specifies the converter Λ^g. To complete the proof of Lemma 5.1, it now suffices to show

Lemma 6.3 *$D'\Lambda^g$ is $\epsilon/2$-near to a block-wise $(\pi^*, \delta_2 n)$-source on $\{0,1\}^r$.*

6.3 The Correctness Proof of the Conversion

We need to show that the distribution $D'\Lambda^g$ is within $\epsilon/2$ of a block-wise $(\pi^*, \delta_2 n)$-source. We start with some facts about convex combinations of distributions. Recall that we often view a distribution D on a finite set X as a (row) vector indexed by the set X. We say that a family of distributions $\mathcal{D}$ on X is *convex* if the associated set of vectors is convex. Two lemmas will be useful:

Lemma 6.4 *Let $\mathcal{D}$ be a convex family of distributions on X and suppose that $D_1, D_2, \ldots, D_k$ are distributions on X such that D_i is ϵ_i-near to the family $\mathcal{D}$. Suppose that $\lambda_1, \lambda_2, \ldots, \lambda_k$ are nonnegative reals summing to 1, and let $D = \sum_{i=1}^k \lambda_i D_i$. Then D is ϵ-near to the family $\mathcal{D}$, where $\epsilon = \sum_{i=1}^k \epsilon_i \lambda_i$.*

Proof: Let i be any integer between 1 and k. By assumption, D_i is ϵ_i-near to $\mathcal{D}$ and therefore there exists a $D_i' \in \mathcal{D}$ such that $\|D_i - D_i'\| \leq \epsilon_i$. Let $D' = \sum_{i=1}^k \lambda_i D_i'$. Since $\mathcal{D}$ is convex, $D' \in \mathcal{D}$. Furthermore,

$$\|D - D'\| \leq \sum_{i=1}^k \lambda_i \|D_i - D_i'\| \leq \sum_{i=1}^k \lambda_i \epsilon_i,$$

which concludes the proof. □

Lemma 6.5 *Let n, s be integers with $n \geq s$. Then for any (n,s)-segmentation π and real number $b > 0$, the class of block-wise (π, b)-sources on $\{0,1\}^n$ is convex.*

Proof: Let $\pi = (B_1, B_2, \ldots, B_s)$. Suppose D_1, D_2 are arbitrary block-wise (π, b)-sources on $\{0,1\}^n$ and λ_1, λ_2 are arbitrary nonnegative reals summing to 1. Let $D = \sum_{i=1}^2 \lambda_i D_i$. We want to show that for any $x \in \{0,1\}^n$ and $1 \leq k \leq s$, $D(x_{B_k} | x_{B_{[1,k-1]}}) \leq 2^{-b}$. Now,

$$
\begin{aligned}
D(x_{B_k} | x_{B_{[1,k-1]}}) &= D(x_{B_{[1,k]}})/D(x_{B_{[1,k-1]}}) \\
&= \left(\sum_{i=1}^2 \lambda_i D_i(x_{B_{[1,k]}})\right)/\left(\sum_{i=1}^2 \lambda_i D_i(x_{B_{[1,k-1]}})\right) \\
&\leq \max_i \{D_i(x_{B_{[1,k]}})/D_i(x_{B_{[1,k-1]}})\} \\
&\leq 2^{-b},
\end{aligned}
$$

where the last inequality follows from the assumption that the D_i are block-wise (π, b)-sources. $\square$

To show that the distribution $D'\Lambda^g$ is within $\epsilon/2$ of a block-wise $(\pi^*, \delta_2 n)$-source, we will define a collection of distributions $(E_\pi^* : \pi \in \Pi(n,s,d))$, and express $D'\Lambda^g$ as a convex combination of such distributions. We will then use Lemma 6.4 to upper bound the distance of $D'\Lambda^g$ from a block-wise $(\pi^*, \delta_2 n)$-source, in terms of the distances of each E_π^* from such a source.

To define the distributions E_π^*, we first define for each $\pi \in \Pi(n,s,d)$ the *segmentation class* of π to be $S_\pi = \{x \in \mathrm{supp}(D') | \pi_x = \pi\}$. The next proposition follows easily from Corollary 6.1:

Proposition 6.1 *Suppose $\pi = (B_1, \ldots, B_s) \in \Pi(n,s,d)$. Then S_π is not empty implies that $|B_i| \geq \delta_1 n$ for all $1 \leq i \leq s$. Moreover, for any $x \in S_\pi$ and $1 \leq k \leq s$, we have $D'(x_{B_k} | x_{B_{[1,k-1]}}) \leq 2^{-\delta_1 n}$, i.e., $I_{B_k}^{D'}(x) \geq \delta_1 n$.*

Let $\kappa_\pi = D'(S_\pi)$. By definition, the map g acts on all $x \in S_\pi$ as follows: segment the bits of x according to π, and pad enough 0's to each segment so that its length is n. That is, $g(x) = f_\pi(x)$ for $x \in S_\pi$. Note that while the map g is not necessarily one-to-one on $\{0,1\}^n$, it is one-to-one when restricted to any S_π. Since $\{S_\pi : \ \pi \in \Pi(n,s,d)\}$ clearly forms a partition of $\mathrm{supp}(D')$, we have $\sum_\pi \kappa_\pi = 1$.

If $\kappa_\pi > 0$, we define E_π to be the probability distribution over $\{0,1\}^n$ conditioned on S_π, i.e., for each $x \in \{0,1\}^n$,

$$E_\pi(x) = \begin{cases} \frac{D'(x)}{\kappa_\pi} & \text{if } x \in S_\pi \\ 0 & \text{otherwise.} \end{cases}$$

For $\kappa_\pi = 0$, we take E_π to be an arbitrary distribution. It is easy to see that $D' = \sum_\pi \kappa_\pi E_\pi$.

We define E_π^* to be the probability distribution on $\{0,1\}^r$ such that for $y \in \{0,1\}^r$:

$$E_\pi^*(y) = \begin{cases} E_\pi(x) & \text{if } \exists\ x \in S_\pi \text{ such that } g(x) = y \\ 0 & \text{otherwise.} \end{cases}$$

From the fact that g is one-to-one on S_π through f_π, it is easy to check that E_π^* is well-defined and that $E_\pi^* = E_\pi \Lambda^g = E_\pi \Lambda^{f_\pi}$.

We can now express $D'\Lambda^g$ as a convex combination of the E_π^*:

$$D'\Lambda^g = \sum_\pi \kappa_\pi E_\pi \Lambda^g = \sum_\pi \kappa_\pi E_\pi^*.$$

We want to use this to upper bound the distance of $D'\Lambda^g$ from a block-wise $(\pi^*, \delta_2 n)$ source. Let ϵ_π denote the distance from E_π to the family of block-wise $(\pi, \delta_2 n)$-sources on $\{0,1\}^n$. Thus for each $\pi \in \Pi(n,s,d)$ there is a block-wise $(\pi, \delta_2 n)$-source F_π on $\{0,1\}^n$ that is ϵ_π-near to E_π. Let F_π^* be the distribution on $\{0,1\}^r$ defined as follows: for $y \in \{0,1\}^r$,

$$F_\pi^*(y) = \begin{cases} F_\pi(x) & \text{if } f_\pi(x) = y \\ 0 & \text{otherwise.} \end{cases}$$

Since f_π clearly defines a one-to-one function on $\{0,1\}^n$, we can see that $F_\pi^* = F_\pi \Lambda^{f_\pi}$. Following the definition of f_π, we can also see that for any n-bit string x and $1 \leq i \leq s$, the information in the i-th block (w.r.t. π) of x relative to F_π is exactly the same as the information in the i-th block (w.r.t. π^*) of $f_\pi(x) \in \{0,1\}^r$ relative to F_π^*. Therefore, since F_π is a block-wise $(\pi, \delta_2 n)$-source on $\{0,1\}^n$, F_π^* is a block-wise $(\pi^*, \delta_2 n)$-source on $\{0,1\}^r$.

Observe:

$$\|E^*_\pi - F^*_\pi\| = \|E_\pi \Lambda^{f_\pi} - F_\pi \Lambda^{f_\pi}\| \leq \|E_\pi - F_\pi\| = \epsilon_\pi.$$

Now by Lemma 6.4 and Lemma 6.5,

Lemma 6.6 *$D'\Lambda^g$ is γ-near to a block-wise $(\pi^*, \delta_2 n)$-source on $\{0,1\}^r$, where γ is defined to be $\sum_{\pi \in \Pi(n,s,d)} \kappa_\pi \epsilon_\pi$.*

So it suffices to upper bound γ in this lemma by $\epsilon/2$. We will prove:

Lemma 6.7 *For any $\pi \in \Pi(n,s,d)$ with $\kappa_\pi > 0$, $\epsilon_\pi \leq s2^{(\delta_2 - \delta_1)n}/\kappa_\pi$.*

Combined with Lemma 6.6, we obtain $\gamma \leq |\Pi(n,s,d)| s 2^{(\delta_2-\delta_1)n}$. Thus to show $\gamma \leq \epsilon/2$ it suffices to show: $(\delta_1 - \delta_2)n \geq 1 + \log \epsilon^{-1} + \log s + \log |\Pi(n,s,d)|$. The left hand side is $n^{\Omega(1)}$ and the right hand side is bounded by a constant times $\log n$, so the desired bound on γ holds for sufficiently large n.

Thus it remains to prove Lemma 6.7.

Proof of Lemma 6.7:

Fix any $\pi \in \Pi(n,s,d)$ with $\kappa_\pi > 0$. We need to upper bound the distance of E_π from the family of block-wise $(\pi, \delta_2 n)$-sources on $\{0,1\}^n$. We first prove a general upper bound on the distance of an arbitrary n-bit source F from the family of block-wise (π, b)-sources on $\{0,1\}^n$.

The reader will find it useful to recall the definition of the π-tree representation of a distribution F on $\{0,1\}^n$. Recall that F is a block-wise (π, b)-source if and only if every edge label in this tree is at most 2^{-b}. With this in mind, we make the following definitions:

Definition 6.1 *Suppose F is an n-bit source. Let $\pi = [B_1, \ldots, B_s]$ be an (n,s)-segmentation and k be an integer satisfying $1 \leq k \leq s$. $x \in \{0,1\}^n$ is said to be* (π,b)-bad at block k relative to distribution F *if $F(x_{B_k}|x_{B_{[1,k-1]}}) > 2^{-b}$, i.e., the label of the k-th edge on the path from the root to x in T^π is greater than 2^{-b}. $x \in \{0,1\}^n$ is* (π,b)-bad relative to F *if there is a k such that x is (π,b)-bad at block k. We denote by $\beta_F(\pi,b;k)$ the probability that an n-bit string randomly chosen according to F is (π,b)-bad at block k and by $\beta_F(\pi,b)$ the probability that an n-bit string randomly chosen according to F is (π,b)-bad.*

So F is a block-wise (π,b)-source on $\{0,1\}^n$ if and only if none of the strings is (π,b)-bad. Generally, we have the following bound on the distance of F from a block-wise (π,b)-source:

Lemma 6.8 *Let F be an n-bit source, let b be a positive real number and let $\pi = [B_1, \ldots, B_s]$ be an (n,s)-segmentation in which each block has at least b elements. Then F is $\beta_F(\pi,b)$-near to a block-wise (π,b)-source.*

Proof: We will construct a block-wise (π,b)-source F' with the property that if $F'(x) \leq F(x)$ for an arbitrary $x \in \{0,1\}^n$, then x is (π,b)-bad relative to F. By Proposition 2.3(1), this implies that F' is $\beta_F(\pi,b)$-near to F.

Consider the π-tree representation for F. We modify the probability labels of this tree to obtain the distribution F'. The tree for F' must satisfy that the conditional probability assigned to each edge is at most 2^{-b}. Consider an internal node z at depth $k < s$. The edges from z correspond to binary strings of length $|B_{k+1}| \geq b$. Thus z has at least 2^b edges coming from it. Therefore it is possible to choose $F'(y|z)$ for $y \in \{0,1\}^{|B_{k+1}|}$ so that $F'(y|z) = 2^{-b}$ if $F(y|z) > 2^{-b}$,

$F(y|z) \leq F'(y|z) \leq 2^{-b}$ if $F(y|z) \leq 2^{-b}$, and the sum of $F'(y|z)$ over strings y of length $|B_{k+1}|$ is 1. Doing this for all internal nodes z yields the desired F'. It is easy to see that if x is not (π, b)-bad relative to F then the conditional probability labels on the path to x are at least as big for F' as for F, and so $F'(x) \geq F(x)$. $\square$

By Proposition 6.1, for any $\pi \in \Pi(n, s, d)$ with $\kappa_\pi > 0$, each block in π has at least $\delta_1 n$ elements. Then the above lemma tells us that $\beta_{E_\pi}(\pi, \delta_2 n) \geq \epsilon_\pi$. Since $\beta_{E_\pi}(\pi, b) \leq \sum_{k=1}^{s} \beta_{E_\pi}(\pi, b; k)$, we will complete the proof of Lemma 6.7 by showing:

$$\beta_{E_\pi}(\pi, b; k) \leq \frac{2^{b-\delta_1 n}}{\kappa_\pi}. \tag{4}$$

By definition, $\beta_{E_\pi}(\pi, b; k)$ is equal to the sum of $E_\pi(x)$ over all the $x \in \text{supp}(E_\pi)$ that are (π, b)-bad at block k relative to E_π. We know that for any $x \in \{0,1\}^n$, x is (π, b)-bad at block k relative to E_π if $E_\pi(x_{B_k} | x_{B_{[1,k-1]}}) > 2^{-b}$. Moreover, for $x \in S_\pi$, we have:

$$\begin{aligned}
E_\pi(x_{B_k} | x_{B_{[1,k-1]}}) &= \frac{E_\pi(x_{B_{[1,k]}})}{E_\pi(x_{B_{[1,k-1]}})} \\
&= \frac{D'(x_{B_{[1,k]}})}{\kappa_\pi E_\pi(x_{B_{[1,k-1]}})} \frac{D'(x_{B_{[1,k-1]}})}{D'(x_{B_{[1,k-1]}})} \\
&= D'(x_{B_k} | x_{B_{[1,k-1]}}) \frac{D'(x_{B_{[1,k-1]}})}{\kappa_\pi E_\pi(x_{B_{[1,k-1]}})} \\
&\leq \frac{2^{-\delta_1 n}}{\kappa_\pi} \frac{D'(x_{B_{[1,k-1]}})}{E_\pi(x_{B_{[1,k-1]}})},
\end{aligned}$$

where the inequality follows from Proposition 6.1. Putting these together, we have that a necessary condition for $x \in S_\pi$ to be (π, b)-bad at block k relative to E_π is:

$$E_\pi(x_{B_{[1,k-1]}}) \leq \frac{2^{b-\delta_1 n}}{\kappa_\pi} D'(x_{B_{[1,k-1]}}). \tag{5}$$

Let X be the set of all $x \in S_\pi = \text{supp}(E_\pi)$ that satisfy (5). The sum of $E_\pi(x)$ over $x \in X$ is then an upper bound on $\beta_{E_\pi}(\pi, b; k)$. Define $q = |B_{[1,k-1]}|$, and note that condition (5) depends only on the first q bits of x. Recall that $E_\pi^{(q)}$ denotes the q-bit truncation of E_π and $D'^{(q)}$ is the q-bit truncation of D'. Let Y be the set of all $y \in \{0,1\}^q$ such that $E_\pi^{(q)}(y) \leq 2^{b-\delta_1 n} D'^{(q)}(y)/\kappa_\pi$. By the definition of q-bit truncation, we know that for any $y \in \{0,1\}^q$, $E_\pi^{(q)}(y) = E_\pi(y)$ and $D'^{(q)}(y) = D'(y)$. It is then clear that the sum of $E_\pi^{(q)}(y)$ over $y \in Y$ is equal to the sum of $E_\pi(x)$ over $x \in X$. So finally we have:

$$\begin{aligned}
\beta_{E_\pi}(\pi, d; k) &\leq \sum_{y \in Y} E_\pi^{(q)}(y) \\
&\leq \sum_{y \in Y} \frac{2^{b-\delta_1 n} D'^{(q)}(y)}{\kappa_\pi} \\
&= \frac{2^{b-\delta_1 n}}{\kappa_\pi} \sum_{y \in Y} D'^{(q)}(y)
\end{aligned}$$

$$\leq \frac{2^{b-\delta_1 n}}{\kappa_\pi},$$

which establishes inequality (4), and completes the proof of Lemma 6.7. □

7 Converting Block-wise Sources to a Uniform Source

In this section we show Lemma 5.2 for G_C. The proof we present is a variant of the correctness proof of the extractor (Function C) in [NZ96].

In Section 4.2 we defined the parameters $t_s = 32$ and $q = p_s = 4t_s + \lceil 6 \log n \rceil$ (thus $q = O(\log n)$). Define Λ to be the matrix on $Z \times W$ such that for $z \in Z, w \in W$,

$$\Lambda(z, w) = |\{y \in \{0,1\}^q | \ C(z, y) = w\}|/2^q.$$

It is clear that Λ is a converter and that G_C is the support of Λ. To prove Lemma 5.2, now it suffices to show that Λ transforms every distribution in $\mathcal{B}(Z, \pi^*, \delta_2 n)$ to a distribution that is ϵ-near to U_W. Fix any $D \in \mathcal{B}(Z, \pi^*, \delta_2 n)$; we want to show that $\|D\Lambda - U_W\| \leq \epsilon$.

First we notice that, in fact,

$$\Lambda(z, w) = \sum_{y \in \{0,1\}^q} \Lambda^C((z, y), w) U_q(y),$$

where Λ^C is the converter from $Z \times \{0,1\}^q$ to W induced by the function C. Therefore $D\Lambda = (D \times U_q)\Lambda^C$. As we know, C is defined to be the m-bit truncation of C^*. Now Lemma 2.1 implies that to prove $\|(D \times U_q)\Lambda^C - U_W\| \leq \epsilon$, it suffices to show $\|(D \times U_q)\Lambda^{C^*} - U_{p_0}\| \leq \epsilon$. Furthermore, since $C^* = C_1 \circ C_2 \circ \ldots \circ C_s$, Proposition 2.2 says that this is equivalent to showing $\|(D \times U_q)\Lambda^{C_s} \ldots \Lambda^{C_2}\Lambda^{C_1} - U_{p_0}\| \leq \epsilon$.

Let k be any integer between 1 and s. For each $y \in \{0,1\}^{(k-1)n}$, let $D_{k|y}$ denote the distribution on $\{0,1\}^n$ such that for $z \in \{0,1\}^n$, $D_{k|y}(z) = D(z|y)$. Recall that for any integer $j \leq sn$, $D^{(j)}$ denotes the j-bit truncation of D. By definition,

$$D^{((k-1)n)}(y) D_{k|y}(z) = D^{(kn)}(yz) = D(yz).$$

As we know $D = D^{(sn)}$, we want to show

$$\|(D^{(sn)} \times U_{p_s})\Lambda^{C_s} \ldots \Lambda^{C_2}\Lambda^{C_1} - U_{p_0}\| \leq \epsilon. \tag{6}$$

We will prove that for each integer k between 1 and s,

$$\|(D^{(kn)} \times U_{p_k})\Lambda^{C_k} - D^{((k-1)n)} \times U_{p_{k-1}}\| \leq 2^{1-t_k/8}. \tag{7}$$

Assume this is true. Then by applying Proposition 2.3(2) and (3), a straightforward induction shows that the left hand side of (6) is at most $\sum_{k=1}^{s} 2^{1-t_k/8} \leq 2 \cdot 2^{1-t_s/8} \leq \epsilon$, where the last inequality follows from $t_s \geq 32$ and $\epsilon = 1/4$. This will complete the proof of the lemma.

So it remains to prove (7). Since D is a block-wise $(\pi^*, \delta_2 n)$-source on Z, by definition, for any $1 \leq k \leq s$ and any $y \in \{0,1\}^{(k-1)n}$, $D_{k|y}$ is an n-bit source with min-entropy $\delta_2 n$. This is at least t_k since $t_0 \leq (\frac{9}{8})^s t_s + \sum_{i=0}^{s-1} (\frac{9}{8})^i \leq \delta_2 n$ for n large enough and $t_0 > t_i$ for $1 \leq i \leq s$. Now Corollary 4.1 says that the distribution $\tau(D_{k|y} \times U_{p_k})$ on $\{0,1\}^{p_{k-1}}$ is quasi-random to within $2^{1-t_k/8}$. Finally,

$$
\begin{aligned}
& \|(D^{(kn)} \times U_{p_k})\Lambda^{C_k} - D^{((k-1)n)} \times U_{p_{k-1}}\| \\
= \; & \frac{1}{2} \sum_{y \in \{0,1\}^{(k-1)n}, u \in \{0,1\}^{p_{k-1}}} D^{((k-1)n)}(y) |\tau(D_{k|y} \times U_{p_k})(u) - U_{p_{k-1}}(u)| \\
= \; & \sum_{y \in \{0,1\}^{(k-1)n}} D(y) \|\tau(D_{k|y} \times U_{p_k}) - U_{p_{k-1}}\| \\
\leq \; & 2^{1-t_k/8} \sum_{y \in \{0,1\}^{(k-1)n}} D(y) = 2^{1-t_k/8}.
\end{aligned}
$$

This completes the proof. □

Remark: In the case that $\epsilon = n^{-O(1)}$, we can take $t_s = 8 \log \frac{1}{\epsilon} + 16$ so that q is still $O(\log n)$ and $2 \cdot 2^{1-t_s/8} \leq \epsilon$.

8 Summary of Parameters

Here we give a summary of the parameters that appeared in the construction and the proof:

$1 \geq \xi > \lambda \geq 0$;
$n_0(\xi, \lambda)$ is sufficiently large and $n \geq n_0$;
$\log T \geq n^{\xi}$ and $m \leq n^{\lambda}$;
for $0 \leq i \leq 3$, $\eta_i = \xi(1 - i/3) + \lambda(i/3)$ and $\delta_i = \delta_i(n) = n^{\eta_i - 1}$;
$a = \frac{\lambda}{\log(9/8)}$, $s = \lceil a \log n \rceil$ and $r = sn$;
$\varphi = 1 - 2^{(\eta_1 - \eta_0)/(3a)}$ and $d = 2^{(3 \log \varphi^{-1})/(\eta_0 - \eta_1)}$;
$\epsilon = 1/4$;
$t_s = 8 \log \epsilon^{-1} + 16 = 32$ and $t_{k-1} = \lceil \frac{9}{8} t_k \rceil$ for $0 < k \leq s$;
$p_k = 4t_k + \lceil 6 \log n \rceil$ for $0 \leq k \leq s$;
$q = p_s = O(\log n)$.

Concluding Remark

As mentioned in the introduction, the elegant new result of [ACRT97] shows a polynomial-time simulation of BPP using η-minimally random sources: the work of [ACR96] and our construction of OR-dispersers, are key ingredients of their work.

Acknowledgements. Aravind Srinivasan thanks David Zuckerman for their past joint work [SZ94], through which he could learn from David's deep understanding of computing with imperfect sources of randomness. Shiyu Zhou would like to thank Endre Szemerédi for introducing him to this subject. We thank Andrea Clementi for explaining to us the exciting results of [ACR96, ACR97, ACRT97], and thank the referee for her/his helpful comments.

References

[ACR96] A. E. Andreev, A. E. F. Clementi, and J. P. D. Rolim, "A new general de-randomization method", submitted to *J. ACM.* Preliminary version in *Proc. International Colloquium on Automata, Languages and Programming*, 1996, pp. 357–368.

[ACR97] A. E. Andreev, A. E. F. Clementi, and J. P. D. Rolim "Worst-case hardness suffices for derandomization: a new method for hardness-randomness trade-offs", to appear in *Proc. International Colloquium on Automata, Languages and Programming*, 1997, pp. 177–187.

[ACRT97] A. E. Andreev, A. E. F. Clementi, J. P. D. Rolim, and L. Trevisan, "Weak Random Sources, Hitting sets, and BPP Simulations", Technical Report TR97-011, *Electronic Colloquium on Computational Complexity*, 1997. URL: `http://www.eccc.uni-trier.de/eccc/`. Also in *Proc. IEEE Symposium on Foundations of Computer Science*, 1997, pp. 264–272.

[Blu86] M. Blum, "Independent Unbiased Coin Flips from a Correlated Biased Source: a Finite Markov Chain," *Combinatorica*, 6(2): 97-108, 1986.

[CG88] B. Chor and O. Goldreich, "Unbiased Bits from Sources of Weak Randomness and Probabilistic Communication Complexity," *SIAM J. Comput.*, 17(2):230-261, 1988.

[CW89] A. Cohen and A. Wigderson, "Dispersers, Deterministic Amplification, and Weak Random Sources," *Proc. IEEE Symposium on Foundations of Computer Science*, 1989, pp. 14–19.

[FLW92] A. M. Ferrenberg, D. P. Landau, and Y. J. Wong, "Monte Carlo simulations: Hidden errors from "good" random number generators," *Physical Review Letters*, 69(23):3382–3384, 1992.

[FN93] A. Fiat and M. Naor, "Implicit $O(1)$ probe search," *SIAM J. Comput.*, 22:1-10, 1993.

[HRD94] T.-s. Hsu, V. Ramachandran, and N. Dean, "Parallel implementation of algorithms for finding connected components," *Proc. DIMACS International Algorithm Implementation Challenge*, 1994, pp. 1–14.

[Hsu93] T.-s. Hsu, "Graph augmentation and related problems: theory and practice," PhD thesis, Department of Computer Sciences, University of Texas at Austin, October 1993.

[IW97] R. Impagliazzo and A. Wigderson, "P = BPP unless E has Sub-Exponential Circuits: Derandomizing the XOR Lemma", *Proc. ACM Symposium on Theory of Computing*, 1997, pp. 220-229.

[IZ89] R. Impagliazzo and D. Zuckerman, "How to Recycle Random Bits", *Proc. 30th Symposium on Foundations of Computer Science*, 1989, pp. 248-253.

[ILL89] R. Impagliazzo, L. Levin, and M. Luby, "Pseudo-Random Generation from One-Way Functions," *Proc. ACM Symposium on Theory of Computing*, 1989, pp. 12–24.

[Neu63] J. von Neumann, "Various techniques for use in connection with random digits", in von Neumann's Collected works, pp 768-770, Pergaman, New York, 1963.

[Nis96] N. Nisan, "Extracting Randomness: How and Why", *Proc. IEEE Conference on Computational Complexity (formerly "Structure in Complexity Theory")*, 1996, pp. 44–58.

[NZ96] N. Nisan and D. Zuckerman, "Randomness is Linear in Space," *Journal of Computer and System Sciences*, 52:43–52, 1996.

[San87] M. Santha, "On Using Deterministic Functions in Probabilistic Algorithms," *Information and Computation*, 74(3): 241-249, 1987.

[SV86] M. Santha and U. Vazirani, "Generating Quasi-Random Sequences from Slightly Random Sources," *Journal of Computer and System Sciences*, 33:75–87, 1986.

[Sip88] M. Sipser, "Expanders, Randomness, or Time versus Space," *Journal of Computer and System Sciences*, 36: 379-383, 1988.

[SZ94] A. Srinivasan and D. Zuckerman, "Computing with Very Weak Random Sources", *Proc. IEEE Symposium on Foundations of Computer Science*, 1994, pp. 264–275. Full version available as Technical Report TRA4/96, Department of Information Systems and Computer Science, National University of Singapore, April 1996.

[TaS96] A. Ta-Shma, "On extracting randomness from weak random sources", *Proc. ACM Symposium on Theory of Computing*, 1996, pp. 276–285.

[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," *Proc. ACM Symposium on Theory of Computing*, 1987, pp. 160–168.

[Vaz87b] U. Vazirani, "Strong Communication Complexity or Generating Quasi-Random Sequences from Two Communicating Semi-Random Sources," *Combinatorica*, 7 (4): 375-392, 1987.

[VV85] U. Vazirani and V. Vazirani, "Random Polynomial Time is Equal to Slightly-Random Polynomial Time," *Proc. IEEE Symposium on Foundations of Computer Science*, 1985, pp. 417-428. See also U. Vazirani and V. Vazirani, "Random polynomial time is equal to semi-random polynomial time", Technical Report 88-959, Department of Computer Science, Cornell University, 1988.

[WZ93] A. Wigderson and D. Zuckerman, "Expanders that Beat the Eigenvalue Bound: Explicit Construction and Applications," *Proc. ACM Symposium on Theory of Computing*, 1993, pp. 245-251.

[Zuc90] D. Zuckerman, "General Weak Random Sources," *Proc. IEEE Symposium on Foundations of Computer Science*, 1990, pp. 534-543.

[Zuc91] D. Zuckerman, "Simulating BPP Using a General Weak Random Source," *Proc. IEEE Symposium on Foundations of Computer Science*, 1991, pp. 79-89. Final version in: *Algorithmica*, 16:367-391, 1996.

[Zuc93] D. Zuckerman, "NP-complete problems have a version that's hard to approximate," *Proc. IEEE Conference on Structure in Complexity Theory*, 1993, pp. 305–312.

[Zuc96] D. Zuckerman, "Randomness-optimal sampling, extractors and constructive leader election", *Proc. ACM Symposium on Theory of Computing*, 1996, pp. 286–295.