Near-Optimal conversion of Hardness into Pseudo-Randomness

Russell Impagliazzo
Computer Science and Engineering
UC, San Diego
9500 Gilman Drive
La Jolla, CA 92093-0114
russell@cs.ucsd.edu

Ronen Shaltiel
Department of Computer Science
Hebrew University
Jerusalem, Israel
ronens@cs.huji.ac.il

Avi Wigderson
Department of Computer Science
Hebrew University
Jerusalem, Israel
avi@cs.huji.ac.il

February 11, 2003

Abstract

Various efforts ([**?**, **?**, **?**]) have been made in recent years to derandomize probabilistic algorithms using the complexity theoretic assumption that there exists a problem in $E = dtime(2^{O(n)})$, that requires circuits of size $s(n)$, (for some function s). These results are based on the NW-generator [**?**]. For the strong lower bound $s(n) = 2^{\epsilon n}$, [**?**], and later [**?**] get the optimal derandomization, $P = BPP$. However, for weaker lower bound functions $s(n)$, these constructions fall far short of the natural conjecture for optimal derandomization, namely that $bptime(t) \subseteq dtime(2^{O(s^{-1}(t))})$. The gap in these constructions is due to an inherent limitation on efficiency in NW-style pseudo-random generators.

In this paper we are able to get derandomization in almost optimal time using *any* lower bound $s(n)$. We do this by using the NW-generator in a new, more sophisticated way. We view any failure of the generator as a reduction from the given "hard" function to its restrictions on smaller input sizes. Thus, either the original construction works (almost) optimally, or one of the restricted functions is (almost) as hard as the original. Any such restriction can then be plugged into the NW-generator recursively. This process generates many "candidate" generators - all are (almost) optimal, and at least one is guaranteed to be "good". Then, to perform the approximation of the acceptance probability of the given circuit (which is the key to derandomization), we use ideas from [**?**]: we run a tournament between the "candidate" generators which yields an accurate estimate.

Following Trevisan, we explore information theoretic analogs of our new construction. Trevisan [**?**] (and then [**?**]) used the NW-generator to construct efficient extractors. However, the inherent limitation of the NW-generator mentioned above makes the extra randomness required by that extractor suboptimal (for certain parameters). Applying our construction, we show how to use a weak random souce with optimal amount of extra randomness, for the (simpler than extraction) task of estimating the probability of any event (which is given by an oracle).

1 Introduction

This paper addresses the question of hardness versus randomness trade-offs. Such results show that probabilistic algorithms can be efficiently simulated deterministically under some complexity theoretic assumptions. A number of such results are known under a "worst-case circuit complexity assumption":

The s-worst-case circuit complexity assumption: There exists a function $f = \{f_n\}$ which is computable in time $2^{O(n)}$, yet for all n, circuits of size $s(n)$ cannot compute f_n.

The conclusion we are after is of the following type: "Any probabilistic algorithm that runs in time t, can be simulated deterministically in time $T(t)$". Such results were previously proven by [**?**, **?**, **?**], and our contribution is a construction that gives a better tradeoff between the simulation quality T and the assumption strength $s(n)$.

Result Comparison:
All results assume the s-worst-case complexity hardness assumption.

Reference	Conclusion for arbitrary s	Conclusion for $s(n) = 2^{n^\epsilon}$
[**?**]	$bptime(t) \subseteq dtime(2^{O((s^{-1}(t))^2 \log t)})$	$bptime(t) \subseteq dtime(2^{O(\log^{\frac{2}{\epsilon}+1} t)})$
[**?**]	$bptime(t) \subseteq dtime(2^{O(\frac{(s^{-1}(t))^4}{\log^3 t})})^a$	$bptime(t) \subseteq dtime(2^{O(\log^{\frac{4}{\epsilon}-3} t)})$
[**?**]	$bptime(t) \subseteq dtime(2^{O(\frac{(s^{-1}(t))^2}{\log t})})$	$bptime(t) \subseteq dtime(2^{O(\log^{\frac{2}{\epsilon}-1} t)})$
this paper	$bptime(t) \subseteq dtime(2^{O(s^{-1}(t^{O(\log\log t)}))})^b$	$bptime(t) \subseteq dtime(2^{O(\log^{\frac{1}{\epsilon}} t \cdot \log\log\log t)})$
optimal[c]	$bptime(t) \subseteq dtime(2^{O(s^{-1}(t))})$	$bptime(t) \subseteq dtime(2^{O(\log^{\frac{1}{\epsilon}} t)})$

[a] Impagliazzo and Wigderson state their result only for $s(n) = 2^{\Omega(n)}$, and their result puts BPP in P, for such a lower bound.

[b] Our result is a bit better, but we cannot state it in this notation.

[c] The best we can hope for with current techniques.

1.1 Background

Following [**?**], the task of derandomizing probabilistic algorithms reduces to the problem of deterministically approximating the fraction of the inputs which a given circuit accepts. We call such machines approximators, and our task becomes constructing efficient (in terms of running time) approximators. Previous results constructed efficient approximators by constructing pseudo-random generators.[1] Indeed, with a pseudo-random generator in hand, one can easily construct an efficient approximator. Simply run the generator over all possible seeds to construct a small discrepancy set,[2] and then run the circuit over all the inputs in the discrepancy set. It is clear from this discussion that the main cost of this process comes from constructing the discrepancy set which is of size exponential in the generator's seed size.[3]

[1] Informally, a pseudo-random generator is a machine that transforms a short seed of truly random bits into a long string of bits that "appear" random to small circuits.

[2] Informally, A discrepancy set is a small set such that no small circuit can distinguish between an element chosen uniformly from the set and a truly uniform element.

[3] Another observation is that no harm is done in allowing such generators to run in time exponential in the seed length.

Yao [?] used the Blum-Micali generator, on a cryptographic assumption much stronger than the corresponding worst-case circuit complexity assumption to give the first non-trivial generator for derandomization. Nisan and Wigderson [?] weakened Yao's assumption to the following distributional circuit complexity assumption, which is still seemingly stronger than the worst-case circuit compexity assumptions above:

The v-distributional complexity hardness assumption: There exists a function $h = \{h_n\}$ which is computable in time $2^{O(n)}$, yet for all n, every circuit of size $v(n)$ computes f_n correctly on at most $1/2 + 1/v(n)$ fraction of the inputs.

Previous results using worst-case assumptions ([?, ?, ?]) focused on "hardness amplification", that is showing the v-distributional complexity hardness assumption follows from the s-worst case hardness assumption. Recently, [?] came up with an almost optimal hardness amplification scheme. Informally speaking, they show that given a function $f : \{0,1\}^n \to \{0,1\}$ that cannot be computed by circuits of size s, one can construct a function $h : \{0,1\}^{O(n)} \to \{0,1\}$ for which every circuit of size $s^{\Omega(1)}$ computes g correctly on at most $1/2 + 1/s^{\Omega(1)}$ fraction of the inputs. With h in hand they activate the NW-generator, and build an efficient approximator, especially when $s(n)$ is exponential. However, there are some inherrent limits to the NW-generator that make these derandomizations sub-optimal for functions $s(n)$ which are not exponential.

1.2 Our result

The main point of the previous section is that having pushed the hardness amplification phase to the limit, the remaining inefficiency is caused by the NW-generator. When assuming the s-worst case hardness assumption, one may hope to get a generator

$$G : \{0,1\}^{O(n)} \to \{0,1\}^{s(n)^{\Omega(1)}}$$

that fools circuits of size $s(n)^{\Omega(1)}$. However, the best result using the NW-generator takes larger seed

$$NW : \{0,1\}^{O(\frac{n^2}{\log s(n)})} \to \{0,1\}^{s(n)^{\Omega(1)}}$$

For the same task. Recall that the parameter that dominates the time of the derandomization is the seed size.

In this paper we are able to minimize the seed size to the optimal $m = O(n)$. However, we do lose something. The first loss is that we are only able to "fool" circuits of size $t = s(n)^{\Omega(\frac{1}{\log \log n})}$ rather than $s(n)^{\Omega(1)}$.[4] The second loss is that rather than constructing a discrepancy set, we construct $2^{O(n)}$ sets where at least one of them is a discrepancy set. We don't know how to find the "right" discrepancy set in the huge collection of sets. However, we will show that this collection is still useful to construct an approximator.

To explain the inherent inefficiency in the NW construction, and how we overcome it, we briefly describe it.

The NW-generator: As mentioned before, to use the NW-generator one needs a hard function h. Using the optimal hardness amplification of [?], we may assume that this function is $s(n)^{\Omega(1)}$-distributional complexity hard. The NW-generator constructs a "design", that is t sets $S_1, .., S_t$ of size n in $\{1, .., m\}$, where the size of the intersection of any two sets is at most k. It is very

[4] This means that in order to derandomize a probabilistic algorithm that runs in time t, we need $n \geq (s^{-1}(t^{\log \log n}))$ rather than $n \geq s^{-1}(t^{O(1)})$.

important observation that is not hard to prove that this requirement forces $m = \Omega(n^2/k)$. The NW-generator is a function $NW_h : \{0,1\}^m \to \{0,1\}^t$. The i'th bit in the output of NW_h is simply $h(x|_{S_i})$, where $x|_{S_i}$ stands for the n bits of x such that their indices are in S_i. The main lemma of [**?**] says that if the generator "fails" then h is easy. More precisely, the statement is that if the set $\{NW_h(x)|x \in \{0,1\}^m\}$ is not a discrepancy set for circuits of size t, then there exists a circuit of size roughly $t2^k$ which computes h. So the size of the circuit which the generator fools is $t = s/2^k$. We want $t = s^{\Omega(1)}$, we choose $k = \Theta(\log s)$. As mentioned before small intersection size forces large seed, and one ends up with $m = \Omega(n^2/\log s)$.

On one hand we must have k small, because a factor of 2^k is lost when getting t from s. On the other, small k forces large m since $m = \Omega n^2/k$. One may hope to build designs with smaller intersection by allowing a more general concept of designs, which still accommodates the NW main lemma. This possibility is ruled out in [**?**]. The only option left is to reduce the factor 2^k lost in the circuit complexity, allowing one to pick $k = \omega(\log t)$, and hence decreasing the value of $m = \Theta(n^2/k)$. This is basically the approach we use here, which leads to some interesting complications.

The new idea: One possible view of the proof of the NW-lemma is that the NW-generator specifies family of $2^{O(n)}$ functions over k bits, (which are restrictions of h to k bits). Such that if the set $\{NW_h(x)|x \in \{0,1\}^m\}$ is not a discrepency set for circuits of size t, then one of the specified functions requires circuit size $s/poly(t)$. The former proof used the fact that any function over k bits can be computed by a circuit of size 2^k, they chose $t = s/2^k$, and concluded that the set above is indeed a discrepancy set for circuits of size t. We replace that argument by considering two cases:

1. All the functions specified by the NW-generator have "small" (size $s/poly(t)$) circuits. In such a case we know that the NW set is a discrepancy set for circuits of size t, and we don't lose the 2^k factor.
2. At least one of the specified functions cannot be computed by a circuit of size $s/poly(t)$. In this case it may be that the NW set is not a discrepancy set. However, we have at hand a function on much fewer bits (k instead of n) than the original hard function, that is almost as hard. From the point of view of hardness to input size ratio, this function is harder than the one we started with. We can "plug" it to the NW-generator and enjoy the better lower bound. This approach can be used recursively until we end up with parameters that the former proof can handle.

The construction: We don't know which of the two cases happened, and even worse, in case 2, we don't know which of the $2^{O(n)}$ specified functions is the hard function. Thus, we try all possibilities. we construct sets (that are candidates to be discrepancy sets) from the initial function and all it's specified functions. We continue this recursively until we are sure that one of the functions we consider is "hard" but all specified functions are "easy". This can be shown to happen after at most $\log\log n$ levels, and at this point we have $2^{O(n)}$ candidates. This process involves some loss. At each level we lose a $poly(t)$ factor from s, and so we end up choosing $t = s^{\Theta(\frac{1}{\log\log n})}$.

Our final move is approximating a given circuit having $2^{O(n)}$ candidates where at least one of them is a discrepancy set. To do this we use an idea from [**?**]. We construct a matrix with entries for each pair of sets. For each such pair we run C on all possible xors of elements from the two sets, and compute the fraction of inputs accepted by C. Note that it if one of the two sets is a discrepancy set then the set of the xors is also a discrepancy set. This means that in the row of the discrepancy set, all entries are good approximations of the "correct" value, and hence lie in a small interval. For each of the other rows, the entry in the column of the discrepancy set is close

to the "correct" value. This means that any row in which all entries lie in a small interval contains entries which are good approximations of the fraction of inputs accepted by C. Using the above process we complete the proof.

An information theoretic analog a la Trevisan: Recently, Trevisan [**?**] used the NW-generator to construct an extractor. An ϵ-extractor of t bits from r bits using m bits is an efficiently computable function $Ext : \{0,1\}^l \times \{0,1\}^m \rightarrow \{0,1\}^t$, such that for all distributions D on $\{0,1\}^l$ having min-entropy[5] r, the distribution obtained by sampling f according to D and x uniformly from $\{0,1\}^m$ and computing $Ext(f,x)$ is at most ϵ statistical distance from the uniform distribution on t bits. Trevisan's extractor works by treating f as a function over $n = \log l$ bits, "amplifying" its hardness, and applying $NW_f(x)$. If an event T distinguishes between the output of the extractor and a uniform distribution, then it must do so for many f's. For every f such that $NW_f(\cdot)$ is "caught" by T, there is a small circuit C which uses T gates[6] , and computes f, (This is the main lemma of the NW-generator). This bounds the number of f's such that $NW_f(\cdot)$ is caught by T, and the proof is done by realizing that high min-entropy says that there are a lot of f's. We focus our attention to constructing extractors with minimal m. Trevisan's construction suffers from the same inefficiency of the NW-generator which we treat here. Namely, the size of the constructed circuit ($t2^k$) must be small compared to the min-entropy r. As explained in the previous paragraphs, the need to decrease k is met by increasing m, and one ends up with $m = O(\frac{\log^2 l}{\log r})$. We may hope that using our technique we could do with the optimal $m = O(\log l)$ for any r. This is indeed the case. However we don't get an extractor since our construction is not a generator. Instead, we get the information theoretic analog of an approximator. This is a deterministic machine that approximates the probability of any given event $T : \{0,1\}^t \rightarrow \{0,1\}$ (which the machine accesses as an oracle), using one sample from a distribution D with min entropy r. The machine runs in time $l^{O(1)}$, and $t = r^{O(\frac{1}{\log\log\log l})}$. To see the connection to extractors, note that with an extractor in hand one can perform this task by going over all strings in $\{0,1\}^m$, and using the set $\{Ext(f,x)|x \in \{0,1\}^m\}$ to approximate the given event T.

2 Definitions and History

2.1 Hard functions

We start by defining "hardness" in both worst-case complexity and distributional complexity settings.

Definition 1 *For a function $f : \{0,1\}^n \rightarrow \{0,1\}$, we define:*

1. $S(f) = min\{size(C)|$ *circuits C that compute f correctly on every input*$\}$
2. $SUC_s(f) = max\{Pr_{x \in_R \{0,1\}^n}(C(x) = f(x))|$ *circuits C of size s*$\}$
3. $ADV_s(f) = 2SUC_s(f) - 1$

When invoking The NW-generator against circuits of size $t(n)$, one needs a function $h = \{h_n\}$, with $ADV_{t(n)}(h_n) \leq \frac{1}{t(n)}$. Much work has been done on building such a function from a worst-case circuit lower bound, (see for example [**?**, **?**, **?**]). In this paper we use the current result by [**?**], which we state in our notation.

[5]The min-entropy of a distribution D is $-\log(min_x D(x))$.

[6]Note that T is simply a function $T : \{0,1\}^t \rightarrow \{0,1\}$, and so we can think about it as a gate.

Theorem 1 *[?] For every function $f : \{0,1\}^n \to \{0,1\}$, and ϵ, there exists a function $h : \{0,1\}^{4n} \to \{0,1\}$, such that, for $v = S(f)(\frac{\epsilon}{n})^{O(1)}$*

1. *h can be computed in time $2^{O(n)}$, given an oracle to f.*
2. *$ADV_v(h) \leq \epsilon$*

2.2 Generators, Discrepancy sets and Approximators

In this section we define pseudo-random generators, and machines we call approximators. It is convenient to define both using the notion of discrepancy sets.

Definition 2 *For a circuit C on t bits define: $\mu(C) = Pr_{w \in_R \{0,1\}^t}(C(w) = 1)$*

Definition 3 *A (t, ϵ)-discrepancy set, is a multi-set $D \subseteq \{0,1\}^t$, such that for all circuits C of size t:*

$$|Pr_{w \in_R D}(C(w) = 1) - \mu(C)| \leq \epsilon$$

In the above definition, we did not specify a parameter for the input size of the circuit. As far as we are concerned a circuit of size t may take t bits as input. We proceed and define pseudo-random generators. For the purpose of derandomizing probabilistic algorithms, generators may be allowed to run in time exponential in their input.

Definition 4 *A ϵ-Generator G is a family of functions[7] $G_t : \{0,1\}^{m(t)} \to \{0,1\}^t$, such that:*

1. *For all t, the set $D = \{G_r(x) | x \in \{0,1\}^{m(t)}\}$ is a (r, ϵ)-discrepancy set.*
2. *G is computable in time $2^{O(m)}$, (exponential in the size of the input).*

The existence of "good" generators implies a non-trivial deterministic simulation of probabilistic algorithms. However, the proof works by building the following device (which appears implicitly since [?] and is also used implicitly in other efforts to derandomize BPP such as [?]).

Definition 5 *A ϵ-approximator is a deterministic machine that takes as input a circuit C and outputs an approximation to $\mu(C)$, that is, a number q such that $|\mu(C) - q| \leq \epsilon$*

The following two implications are standard:

Lemma 1 *([?])*

1. *If there exists a (m, ϵ)-generator then there exists a ϵ-approximator that (on a circuit of size t) runs in time $2^{O(m(t))} t^{O(1)}$.*
2. *If there exists a $1/10$-approximator that runs in time $p(t)$ on circuits of size t, then $bptime(t) \subseteq dtime(p(t^2))$.*

Proof: (sketch)
Having a generator, one can run the given circuit on all possible outputs of the generator. This is indeed an efficient approximator. Having an approximator, and given a probabilistic algorithm $M(x, y)$, (where x is the input, and y is the random string), simply construct the circuit $C_x(y) = M(x, y)$, and approximate it's success probability. •

As seen from lemma **??**, the task of derandomizing probabilistic algorithms, reduces to constructing efficient generators. Where efficiency means small as possible seed size.

[7] In the next sections we will have m be a function of n (the input size of the hard function), rather than a function of t

2.3 The NW-generator

In this section we present the NW-generator, it's best known consequences for derandomization, and explain it's inherent inefficiency when used with a sub-exponential lower bound.

Theorem 2 *(Construction of nearly disjoint sets [?]) There exists an algorithm that given numbers n, m, t, such that $t = 2^{O(\frac{n^2}{m})}$, constructs a (n, m)-**design**, that is: sets $S_1, .., S_t$, such that:*

1. *For all $1 \leq i \leq t$, $S_i \subseteq [m]$, and $|S_i| = n$.*
2. *For all $1 \leq i < j \leq t$, $|S_i \cap S_j| \leq k = c\frac{n^2}{m}$, for some constant c.*
3. *The running time of the algorithm is exponential in m.*

Definition 6 *(The NW-generator [?]) Given some function $h = \{h_n\}$, and n, m, the NW-generator works by building an (n, m)-design, $S_1, .., S_t$. It takes as input m bits, and outputs t bits.*

$$NW_h^{n,m}(x) = (h(x|_{S_1}), .., h(x|_{S_t}))$$

The thing to do now, is prove that if one "plugs" a hard enough h into the NW-generator, it fools circuits of some size.

Lemma 2 *[?] Fix n, m, t, v, ϵ. Let $S_1, .., S_t$ be the (n, m)-design promised by theorem **??**, and let $k = c\frac{n^2}{m}$, be the promised bound on the intersection size. Let $h : \{0,1\}^n \to \{0,1\}$, be a function such that $ADV_v(h) \leq \frac{\epsilon 2^{k+1}}{v}$. The set:*

$$D = \{NW_h^{n,m}(x) |\ x \in \{0,1\}^m\}$$

is a (t, ϵ)-discrepancy set, with $t = min(2^{O(\frac{n^2}{m})}, \frac{v}{2^{k+1}})$.

The drawback in lemma **??**, is that $t = O(\frac{v}{2^k})$, and a factor of 2^k is lost when getting t from v. To cope with this k must be decreased, (particularly k must satisfy $2^k < v$). Since k is roughly $\frac{n^2}{m}$, m must be increased to roughly $\frac{n^2}{\log v}$, resulting in a generator that takes a large seed for weak lower bounds v, and an non-efficient approximator. (Recall that the running time of an approximator is exponential in the generator's seed size).

Using theorems **??**,**??** and lemma **??** with the parameters described above [**?**] prove the following theorem.

Theorem 3 *([?]) If there exists a function $f = \{f_n\}$ that is computable in time $2^{O(n)}$ and for all n, $S(f_n) \geq s(n)$, then there exists a $s^{-\Omega(1)}$-generator, $G : \{0,1\}^{O(\frac{n^2}{\log s})} \to \{0,1\}^t$, which fools circuits of size t, for $t = s^{\Omega(1)}$.*

We may still expect to have an optimal generator, that is $G : \{0,1\}^{O(n)} \to \{0,1\}^{s^{\Omega(1)}}$ which fools circuits of size $s^{\Omega(1)}$. This cannot be achieved by improving the design, as the next lemma shows that the current construction of designs is optimal.

Lemma 3 *If $S_1, .., S_t \subseteq [m]$, and for all $1 \leq i \leq t$, $|S_i| = n$, and for all $1 \leq i < j \leq t$, $|S_1 \cap S_j| \leq k$, and $t \geq \frac{n}{2k}$, then $m \geq \frac{n^2}{4k}$*

Proof: It is enough to prove the lemma for $t = \frac{n}{2k}$. Using the first two terms in the inclusion-exclusion formula we get that:

$$m \geq |\cup_{1 \leq i \leq t} S_i| \geq \sum_{1 \leq i \leq t} |S_i| - \sum_{1 \leq i < j \leq t} |S_i \cap S_j|$$

Which is the required bound, for our choice of parameters. •

[**?**] prove an information theoretic generalization of the inclusion exclusion bound. This rules out the possibility of obtaining better parameters by "relaxing" the notion of a design to a weaker one for which lemma **??** can be proven.

3 The new approximator

The main result of this paper is a construction of an approximator that "corresponds" to an almost optimal generator:

Theorem 4 *If $f = \{f_n\}$ is a function computable in time $2^{O(n)}$, such that for all n, $S(f_n) \geq s(n)$, then there exists a ϵ-approximator, such that on circuits of size $s(n)^{O(\frac{1}{\log \log n})}$ runs in time $2^{O(n)}$, with $\epsilon = s^{-\Omega(\frac{1}{\log \log n})}$.*

A particularly interesting case is $s(n) = 2^{n^\epsilon}$, for which we obtain:

$$bptime(t) \subseteq dtime(2^{O((\log t)^{\frac{1}{\epsilon}} \log \log \log t)})$$

For more general functions $s(n)$, the above equation does not seem to have a nice closed-form solution. However, since the derandomization takes time 2^n, we can pick $n = \min\{t, s^{-1}(t^{O(\log \log t)})\}$. Then since $n \leq t$, $t^{\log \log t} > t^{\log \log n}$. This gives:

Theorem 5 *Let $f = \{f_n\}$ be a function computable in time $2^{O(n)}$, such that for all n, $S(f_n) \geq s(n)$, then $bptime(t) \subseteq dtime(2^{s^{-1}(t^{O(\log \log t)})})$.*

This approximator should be compared to the one of [**?**], which is constructed by applying theorem **??** and lemma **??** in sequence. [**?**]'s approximator runs in time $2^{O(\frac{n^2}{\log s})}$ on circuits of size $s^{\Omega(1)}$.

3.1 A new lemma

In this paper we replace lemma **??**, by a new lemma, saying that: "either we can build a discrepancy set for a large t, or we have at our disposal a hard function on smaller input". We could plug this function into the NW-generator, and since it's input size n is decreased, we will be able to build designs with smaller k.

Definition 7 *Given a function $h : \{0,1\}^n \to \{0,1\}$, two sets $S_1, S_2 \subseteq [m]$, where: $|S_1| = |S_2| = n$, $k = |S_1 \cap S_2|$, and $\alpha \in \{0,1\}^{n-k}$, we define a function $b_h^{S_1,S_2,\alpha} : \{0,1\}^k \to \{0,1\}$, in the following way:*

$$b_h^{S_1,S_2,\alpha}(z) = h(z;\alpha)$$

We think of S_1, as the n input bits to h. z is placed in the bits which correspond to $S_1 \cap S_2$, and α is used to "fill" the remaining $n-k$ bits. Note that the definition is not symmetric in S_1, S_2.

Lemma 4 *Fix n, m, v, t, ϵ, such that $t < min(2v, 2^{O(\frac{n^2}{m})})$. Let $S_1, .., S_t$ be the (n, m)-design promised by theorem* **??**, *and let $k = c\frac{n^2}{m}$. Let $h : \{0,1\}^n \to \{0,1\}$, be a function such that $ADV_v(h) \leq \frac{\epsilon}{t}$. Consider the set:*

$$D = \{NW_h^{n,m}(x) |\ x \in \{0,1\}^m\}$$

If D is not a (t, ϵ)-discrepancy set then there exist some $1 \leq i, j \leq t$, and a fixing $\alpha \in \{0,1\}^{n-k}$, such that: $S(b_h^{S_i,S_j,\alpha}) \geq \frac{v}{2t}$).

Remark 1 *It is worthwhile to notice that lemma* **??** *indeed follows from lemma* **??**. *Simply take $t = \frac{v}{2^{k+1}}$. This matches the assumption about h. None of the restricted functions can require circuit complexity $\frac{v}{2t} \geq 2^k$, since they are functions over k bits. So it must be the case that D is a (t, ϵ)-discrepancy set.*

The proof of lemma **??** appears in appendix **??**.

3.2 The construction

In this section, we start presenting the new approximator. The first step will be building a machine that takes a circuit size as input, and constructs a collection of small sets, where at least one of them is a "good" discrepancy set. In the next section we deal with the problem of approximating the fraction of the inputs accepted by a given circuit with such a collection.

To build the generator we need a function $f = \{f_n\}$, such that:

1. f is computable in time $2^{O(n)}$.
2. For all n, $S(f_n) \geq s(n)$. (One can replace the "For all n" by "For infinitely many n", to get a weaker result).

Parameters for the construction:

m	-	the seed length.
t	-	the length of the "pseudo-random" string, (which is also the size of the circuit we want to fool).
ϵ	-	a bound on the error of the generator.
n	-	an input length on which f_n is hard.
s	-	the lower bound known on f_n, (that is a number such that: $S(f_n) \geq s$).

The construction works by recursively calling the procedure **construct**(l, n, s, g), (where l, n, s are integers and g is a function from $\{0,1\}^n$ to $\{0,1\}$, represented as a truth table). The first call is to **construct**$(1, n, s, f_n)$

construct(l, n, s, g)

1. Use theorem **??** to create a function $h : \{0,1\}^{4n} \to \{0,1\}$, such that if $S(g) = s$, then $ADV_v(h) \leq \frac{\epsilon}{t}$. (This can be achieved with $v = s(\frac{\epsilon}{tn})^{O(1)}$).
2. Use theorem **??** to create a $(4n, m) - design$, $S_1, .., S_t$.
3. Let $k = c\frac{(4n)^2}{m}$ be the bound on the intersection size.

4. Output $D = \{NW_h^{4n,m}(x) | x \in \{0,1\}^m\}$.
5. If $\frac{v}{2^{k+1}} \geq t$, return.
6. For all $i \neq j \in [t]$, and for all $\alpha \in \{0,1\}^{n-k}$, Call **construct(** $l+1, k, \frac{v}{2t}, b_h^{S_i,S_j,\alpha}$**)**

Note that for each instantiation of **construct**, l is the level of the instantiation in the recursion tree. The values of n, s, v, k depend only on l, and so we call them n_l, s_l, v_l, k_l respectively.

Theorem 6 *Under the following assumptions:*

1. $S(f_n) \geq s$.
2. f *is computable in time* $2^{O(n)}$.
3. $t = s^{\frac{1}{2 \log \log n}} > n$.
4. $\epsilon = t^{-O(1)}$.
5. $m = 2cn$.

The process described runs in time $2^{O(n)}$*, and at least one of the sets* D *generated during runtime is a* (t, ϵ)*-discrepancy set.*

The proof of theorem **??** appears in appendix **??**.

4 A tournament of generators

In the previous section we constructed a collection of $2^{O(n)}$ sets, where one of them is a (t, ϵ)-discrepancy set. To complete the construction of the approximator, we need to be able to approximate the success probability of a given circuit, using such a collection. We achieve this using an idea from [**?**]. (See also [**?**]).

Theorem 7 *There exists an algorithm for the following computational problem:*
Input:

- *A circuit* C *of size* t.
- *A collection of multi-sets* $D_1, .., D_l$*, such that for* $1 \leq i \leq l$*,* $D_i \subseteq \{0,1\}^t$*, and* $|D_i| \leq M$*. Moreover, at least one of the* D*'s is a* (t, ϵ)*-discrepancy set.*

Output: *A number* α*, such that* $|\alpha - \mu(C)| \leq 2\epsilon$
The algorithm solves the problem. and runs in time polynomial in t, l, M.

The proof of theorem **??** appears in appendix **??**. By applying theorems **??**,**??** in sequence we get the approximator, and prove theorem **??**.

5 An information theoretic analog a-la Trevisan

Recently, Trevisan [**?**] used the NW-generator to construct an extractor. Trevisan's suffers from the same inefficiency of the NW-generator. In this section we use our technique to build an information theoretic analog of an approximator.

Definition 8 *An ϵ-extractor is a function $Ext : \{0,1\}^l \times \{0,1\}^m \to \{0,1\}^t$, which can be computed in polynomial time, such that for all distributions Source on $\{0,1\}^l$ having min-entropy*[8] *r. The distribution obtained by applying $Ext(f,x)$, where f is sampled from Source and x is sampled uniformly from $\{0,1\}^m$ is ϵ-close*[9] *to the uniform distribution on t bits.*

Definition 9 *Given an event $T \subseteq \{0,1\}^t$ (We will think about such an event as a function $T : \{0,1\}^t \to \{0,1\}$), define: $\mu(T) = Pr_{w \in_R \{0,1\}^t}(T(w) = 1)$*

One possible use to an extractor is to approximate $\mu(T)$ for a given event $T \subseteq \{0,1\}^t$.

Definition 10 *A (ϵ, δ)-approximator is a deterministic machine App, such that for any event $T \subseteq \{0,1\}^t$, and distribution Source with min-entropy $r \geq t$,*

$$Pr_{f \in Source}\left[|App^T(f) - \mu(T)| > \epsilon\right] < \delta$$

In words, the approximator takes one sample from a distribution with min-entropy r, and uses it to approximate the probability of any event T. App uses T as an oracle.

The important parameter is the running time of the approximator. We stress that any call to the oracle T takes one time unit. The following lemma (which is an analog of lemma **??**) shows that this device is the analog of an approximator for our setting.

Lemma 5 *If there exists a ϵ-extractor (for some parameters l, m, t, r), then for all $\delta < \epsilon$ there exists a $(\frac{\epsilon-\delta}{1-\delta}, \delta)$-approximator that runs in time $l^{O(1)}2^m$.*

Proof: With an extractor in hand, construct the set $D = \{Ext(f,x) | x \in \{0,1\}^m\}$, check weather $y \in T$, for all $y \in D$, and output the proportion of y's that are in T. The distribution induced by the extractor approximates $\mu(T)$ with error at most ϵ. Therefore, the fraction of the f's such that $Ext(f,\cdot)$ gives an approximation with error greater than $\frac{\epsilon-\delta}{1-\delta}$ is bounded by δ. •

Our interest is constructing extractors with minimal m. We cannot construct an extractor using our technique, Instead, we construct efficient (in terms of running time) approximators.

Theorem 8 *For any l, r and δ there exists an (ϵ, δ)-approximator, with $t = (r + \log \delta)^{\Omega(\frac{1}{\log \log \log l})}$, which runs in time $l^{O(1)}$, and $\epsilon = (r + \log \delta)^{-\Omega(\frac{1}{\log \log \log l})}$.*

¿From the point of view of extractors, this corresponds to an extractor with $m = O(\log l)$, which is optimal. One use of an approximator is to simulate a probabilistic algorithm given a distribution with some min-entropy. For simplicity, we phrase the next theorem only for constant error.

Theorem 9 *Given a distribution Source on $\{0,1\}^l$ with min-entropy r, and a probabilistic algorithm A that runs in time q, and uses $r^{O(\frac{1}{\log \log \log l})}$ random bits, there exists a deterministic algorithm that runs in time $l^{O(1)}q$, and given one sample from Source and an input x outputs $A(x)$, with arbitrary constant error.*

This is somewhat better (for some choices of parameters) from the approximator constructed using lemma **??** from Trevisan's extractor. As this approximator runs in time $2^{\frac{\log^2 l}{\log r}} q$. The proof of theorem **??** is very similar to that of theorem **??** and appears in appendix **??**.

Acknowledgements

We thank Oded Goldreich for a conversation that started us working on this paper.

[8] The min-entropy of a distribution D is $-\log(min_x D(x))$.

[9] The distance between two distributions D_1, D_2 on X is defined to be $max_{A \subseteq X}|D_1(A) - D_2(A)|$.

A Proofs

A.1 Proof of the main lemma

Proof: (Of lemma **??**)
If D is not a (t,ϵ)-discrepancy set, then there exists a circuit A of size t such that:

$$|Pr_{w\in_R D}(A(w)=1)-\mu(A)|>\epsilon$$

By using a standard "hybrid argument" as in [**?**], we get that there exists a circuit C of size t, and $1\leq j\leq t$, such that C predicts the j'th bit of the generator's output from the previous $j-1$ of the bits, namely:

$$Pr_{w\in_R D}(C(w_1,..,w_{j-1})=w_j)>\frac{1}{2}+\frac{\epsilon}{t}$$

Choosing a random w in D, amounts to choosing a random x in $\{0,1\}^m$, and applying $NW_h^{n,m}$. This means that w_j is nothing but $h(x|_{S_j})$. We get that:

$$Pr_{x\in_R\{0,1\}^m}(C(NW_h^{n,m}(x)|_{1..j-1})=h(x|_{S_j}))>\frac{1}{2}+\frac{\epsilon}{t}$$

There exists a fixing $\beta\in\{0,1\}^{m-n}$ to the bits outside of S_j, such that:

$$Pr_{y\in_R\{0,1\}^n}(C(NW_h^{n,m}(y;\beta)|_{1..j-1})=h(y))>\frac{1}{2}+\frac{\epsilon}{t} \tag{1}$$

For $i<j$, if we set $\alpha_i=\beta|_{S_i\setminus S_j}$, we get:

$$NW_h^{n,m}(y;\beta)|_i=h((y;\beta)|_{S_i})=b_h^{S_i,S_j,\alpha_i}((y;\beta)|_{S_i\cap S_j})$$

Therefore, if it is the case that all the $b_h^{S_i,S_j,\alpha}$'s had low circuit complexity, then there are size $\frac{v}{2t}$ circuits which compute $NW_h^{n,m}(y;\beta)|_i$ for all $i<j$. Combining these with C, and using (**??**), we get a circuit D of size $t+t\cdot\frac{v}{2t}\leq v$, such that:

$$Pr_{y\in_R\{0,1\}^n}(D(y)=h(y))>\frac{1}{2}+\frac{\epsilon}{t}$$

Which is a contradiction. •

A.2 Proof of theorem ??

The theorem will follow from a sequence of claims.

Claim 1 *Using the conditions in theorem* **??**, *It is easy to get the following equations.*

- $v_l=\frac{s_l}{t^{O(1)}}$.
- $s_l=\frac{s}{t^{O(l)}}$.
- $n_l=O(\frac{n}{2^{2^{l-1}}})$

Claim 2 *The process described can be performed in time* $2^{O(n)}$.

Proof: We have already fixed $m = O(n)$. The work done in each instantiation of **construct** can be done in time $2^{O(m)} = 2^{O(n)}$. We will bound the size of the recursion tree. The degree of the recursion tree at level l is bounded by $t^2 2^{n_l}$. Having fixed $t = s^{\frac{1}{2\log\log n}}$, we note that in all levels but the last one $t^2 \leq 2^{O(n_l)}$. Otherwise,

$$\frac{v_l}{t} > t^2 > 2^{n_l} > 2^{k_l+1}$$

and the process should stop. Using the fact that for all l, $n_{l+1} \leq \frac{n_l}{2}$, we can bound the degree of the recursion tree at level l by $2^{O(n_l)} = 2^{O(\frac{n}{2^l})}$. This means that the total number of instantiations is bounded by

$$\prod_l 2^{O(\frac{n}{2^l})} = 2^{n\sum_l \frac{1}{2^l}} = 2^{O(n)}$$

•

Claim 3 *The depth of the recursion tree is bounded by* $O(\log\log n)$.

Proof: We simply have to estimate l such that $\frac{v_l}{2^{k_l}} \geq t$
Using our former equations this translates to:

$$\frac{s}{s^{O(\frac{l}{2\log\log n})} 2^{\frac{n}{2^{2^l}}}} \geq s^{\frac{1}{2\log\log n}}$$

Which is satisfied by taking $l = \Theta(\log\log n)$. •

Claim 4 *Suppose that all the sets D produced up to level $l-1$, are not (t,ϵ)-discrepancy sets, then there exists a g in level l such that $S(g) \geq s_l$.*

Proof: The proof uses induction on l. The claim is certainly true for $l = 1$. For $l > 1$, We know that in levels 1 up to l there is no (t,ϵ)-discrepancy set. Using the induction hypothesis for levels 1 up to $l-1$, we know that there is some function g_{l-1}, in level $l-1$ such that $S(g_{l-1} \geq s_{l-1}$. This means that in the same instantiation of **construct**, the function h_{l-1} had $ADV_{v_{l-1}}(h_{l-1}) \leq \frac{\epsilon}{t}$. Using lemma **??**, we get that if the D produced at the current instantiation of **construct** is not a (t,ϵ)-discrepancy set, then there exists a restriction $b = b_{h_{l-1}}^{S_i,S_j,\alpha}$, (for some choice of i,j,α), such that:

$$S(b) \geq \frac{v_{l-1}}{2t} = s_l$$

•

Proof: (Of theorem **??**) We have already bounded the running time in claim **??**. Let d be the depth of the recursion tree. ¿From claim **??** we get that if non of the D's in levels 1 up to d is a (t,ϵ)-discrepancy set then one of the g's in the last level has $S(g) \geq s_H$. And so, $ADV_{v_t}(h) \leq \frac{\epsilon}{t}$. At the last level we can afford the price of using lemma **??**. Using it we get that D is a $(\frac{v_d}{2^{k+1}},\epsilon)$-discrepancy set. Using the fact that at level d, $\frac{v_d}{2^{k+1}} \geq t$, we get that D is a (t,ϵ)-discrepancy set.
•

A.3 Proof of theorem ??

Proof: For $y \in \{0,1\}^t$, define $C_y(w) = C(w \oplus y)$. Note that for all $y \in \{0,1\}^t$, C_y is of roughly the same size as C. For $1 \leq i \leq l$, define:

$$\alpha_i(y) = Pr_{w \in_R D_i}(C(w \oplus y) = 1)$$

For $i, j \in [l]$ define:

$$\alpha_{ij} = E_{y \in_R D_j}(\alpha_i(y))$$

Let k be an index such that D_k is a (t, ϵ)-discrepancy set. For all $y \in \{0,1\}^t$, $\mu(C) = \mu(C_y)$, and $|\alpha_k(y) - \mu(C_y)| \leq \epsilon$. ¿From this we have that for all $1 \leq j \leq l$:

$$|\alpha_{kj} - \mu(C)| \leq \epsilon$$

Note that for all $i, j \in [l]$, $\alpha_{ij} = \alpha_{ji}$. This is because both amount to taking all pairs a_1, a_2 from D_i, D_j and running $C(a_1 \oplus a_2)$. The algorithm computes α_{ij} for all $i, j \in [l]$. and picks a row r such that all the numbers in $I = \{\alpha_{rj} | j \in [l]\}$ lie on an interval of length 2ϵ. It then returns α, the middle of the interval of I. Such an r exists, because k has that property. For all i, we have that $|\alpha_{ik} - \mu(C)| \leq \epsilon$, and therefore all the numbers in I, are at a distance of 3ϵ from $\mu(C)$. ¿From this we have that $|\alpha - \mu(C)| \leq 2\epsilon$. •

A.4 Proof of theorem ??

The following section is devoted to proving theorem **??** The construction and proof are almost identical to the previous sections. Given f which is sampled from *Source*, we think about it as a function $f : \{0,1\}^n \to \{0,1\}$, where $n = \log l$. We fix:

- $m = 2cn$.
- $s = \sqrt{r + \log \delta}$
- $t = s^{\frac{1}{2 \log \log n}}$.

The actual approximation is done by calling **construct(1,n,s,f)**. We end up with some $2^{O(n)} = l^{O(1)}$ sets. We will then use similar arguments to the previous sections to approximate T. We will require some new notation.

Definition 11 *For functions $f : \{0,1\}^n \to \{0,1\}$, $T : \{0,1\}^t \to B$ we define:*

1. $S_T(f) = min\{size(C) |$ *circuits C that use T-gates and compute f correctly on every input*$\}$
2. $SUC_{s,T}(f) = max\{Pr_{x \in_R \{0,1\}^n}(C(x) = f(x)) |$ *circuits C of size s that use T-gates*$\}$
3. $ADV_{s,T}(f) = 2SUC_{s,T}(f) - 1$

By T-gates, we mean that the circuit can compute the function T at the cost of one gate.

Definition 12 *Given a function $T : \{0,1\}^t \to \{0,1\}$, a (T, ϵ)-discrepancy set, is a multi-set $D \subseteq \{0,1\}^t$, such that for all $y \in \{0,1\}^t$,*

$$|Pr_{w \in_R D}(T(w \oplus y) = 1) - \mu(T)| \leq \epsilon$$

The motivation for this definition is given by the fact that when proving theorem **??**, that enabled us to approximate a given circuit T using a collection of sets where one of them is a discrepancy set, we actually used only that one of the sets is a (T, ϵ)-discrepancy set. This is because we only used the sets D to approximate specific circuits of the form $T_y(w) = T(w \oplus y)$.

Theorem 10 *(Analog of theorem **??**)*
There exists an algorithm for the following computational problem:
Input:

- *A function $T : \{0,1\}^t \to \{0,1\}$, which the algorithm may use as an oracle.*
- *A collection of multi-sets $D_1, .., D_l$, such that for $1 \leq i \leq l$, $D_i \subseteq \{0,1\}^t$, and $|D_i| \leq M$. Moreover, at least one of the D's is a (T, ϵ)-discrepancy set.*

Output: *A number α, such that $|\alpha - \mu(T)| \leq 2\epsilon$*
The algorithm solves the problem. and runs in time polynomial in l, M.

Using this notation we can rephrase theorem **??**.

Theorem 11 *(Analog of theorem **??**)*
For every function $f : \{0,1\}^n \to \{0,1\}$, and ϵ, there exists a function $h : \{0,1\}^{4n} \to \{0,1\}$, such that:

1. *h can be computed in time $2^{O(n)}$, given an oracle to f.*
2. *$ADV_{v,T}(h) \leq \epsilon$*
3. *$v = S_T(f)(\frac{\epsilon}{n})^{O(1)}$*

This theorem is essentially identical to theorem **??**. The proof works by assuming that the conclusion of the theorem is false. It then constructs a circuit that shows that the assumption is false. ¿From our point of view any copies of T that were in the first circuit are used just the same in the second circuit. The same idea is used to prove the analog of lemma **??**.

Lemma 6 *(Analog of lemma **??**)*
*Fix $T : \{0,1\}^t \to \{0,1\}$. Let n, m, v, t, ϵ, such that $t < min(2v, 2^{O(\frac{n^2}{m})})$. Let $S_1, .., S_t$ be the (n, m)-design promised by theorem **??**, and let $k = c\frac{n^2}{m}$. Let $h : \{0,1\}^n \to \{0,1\}$, be a function such that $ADV_{v,T}(h) \leq \frac{\epsilon}{t}$. Consider the set:*

$$D = \{NW_h^{n,m}(x) |\ x \in \{0,1\}^m\}$$

If D is not a (T, ϵ)-discrepancy set then there exist some $1 \leq i, j \leq t$, and a fixing $\alpha \in \{0,1\}^{n-k}$, such that: $S_T(b_h^{S_i,S_j,\alpha}) \geq \frac{v}{2t}$).

Suppose D is not a (T, ϵ)-discrepancy set. then there exists some $y \in \{0,1\}^t$ such that $T(\cdot \oplus y)$ is not fooled by D. but this circuit has size $O(t)$ when viewed as a circuit with T-gates. The proof of lemma **??** constructs a circuit for h using this circuit, and can proceed unchanged from this point. We are now ready to prove theorem **??**.

Proof: (Of theorem **??**) Consider the function f selected from the distribution *Source*. We claim that with probability $1 - \delta$, $S_T(f) \geq s$. This is because the number of circuits of size s is bounded

by $2^{s^2)}$. and the fact that *Source* has min-entropy r, says that any set of size 2^{s^2} has probability at most $2^{s^2}2^{-r} < \delta$. Assuming that an f such that $S_T(f) \geq s$ was selected from the source, we can use lemma **??** recursively, as in theorem **??** to conclude that one of the sets D constructed by the process is a (T, ϵ)-discrepancy set of size $2^{O(n)}$, with element size $t = s^{\Omega(\frac{1}{\log\log n})} = (r + \log\delta)^{\Omega(\frac{1}{\log\log\log l})}$. Using theorem **??** we can approximate the probability of T. •