Paper Summary: Extractors and Pseudorandom Generators

Original work by
Luca Trevisan
University of California at Berkeley

Summarized by
Rachit Garg - CS14B050
IIT Madras

May 15, 2018

Contents

1 Introduction 1

2 Some Definitions 2

3 Prior Work 3

4 Techniques Overview 4

5 Impagliazzo and Wigderson PRG to Extractor 4

6 Nisan Wigderson PRG to Extractor 5

7 Follow up work by Raz, Reingold and Vadhan 8

1 Introduction

Extractors are algorithms that convert a weakly random distribution into an almost uniform distribution by using additional truly random bits. The author showed how to construct such an explicit extractor using a particular pseudorandom generator construction. This construction improved on the previous constructions and was simpler than them. Since pseudorandom generators are known for computational indistinguishability, using its construction to show a connection to an explicit extractor which guarantees statistically close distributions was an interesting idea that this paper explored.

2 Some Definitions

k-source random variables: We say that (the distribution of) a random variable X of range $\{0,1\}^n$ has min entropy at least k if for every $x \in \{0,1\}^n$ it holds $Pr[X = x] \leq 2^k$. The random variable on such distributions is known as having k source. We use such sources to characterize our weakly random distribution.

Seeded extractors: A function is called a (k, ϵ) extractor if:

$$Ext : \{0,1\}^n \times \{0,1\}^t \to \{0,1\}^m$$
$$\forall X \ H_{\infty(X)} \geq k \text{ and } \Delta(Ext(X, \mathbb{U}_t), \mathbb{U}_m) \leq \epsilon$$

where $H_{\infty(X)} \geq k$ implies that the min entropy is greater than equal to k.
Extractors are used in theory to simulate BPP(error on both sides) algorithms. Ideally, what we would like the extractor to do is that make any algorithm A that works correctly when fed perfectly random bits $\mathbb{U}_m$, and produce a new algorithm A' that will work even if it is fed random bits $X \in \{0,1\}^n$ that come from a weak random source.

Disperser: Dispersers capture a weaker notion than extractors. A function is called a (k, ϵ) disperser if:

$$Disp : \{0,1\}^n \times \{0,1\}^t \to \{0,1\}^m$$
$$\forall X \ H_{\infty(X)} \geq k \text{ and } |Supp(Disp(X, \mathbb{U}_t))| \geq (1 - \epsilon)2^m$$

where $H_{\infty(X)} \geq k$ implies that the min entropy is greater than equal to k.
While extractors can be used to simulate BPP algorithms with a weak random source, dispersers can be used to simulate RP(error on one side) algorithms with a weak random source.

Pseudorandom Generator: Random variables X and Y with the same range $\{0,1\}^n$ are (S, ϵ)-indistinguishable if for every $T : \{0,1\}^n \to \{0,1\}$ computable by a circuit of size S it holds:

$$|Pr[T(X) = 1] - Pr[T(Y) = 1]| \leq \epsilon$$

A pseudorandom generator is an algorithm $G : \{0,1\}^t \to \{0,1\}^m$ where $t << m$ and $G(\mathbb{U}_t)$ is (S, ϵ)-indistinguishable from $\mathbb{U}_m$, for large S and small ϵ.

Trevisan in this paper discovered that the NisanWigderson pseudorandom generator **?**, previously only used in a computational setting, could be used to construct extractors. For certain settings of the parameters, Trevisans extractor is optimal and improves on previous constructions. Trevisans extractor improves over previous constructions in the case of

extracting a relatively small number of random bits (e.g., extracting $k^{1-\alpha}$ bits from source with k bits of randomness, where $\alpha > 0$ is an arbitrarily small constant) with a relatively large statistical difference from uniform distribution. More formally it is stated that:

Theorem 2.1. *There is an algorithm that on input parameters $n, k \leq n, 36 \leq m < k/2, 0 < \epsilon < 2^{-k/12}$ computes in poly$(n, 2^t)$ time a (k, ϵ)-extractor*

$$Ext : \{0,1\}^n \times \{0,1\}^t \to \{0,1\}^m \text{ where } t = O(\frac{(\log n/\epsilon)^2}{\log(k/2m)} \cdot e^{\frac{\ln m}{\log(k/2m)}})$$

3 Prior Work

There have been several attempts to study the theory of extractors. Ideally we want that we extract a lot of randomness using very few truly random bits. There are a few bounds on the parameters n, t, m which were studied to show how well we can achieve this. The main question that was studied in the field of extractors was about simulation of randomized algorithms using weak random sources can be stated as follows: suppose that for every n we have access to a $k(n)$-source, and that we are given a polynomial-time randomized algorithm that we want to simulate given only one access to one of the sources: what is the most slowly growing function $k(\cdot)$ such that we can have polynomial-time simulations? Table [1] mentions the comparison of the result of this paper with prior work. This paper shows results that construct an extractor with a better bound on the additional randomness as compared to prior work. It is also worth noting that this paper constructs an extractor rather than a disperser constructed by **??**.

Reference	**Min entropy** k	**Output length** m	**Additional randomness** t
?	$n - a$	$n - \Theta(a)$	$O(a)$
?	$\Omega(n)$	$(1-\alpha)k$	$O(\log n)$
?	any k	k	$O((\log n)^9)$
?	$n^{\Omega(1)}$	$k^{1-\alpha}$	$O(\log n \log\log n)$
? (disperser)	$n^{\Omega(1)}$	$k^{1-\alpha}$	$O(\log n)$
? (disperser)	any k	$k-$ poly$(\log n)$	$O(\log n)$
This Paper	$n^{\Omega(1)}$	$k^{1-\alpha}$	$O(\log n)$
	any k	$k^{1-\alpha}$	$O((\log^2 n)/\log k)$
Optimal non-explicit constructions	any k	k	$O(\log n)$

Table 1: Comparison of the extractor parameters to prior work

4 Techniques Overview

The paper contained two broad techniques. First involved a connection between a certain kind of psudorandom generator and an extractor. The main contribution is the statement of the result, rather than its proof, since it involves a new, more general, way of looking at pseudorandom generator constructions. The Impagliazzo-Wigderson generator **?** is used in their construction. The analysis of its constructions shows that if the predicate is hard, then it is also computationally hard to distinguish the output of the generator from the uniform distribution. This implication is proved by means of a reduction showing how a circuit that is able to distinguish the output of the generator from the uniform distribution can be transformed into a slightly larger circuit that computes the predicate. The paper mentions a stronger assumption of the predicate being chosen randomly and claims that under this assumption the output is statistically close to uniform.
Second technique involves converting the Nisan-Wigderson generator **?** to an extractor. This generator is more simpler and easier to analyse. When we move the proof along similar lines as in the Impagliazzo-Wigderson generator we see that at one point the proof fails as a bound is not satisfied.[See 6 for further explanation]. To fix this the author uses error correcting codes to achieve a good extractor with better parameters to prior work and the Impagliazzo-Wigderson generator.

5 Impagliazzo and Wigderson PRG to Extractor

The main theorem by Impagliazzo and Wigderson in their paper **?** was that assuming there exists a family of predicates $P_l : \{0,1\}^l \to \{0,1\}$ that can be decided in time $2^{O(l)}$ and are hard i.e. have circuit complexity at least $2^{\gamma l}$ for some $\gamma > 0$. Then for every constant $\epsilon > 0$ and m there exists a $(O(m), \epsilon)$ PRG and $P = BPP$.
This was proved in their paper using a lemma as follows:

Lemma 5.1. *Suppose we have an oracle access to a predicate $P : \{0,1\}^l \to \{0.1\}$ and there exists an algorithm that computes a function $IW_P : \{0,1\}^t \to \{0,1\}^m$ in poly(m) time, where $t = O(l)$ and $m = 2^{\alpha l}$ where $0 < \alpha < \delta$ such that for every $T : \{0,1\}^m \to \{0,1\}$ if*

$$|Pr[T(IW_P(\mathbb{U}_t)) = 1] - Pr[T(\mathbb{U}_m) = 1]| > \epsilon$$

then P can be computed by a circuit A, that uses T-gates(meaning T is computed with unit cost) and whose size is at most $2^{\delta l}$.

We can see how this lemma gives the main theorem as if we have access to predicates that are hard that have complexity atleast $2^{2\delta l}$, then we can look at those T that have size $2^{\delta l}$. If for them the PRG property is not satisfied. (i.e. if $(2^{2\delta l}, \epsilon)$ distinguishable) $|Pr[T(IW_P(\mathbb{U}_t)) = 1] - Pr[T(\mathbb{U}_m) = 1]| > \epsilon$. This gives a contradiction as the statement of the lemma implies that P can be computed by a circuit of size at most $2^{2\delta l}$ and hence

not hard.
Next, they use the property of the lemma to show that if the predicate is random instead of hard we can use it to create an extractor. We note here that the notion of statistical indistinguishability follows from the fact that the statement of the lemma considers every T.

Lemma 5.2. *The function* $Ext : \{0,1\}^n \times \{0,1\}^t \to \{0,1\}^m$ *defined as:*

$$Ext(x,s) = IW^{(m)}_{\langle x \rangle}(s)$$

where $\langle x \rangle : \{0,1\}^l \to \{0,1\}$, $n = 2^l$ *and* $t = O(l)$ *and* $m = n^\alpha$. *Then extractor is a* $(m\delta n^\delta + \log(1/\epsilon), 2\epsilon)$ *- extractor.*

The main idea here is that we count the number of strings for which we can distinguish with probability greater than ϵ. The count of such strings using our choice of parameters is less than $\epsilon \cdot 2^k$(the parameters are chosen such) because the size of the circuit that computes $\langle x \rangle$ is fixed. (From the statement of lemma 5.1, we have that there is a circuit of size $2^{\delta l}$ that uses T-gates and computes $\langle x \rangle$, hence the count of all possible circuits is a bound on $\langle x \rangle$). When the random variable is chosen from a k-source, we can conclude that the probabilty that we choose a string that has probability greater than ϵ is bounded by ϵ. We can see using a markov argument that the statistical difference is less than 2ϵ over all X. Hence it is a valid $(m\delta n^\delta + \log(1/\epsilon), 2\epsilon)$ extractor.

6 Nisan Wigderson PRG to Extractor

The construction in Nisan Wigderson PRG **?** moves along similar lines(5.1) and is proved by means of a reduction that shows that if T is a test that distinguishes the output of the generator with predicate P from uniform, then there is a small circuit with one T-gate that approximately computes P. That is, the circuit computes a predicate that agrees with P on a fraction of inputs noticeably bounded away from $1/2$. As in the previous case we define a bad set of strings where the statistical distance is greater than ϵ. But previously we could say that each circuit should give a different bad string, and hence the size of the strings with the bad set is bounded by the number of possible circuit. But now the result states that the circuit with a fixed size can only approximately evaluate $\langle x \rangle$. Hence for each such $\langle x \rangle$ we can say that there is a circuit of size S that describes a string whose Hamming distance from $\langle x \rangle$ is noticeably less than $1/2$. Since there are about 2^S such circuits, the total number of bad strings is at most 2^S times the number of strings that can belong to a Hamming sphere of radius about $1/2$. This is the only point where the proof breaks down. Hence the idea here is that we bound the number of circuits in a hamming ball of radius almost half by using error correcting codes. This completes the proof. We assume the following lemma to be true.

Lemma 6.1. *(Error Correcting Codes). For every n and δ there is a polynomial-time computable encoding $EC : \{0,1\}^n \to \{0,1\}^{\tilde{n}}$ where $\tilde{n} = poly(n, 1/\delta)$ such that every ball of Hamming radius $(1/2 - \delta)\tilde{n}$ in $\{0,1\}^{\tilde{n}}$ contains at most $1/\delta^2$ codewords. Furthermore $\tilde{n}$ can be assumed to be a power of 2.*

Their are many constructions of the error correcting code that achieve the requirements. In particular one can use a Reed-Solomon code concatenated with a Hadamard code. We assume that such a construction is available to us.

We now look at the definition for (m, l, a) design which will be useful for the construction of the generators.

Design: For positive integers $m, l, a \leq l$, and $t \geq l$, $a(m, t, l, a)$ design is a family $S = S_1, ..., S_m$ of sets such that

- $S_i \subseteq [t]$,
- $|S_i| = l$,
- $\forall i \neq j \in [m], |S_i \cap S_j| \leq a$.

Lemma 6.2. *For every positive integers m, l, and $a \leq l$ there exists a (m, t, l, a) design where $t = e^{\frac{\ln m}{a}+1} \cdot \frac{l^2}{a}$. Such a design can be computed deterministically in $O(2^t m)$ time.*

Nisan and Wigderson showed an explicit construction in their paper. **?**. They showed this construction for $a = \log m$.

We now describe a notation that will be useful in defining the Nisan Wigderson generator. If $S \subseteq [t]$, with $S = \{s_1, ..., s_l\}$ (where $s_1 < s_2 < \ldots < s_l$) and $y \in \{0,1\}^t$, then we denote by $y_{|S} \in \{0,1\}^l$ the string $y_{s_1} y_{s_2} \ldots y_{s_l}$. We now define the NW generator.

Definition 6.3. For a function $f : \{0,1\}^l \to \{0,1\}$ and an (m, t, l, a)-design $S = (S_1, ..., S_m)$, the Nisan-Wigderson generator $NW_{f,S} : \{0,1\}^t \to \{0,1\}^m$ is defined as:

$$NW_{f,S}(y) = f(y_{|S_1}) \ldots f(y_{|S_m})$$

The intuition behind the Nisan and Wigderson generator is that if f is hard-on average then f being evaluated at random points will be hard to distinguish by a bounded adversary. In order to reduce the seed, evaluation points are not chosen independently but based on a small corelation generated by the design. Nisan and Wigderson show that such a construction is indeed a PRG. The main lemma used is as follows:

Lemma 6.4. *Let S be an (m, l, a)-design, and $T : \{0,1\}^m \to \{0,1\}$. Then there exists a family $\mathbb{G}_T$ (depending on T and S) of at most $2^{m2^a + \log m + 2}$ functions such that for every*

function $f : \{0,1\}^l \to \{0,1\}$ *satisfying:*

$$|Pr_{y\in\{0,1\}^t}[T(NW_{f,S}(y)) = 1] - Pr_{r\in\{0,1\}^m}[T(r) = 1]| \geq \epsilon$$

then there exists a function $g : \{0,1\}^l \to \{0,1\}, g \in \mathbb{G}_T$ *, such that* $g(\cdot)$ *approximates* $f(\cdot)$ *within* $1/2 + \epsilon/m$*.*

Nisan and Wigderson proved a similar lemma along these lines and showed that if the function were not a PRG then by using a hybrid argument one could construct a circuit that predicts a bit of the output by seeing its preceding bits. They then use this circuit to contradict the hardness of f.

Trevisan showed that this lemma along with the error correcting code is enough to prove that the generator is an extractor. To see this observe that every bad string($\langle x\rangle$) will now have a neighbour in $\mathbb{G}_T$ that is within $1/2 + \epsilon/m$ distance from the encoding of $\langle x\rangle$. Hence the number of bad strings are bounded by the total number of strings in $\mathbb{G}_T$ times the number of codewords that are within the ball of length $1/2 - \epsilon/m$. Carefully choosing the parameters proves that the Nissan Wigderson generator is a $(k, 2\epsilon)$ extractor for $t = O(e^{\log(k/2m)} \cdot \frac{l^2}{\log(k/2m)})$.
The final constructor of the extractor function is as follows:

The construction has parameters $n, k \leq n$, $36 \leq m \leq k/2$ and $0 < \epsilon < 2^{-k/12}$. It can be verified that the constraints on the parameters imply that $2 + 3\log m + 3\log(1/\epsilon) < k/2$ (because we have $k/4 > 3\log 1/\epsilon$ and $k/4 \geq m/2 \geq 2 + 3\log m$ for $m \geq 36$).
Let $EC : \{0,1\}^n \to \{0,1\}^{\tilde{n}}$ be as in Lemma 6.1, with $\delta = \epsilon/m$, so that $\tilde{n} = poly(n, 1/\epsilon)$, and define $l = \log \tilde{n} = O(\log n/\epsilon)$.
For an element $u \in \{0,1\}^n$, define $\tilde{u} = \langle EC(u)\rangle : \{0,1\}^l \to \{0,1\}$. Let $S = S_1, ..., S_m$ be as in Lemma 6.2, such that

- $S_i \subseteq [t]$,
- $|S_i| = l$,
- $\forall i \neq j \in [m], |S_i \cap S_j| \leq a = \log(k/2m)$
- $t = O(e^{\log(k/2m)} \cdot \frac{l^2}{\log(k/2m)})$.

Then we observe that from our choice $m > t$ and $Ext : \{0,1\}^n \times \{0,1\}^t \to \{0,1\}^m$ as

$$Ext(u, y) = NW_{\tilde{u},S}(y) = \tilde{u}(y_{|S_1}) \ldots \tilde{u}(y_{|S_m}).$$

7 Follow up work by Raz, Reingold and Vadhan

After Trevisan's paper was published further work was done by Raz et al **?** who devised an improvement on their constructions. Along with some bound improvements they also have an improvement on the running time of the algorithm from $poly(n, 2^t)$ to $poly(n, t)$. They state that Trevisans extractor performs poorly when one wants to extract more than a small fraction of the randomness from the weak random source, or when one wants to achieve a small statistical difference from uniform distribution. They give two ideas, the first idea allows one to extract more than a small fraction of the randomness from a weakly random source. The second idea improves Trevisans construction in the case where the output bits are required to be of a relatively small statistical difference from uniform distribution.