Hard Sets and Pseudo-Random Generators for Constant Depth Circuits

Manindra Agrawal

Department of Computer Science
IIT Kanpur
Kanpur 208016, India
manindra@iitk.ac.in

Abstract. It is shown that the existence of a set in E that is hard for constant depth circuits of subexponential size is equivalent to the existence of a true pseudo-random generator against constant depth circuits.

1 Introduction

Pseudo-random generators against a class of circuits are functions that take a random seed as input and output a sequence of bits that cannot be distinguished from a truly random sequence by any circuit in the class. They play an important role in many areas, particularly in cryptography and derandomization (see, e.g., [BM84,Yao82]). In this paper, we will be interested in derandomization aspect of pseudo-random generators, and therefore, will use the following definition (as given in [NW94]):

Definition 1. *For the class of circuits $\mathcal{C}$, function G is called a $(\ell \mapsto n)$* pseudo-random generator *against $\mathcal{C}$ if*

- $G = \{G_n\}_{n>0}$ *with* $G_n : \{0,1\}^{\ell(n)} \mapsto \{0,1\}^n$,
- G_n *is computable in time* $2^{O(\ell(n))}$,
- *for every n, and for every circuit $C \in \mathcal{C}$ having n input bits,*

$$|\ \mathrm{prob}_{x \in \{0,1\}^n}\{C(x) = 1\} - \mathrm{prob}_{y \in \{0,1\}^{\ell(n)}}\{C(G_n(y)) = 1\}\ | \leq \frac{1}{n}.$$

To derandomize a randomized algorithm, one uses a $(\ell \mapsto n)$ pseudo-random generator against a class of circuits that include the circuit family coding the algorithm, and feed the output of the generator as random input bits to the algorithm for each value of the seed, and then calculate the fraction of ones in the output. Of course, this modified algorithm takes more time—the time taken to compute the generator for every seed value times the time to run the algorithm on every output of the generator. To minimize the time taken, one needs to reduce $\ell(n)$: the best that can be achieved is $\ell(n) = O(\log n)$ and then the increase in time complexity is by a factor of polynomial only. Pseudo-random generators that achieve this seed size are called true pseudo-random generators:

Definition 2. *For the class of circuits $\mathcal{C}$, function G is called a* true pseudo-random generator against $\mathcal{C}$ *if G is a $(\ell \mapsto n)$ pseudo-random generator against $\mathcal{C}$ with $\ell(n) = O(\log n)$.*

While true pseudo-random generators against specific algorithms (i.e., the class against which the generator works include circuits for a specific algorithm only) are known, very few *unconditional* pseudo-random generators are known against natural classes of circuits. Perhaps the most notable amongst these are $((\log n)^{O(d)} \mapsto n)$ pseudo-random generators against the class of depth d and size n circuits [Nis91].

In a seminal work, Nisan and Wigderson [NW94] exhibited a connection between pseudo-random generators and *hard-to-approximate* sets in E:

Definition 3. *For a set A and circuit C with n input bits, let*

$$\mathrm{adv}_C(A) = \mid \mathrm{prob}_{x\in\{0,1\}^n}[C(x) = A(x)] - \mathrm{prob}_{x\in\{0,1\}^n}[C(x) \neq A(x)] \mid .$$

Here we identify A with its characteristic function. For a size bound $s(n)$ of circuits, let $\mathrm{adv}_{s(n)}(A)$ be the maximum of $\mathrm{adv}_C(A)$ where C varies over all size $s(n)$ circuits.

Set $A \in \mathrm{E}$ is hard-to-approximate *by circuits of size $s(n)$ if $\mathrm{adv}_{s(n)}(A) \leq \frac{1}{s(n)}$.*

Nisan and Wigderson showed that:

Nisan-Wigderson Theorem 1. [**NW94**] *There exist $(\ell \mapsto s(\ell^c))$ pseudo-random generators against class of size $s(\ell^c)$ circuits (for some size bound s and constant $c > 0$) if and only if there exist sets in E that are hard-to-approximate by circuits of size $s(\ell^d)$ (for some constant $d > 0$).*

In fact, the pseudo-random generator of [Nis91] is constructed using the above theorem and the fact that there exists a set (e.g. PARITY [Has86]) that is hard-to-approximate by circuits of size $2^{\ell^{\frac{1}{O(d)}}}$ and depth d (the above theorem of Nisan and Wigderson holds in the presence of depth restriction too).

An interesting special case is that of true pseudo-random generators, i.e., when $s(\ell) = 2^{O(\ell)}$. In that case, [NW94] showed that both the constants c and d can be set to one, and thus we get:

Nisan-Wigderson Theorem 2. [**NW94**] *There exist true pseudo-random generators against class of size $2^{\delta\cdot\ell}$ circuits for some constant $0 < \delta < 1$ if and only if there exist sets in E that are hard-to-approximate by circuits of size $2^{\epsilon\cdot\ell}$ for some constant $0 < \epsilon < 1$.*

One of the major implication of the existence of above true pseudo-random generators is that BPP = DP. In the following, we restrict our attention to true pseudo-random generators only as these have the most interesting implications. So, $n = 2^{O(\ell)}$ throughout the paper.

Although [NW94] provides evidence that true pseudo-random generators exist, it is not clear that hard-to-approximate sets, as required, do exist in E. On the other hand, it is easier to believe that there exist sets in E that cannot be solved by subexponential size circuits—in other words, there is a set in E such that $\mathrm{adv}_{2^{\epsilon\cdot\ell}}(A) < 1$ for some $0 < \epsilon < 1$. Therefore, a major line of research in the last ten years has been to construct true pseudo-random generators from this weaker assumption. The approach taken was to start with a set A in E with $\mathrm{adv}_{2^{\epsilon\cdot\ell}}(A) < 1$, and derive another set $B \in$ E from A such that B is hard-to-approximate by $2^{\epsilon'\cdot\ell}$ size circuits as required in the above theorem.

The above aim was achieved in three steps. First, [BFNW93] constructed—starting from a set $A^1 \in$ E with $\mathrm{adv}_{2^{5\epsilon\ell}}(A^1) < 1$—a set $A^2 \in$ E such that $\mathrm{adv}_{2^{3\epsilon\ell}}(A^2) < 1 - \frac{1}{\ell^2}$. Then, in [Imp95], a third set A^3 was constructed from A^2 with $\mathrm{adv}_{2^{2\epsilon\ell}}(A^2) < \frac{5}{6}$, and finally in [IW97] a set A^4 was constructed from A^3 with $\mathrm{adv}_{2^{\epsilon\ell}}(A^4) \leq \frac{1}{2^{\epsilon\cdot\ell}}$ thus achieving the desired generalization of the Nisan-Wigderson Theorem 2. In [STV99] two alternative constructions were given for the same result.

The work in this paper is motivated by the following question: what is the hardness condition needed for constructing true pseudo-random generators against classes of circuits more restricted than the class of polynomial-sized circuits (the class of circuits in the Nisan-Wigderson Theorem 2 is polynomial-sized in the generator output size, and exponential-sized in the generator input size)? A natural way of defining such circuits is by restricting their depth. So we can pose this question for several natural classes of small depth circuits, e.g., AC^0, TC^0, NC^1, NC, etc. In analogy with the above result, we should perhaps expect that to construct pseudo-random generators against polynomial-sized circuits of depth d, we need a hard set against subexponential sized circuits of depth $O(d)$.

We first observe that the constructions given in [BFNW93,Imp95,IW97] have the following property: starting with a set that is hard to compute by the class of circuits of size $2^{5\epsilon\ell}$ and depth d, the constructed set is hard-to-approximate by circuits of size $2^{\epsilon\cdot\ell}$ and depth $d - O(1)$ (for some $\epsilon, \alpha > 0$) *provided the majority gate* is allowed in the original class of circuits. This implies that for all circuit classes $\mathcal{C}$ that include TC^0, one can construct true pseudo-random generators against $\mathcal{C}$ using a set in E that is hard to compute by subexponential sized circuits of the same depth (within a constant factor) as in $\mathcal{C}$.

Therefore, our question is answered for all the well-known circuit classes except for the class AC^0. AC^0 circuits are polynomial-sized constant depth circuits and it is known that they *cannot* compute the majority function [Has86]. Therefore, the construction of [BFNW93,Imp95,IW97] does not give the expected result. Further, this seems to be a fundamental bottleneck as the other two constructions given in [STV99] also require at least threshold gates. So we have a intriguing situation here: even though there exist nearly true pseudo-random generators against AC^0 circuits (given by Nisan [Nis91]) that are *unconditional*, we *do not* seem to get *conditional* true pseudo-random generators against AC^0 under a condition whose stronger forms give true pseudo-random generators against larger classes of circuits! It is useful to note here that true pseudo-random generators against AC^0 circuits are interesting in their own right: their existence would imply that approximate DNF-counting can be derandomized [KL83].

In this paper, we close this gap in our knowledge to show that:

Theorem 1. *There exist true pseudo-random generators against class of size $2^{\delta\cdot\ell}$ and depth $O(d)$ AC^0 circuits for some constant $0 < \delta < 1$ if and only if there exist a set in E that cannot be computed by AC^0 circuits of size $2^{\gamma\cdot\ell}$ and depth $O(d)$ for some constant $0 < \gamma < 1$.*

The idea is to exploit the unconditional pseudo-random generators of Nisan. The generator of Nisan stretches a seed of size $(\log n)^{O(d)}$ to n bits and works against depth d, size n AC^0 circuits. Moreover, every output bit of the generator is simply a parity of a subset of seed bits. Now the crucial observation is that parity of $\mathrm{poly}(\log \mathrm{n})$ bits *can* be computed by AC^0 circuits, and so if we compose the Nisan generator with any given circuit C of depth d and size n, we get another AC^0 circuit of a (slightly) larger depth and size that has only $\mathrm{poly}(\log \mathrm{n})$ input bits (as opposed to n in C) and yet the circuit accepts roughly the same fraction of inputs as C. A careful observation of the constructions of [BFNW93,Imp95,NW94,NW94] yields that if the pseudo-random generator constructed through them needs to stretch a seed of ℓ bits to only $\mathrm{poly}(\ell)$ bits (instead of $2^{\epsilon\cdot\ell}$ bits), then we need to start from a set in E that is hard to compute by circuits of size $2^{\delta'\cdot\ell}$, depth d that have majority gates over only $\mathrm{poly}(\ell)$ bits (instead of over $2^{O(\ell)}$ bits). Such majority gates can be replaced by AC^0 circuits of size $2^{o(\ell)}$. Therefore, we only require sets in E that are hard to compute by size $2^{\delta\cdot\ell}$ and depth d' AC^0 circuits! A minor drawback of the result is that the true pseudo-random generators that we obtain approximate the fraction of inputs accepted by a circuit C within $\frac{1}{\mathrm{poly}(\log \mathrm{n})}$ as opposed to $\frac{1}{n}$ in all the other cases. However, for many applications, e.g., derandomizing approximate DNF-counting, this weaker approximation is sufficient.

The organization of the paper is as follows: in the next section we analyze the existing constructions and in Section 3 we give our construction.

2 Depth increase in existing constructions

The construction in [BFNW93,Imp95,IW97,NW94] can be divided into five stages:

Stage 1. Given a set A_1 in E such that $\mathrm{adv}_{2^{\epsilon_1\ell}}(A_1) < 1$, construct a function $f = \{f_\ell\}$ in E such that for any $\epsilon_f < \epsilon_1$, and for any circuit C of size $2^{\epsilon_f\ell}$, the fraction of inputs on which C can compute f_ℓ correctly is at most $1 - \frac{1}{\ell^2}$. This construction was given in [BFNW93].

Stage 2. From the function f construct a set $A_2 \in \mathrm{E}$ such that for any $\epsilon_2 < \epsilon_f$, $\mathrm{adv}_{2^{\epsilon_2\ell}}(A_2) < 1 - \frac{1}{\ell^3}$. This construction was given in [GL89].

Stage 3. From the set A_2 construct a set $A_3 \in \mathrm{E}$ such that for any $\epsilon_3 < \epsilon_2$, $\mathrm{adv}_{2^{\epsilon_3\ell}}(A_3) < \frac{15}{16}$. This construction was given in [Imp95].

Stage 4. From the set A_3 construct a set $A_4 \in \mathrm{E}$ such that for any $\epsilon_4 < \epsilon_3$, $\mathrm{adv}_{2^{\epsilon_4 \ell}}(A_4) < \frac{1}{2^{\epsilon_4 \ell}}$. This construction was given in [IW97].

Stage 5. using the set A_4, construct a true pseudo-random generator $G = \{G_n\}$ with $G_n : \{0,1\}^{O(\log n)} \mapsto \{0,1\}^n$ against circuits of size n. This, of course, was given in [NW94].

We now describe each of these constructions. The correctness of all the constructions is shown using the contrapositive argument: given a circuit family that solves the constructed set (or function) with the specified advantage, we construct a circuit family that solves the original set (or function) with an advantage that contradicts the hardness assumption about the set. For our purposes, the crucial part in these arguments would be the depth and size increase in the constructed circuit family over the given circuit family. We do not need to worry about the complexity of the constructing the new set from the original one—this is an important to keep in mind as often this complexity is very high (e.g., in Stage 1 and Stage 4).

Several times in the constructions below, we make use of the following (folklore) fact about computing parity or majority of ℓ bits:

Proposition 1. *The parity or majority of ℓ bits can be computed by* AC^0 *circuits of size* $O(2^{\ell^{\frac{4}{d}}})$ *and depth* d.

Hastad [Has86] provided a (fairly tight) corresponding lower bound:

Lemma 1. *The parity or majority of ℓ bits cannot be computed by* AC^0 *circuits of size* $2^{\ell^{\frac{1}{d+1}}}$ *and depth* d.

2.1 Stage 1: Analyzing Babai-Fortnow-Nisan-Wigderson's construction

Construction of f Function f is an small degree, multi-variate polynomial extension of the set A_1 over a suitable finite extension field of F_2. More specifically, function $f(x)$, $|x| = \ell$, is defined as follows (we assume ℓ to be a power to two for convenience):

Fix field $F = F_{\ell^2}$. Let $k = \frac{\ell}{2\log \ell}$. Define polynomial $P(y_1, y_2, \ldots, y_k)$ over F as:

$$P(y_1, y_2, \ldots, y_k) = \sum_{v_1 : |v_1| = \log \ell} \cdots \sum_{v_k : |v_k| = \log \ell} A_1(v_1 v_2 \cdots v_k) \cdot \prod_{i=1}^{k} \delta_{v_i}(y_i),$$

where

$$\delta_{v_i}(y_i) = \frac{\prod_{v : |v| = \log \ell \wedge v \neq v_i} (y_i - v)}{\prod_{v : |v| = \log \ell \wedge v \neq v_i} (v_i - v)}.$$

Let $x = x_1 x_2 \cdots x_k$ with $|x_i| = 2\log \ell$. Then,

$$f(x) = P(x_1, x_2, \ldots, x_k).$$

Polynomial P has $k = \frac{\ell}{2\log \ell}$ variables and each variable has degree at most ℓ.

Correctness of construction Suppose that a family of circuits $\{C_\ell\}$ of size $2^{\epsilon_f \ell}$ exists such that for every ℓ, C_ℓ can correctly compute f on more than $1-\frac{1}{\ell^2}$ fraction of inputs. We use this circuit family to construct a circuit family that correctly decides f, and therefore A_1, everywhere.

Fix ℓ and x, $|x| = \ell$. String x can be viewed as a point in the k-dimensional vector space over F_{ℓ^2}. Select a random line passing through x in this space. It is easily argued that with probability at least $\frac{3}{4}$, on such a line, C_ℓ will correctly compute f on at least $1 - \frac{4}{\ell^2}$ fraction of points. Notice that when restricted to such a line, polynomial P reduces to a univariate polynomial P' of degree at most $\frac{\ell^2}{2\log \ell}$. Randomly select $\frac{\ell^2}{2\log \ell} + 1$ points on this line and use circuit C_ℓ to find out the value of f on these. Clearly, with probability at least $1 - \frac{3}{\log \ell}$ the computed value of f would be correct on all the points. Interpolate polynomial P' using these values and then compute the value of $f(x)$ using P'. The probability that $f(x)$ is correctly computed is at least $\frac{3}{4} \cdot (1 - \frac{3}{\log \ell}) > \frac{2}{3}$. Repeat the same computation with different random choices ℓ^2 times and take the value occurring maximum number of times as the value of $f(x)$. The probability that this is wrong would be less than $\frac{1}{2^\ell}$. Finally, fix a setting of random bits that work for all 2^ℓ different x's. The circuit implementing this algorithm correctly computes f everywhere (the circuit is non-uniform though).

Let us now see what is the size and depth of this circuit, say C', as compared to C_ℓ. Once all the random choices are fixed, C' just needs to use C_ℓ on $\frac{\ell^2}{2\log \ell} + 1$ different inputs (computed by xoring a fixed string to x), and then take a linear combination of the output values[1]. As there are $O(\ell^2)$ outputs each of size ℓ, this can be done by a AC^0 circuit of size $2^{o(\ell)}$. Thus the size of the circuit C' is at most $2^{\epsilon_1 \ell}$ as long as $\epsilon_1 > \epsilon_f$ contradicting the assumption. Notice that depth of C' is only a constant more than of C_ℓ and C does not have any majority gate except those already present in C_ℓ.

2.2 Stage 2: Analyzing Goldreich-Levin's construction

Construction of A_2 Set A_2 is defined as: $xr \in A_2$ iff $|x| = |r| = \ell$ and $f(x) \cdot r = 1$ where '$\cdot$' is the inner product.

Correctness of the construction Assume that a circuit family $\{C_\ell\}$ of size $2^{\epsilon_2 \ell}$ is given such that $\mathrm{adv}_{C_\ell}(A_2) \geq 1 - \frac{1}{\ell^3}$. As we later need this result for smaller advantages too, we give the construction assuming that $\mathrm{adv}_{C_\ell}(A_2) \geq \zeta$. Fix ℓ and x, $|x| = \ell$. Define circuit C' as follows:

[1] This linear combination is the degree zero coefficient of the interpolated polynomial P'. Notice that circuit C' does not need to interpolate P' (which actually may not be possible to do by subexponential sized constant depth circuit) since the points at which values of f are given are fixed (once the random choices are fixed) and therefore, the inverse of the corresponding van der Monde matrix can simply be hardwired into C'.

Let $t = c \cdot \log \ell$ for a suitable $c > 1$. Randomly choose t strings $r_1, \ldots, r_t$ with $|r_i| = \ell$. For each non-empty subset J of $\{1, \ldots, t\}$, let $r_J = \oplus_{i \in J} r_i$ (these r_J's are pairwise independent and this is exploited in the proof). Fix s, $|s| = t$ and compute $\sigma_J = \oplus_{i \in J} s_i$ where s_i is the i^{th} bit of s. Now compute i^{th} bit of $f(x)$ as the majority of the $2^t - 1$ values (obtained by varying J) $\sigma_J \oplus C_{2\ell}(x, r_J \oplus e^i)$ where e^i is an ℓ-bit vector with only the i^{th} bit one. Finally, output the guess for $f(x)$ thus computed for *each* of the $2^t - 1$ values of s.

It was shown in [GL89] that for at least ζ^2 fraction of inputs x, $f(x)$ is present in the list of strings output by the circuit C' with probability close to one. Now there are two ways to design a circuit C'' that outputs $f(x)$ depending on the value of ζ. If $\zeta = \frac{1}{\ell^{\Omega(1)}}$ then C'' randomly picks one string output by C' and outputs it. The probability that it succeeds is close to $\frac{1}{2^t - 1} = \frac{1}{\ell^{O(1)}}$. Fix the internal random bits used by this circuit by averaging. The resulting circuit correctly outputs $f(x)$ on at least ζ^3 fraction of inputs.

The second way is for $\zeta \geq \frac{5}{6}$. In this case, C'' selects the right string from the output list of C' as follows (suggested in [Imp95]): randomly choose $O(\ell)$ many strings $r \in \{0,1\}^\ell$ and for each string u output by C' test if $u \cdot r = C_{2\ell}(x, r)$ and output the string u for which the largest number of r's satisfy the test. It was shown in [Imp95] that suitably fixing random strings r, if $f(x)$ appears in the output list then C'' would certainly output it. Therefore, the fraction of inputs on which C'' is correct is at least ζ^2.

In either case, the depth of the circuit C'' is only a constant more than of $C_{2\ell}$. Although C'' uses majority gates, they are only over $\ell^{O(1)}$ many inputs and so can be replaced by constant depth subexponential AC^0 size circuits.

Notice that the above two constructions cannot handle ζ between $\frac{1}{\ell^{o(1)}}$ and $o(1)$. However, these values of ζ are never required in the constructions[2].

2.3 Stage 3: Analyzing Impagliazzo's hard-core construction

This stage has three substages. In the first substage, starting from set A_2 with $\mathrm{adv}_{2^{\epsilon_2 \ell}}(A_2) \leq 1 - \frac{1}{16\ell^3}$, set A' is constructed with $\mathrm{adv}_{2^{\epsilon' \ell}}(A') \leq 1 - \frac{1}{16\ell^2}$ for any $\epsilon' < \epsilon_2$. In the next stage, set A'' is constructed from A' with $\mathrm{adv}_{2^{\epsilon'' \ell}}(A'') \leq 1 - \frac{1}{16\ell}$ for any $\epsilon'' < \epsilon'$. And in the third substage, from A'', set A_3 is constructed with $\mathrm{adv}_{2^{\epsilon_3 \ell}}(A^3) \leq \frac{15}{16}$ for any $\epsilon_3 < \epsilon''$.

All the three substages are identical. We describe only the first one.

Construction of A' Set A' is defined as: $rs \in A'$ iff $|r| = c \cdot \ell$, $|s| = 2\ell$ and $r \cdot g(s) = 1$ where $g(s) = A_2(x_1)A_2(x_2) \cdots A_2(x_{c \cdot \ell})$ with $x_1, \ldots, x_{c \cdot \ell}$, $|x_i| = \ell$, (for an appropriate constant c) generated from s in a pairwise-independent fashion—let $s = s_1 s_2$ with $|s_1| = |s_2| = \ell$, then $x_i = s_1 \cdot i + s_2$ in the field F_{2^ℓ}.

[2] In fact there is a third way that works for all values of ζ. However, it uses error-correcting codes and decoding these appears to require more than constant depth subexponential size circuits. So we cannot use it.

Correctness of the construction Let a circuit family $\{C_\ell\}$ of size $2^{\epsilon'\ell}$ be given such that $\mathrm{adv}_{C_\ell}(A') \geq 1 - \frac{1}{16\ell^2}$. First invoke the (second) Goldreich-Levin construction to conclude that there exists a circuit family $\{C'_\ell\}$ of size $2^{\delta'\ell}$ for $\epsilon' < \delta' < \epsilon_2$ that computes function $g(s)$ on at least $(1-\frac{1}{16\ell^2})^2 \geq 1 - \frac{1}{8\ell^2}$ fraction of inputs. Fix an ℓ. Define a circuit C'' as:

> On input x, $|x| = \ell$, randomly select an i, $1 \leq i \leq c \cdot \ell$. Then randomly select first half s_1 of the seed s and let $s_2 = x + s_1 \cdot i$ (this ensures that x occurs as x_i). Use s to generate $x_1, \ldots, x_{c\cdot\ell}$. Output the i^{th} bit of $C'_{c\cdot\ell^2}$ as guess for $A_2(x)$.

It was shown in [Imp95] that, for any given set $S \subset \{0,1\}^\ell$ with $|S| \geq \frac{2^\ell}{16\ell^3}$, when input x is randomly selected from S, the probability that $C''(y) = A_2(y)$ is at least $\frac{3}{5}$.

From the circuit C'', construct another circuit C''' as: take ℓ^2 copies of C'' (using different random bits for each one), and take the majority of their output values. For any x, if the probability of C''' incorrectly computing $A_2(x)$ is more than $\frac{1}{2^\ell}$ then it must be that C'' incorrectly computes $A_2(x)$ with probability more than $\frac{2}{5}$. By the above property of C'', there cannot be more than $\frac{2^\ell}{16\ell^3}$ such x's. Therefore, on at least $1 - \frac{1}{16\ell^3}$ fraction of inputs, C''' computes A_2 correctly with probability at least $1 - \frac{1}{2^\ell}$. Now fix the random bits of C''' such that the resulting circuit computes A_2 correctly on at least $1 - \frac{1}{16\ell^3}$ fraction of inputs.

As for the size and depth increase, circuit C''' (as well as the final circuit) uses one majority gate (on ℓ^2 inputs) at the top and one bottom layer of parity gates (on ℓ inputs). It also uses ℓ^2 copies of C'' in parallel. Therefore, the size of the circuit is at most $2^{\epsilon_2\ell}$ since $\epsilon_2 > \delta'$ and depth is only a constant more. This contradicts the assumption about A_2.

The above construction of circuit C''' is used again later with different parameters: starting with a circuit C'' that computes the given set with probability at least $\frac{1}{2} + \epsilon$ on any subset of strings of size $O(2^\ell)$, we can use the above construction to obtain a circuit C''' that computes the set on a constant fraction of inputs in a similar fashion. This circuit is constructed by taking the majority of $O(\frac{\ell}{\epsilon^2})$ copies of another circuit. The value of ϵ would be crucial in our calculations there.

2.4 Stage 4: Analyzing Impagliazzo-Wigderson's construction

Construction of A_4 Set A_4 is defined as: $rs \in A_4$ iff $|r| = \ell$, $|s| = k\ell$, and $r \cdot g'(s) = 1$ where $g'(s) = A_3(x_1)A_3(x_2)\cdots A_3(x_\ell)$ with x_is generated from s via a generator whose output is XOR of the outputs of an expander graph based generator and a NW-design based generator.

Correctness of the construction The construction is this stage is very similar to the one in previous stage. Let a circuit family $\{C_\ell\}$ of size $2^{\epsilon_4\ell}$ be given such that $\mathrm{adv}_{C_\ell}(A_4) \geq \frac{1}{2^{\epsilon_4\ell}}$. Invoke the (first) Goldreich-Levin construction to obtain a circuit family $\{C'_\ell\}$ of size $2^{\epsilon' \ell}$ computing function g' on $\frac{1}{2^{\epsilon' \ell}}$ fraction of inputs for $\epsilon' > \epsilon_4$.

Fix and ℓ. Construct a circuit C'' in a similar fashion (although the analysis becomes different) that computes A_3 with probability at least $\frac{1}{2} + \frac{1}{2^{\epsilon'' \ell}}$ (for any $\epsilon'' > \epsilon'$) on any given set of size $\frac{2^\ell}{16}$ and then construct C''' from C'' by taking the majority of $O(2^{2\epsilon'' \ell})$ copies of C''. As before, it can be shown that C''' computes A_3 correctly with probability more than $1 - \frac{1}{2^\ell}$ on all but $\frac{1}{16}$ fraction of inputs. Fixing random bits of C''' suitably gives a circuit that correctly computes A_3 on at least $\frac{15}{16}$ fraction of inputs.

The size of circuit C''' is at most $2^{\epsilon_3 \ell}$ since $\epsilon_3 > \epsilon''$. The depth of C''' is still only a constant more than that of C_ℓ since the output of the generator used in construction of A_4 can be easily computed: the output of the generator gets fixed upon fixing the random bit values apart from ℓ fixed positions where the string x is written.

However, the majority gate at the top of C''' has $2^{\Omega(\ell)}$ inputs. This *cannot* be done using AC^0 circuits in constant depth and $2^{\delta n}$ size for any $\delta > 0$. In fact, this is the only place where the depth condition is violated.

2.5 Stage 5: Analyzing Nisan-Wigderson's construction

Construction of generator Pseudo-random generator G_n is defined as: given $k \cdot \log n$ length seed s, compute n "nearly disjoint" subsets of bit positions in the seed of size $t \cdot \log n$ each ($t < k$). Let the strings written in these positions be $x_1, \ldots, x_n$, $|x_i| = t \cdot \log n$. Output $A_4(x_1)A_4(x_2) \cdots A_4(x_n)$.

Correctness of the construction Let C be a circuit of size n such that

$$| \operatorname{prob}_{x \in \{0,1\}^n} \{C(x) = 1\} - \operatorname{prob}_{s \in \{0,1\}^{k \cdot n}} \{C(G_n(s)) = 1\} | \geq \frac{1}{n}.$$

Define circuit C^i as:

> On input x_i *and* $A_4(x_1) \cdots A_4(x_{i-1})$, randomly select a bit b and a string r of length $n - i$. Compute $o = C(A_4(x_1) \cdots A_4(x_{i-1})br)$. Output $b \oplus o$.

It was shown in [NW94] that for at least one i, C^i correctly computes $A_4(x_i)$ on at least $\frac{1}{2} + \frac{1}{n^2}$ fraction of inputs.

Exploiting the property that the subsets of bit positions determining each of $x_1, \ldots, x_{i-1}$ are nearly disjoint from those determining x_i, one can fix the random bits of C^i and of the seed s except for those bits that determine x_i such that the advantage of C^i in computing $A_4(x_i)$ is preserved *and* the value of $A_4(x_j)$ (for $j < i$) is needed by the circuit (as x_i varies) for at most n different inputs. So all values of A_4 needed by the circuit (at most n^2) can be hardwired into it, thus eliminating the need of providing $A_4(x_1) \cdots A_4(x_{i-1})$ as part of the input.

Let the final circuit be C''. The size of C'' is $O(n^2)$ and $\text{adv}_{C''}(A_4) > \frac{1}{n^2}$ on inputs of size $t \cdot \log n$. For a suitable choice of t and k, this contradicts the hardness of A_4. The depth of C'' is only a constant more than the depth of C as the only additional computation needed is to select the correct hardwired values of $A_4(x_1), \ldots, A_4(x_{i-1})$ depending on the input x_i (this is a simple table lookup).

2.6 Analyzing constructions of Sudan-Trevisan-Vadhan

The above bottleneck prompts us to look at other constructions of true pseudo-random generators present in the literature: there are two such constructions known given in [STV99]. However, both these constructions have similar bottlenecks. We point out these bottlenecks below:

First construction. This construction uses a *false entropy generator*. This generator makes use of the hard-core result of Impagliazzo [Imp95]. The value of ϵ that the construction requires in the hard-core result is $\frac{1}{2^{O(\ell)}}$. So this has the same problem as the construction of [IW97]: it requires to compute the majority of $2^{O(\ell)}$ bits.

Second construction. This construction actually shows that stage 3 and 4 above can be *bypassed*. In other words, the multivariate polynomial P has enough redundancy to directly ensure that no circuit family of size $2^{\beta\ell}$ can compute the function on more than $\frac{1}{2^{\beta\ell}}$ fraction of inputs. However, the proof for this result is far more involved than the proof of stage 1. In the proof, to interpolate the polynomial correctly on a random line, at least $2^{\kappa\ell}$ samples are needed. This requires, amongst other things, xoring of $2^{\kappa\ell}$ bits and also computing $2^{\kappa\ell}$th power of a given element in a field of size $2^{O(\ell)}$. None of these can be performed by constant depth $2^{O(\ell)}$ sized AND-OR circuits.

3 Proof of Theorem 1

The problem in working with AC^0 circuits is that they are too weak to do even simple computations. But we can use this drawback to our advantage! Since good lower bounds for AC^0 circuits are known [Has86], one can construct unconditional pseudo-random generator against such circuits. In [Nis91], Nisan used lower bounds on parity function to obtain pseudo-random generators against depth d, size n AC^0 circuits that stretch seeds of size $(\log n)^{O(d)}$ to n bits. Moreover, each output bit of these generators is simply parity of some of the seed bits. Therefore, each output bit of the generator can be computed by an AC^0 circuit of size $n^{o(1)}$ and depth $O(d)$.

So, given an AC^0 circuit C of depth d and size n that accepts δ fraction of inputs, when we combine this circuit with the pseudo-random generator of [Nis91], we get *another* AC^0 circuit of depth $O(d)$ and size $O(n^2)$ that has only $(\log n)^{O(d)}$ input bits and still accepts $\delta \pm \frac{1}{n}$ fraction of inputs. Let us try to construct a true pseudo-random generator against *such* a circuit using the Nisan-Wigderson construction. This generator needs to stretch $O(\log n)$ bits to $(\log n)^{O(d)}$ bits. If we examine the Nisan-Wigderson construction of the generator, it is apparent that—if we fix the approximation error to $\frac{1}{(\log n)^{O(1)}}$ instead of $\frac{1}{n}$—such a generator can be constructed *provided* there exists a set $A \in \mathrm{E}$ such that for any depth $O(d)$ circuit family $\{C_\ell\}$ of size $2^{\delta\ell}$, $\mathrm{adv}_{C_\ell}(A) < \frac{1}{\ell^{O(d)}}$. Now notice that such a set can be easily constructed by modifying the stage 4 of the construction! Since instead of $\epsilon = \frac{1}{2^{O(\ell)}}$ we now have $\epsilon = \frac{1}{\ell^{O(d)}}$, the majority gate needed in the construction will have a fan-in of only $\ell^{O(d)}$, and this *can* be done by constant depth AND-OR circuits of subexponential size[3]. Hence the overall construction now becomes six stage one: first three stages are identical to the ones described above; the fourth stage is modified for weaker approximation needed; the fifth stage uses Nisan-Wigderson construction for pseudo-random generator that stretches the seed only polynomially; this stretched seed acts as seed for the Nisan generator in the final stage that stretches the output to n bits.

It is interesting to note that each output bit of this pseudo-random generator is simply an XOR of several bits of the characteristic function A_1: the multivariate polynomial construction in Stage 1 is just an XOR of some input bits; Stage 2, 3, and 4 constructions are clearly simple XORs (computing which bits to XOR requires some effort though); the fifth stage merely copies some bits from input to output; and the last stage (it uses parity function) is also xoring some bits.

Acknowledgements

An anonymous referee's suggestions have helped to make the writeup more readable. Eric Allender and Adam Klivans pointed out a wrong claim made in an earlier version of the paper.

References

[BFNW93] L. Babai, L. Fortnow, N. Nisan, and A. Wigderson. BPP has subexponential time simulations unless EXPTIME has publishable proofs. *Computational Complexity*, 3(4):307–318, 1993.

[BM84] M. Blum and S. Micali. How to generate cryptographically strong sequences of pseudo-random bits. *SIAM Journal on Computing*, 13:850–864, 1984.

[GL89] O. Goldreich and L. A. Levin. A hardcore predicate for all one-way functions. In *Proceedings of Annual ACM Symposium on the Theory of Computing*, pages 25–32, 1989.

[Has86] J. Hastad. *Computational limitations on small depth circuits.* PhD thesis, Massachusetts Institute of Technology, 1986.

[Imp95] R. Impagliazzo. Hard-core distributions for somewhat hard problems. In *Proceedings of Annual IEEE Symposium on Foundations of Computer Science*, pages 538–545, 1995.

[3] Alternatively, one can use a construction (given in [Imp95]) that decreases the advantage to $\frac{1}{\ell^{O(1)}}$ using k-wise independent generation of strings. This avoids the construction of [IW97] altogether.

[IW97] R. Impagliazzo and A. Wigderson. P = BPP if E requires exponential circuits: Derandomizing the XOR lemma. In *Proceedings of Annual ACM Symposium on the Theory of Computing*, pages 220–229, 1997.

[KL83] R. M. Karp and M. Luby. Monte-Carlo algorithms foe enumeration and reliability problems. In *Proceedings of Annual IEEE Symposium on Foundations of Computer Science*, pages 56–64, 1983.

[Nis91] N. Nisan. Pseudo random bits for constant depth circuits. *Combinatorica*, 11(1):63–70, 1991.

[NW94] N. Nisan and A. Wigderson. Hardness vs. randomness. *J. Comput. Sys. Sci.*, 49(2):149–167, 1994.

[STV99] M. Sudan, L. Trevisan, and S. Vadhan. Pseudorandom generators without the XOR lemma. In *Proceedings of Annual ACM Symposium on the Theory of Computing*, pages 537–546, 1999.

[Yao82] A. C. Yao. Theory and applications of trapdoor functions. In *Proceedings of Annual IEEE Symposium on Foundations of Computer Science*, pages 80–91, 1982.