Proving Lower Bounds via Pseudo-Random Generators

Manindra Agrawal

Department of Computer Science
Indian Institute of Technology, Kanpur
manindra@iitk.ac.in

Abstract. In this paper, we formalize two stepwise approaches, based on *pseudo-random generators*, for proving P $\neq$ NP and its arithmetic analog: *Permanent requires superpolynomial sized arithmetic circuits.*

1 Introduction

The central aim of complexity theory is to prove lower bounds on the complexity of problems. While the relative classification of problems (via reductions) has been very successful, not much progress has been made in determining their absolute complexity. For example, we do not even know if NE admits nonuniform NC^1 circuits.

Initial attempts (in 1970s) to prove lower bounds centered on using the diagonalization technique that had proven very useful in recursion theory. However, a series of relativization results soon showed that this technique cannot help in its standard guise [6]. Very recently, the technique has been used to prove certain simultaneous time-space lower bounds [7], however, its usefulness for single resource lower bounds remains unclear.

In the 1980s, the results of Razborov [15] (lower bounds on monotone circuits) and Håstad [8] (lower bounds on constant depth circuits) gave rise to the hope of proving lower bounds via combinatorial arguments on boolean circuit model of complexity classes. However, there was little progress since mid-80s and ten years later Razborov and Rudich [16] explained the reason for this: they showed that combinatorial arguments used for previous lower bounds cannot be extended to larger classes.

Over the last ten years, a new paradigm is slowly emerging that might lead us to strong lower bounds: *pseudo-random generators*. These were introduced in 1980s by Yao [22], Blum, and Micali [4], Nisan and Wigderson [14] to formulate the hardness of cryptographic primitives (in the first two references) and to derandomize polynomial-time randomized algorithms (in the last reference).

It was known from the beginning that existence of pseudo-random generators implies lower bounds on boolean circuits. In fact, they can be viewed as a strong form of diagonalization. Attempts were then made to prove the other (and seemingly more interesting) direction: lower bounds on boolean circuits imply existence of pseudo-random generators. This was achieved after a lot of effort:

Håstad, Impagliazzo, Levin and Luby [9] showed that pseudo-random generators of *polynomial stretch* are equivalent to *one-way functions*, Impagliazzo and Wigderson [10] showed that pseudo-random generators of *exponential stretch* are equivalent to hard sets in E.

Some recent advances suggest that the first (and easier) direction of the above equivalence may in fact hold the key to obtaining lower bounds. Very recently, Omer Reingold used expander graphs (these are one of the fundamental tools in derandomization) to search in an undirected graph using logarithmic space [17]. This proves SL = L, resolving the complexity of the class SL. Although Reingold's result does not yield a pseudo-random generator or a lower bound, it suggests that one can do derandomization without appealing to lower bounds, and a strong enough derandomization will result in a lower bound.

Lower bounds for arithmetic circuits have also been investigated, but again, without much success. It appears that obtaining lower bounds for these circuits should be easier than boolean circuits since boolean circuits can simulate arithmetic circuits but not vice versa. Kabanets and Impagliazzo [11] have recently observed a connection between lower bounds on arithmetic circuits and derandomizations of *polynomial identity testing problem* (given a multivariate polynomial computed by an arithmetic circuit, test if it is identically zero). This connection, however, is not as tight as for boolean circuits. For these circuits too there is some evidence that proving lower bounds via derandomization might work: the primality testing algorithm of Agrawal, Kayal, Saxena [2] essentially derandomizes a certain polynomial identity.

Admittedly, the evidence for the success of "pseudo-random generator" approach is weak: neither Reingold's result nor the primality testing algorithm yield a lower bound (the AKS "derandomization" works only for a problem, not a class). However, this is one of the, if not the, most promising approach that we have presently for obtaining boolean and arithmetic circuit lower bounds and so needs to be investigated seriously. In this article, we formulate, based on pseudo-random generators, stepwise approaches to resolve two of the most important conjectures in complexity theory: P $\neq$ NP and its arithmetic analog *Permanent requires superpolynomial-sized arithmetic circuits*. For arithmetic circuits, the result of Kabanets and Impagliazzo is not strong enough to show that derandomization implies second conjecture. To make it work, we define pseudo-random generators for arithmetic circuits and show that certain generators imply the desired lower bound on arithmetic circuits.

2 Pseudo-Random Generators for Boolean Circuits

Let $\mathcal{C}(s(n), d(n))$ denote the class of circuits of size $s(n)$ and depth $d(n)$ on inputs of size n. We will assume that all our circuits are *layered* with layers alternating between AND and OR gates.

We begin with the definition of a pseudo-random generator for boolean circuits.

Definition 21 Function f, $f : \{0,1\}^* \mapsto \{0,1\}^*$, is a $(\ell(n), n)$-*pseudo-random generator against* $\mathcal{C}(s(n), d(n))$ if:

- $f(\{0,1\}^{\ell(n)}) \subseteq \{0,1\}^n$ with $\ell(n) < n$ for all n.
- For any circuit $C \in \mathcal{C}(s(n), d(n))$,

$$\mid \Pr_{x \in \{0,1\}^n}[C(x) = 1] - \Pr_{y \in \{0,1\}^{\ell(n)}}[C(f(y)) = 1] \mid \leq \frac{1}{n}.$$

The difference between output and input length of a pseudo-random generator, $n - \ell(n)$, is called the *stretch* of the generator. A simple counting argument shows that there exist $(O(\log s(n)), n)$-pseudo-random generators against $\mathcal{C}(s(n), d(n))$:

> Define f by randomly assigning strings of length n for inputs of size $4 \log s(n)$. Let y be a string of size $4 \log s(n)$. For a circuit $C \in \mathcal{C}(s(n), d(n))$ with input size n, define random variable Y to be $C(f(y))$. The expected value of Y is precisely the fraction of strings accepted by C. We have $s^4(n)$ such independent random variables with the same expected value. Hence, by Chernoff's bound, the probability that the average of these variables differs from the expectation by more than $\frac{1}{n}$ is less than $\frac{1}{2^{s^3(n)}}$. Since there are less than $2^{s^2(n)}$ circuits of size $s(n)$, most of the choices of f will be pseudo-random.

It is also easy to see (via a similar counting argument) that $(o(\log s(n)), n)$-pseudo-random generators cannot exist against $\mathcal{C}(s(n), d(n))$. This motivates the following definition.

Definition 22 A $(O(\log s(n)), n)$-pseudo-random generator against $\mathcal{C}(s(n), d(n))$ is called an *optimal* pseudo-random generator against $\mathcal{C}(s(n), d(n))$.

So far, we have not examined the computability aspect of pseudo-random generators. It is easy to see that optimal pseudo-random generators against $\mathcal{C}(s(n), d(n))$ can be computed in time $O(2^{s^2(n)})$. To make the notion interesting, we need to compute them faster. There are two ways in which the time complexity of a generator can be measured: as a function of output size or as a function of input size. We choose to express it in terms of input size.

Definition 23 A $(\ell(n), n)$-pseudo-random generator f against $\mathcal{C}(s(n), d(n))$ is $t(m)$-*computable* if there is an algorithm running in time $t(m)$ that on input (y, i) with $|y| = m = \ell(n)$ and $1 \leq i \leq n$, outputs i^{th} bit of $f(y)$.

In the above, we have defined the complexity of f slightly differently – usually it is defined to be the complexity of computing the entire $f(y)$. Our definition has an advantage when both $\ell(n)$ and $t(\ell(n))$ are substantially smaller than n. In that case, the first few bits of f can be computed very quickly and this fact would be useful later.

3 Boolean Circuit Lower Bounds via Pseudo-Random Generators

A $(\ell(n), n)$-pseudo-random generator against $\mathcal{C}(s(n), d(n))$ that is $2^{O(m)}$-computable yields a lower bound on $\mathcal{C}(s(\ell^{-1}(n)), d(\ell^{-1}(n)))$.

Theorem 31 ([14]) *Let f be a $t(m)$-computable $(\ell(n), n)$-pseudo-random generator against $\mathcal{C}(s(n), d(n))$. Then there is a set in* $\mathrm{Ntime}(t(m)\cdot m) \cap \mathrm{Dtime}(t(m)\cdot 2^m)$ *that cannot be accepted by any circuit family in $\mathcal{C}(s(\ell^{-1}(n)), d(\ell^{-1}(n)))$.*

Proof. Define a set A as follows:

> On input x, $|x| = m = \ell(n) + 1$ for some n, find if there exists a y, $|y| = \ell(n)$ such that x is a prefix of $f(y)$. If yes, accept otherwise reject.

The set A is in $\mathrm{Ntime}(t(m) \cdot m)$: guess a y of size $m-1$ and compute first m bits of $f(y)$. The set of also in $\mathrm{Dtime}(t(m) \cdot 2^m)$: for every y of size $m-1$ compute the first m bits of $f(y)$ and check if any matches.

Suppose there is a circuit family in $\mathcal{C}(s(\ell^{-1}(n)), d(\ell^{-1}(n)))$ that accepts A. Fix an input size $m = \ell(n)+1$ for some n and consider the corresponding circuit C from the family. Construct a new circuit, say D, on input size n as follows. Circuit D simply simulates circuit C on the first m bits of its input (ignoring the remaining bits). By the definition of A, it follows that for every y, $|y| = m-1$, $f(y)$ is accepted by D. In addition, circuit D rejects at least half of its inputs (because the number of prefixes of m bits of $f(y)$'s is at most 2^{m-1}). Circuit D is in $\mathcal{C}(s(n), d(n))$ since the input size has grown from m (for circuit C) to n (for circuit D). Therefore,

$$\left| \Pr_{x\in\{0,1\}^n}[D(x) = 1] - \Pr_{y\in\{0,1\}^{m-1}}[D(f(y)) = 1] \right| \leq \frac{1}{n}.$$

However, the first probability is less than $\frac{1}{2}$ while the second is 1 as argued above. This is a contradiction. □

For optimal generators, we get the following corollary.

Corollary 32 *Let f be a $t(m)$-computable optimal pseudo-random generator against $\mathcal{C}(s(n), d(n))$. Then there is a set in* $\mathrm{Ntime}(t(m)\cdot m) \cap \mathrm{Dtime}(t(m)\cdot 2^m)$ *that cannot be accepted by any circuit family in $\mathcal{C}(2^{\epsilon n}, d(s^{-1}(2^{\epsilon n})))$ for some $\epsilon > 0$.*[1]

As of now, the best pseudo-random generator known is the following.

Lemma 33 ([8, 14]) *For any $d > 0$, there exists a $m^{O(1)}$-computable $(\log^{O(d)} n, n)$-pseudo-random generator against $\mathcal{C}(n, d)$, the class of size n, depth d circuits.*

[1] The converse of this corollary was shown by Impagliazzo and Wigderson [10].

The above generator is constructed by taking Håstad's lower bound [8] on constant-depth circuits and applying Nisan-Wigderson's construction [14] of pseudo-random generators on it. This generator is clearly not an optimal generator, but comes close – its input length is $\log^{O(d)} n$ instead of $O(\log n)$. If one can reduce the input length to, say, $O(t(d)\log n)$ for any function $t(\cdot)$, then we get an optimal generator. This is our first step:

Step 1. *Obtain a $2^{O(m)}$-computable optimal pseudo-random generator against $\mathcal{C}(n,d)$ for each $d > 0$.*

Such generators will already yield an interesting lower bound.

Lemma 34 *If there exists a $2^{O(m)}$-computable optimal pseudo-random generator against $\mathcal{C}(n,d)$ then there is a set in* E *that cannot be accepted by any circuit family in $\mathcal{C}(2^{\epsilon n},d)$ for some $\epsilon > 0$.*

Proof. Direct from Corollary 32. □

It is worth mentioning at this point that exponential lower bounds are not known even for *depth three* circuits! The above lemma implies another lower bound:

Corollary 35 *If there exists a $2^{O(m)}$-computable optimal pseudo-random generator against $\mathcal{C}(n,d)$ then there is a set in* E *that cannot be accepted by a non-uniform semiunbounded circuit family of size $n^{d-\epsilon}$, depth $(d-\epsilon)\log n$ for any $\epsilon > 0$.*

Proof. Take any size $n^{d-\epsilon}$, depth $(d-\epsilon)\log n$ (for some $\epsilon > 0$) circuit C on n inputs with unbounded fanin OR-gates. The circuit can be converted into a subexponential size depth d circuit as follows. Cut C into $\frac{d}{2}$ layers of depth $\frac{2(d-\epsilon)}{d}\log n$ each. In each layer, write each topmost gate as OR-of-ANDs of bottommost gates. A direct counting shows that each such OR-of-ANDs will have $O(n^{2dn^{1-\frac{\epsilon}{d}}})$ gates. There are at most $n^{d-\epsilon}$ OR-of-ANDs, and therefore, the size of the resulting circuit is at most $2^{O(\log n \cdot n^{1-\frac{\epsilon}{d}})} = 2^{o(n)}$. The depth of the circuit is d. The existence of a $2^{O(m)}$-computable optimal pseudo-random generator against $\mathcal{C}(n,d)$ implies the existence of a set A in E that cannot be accepted by any family of circuits from $\mathcal{C}(2^{\delta n},d)$ for suitable $\delta > 0$ by the above lemma. This means that circuit C cannot accept $\{A\}_{=n}$. □

By improving the complexity of the generator, we can get a better lower bound. This is our second step:

Step 2. *Obtain a $m^{O(1)}$-computable optimal pseudo-random generator against $\mathcal{C}(n,d)$ for each $d > 0$.*

The better time complexity of the generator implies that the set A will now belong to the class NP instead of E. Thus we get:

Corollary 36 *If there exists a $m^{O(1)}$-computable optimal pseudo-random generator against $\mathcal{C}(n,d)$ then there is a set in* NP *that cannot be accepted by a non-uniform semiunbounded circuit family of size $n^{d-\epsilon}$, depth $(d-\epsilon)\log n$ for any $\epsilon > 0$.*

The next aim is to construct an optimal pseudo-random generator against a larger class of circuits: class of logarithmic depth circuits of fanin two, i.e., NC^1. This class contains constant depth circuits and is only "slightly higher" (although *no* lower bounds are known for this class).

Step 3. *Obtain a $m^{O(1)}$-computable optimal pseudo-random generator against $\mathcal{C}(n,\log n)$.*

This generalization improves the lower bound *substantially.*

Lemma 37 *If there exists a $m^{O(1)}$-computable optimal pseudo-random generator against $\mathcal{C}(n,\log n)$ then there is a set in* NP *that cannot be accepted by any non-uniform family of sublinear depth and subexponential size circuits.*

Proof. Directly from Corollary 32. □

The last step is to push the class of circuits further up to all polylog depth circuits, i.e., the class NC. This class is believed to be substantially smaller than the class of all polynomial sized circuits.

Step 4. *Obtain a $m^{O(1)}$-computable optimal pseudo-random generator against $\mathcal{C}(n,\log^{O(1)} n)$.*

The following lemma follows immediately.

Lemma 38 *If there exists a $m^{O(1)}$-computable optimal pseudo-random generator against $\mathcal{C}(n,\log^{O(1)} n)$ then there is a set in* NP *that cannot be accepted by any non-uniform family of polynomial depth and subexponential size circuits.*

As a corollary of above, we have:

Corollary 39 *If there exists a $m^{O(1)}$-computable optimal pseudo-random generator against $\mathcal{C}(n,\log^{O(1)} n)$ then* $\mathrm{P} \neq \mathrm{NP}$.

4 Pseudo-Random Generators for Arithmetic Circuits

Lower bounds for arithmetic circuits are even less understood than boolean circuits. For example, we do not even know lower bounds on depth four arithmetic circuits. Mulmuley and Sohoni [12] have been trying to use algebraic geometric techniques for proving arithmetic circuit lower bounds. Here, we formulate an alternative way using pseudo-random generators.

Let $\mathcal{A}(n,F)$ be the class of arithmetic circuits over field F such that any circuit $C \in \mathcal{A}(n,F)$ has n addition, subtraction, and multiplication gates over

the field F. We assume that all arithmetic circuits are *layered* and the layers alternate between multiplication and addition/subtraction gates. Circuit C has at most n input variables and computes a polynomial over F of degree at most 2^n. Note that the number of input variables for arithmetic circuit is not as important parameter as for boolean circuits. Even single variable circuits can compute very complex polynomials. Kabanets and Impagliazzo [11] showed a connection between *polynomial identity testing* and lower bounds on arithmetic circuits. They proved that if there is a polynomial-time deterministic algorithm for verifying polynomial identities then NEXP cannot have polynomial-sized arithmetic circuits. They also proved a partial converse: if Permanent cannot be computed by polynomial-sized arithmetic circuits, then polynomial identity testing can be done in subexponential time.

We make this relationship between identity testing and lower bounds stronger via an appropriate notion of pseudo-random generator against arithmetic circuits.

Definition 41 Function $f : \mathbb{N} \mapsto (F[y])^*$ is a $(\ell(n), n)$-*pseudo-random generator* against $\mathcal{A}(n, F)$ if:

- $f(n) \in (F[y])^{n+1}$ for every $n > 0$.
- Let $f(n) = (f_1(y), \ldots, f_n(y), g(y))$. Then each $f_i(y)$ as well as $g(y)$ is a polynomial of degree at most $2^{\ell(n)}$.
- For any circuit $C \in \mathcal{A}(n, F)$ with $m \leq n$ inputs:
  $C(x_1, x_2, \ldots, x_m) = 0$ iff $C(f_1(y), f_2(y), \ldots, f_m(y)) = 0 \pmod{g(y)}$.

A direct application of Schwartz-Zippel lemma [18, 23] shows that there always exist $(O(\log n), n)$-pseudo-random generators against $\mathcal{A}(n, F)$:

> For every $n > 0$, define $f(n)$ to be the sequence $(f_1(y), f_2(y), \ldots, f_n(y), g(y))$ where each $f_i(y)$ is a random degree $n^3 - 1$ polynomial and $g(y)$ is an irreducible polynomial of degree n^3 over F. In addition, if F is infinite, all these polynomials have coefficients bounded by 2^n. Let $C \in \mathcal{A}(n, F)$ compute a non-zero polynomial on $m \leq n$ variables. Let $\hat{F}$ be the extension field $F[y]/(g(y))$. Polynomial $f_i(y)$ can be thought of as a random element of the field $\hat{F}$. Now by Schwartz-Zippel lemma, the probability that $C(f_1(y), f_2(y), \ldots, f_m(y)) = 0 \pmod{g(y)}$ is at most $\frac{\deg C}{2^{n^3}} \leq \frac{1}{2^{n^3-n}}$ since $\deg C \leq 2^n$. Since there are at most 2^{n^2} circuits of size n, the probability that the generator fails against any such circuit is at most $\frac{1}{2^{n^3-n^2-n}}$. Therefore, most of the choices of f are pseudo-random.

As in the case of boolean circuits, we call such generators *optimal* pseudo-random generators. There are, however, a few of crucial differences between the boolean and arithmetic cases. Firstly, the pseudo-random generator against arithmetic circuits does not approximate the number of zeroes of the polynomial computed by the circuit. Secondly, it computes a polynomial for each input instead of a bit value and a moduli polynomial. Finally, it outputs only *one* sequence of $n + 1$ polynomials as opposed to a polynomial number of strings of length

n in the boolean case. The degree of each output polynomial is $2^{\ell(n)}$ which equals $n^{O(1)}$ for optimal generator. Therefore, the time needed to compute such a generator is $2^{\Omega(\ell(n))}$ ($= n^{\Omega(1)}$ for optimal case). This can be exponentially larger than the input size of the generator. Hence we do not have as much freedom available to vary the time complexity of the generator. This motivates the following definition.

Definition 42 A $(\ell(n), n)$-pseudo-random generator f against $\mathcal{A}(n, F)$ is *efficiently computable* if $f(n)$ is computable in time $2^{O(\ell(n))}$.

This definition of pseudo-random generators is the right one from the perspective of derandomization of identity testing.

Theorem 43 *Suppose there exists an efficiently computable $(\ell(n), n)$-pseudo-random generator against $\mathcal{A}(n, F)$. Then polynomial identity testing can be done deterministically in time $n \cdot 2^{O(\ell(n))}$.*

Proof. Let $C \in \mathcal{A}(n, F)$ be a circuit of size n computing a possible identity over F on $m \leq n$ variables. Then $C(f_1(y), f_2(y), \ldots, f_m(y)) \pmod{g(y)}$ can be computed in time $2^{O(\ell(n))}$: each $f_i(y)$ and $g(y)$ is of degree $2^{O(\ell(n))}$ and can be computed in the same time; and then the circuit C can be evaluated modulo $g(y)$ in $n \cdot 2^{O(\ell(n))}$ time. □

Corollary 44 *Suppose there exists an efficiently computable optimal pseudo-random generator against $\mathcal{A}(n, F)$. Then polynomial identity testing can be done in* P.

A further evidence of "correctness" of the definition is provided by the AKS primality test [2] which can be viewed as derandomization of a specific identity. The identity is $C(x) = (1 + x)^n - x^n - 1$ over Z_n and, as shown in [1], the function $f(n) = (x, g(x))$ with

$$g(x) = x^{16 \log^5 n} \cdot \prod_{r=1}^{16 \log^5 n} \prod_{a=1}^{4 \log^4 n} ((x - a)^r - 1)$$

($g(x)$ is of degree $O(\log^{14} n)$) is an efficiently computable optimal "pseudo-random generator" against $C(x) \in \mathcal{A}(O(\log n), Z_n)$ (it is not really a pseudo-random generator since it works only against a subset of circuits in $\mathcal{A}(O(\log n), Z_n)$).

5 Arithmetic Circuit Lower Bounds via Pseudo-Random Generators

As in the case of boolean circuits, an efficiently computable pseudo-random generator implies a lower bound:

Theorem 51 *Let f be an efficiently computable $(\ell(n), n)$-pseudo-random generator against $\mathcal{A}(n, F)$. Then there is a multilinear polynomial computable in time $2^{O(\ell(n))}$ that cannot be computed by any circuit family in $\mathcal{A}(n, F)$.*[2]

Proof. For any $m = \ell(n)$, define polynomial $q(x_1, x_2, \ldots, x_{2m})$ as:

$$q(x_1, x_2, \ldots, x_{2m}) = \sum_{S \subseteq [1,2m]} c_S \cdot \prod_{i \in S} x_i.$$

The coefficients c_S satisfy the condition

$$\sum_{S \subseteq [1,2m]} c_S \cdot \prod_{i \in S} f_i(y) = 0$$

where $f(n) = (f_1(y), f_2(y), \ldots, f_n(y), g(y))$. Such a q always exists as the following argument shows.

> The number of coefficients of q are exactly 2^{2m}. These need to satisfy a polynomial equation of degree at most $2m \cdot 2^m$. So the equation gives rise to at most $2m \cdot 2^m + 1$ homogeneous constraints on the coefficients. Since $(2m \cdot 2^m + 1) < 2^{2m}$ for $m \geq 3$, there is always a non-trivial polynomial q satisfying all the conditions.

The polynomial q can be computed by solving a system of $2^{O(m)}$ linear equations in $2^{O(m)}$ variables over the field F. Each of these equations can be computed in time $2^{O(m)}$ using computability of f. Therefore, q can be computed in time $2^{O(m)}$. Now suppose q can be computed by a circuit $C \in \mathcal{A}(n, F)$. By the definition of polynomial q, it follows that $C(f_1(y), f_2(y), \ldots, f_{2m}(y)) = 0$. The size of circuit C is n and it computes a non-zero polynomial. This contradicts the pseudo-randomness of f. □

As in the case of boolean circuits, optimal pseudo-random generators against constant depth arithmetic circuits is our first goal.

Step 1. *Obtain an efficiently-computable optimal pseudo-random generator against size n arithmetic circuits of depth d over F for each $d > 0$.*

And exactly as in the boolean case, we get a lower bound on log-depth polynomial size circuits with unbounded fanin addition gates.

Lemma 52 *If there exist efficiently-computable optimal pseudo-random generators against size n arithmetic circuits of depth d over F then for there exists a multilinear polynomial computable in* E *that cannot be computed by a nonuniform family of circuits with unbounded fanin addition gates of size $n^{d-\epsilon}$, depth $(d - \epsilon) \log n$ for any $\epsilon > 0$.*

[2] A partial converse of this theorem can also be shown: if there exists a polynomial computable in time $2^{O(\ell(n))}$ that cannot be computed by a circuit family in $\mathcal{A}(n, F)$ then there exists an efficiently computable $(\ell^2(n), n)$-pseudo-random generator against the class of size n circuits over F whose degree is bounded by n.

Proof. A size $n^{d-\epsilon}$, depth $(d-\epsilon)\log n$ arithmetic circuit with unbounded fanin addition gates can be translated, exactly as in proof of Lemma 35, to a subexponential sized depth d circuit. The optimal pseudo-random generator against depth d circuits gives the lower bound. □

The class of arithmetic branching programs is equivalent to the class of polynomials computed by determinants of a polynomial sized matrix [21, 19, 5]. Also, polynomial-sized arithmetic formulas can be expressed as polynomial sized arithmetic branching programs. We get a much stronger lower bound by generalizing the pseudo-random generator to work against polynomial sized branching programs.

Step 2. *Obtain an efficiently-computable optimal pseudo-random generator against size n arithmetic branching programs over F.*

This step nearly achieves our final goal.

Lemma 53 *If there exist efficiently-computable optimal pseudo-random generators against size n arithmetic branching programs over F then there exists a multilinear polynomial computable in* E *that (1) cannot be expressed as the determinant of a subexponential sized matrix and (2) cannot be computed by a $2^{o(\frac{n}{\log n})}$-sized arithmetic circuit.*

Proof. The first part follows directly from Theorem 51 translated for arithmetic branching programs. For the second part, recall that the polynomial q is multilinear and so has polynomial degree. In [20] it is shown that arithmetic circuits of size N and degree D can be transformed to arithmetic circuits of size $N^{O(1)}$ with depth $O(\log N \log D)$. Further, a circuit of depth $O(\log N \log D)$ can be expressed as determinant of a matrix of size $2^{O(\log N \log D)} = N^{O(\log D)}$. Using the lower bound of first part, it follows that the polynomial q cannot be computed by arithmetic circuits of size $2^{o(\frac{n}{\log n})}$. □

To obtain a lower bound on permanent, we need to improve the time complexity of polynomial q. Suppose that each coefficient c_S of the polynomial q can be computed by a #P-function (this will require that all coefficients of each polynomial in $f(n)$ to be computed by a #P-function). Then it follows that the polynomial q can be expressed as permanent of a matrix of size polynomial in m (because permanent captures #P-computations). Let us call such a generator #P-*computable*.

Step 3. *Obtain a* #P*-computable optimal pseudo-random generator against size n arithmetic branching programs over F.*

Corollary 54 *If there exists an efficiently-computable optimal pseudo-random generator against size n arithmetic branching programs over F then the permanent of a $n \times n$ matrix over F (1) cannot be expressed as the determinant of a subexponential-sized matrix over F, (2) cannot be computed by a $2^{o(\frac{n}{\log n})}$-sized arithmetic circuit.*

Of course, the above step cannot be carried out for fields of characteristic two where permanent is equal to the determinant.

6 Will This Approach Work?

In the sequence of steps proposed to prove arithmetic and boolean circuit lower bounds, perhaps the most important one is step 1. Achieving this step will, besides providing strong lower bounds for the first time, will establish the correctness of the approach and increase the possibility that remaining steps can also be achieved.

In this section, we discuss some potential candidates to achieve Step 1 for both boolean and arithmetic circuits.

6.1 Step 1 for Boolean Circuits

Hitting set generators are a weaker form of optimal pseudo-random generators. The difference is that for a circuit C that accepts at least half of its inputs, a hitting set generator is required to generate at least one input x to C such that $C(x) = 1$. (As opposed to this, pseudo-random generators are required to generate appoximately $\Pr_x[C(x) = 1]$ fraction of x's on which $C(x) = 1$.) Both hitting set generators and optimal pseudo-random generators against $\mathcal{C}(s(n), d(n))$ have input of size $O(\log s(n))$.

The proof of Theorem 31 shows that efficiently computable hitting set generators are sufficient to obtain lower bounds. Coupled with the result of [10], this implies that efficiently computable hitting set generators and efficiently computable optimal pseudo-random generators are equivalent.

Let $f : \{0,1\}^{O(\log n)} \mapsto \{0,1\}^n$ be any $\frac{1}{n^{2d}}$*-biased,* $2\log n$*-wise independent* generator. In other words, any $2\log n$ output bits of f, on a random input, are nearly independent (with a bias of at most $\frac{1}{n^{2d}}$). There exist several constructions of such generators [13, 3]. We take one from [3]. Define $f_{B,d}$, $f_{B,d} : \{0,1\}^{(4d+4)\log n} \mapsto \{0,1\}^n$, as:

$$f_{B,d}(x, y) = (x^0 \cdot y)(x^1 \cdot y) \cdots (x^{n-1} \cdot y)$$

where $|x| = |y| = (2d + 2)\log n$, x^i is computed in the field $F_{n^{2d+2}}$ treating x as an element of the field, and '$\cdot$' is inner product modulo 2. It is shown in [3] that $f_{B,d}$ satisfies the required independence property.

Functions $f_{B,d}$ can easily shown to be $m^{O(1)}$ computable. We can prove the following about function $f_{B,2}$:

Lemma 61 *Function $f_{B,2}$ is a hitting set generator against depth 2, size n boolean circuits.*

Proof. Without loss of generality, consider a depth 2, size n circuit C that is an OR-of-ANDs and accepts at least half fraction of inputs. Delete all the AND gates from C of fanin more than $2\log n$. Let the resulting circuit be C'. Any input accepted by circuit C' is also accepted by C and the fraction of inputs accepted by C' is at least $\frac{1}{2} - \frac{1}{n}$ (a deleted AND gate outputs a 1 on at most $\frac{1}{n^2}$ inputs and there are at most n deleted AND gates). Consider any surviving

AND gate in the circuit. Its fanin is at most $2\log n$. Therefore, it outputs a 1 on at least $\frac{1}{n^2}$ inputs. Since the output of $f_{B,2}$ is $2\log n$-wise independent with a bias of at most $\frac{1}{n^4}$, the probability that this AND gate will output a 1, when given $f_{B,2}$ as input, is at least $\frac{1}{n^2} - \frac{1}{n^4} > 0$. Hence $f_{B,2}$ is a hitting set generator against C.

About lemma and Lemma 35 together show that:

Corollary 62 *There is a set in* NP *that cannot be accepted by semiunbounded circuits of size* $n^{2-\epsilon}$ *and depth* $(2-\epsilon)\log n$ *for any* $\epsilon > 0$.

In fact, using a different definition for $f_{B,2}$ from [3] can bring the complexity of the hard set down to SAC^1 from NP. This implies that $f_{B,d}$ *cannot* be a hitting set generator for all d (because SAC^1 circuits can be transformed to subexponential sized constant depth circuits as observed earlier). However, it appears that a combination of $f_{B,d}$ with other derandomization primitives can result in hitting set generators for higher depths.

6.2 Step 1 for Arithmetic Circuits

We need to weaken the definition of pseudo-random generators for arithmetic circuits too.

Definition 63 Function $f : \mathbb{N} \times \mathbb{N} \mapsto (F[y])^*$ is a *hitting set generator* against $\mathcal{A}(n, F)$ if:

- $f(n,k) \in (F[y])^{n+1}$ for every $n > 0$ and $1 \leq k \leq n^{O(1)}$.
- Let $f(n,k) = (f_{1,k}(y), \ldots, f_{n,k}(y), g_k(y))$. Then each $f_{i,k}(y)$ as well as $g_k(y)$ is a polynomial of degree at most $n^{O(1)}$.
- For any circuit $C \in \mathcal{A}(n,F)$ with $m \leq n$ inputs:
  $C(x_1, x_2, \ldots, x_m) = 0$ iff for every k, $1 \leq k \leq n^{O(1)}$,

$$C(f_{1,k}(y), f_{2,k}(y), \ldots, f_{m,k}(y)) = 0 \pmod{g_k(y)}.$$

It is easy to see that a complete derandomization of identity testing can also be done by an efficiently computable hitting set generator. By slightly modifying the definition of polynomial q in the proof of Theorem 51, a similar lower bound can be shown too (q will now need to satisfy $n^{O(1)}$ polynomial equations of degree $n^{O(1)}$ instead of just one; this still translates to $n^{O(1)}$ homogeneous constraints on the coefficients of q).

Define function $f_{A,d}$ as:

$$f_{A,d}(n,k) = (y^{k^0}, y^{k^1}, \ldots, y^{k^{n-1}}, y^r - 1)$$

where $r \geq n^{4d}$ is a prime and $1 \leq k < r$.

Function $f_{A,d}$ is easily seen to be $n^{O(1)}$ computable. Polynomial q, defined for function $f_{A,d}$, can be computed in PSPACE (it is not clear how to compute q in #P). We can prove the following about function $f_{A,2}$:

Lemma 64 *Function $f_{A,2}$ is a hitting set generator against size n, depth 2 arithmetic circuits.*

Proof. Consider a size n, depth 2 arithmetic circuit C computing a non-zero polynomial. If C has a multiplication gate at the top, then it will be non-zero on any sequence $f_{A,2}(n,k)$ such that $k^i \neq k^j \pmod r$ for $1 \leq i < j < n$. Most of the k's (e.g., any k which is a generator for F_r^*) have this property.

Now consider C with an addition gate at the top. C then computes a polynomial of degree up to n with at most n non-zero terms. Let t_1 be the first term of this polynomial (under some ordering) and let t_j be any other term. Then the number of k's for which $t_1 = t_j \pmod{y^r - 1}$ under the substitution of variables according to $f_{A,2}(n,k)$ is at most $n-1$. So the number of k's for which $t_1 = t_j \pmod{y^r - 1}$ for *some* j is at most n^2. As total number of k's is n^4, there exist k's for which $t_1 \neq t_j \pmod{y^r - 1}$ for any $j > 1$ under the substitution $f_{A,2}(n,k)$. C evaluates to a non-zero polynomial modulo $y^r - 1$ on such inputs.

Using Lemma 52, this results in:

Corollary 65 *There is a multilinear function computable in* PSPACE *that cannot be computed by circuits with unbounded fanin addition gates of size $n^{2-\epsilon}$ and depth $(2-\epsilon)\log n$ for any $\epsilon > 0$.*

We conjecture that above lemma holds for all depths:

Conjecture. *Function $f_{A,d}$ is a hitting set generator against depth d, size n arithmetic circuits for every $d > 0$.*

It is to be hoped that the next twenty five years will be more fruitful for lower bounds than the previous ones. One might even hope that all the proposed steps will be achieved answering two of the most fundamental questions in complexity theory.

References

[1] Manindra Agrawal. On derandomizing tests for certain polynomial identities. In *Proceedings of the Conference on Computational Complexity*, pages 355–362, 2003.

[2] Manindra Agrawal, Neeraj Kayal, and Nitin Saxena. PRIMES is in P. *Annals of Mathematics*, 160(2):781–793, 2004.

[3] N. Alon, O. Goldreich, J. Håstad, and R. Peralta. Simple constructions of almost k-wise independent random variables. In *Proceedings of Annual IEEE Symposium on Foundations of Computer Science*, pages 544–553, 1990.

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

[5] C. Damm. DET=$\mathrm{L}^{\#l}$. Technical Report Informatik-preprint 8, Fachbereich Informatik der Humboldt Universität zu Berlin, 1991.

[6] L. Fortnow. The role of relativization in complexity theory. Bulletin of the European Association for Theoretical Computer Science, 1994. Complexity Theory Column.

[7] L. Fortnow. Time-space tradeoffs for satisfiability. *J. Comput. Sys. Sci.*, 60(2):337–353, 2000.

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

[9] J. Håstad, R. Impagliazzo, L. Levin, and M. Luby. A pseudo-random generator from any one-way function. *SIAM Journal on Computing*, pages 221–243, 1998.

[10] 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.

[11] Valentine Kabanets and Russell Impagliazzo. Derandomizing polyonmial identity tests means proving circuit lower bounds. In *Proceedings of Annual ACM Symposium on the Theory of Computing*, pages 355–364, 2003.

[12] K. Mulmuley and M. Sohoni. Geometric complexity theory I: An approach to the P vs. NP and other related problems. *SIAM Journal on Computing*, 31(2):496–526, 2002.

[13] J. Naor and M. Naor. Small-bias probability spaces: Efficient constructions and applications. In *Proceedings of Annual ACM Symposium on the Theory of Computing*, pages 213–223, 1990.

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

[15] A. Razborov. Lower bounds for the monotone complexity of some boolean functions. *Doklady Akademii Nauk SSSR*, 281(4):798–801, 1985. English translation in Soviet Math. Doklady, 31:354-357, 1985.

[16] A. Razborov and S. Rudich. Natural proofs. In *Proceedings of Annual ACM Symposium on the Theory of Computing*, pages 204–213, 1994.

[17] O. Reingold. Undirected s-t-connectivity in logspace. In *Proceedings of Annual ACM Symposium on the Theory of Computing*, pages 376–385, 2005.

[18] J. T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *J. ACM*, 27(4):701–717, 1980.

[19] S. Toda. Counting problems computationally equivalent to the determinant. manuscript, 1991.

[20] L. Valiant, S. Skyum, S. Berkowitz, and C. Rackoff. Fast parallel computation of polynnomials using few processors. *SIAM Journal on Computing*, 12:641–644, 1983.

[21] V Vinay. Counting auxiliary pushdown automata and semi-unbounded arithmetic circuits. In *Proceedings of the Structure in Complexity Theory Conference*, pages 270–284. Springer LNCS 223, 1991.

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

[23] R. E. Zippel. Probabilistic algorithms for sparse polynomials. In *EUROSCAM'79*, pages 216–226. Springer LNCS 72, 1979.