A note on exponential circuit lower bounds from derandomizing Arthur-Merlin games

Scott Aaronson
MIT
aaronson@csail.mit.edu

Barış Aydınlıoğlu
University of Wisconsin-Madison
baris@cs.wisc.edu

Harry Buhrman
CWI & University of Amsterdam
buhrman@cwi.nl

John Hitchcock[*]
University of Wyoming
jhitchco@cs.uwyo.edu

Dieter van Melkebeek[†]
University of Wisconsin-Madison
dieter@cs.wisc.edu

November 12, 2010

Abstract

We present an alternate proof of the recent result by Gutfreund and Kawachi that derandomizing Arthur-Merlin games into P^{NP} implies linear-exponential circuit lower bounds for E^{NP}. Our proof is simpler and yields stronger results. In particular, consider the promise-AM problem of distinguishing between the case where a given Boolean circuit C accepts at least a given number b of inputs, and the case where C accepts less than $\delta \cdot b$ inputs for some positive constant δ. If P^{NP} contains a solution for this promise problem then E^{NP} requires circuits of size $\Omega(2^n/n)$ almost everywhere.

1 Introduction

Derandomization has strong ties with circuit lower bounds. In the setting of randomized decision procedures the existence of pseudorandom generators is equivalent to circuit lower bounds for $\mathrm{E} \doteq \mathrm{DTIME}(2^{O(n)})$. At the high end, pseudorandom generators that yield BPP = P are equivalent to E requiring circuits of linear-exponential size. At the low end, pseudorandom generators that put BPP in deterministic subexponential time are equivalent to E requiring circuits of superpolynomial size.

Whether *any* deterministic simulation of BPP implies the same circuit lower bounds as a simulation through pseudorandom generators would, remains open. In recent years some partial results along those lines were established at the low end of the derandomization spectrum. Impagliazzo et al. [IKW02] showed that simulations of promise-BPP in deterministic subexponential time

[*]Partially supported by an NWO travel grant and by NSF grants 0652601 and 0917417.
[†]Partially supported by NSF grants 0728809 and 1017597.

imply that NE $\doteq$ NTIME$(2^{O(n)})$ requires circuits of superpolynomial size. Kabanets and Impagliazzo [KI04] proved that the same level of derandomization for BPP proper implies that either NE requires circuits of superpolynomial size or else the permanent over the integers requires arithmetic circuits of superpolynomial size. Kinne et al. [KvMS09] (see also [AvM10]) showed that the circuit lower bound for NE in the latter statement can be made linear-exponential and holds for NE$\cap$coNE. However, a genuine high-end result remains elusive, i.e., that a simulation of (promise-)BPP in P implies that E requires circuits of linear-exponential size.

In the setting of Arthur-Merlin games we know that pseudorandom generators that yield nondeterministic simulations are equivalent to nondeterministic circuit lower bounds for NE $\cap$ coNE. As for circuit lower bounds that follow from *any* type of nondeterministic simulation of AM, Impagliazzo et al. [IKW02] showed that a low-end derandomization of MA yields a language in NE that requires deterministic circuits of superpolynomial size. Very recently, Gutfreund and Kawachi [GK10] established the first result at the high end, namely that derandomizing promise-AM into P^{NP} implies that E^{NP} requires deterministic circuits of linear-exponential size.

Theorem 1 (Gutfreund-Kawachi [GK10]). [1] *If promise-*AM *can be simulated in* P^{NP}*, then* E^{NP} *requires circuits of size* $2^{\epsilon n}$ *for every constant* $\epsilon < 1$ *and infinitely many input lengths* n*.*

Gutfreund and Kawachi also argued that the same lower bound follows from derandomizing into NP a specific promise problem that is closely related to approximate counting.

Theorem 2 (Gutfreund-Kawachi [GK10]). [2] *Let* C *denote a Boolean circuit on* m *inputs,* b *an integer in binary, and* c *a positive real. If there is a language in* NP *that contains all instances* (C, b) *where* $|C^{-1}(1)| \geq b$*, and does not contain any instances where* $|C^{-1}(1)| < \delta(m) \cdot b$ *for* $\delta(m) = 1 - 1/m^c$*, then* E^{NP} *requires circuits of size* $2^{\epsilon n}$ *for* $\epsilon = 1 - O(1/c)$ *and infinitely many input lengths* n*.*

Note that the promise problem in Theorem 2 is known to lie in promise-AM [GS89].

2 Statement

In this note we present a simpler proof of Theorems 1 and 2 that leads to the following stronger result.

Theorem 3. *Let* C *denote a Boolean circuit, and* b *an integer in binary. If for some positive constant* δ *there is a language in* P^{NP} *that contains all instances* (C, b) *where* $|C^{-1}(1)| \geq b$ *and does not contain any instances where* $|C^{-1}(1)| < \delta \cdot b$*, then there is a language in* E^{NP} *that requires circuits of size* $\alpha 2^n/n$ *for some positive constant* α *and all but finitely many input lengths* n*.*

Recall that the maximum circuit complexity is $\Theta(2^n/n)$. Theorem 3 is stronger than Theorem 1 because the circuit lower bound is $\Omega(2^n/n)$ rather than $2^{\epsilon n}$ for any constant $\epsilon < 1$, and also because

[1] Gutfreund and Kawachi only state Theorem 1 for *some* constant $\epsilon > 0$. A careful analysis of their proof and some tweaks to the results they rely on reveals that their approach works for *any* constant $\epsilon < 1$.

[2] Gutfreund and Kawachi state their result using an additional unary input a, and set $\delta = 1 - 1/a$. They need $a = m^c$ for $c > 10$ to establish the lower bound for *some* constant $\epsilon > 0$. A careful analysis of their proof and some tweaks to the results they rely on reveals that their approach with $a = m^c$ yields $\epsilon = 1 - \Theta(1/c)$.

the lower bound holds almost everywhere rather than infinitely often. Theorem 3 is stronger than Theorem 2 for the same reasons and also because we only need a derandomization into P^{NP} (rather than into NP) and because the gap between the two cases of the promise problem can be constant (rather than polynomially small).

3 Proof Idea

Apart from the above strengthenings, our proof is significantly simpler than the one in [GK10]. The proof in [GK10] consists of a case distinction and relies on results about strong Karp-Lipton collapses and learning circuits with the use of an oracle for NP. In contrast, our proof is direct and elementary.

Our idea is to follow Kannan's approach [Kan82] and construct a language that requires large circuits by setting the prefix of its characteristic sequence so as to quickly diagonalize against all small circuits. More specifically, we mimic the process of successively setting the next bit of the characteristic sequence to the minority vote of the circuits of size $\alpha 2^n/n$ that are consistent with the sequence constructed thus far. This ideal process would reduce the number of consistent circuits by at least half in each step, implying that we'd be done after $O(\alpha 2^n)$ steps. Now, let A be a language in P^{NP} that solves the promise problem described in the statement of Theorem 3. Using A as an oracle, we can mimic the above ideal process and guarantee that we reduce the number of consistent circuits by some constant factor $\beta < 1$, where β depends on δ. To do so, it suffices to approximate the number of circuits consistent with the sequence thus far extended with a zero, do the same for the extension with a one, and select the extension that gives the smaller estimate. This can be can be done by calling A with an input C that embodies the characteristic sequence, and performing a binary search on the other input b. The process ends after $O(\alpha 2^n/\log(1/\beta))$ steps, which is less than 2^n for sufficiently small α and sufficiently large n. Since A lies in P^{NP}, the resulting process yields a language in E^{NP} that has no circuits of size $\alpha 2^n/n$ for all but finitely many input lengths n.

4 Formal Proof and Parameterized Statement

We now fill in the details and given a formal proof of Theorem 3. We first introduce some parameters that will allow us to state a generalization of Theorem 3.

Let $s(n) \geq n$ denote the circuit lower bound we are shooting for, i.e., we want to construct a language L that requires circuits of size $s(n)$ for all but finitely many input lengths n.

Let A denote a language that solves the promise problem given in the statement of Theorem 3 but where δ can be a function of the number of inputs of the circuits. For a given Boolean circuit C on m inputs and an integer b in binary, A contains (C, b) if $|C^{-1}(1)| \geq b$ and does not contain (C, b) if $|C^{-1}(1)| < \delta(m) \cdot b$. In the middle case where $|C^{-1}(1)| \in [\delta(m) \cdot b, b)$, the membership of (C, b) to A can be arbitrary.

The circuits C we supply to A take as input the description of a circuit D of size $s(n)$ on n inputs. Thus, C takes $m = O(s(n) \log s(n))$ inputs.

We construct L iteratively, where in iteration $i = 0, 1, \ldots,$ we determine χ_i, the ith symbol of the characteristic string χ of L. To explain iteration i we denote by S_i the set of circuits of size $s(n)$

on n inputs that agree with χ up to its ith symbol, i.e., a circuit D is in S_i iff for $j = 0, \ldots, i-1$, D outputs χ_j when given as input the n-bit binary encoding of integer j. In iteration i we first tentatively set χ_i to 0 and use A to obtain an estimate σ_0 on the size of S_{i+1}. Then we set χ_i to 1 and obtain an estimate σ_1. We finalize χ_i to the value $c \in \{0,1\}$ such that $\sigma_c = \min(\sigma_0, \sigma_1)$ (say we set $c = 0$ in case of a tie).

To estimate $|S_{i+1}|$, first we construct a circuit C that recognizes S_{i+1}. The circuit C takes as input a binary string of length m that is the description of a size $s(n)$ circuit D on n inputs, and returns 1 iff $D \in S_{i+1}$. More precisely, C contains as hardcode the characteristic string χ constructed thus far and simulates its input D on inputs $j = 0, \ldots, i$, and accepts iff D agrees with χ for all j. Next, we run a binary search for the largest integer b^* such that $(C, b^*) \in A$. Note that $(C, 0) \in A$ so b^* exists. The binary search returns a value $\tilde{b}$ such that (i) $(C, \tilde{b}) \in A$ and (ii) $(C, \tilde{b}+1) \notin A$. As $A(C, \cdot)$ may not be perfectly monotone, $\tilde{b}$ may differ from b^* but the specification of A guarantees that $|C^{-1}(1)| \geq \delta(m) \cdot \tilde{b}$ (because of (i)) and $|C^{-1}(1)| < \tilde{b} + 1$ (because of (ii)), so our estimate $\tilde{b}$ satisfies $|S_{i+1}| \leq \tilde{b} \leq |S_{i+1}|/\delta(m)$.

At the end of iteration i, if the smaller estimate σ_c turns out 0 then we conclude that the diagonalization is complete and we terminate the iterations. Finally, to decide whether $x \in \{0,1\}^n$ belongs to L, we interpret x as an integer and accept if x is less than the length of χ and the xth symbol of χ is 1; we reject x otherwise. This completes the construction of L.

For the construction to work, we need to make sure that the diagonalization is completed by the time we exhaust the 2^n inputs of length n. Consider the number of circuits eliminated in round i. According to our estimate this number is $\sigma_{\neg c}$, but actually it may be as little as $\delta(m) \cdot \sigma_{\neg c}$. Since the total number of circuits under consideration at round i is at most $2\sigma_{\neg c}$, it follows that at least a $\delta(m)/2$ fraction of those circuits are eliminated during round i. Thus, $|S_i| \leq (1-\delta(m)/2)^i \cdot |S_0| \leq \exp(-i\delta(m)/2) \cdot 2^m$. The latter quantity is less than 1 for $i > 2\ln(2)m/\delta(m)$. So, as long as $2\ln(2)m/\delta(m) < 2^n$, we can complete the diagonalization process as required. Note that the condition is met if $m/\delta(m) = O(2^n)$ for every $m(n) = O(s(n)\log s(n))$.

Let us now analyze the complexity of the resulting language L. The circuits C used in the ith step can be constructed in time $\mathrm{poly}(i, s(n))$. The binary search in each step requires at most m calls to A, as b^* ranges up to 2^m. Assuming A can be decided in time $a(N)$ on inputs of length N (when given access to an oracle for NP), the amount of time for the ith step is $O(m \cdot a(\mathrm{poly}(i, s(n)))$ (when given access to NP). By the previous paragraph, the number of steps is $O(m/\delta(m))$. Hence, given access to NP, the amount of time over all steps is

$$O\left(\frac{m^2}{\delta(m)} \cdot a(\mathrm{poly}(\frac{m}{\delta(m)}, s(n)))\right) = O\left(a(\mathrm{poly}(s(n), \frac{1}{\delta(O(s(n)\log s(n)))}))\right), \tag{1}$$

where we assume that $a(N)$ and $1/\delta(m)$ are monotone and, without loss of generality, that $a(N) \geq N$. If $s(n)$ and $1/\delta(m)$ are constructible as well, (1) also bounds the overall time complexity of L.

We have thus proved the following result, of which Theorem 3 is a special case.

Theorem 4. *Let a, $1/\delta$, and s be functions such that a and $1/\delta$ are monotone, $1/\delta$ and s are constructible, and $m/\delta(m) = O(2^n)$ whenever $m(n) = O(s(n)\log s(n))$. Let C denote a Boolean circuit on m inputs, and b an integer in binary. If there is a language in $\mathrm{DTIME}(a(n))^{\mathrm{NP}}$ that contains all instances (C, b) where $|C^{-1}(1)| \geq b$ and does not contain any instances where $|C^{-1}(1)| < \delta(m) \cdot b$, then there is a language in $\mathrm{DTIME}(t(n))^{\mathrm{NP}}$ that requires circuits of size $s(n)$ for all but finitely* *many input lengths n, where*

$$t(n) = a\left((s(n) \cdot \frac{1}{\delta(O(s(n)\log s(n)))})^{O(1)}\right).$$

Theorem 4 gives the following interesting instantiations that yield that a hard language in E^{NP}.

Corollary 1. *For each combination of the parameters a, δ, and s in the table below, under the hypothesis of Theorem 4, there is a language in E^{NP} that requires circuits of size $s(n)$ for all but finitely many input lengths n.*

$a(n)$	$\delta(n)$	$s(n)$
$n^{O(1)}$	$\Omega(1)$	$\Omega(2^n/n)$
$2^{(\log n)^{O(1)}}$	$1/2^{(\log n)^{O(1)}}$	$2^{n^{\Omega(1)}}$
$2^{n^{o(1)}}$	$1/n^{O(1)}$	$n^{\omega(1)}$

Theorem 3 follows from the first line of the table.

References

[AvM10] Scott Aaronson and Dieter van Melkebeek. A note on circuit lower bounds from derandomization. Technical Report TR 10-105, Electronic Colloquium on Computational Complexity, 2010.

[GK10] Dan Gutfreund and Akinori Kawachi. Derandomizing Arthur-Merlin games and approximate counting implies exponential-size lower bounds. In *Proceedings of the IEEE Conference on Computational Complexity*, pages 38–49, 2010.

[GS89] Shafi Goldwasser and Michael Sipser. Private coins versus public coins in interactive proof systems. In *Randomness and Computation*, volume 5 of *Advances in Computing Research*, pages 73–90. JAI Press, Greenwich, 1989.

[IKW02] Russell Impagliazzo, Valentine Kabanets, and Avi Wigderson. In search of an easy witness: exponential time vs. probabilistic polynomial time. *Journal of Computer and System Sciences*, 65:672–694, 2002.

[Kan82] Ravi Kannan. Circuit-size lower bounds and nonreducibility to sparse sets. *Information and Control*, 55:40–56, 1982.

[KI04] Valentine Kabanets and Russell Impagliazzo. Derandomizing polynomial identity tests means proving circuit lower bounds. *Computational Complexity*, 13:1–46, 2004.

[KvMS09] Jeff Kinne, Dieter van Melkebeek, and Ronen Shaltiel. Pseudorandom generators and typically-correct derandomization. In *Proceedings of the International Workshop on Randomization and Computation*, pages 574–587, 2009.