BPP HAS SUBEXPONENTIAL TIME SIMULATIONS UNLESS *EXPTIME* HAS PUBLISHABLE PROOFS

László Babai, Lance Fortnow,
Noam Nisan and Avi Wigderson

Abstract. We show that *BPP* can be simulated in subexponential time for infinitely many input lengths unless exponential time

- collapses to the second level of the polynomial-time hierarchy,
- has polynomial-size circuits and
- has publishable proofs (*EXPTIME* = *MA*).

We also show that *BPP* is contained in subexponential time unless exponential time has publishable proofs for infinitely many input lengths. In addition, we show *BPP* can be simulated in subexponential time for infinitely many input lengths unless there exist unary languages in *MA* - *P*.

The proofs are based on the recent characterization of the power of multiprover interactive protocols and on random self-reducibility via low-degree polynomials. They exhibit an interplay between Boolean circuit simulation, interactive proofs and classical complexity classes. An important feature of this proof is that it does not relativize.

One of the ingredients of our proof is a lemma that states that if *EXPTIME* has polynomial size circuits then *EXPTIME* = *MA*. This extends previous work by Albert Meyer.

Key words. Complexity Classes, Interactive Proof Systems
Subject classifications. 68Q15

1. Introduction

How much time is necessary to deterministically simulate a probabilistic machine? We could simulate our machine for every possible choice of coin tosses and then compute the probability of acceptance. Simulating a probabilistic polynomial-time Turing machine using this method would require exponential time. Does there exist a faster way?

One solution involves pseudorandom number generators. A pseudorandom number generator maps some set of random bits to a larger set of bits where the larger set of bits is computationally indistinguishable from random. One could use a pseudorandom number generator to reduce the number of random bits needed for a probabilistic algorithm to accept a certain language and thus to reduce the number of computation paths to simulate. Blum and Micali [11] described the first secure pseudorandom generator based on the assumption of the hardness of the discrete log function. These ideas are generalized by Yao [26, 12] who showed that one can convert any one-way permutation into a pseudorandom number generator to show that *BPP* has subexponential time simulations. A series of results [21, 13, 19, 16] show that any one-way function is enough to create a pseudorandom generator that could be used to show that *BPP* is in subexponential time.

Rather than making hardness assumptions on certain types of functions, in this paper we shall consider the effect of complexity theoretic assumptions. Nisan and Wigderson [24] show that if there exists a function computable in exponential time that cannot be approximated by a polynomial-size circuit then there exists a pseudorandom number generator computable in subexponential time that looks random to polynomial-size circuits for infinitely many input lengths. They use this pseudorandom number generator to show that if such a hard function exists then *BPP* has such a weak subexponential simulation.

We generalize this result by showing that a much weaker assumption will still give the same consequence. We show that *BPP* has a weak subexponential simulation, i.e. a simulation in time 2^{n^ϵ} for infinitely many values of n and every $\epsilon > 0$, unless exponential time has polynomial size circuits and is contained in the class *MA* of languages with probabilistically checkable proofs. Since *MA* is known to lie within $\Sigma_2^P \cap \Pi_2^P$ [2], we deduce that subexponential simulations for *BPP* exist unless *EXPTIME* lies within the second level of the polynomial-time hierarchy.

Thus the main theorems of our paper are the following.

Theorem 1.1.

1. *BPP admits weak subexponential time simulations unless EXPTIME* $= MA = \Sigma_2^P \subset P/poly$.

2. *BPP is contained in subexponential time unless MA weakly simulates EXPTIME.*

We will use Theorem 1.1 to also prove the following.

Theorem 1.2. *BPP admits weak subexponential time simulations unless there exist unary languages in $MA - P$.*

From the proof of Theorem 1.2 we will deduce the following corollary.

Corollary 1.3. *If $EH = E$ then $P = BPP$.*

Corollary 1.3 is a rare instance of a collapse at the exponential-time level implying a collapse at the polynomial-time level.

2. Background and Definitions

Nisan and Wigderson [24] showed a general theorem on how certain hard functions could be used to create various pseudorandom number generators. We will use a specific corollary of their work.

We use the notation *EXPTIME* for $\cup_{k>0} DTIME[2^{n^k}]$ and the notation E for $\cup_{c>0} DTIME[2^{cn}]$. We define EH, the exponential-time hierarchy, as $\cup_{k>0} E^{\Sigma_k^P}$.

The class MA, defined by Babai [2] (cf. [5]), denotes the Merlin-Arthur class, the class of languages accepted by an interactive proof system consisting of a single message from the prover followed by probabilistic verification. Formally, we say that $L \in MA$ if there exists a probabilistic polynomial time machine M and a polynomial $q(n)$ such that:

1. For all $x \in L$ there exists a y, $|y| = q(|x|)$ and $M(x,y)$ accepts with probability at least two-thirds, and

2. For all $x \notin L$ and for all y such that $|y| = q(|x|)$, $M(x,y)$ accepts with probability at most one-third.

Arguably this represents the class of languages with *publishable proofs* of membership (not requiring direct interaction between prover and verifier; the verifier can flip coins at any later date). Babai [2] has shown that $\Sigma_2^P \cap \Pi_2^P$ contains MA.

Let $\Sigma = \{0,1\}$. Let f be a function mapping Σ^* to Σ. We say f is $t(n)$-*approximated* by circuits of size $s(n)$ if for all sufficiently large n there exists a circuit C_n of size $s(n)$ on inputs of length n such that

$$\Pr(C_n(x) \neq f(x)) \leq \frac{1}{t(n)}$$

where x is chosen uniformly over Σ^n.

We say that f is *weakly $t(n)$-approximated* by circuits of size $s(n)$ if the above statement holds for infinitely many n. Nisan and Wigderson [24] use "approximated by circuits of size $s(n)$" to mean "weakly n^j-approximated by circuits of size $s(n)$ for any fixed j" in our terminology.

A function f cannot be (weakly) approximated by polynomial-size circuits if it cannot be (weakly) n^j-approximated by circuits of size n^j for any fixed j. A language cannot be approximated if the characteristic function of that language cannot be approximated. A language class cannot be approximated if some language in that class cannot be approximated.

The class *P/poly* consists of all languages recognizable by a (nonuniform) family of polynomial-size circuits.

We say that a machine M *weakly computes* the language $L \subset \Sigma^*$ if for infinitely many values of n,

$$L \cap \Sigma^n = L(M) \cap \Sigma^n.$$

We say that L admits *weak subexponential simulations* if for every $\epsilon > 0$ there exists a 2^{n^ϵ}-time bounded Turing machine which weakly computes L. We say that a language class C admits weak subexponential simulations if each member of C does. A class D weakly simulates a class C if for every language in C there is a language in D that agrees with C for infinitely many input lengths.

We will use and significantly extend the following theorems due to Nisan and Wigderson [24].

Theorem 2.1.

1. *If EXPTIME cannot be approximated by polynomial-size circuits then BPP admits weak subexponential simulations.*

2. *If EXPTIME cannot be weakly approximated by polynomial-size circuits then* $BPP \subseteq \cap_{\epsilon>0} DTIME[2^{n^\epsilon}]$.

To extend these theorems we will use some of the recent work by Babai, Fortnow and Lund [4] on multiple prover interactive proof systems. Interactive proof systems were introduced by Babai [2] and Goldwasser, Micali and Rackoff [15] as a probabilistic extension of *NP*. The model consists of an infinitely powerful but untrustworthy prover that tries to convince a probabilistic polynomial-time verifier that a string is in a certain language. A recent series of results by Lund, Fortnow, Karloff and Nisan [23] and Shamir [25] show that this model accepts exactly those languages recognizable in polynomial space.

Ben-Or, Goldwasser, Kilian and Wigderson [7] generalize this model to have many provers, that cannot communicate with each other or see the communication of the others with the verifier. Babai, Fortnow and Lund [4] show that this model accepts exactly the languages recognizable in nondeterministic exponential time. We will use the following theorem from [4] regarding the power of *EXPTIME* provers.

Theorem 2.2. *Any language in EXPTIME has a multi-prover interactive proof system where the honest provers are limited to computing within deterministic exponential time.*

We note that this result implies that *EXPTIME* admits instance checking in the sense of Blum and Kannan [9], cf. [4].

3. Random Self-Reducibility in High Complexity Classes

Following an idea of Beaver and Feigenbaum [6], Lipton [22] observed that the permanent function has the following random self-reducibility property: if p is a prime greater than n and an oracle tells the value of $n \times n$ permanents over $\mathbb{Z}_p$ correctly for a $1 - \frac{1}{3n}$ portion of the set of inputs, then one can use this oracle to find the correct value with exponentially small probability of error on every input. Blum, Luby and Rubinfeld [10] used similar ideas.

The idea of the proof is that the value of permanent of A can be computed, using interpolation, from the permanents of any $n+1$ matrices $A + \alpha_i B$ where B is a random matrix and $\alpha_i \in \mathbb{Z}_p$.

A similar idea works for arbitrary polynomials of low degree. In particular, by extending a Boolean function f to a multilinear function g over $\mathbb{Z}_p$, we obtain a random self-reducible f-hard and $PSPACE^f$-easy function g.

We prove the following lemmas from [4] for completeness.

Lemma 3.1. *Let $A : \{0,1\}^s \to \mathbb{Q}$ be a function. Then A has a unique multilinear extension $\widetilde{A} : \mathbb{Q}^s \to \mathbb{Q}$.*

Proof. Define $\widetilde{A}$ by

$$\widetilde{A}(x) =: \sum_{b \in \{0,1\}^s} \prod_{i=1}^{s} A(b)\ell_{\beta_i}(\xi_i), \tag{3.1}$$

where $x = (\xi_1, \ldots, \xi_s)$; b is the bit-string $\beta_1 \ldots \beta_s$; and $\ell_0(\xi) = 1 - \xi$, $\ell_1(\xi) = \xi$. Clearly, $\widetilde{A}$ possesses the required properties.

To prove the uniqueness, assume $f : \mathbb{Q}^s \to \mathbb{Q}$ is multilinear and its restriction to $\{0,1\}^s$ is zero. For $x \in \mathbb{Q}^s$, let $k(x)$ denote the number of coordinates different from 0,1. We prove by induction on $k(x)$ that $f(x) = 0$. Indeed this is true by assumption for $k(x) = 0$. Now for some $k(x) > 0$ suppose e.g. that $\xi_1 \notin \{0,1\}$. Replacing ξ_1 by either 0 or 1 we obtain places where f vanishes by the induction hypothesis; but then, by the linearity in ξ_1, it vanishes at x as well. □

Let us call a language L *PSPACE-robust* if $P^L = \mathit{PSPACE}^L$. (The oracle *PSPACE*-machine is restricted to polynomial length queries.) We say that L has a $t(n)$-random self-reduction if L has a random self-reduction that makes at most $t(n)$ queries.

Lemma 3.2. *Every PSPACE-robust language has a Turing-equivalent family of $(n+1)$-random self-reducible multilinear functions over finite fields.*

Proof. Let L be a *PSPACE*-robust language. Let $g_n(x_1, \ldots, x_n)$ be the unique multilinear extension of the characteristic function of $L_n = L \cap \{0,1\}^n$ (Lemma 3.1). The function g_n is $(n+1)$-random self-reducible via the same proof that shows the permanent is $(n+1)$-random self-reducible.

Clearly $L \in P^g$, where $g = \{g_n : n \geq 0\}$. We will describe an alternating polynomial-time Turing machine with access to L computing g. First guess the value $z = g_n(x_1, \ldots, x_n)$. Then existentially guess the linear function $h_1(y) = g(y, x_2, \ldots, x_n)$ and verify that $h_1(x_1) = z$. Then universally choose $t_1 \in \{0,1\}$ and existentially guess the linear function $h_2(y) = g(t_1, y, x_3, \ldots, x_n)$. Keep repeating this process until we have specified $t_1, \ldots, t_n$ and then verify that $t_1 \ldots t_n \in L$. Since a *PSPACE* machine can simulate an alternating polynomial-time Turing machine, if L is *PSPACE*-robust then g is Turing-reducible to L. □

In particular, we have random self-reducible *PSPACE*-complete functions, *EXPTIME*-complete functions, etc. This observation, inspired by [6] and spelled out simultaneously by the authors of [6] and [4], has significant consequences, as we shall see below.

4. Deterministic Simulation of *BPP*

In this section we complete the proof of Theorem 1.1. The proof consists of proving the following two lemmas.

Lemma 4.1.

1. *BPP admits weak subexponential time simulations unless EXPTIME is in P/poly.*

2. *BPP is in subexponential time unless EXPTIME can be weakly simulated by polynomial-size circuits.*

Lemma 4.2.

1. *If EXPTIME $\subseteq$ P/poly then EXPTIME = MA.*

2. *If EXPTIME is weakly simulated by polynomial-size circuits then MA weakly simulates EXPTIME.*

Lemma 4.2 extends a result of Albert Meyer (see [20]): If *EXPTIME* $\subseteq$ *P/poly* then *EXPTIME* $= \Sigma_2^P$. We use entirely different techniques to prove this stronger lemma. This lemma is also an extension of a corollary in [23].

Proof. [of Lemma 4.2] By Theorem 2.2 [4] we know that to prove a language $L \in$ *EXPTIME* with multiple provers, we only need *EXPTIME*-strong provers. Now the *MA* protocol proceeds as follows. Merlin gives Arthur C_1 and C_2 which are the polynomial size circuits (for the input lengths for which such circuits exist) computing the two provers P_1 and P_2, respectively. Arthur then simulates the verifier V using C_1 and C_2 for P_1 and P_2, respectively. $\square$

Proof. [of Lemma 4.1] Let L be an *EXPTIME*-complete language. We encode the set $L \cap \Sigma^n$ as a Boolean function f. Let p be a prime greater than n, and let g be the (unique) multilinear extension of f to $\mathbb{Z}_p^n \to \mathbb{Z}_p$. We require a lemma involving g.

Lemma 4.3.

1. *If BPP does not have a weak subexponential time simulation then there is a family of poly(n) size (nonuniform) circuits computing g for all but a $\frac{1}{3n}$ fraction of the inputs of length n.*

2. *If BPP is not in subexponential time, then for an infinite number of input lengths n, there is a poly(n) size (nonuniform) circuit computing g for all but a $\frac{1}{3n}$ fraction of the inputs of length n.*

Proof. Assume there is no polynomial-size circuit family computing g for all but a $\frac{1}{3n}$ fraction of the inputs. Goldreich and Levin [14] show how to construct from g a one-bit function h that is computable in exponential time but cannot be approximated by polynomial-size circuits. The lemma follows by Theorem 2.1. □

We continue the proof of Lemma 4.1: Assume now that *BPP* does not have a weak subexponential time simulation. By Lemma 4.3 we have a family D_n of poly(n) size circuits computing g for all but a fraction $\frac{1}{3n}$ of the inputs. Since g is $(n+1)$-random self-reducible, create the following randomized polynomial size circuit family for g: C_n will use random inputs to generate the random-self reduction of g and use the D_n circuit for those queries. The probability that the random self-reduction queries one of the strings that D_n fails to compute correctly is bounded by $(n+1)/3n < 2/5$ for almost every n.

Using techniques of [1, 8] we can replace the randomness with non-uniformity: We can use the usual amplification techniques to reduce the error to less than 2^{-n}. Then there must be a single random sequence that gives a correct answer for all inputs. We encode this string into the circuit.

This proves the first part of Lemma 4.1. The second part has virtually the same proof. Theorem 1.1 follows. □

5. Unary Languages

In this section we will prove Theorem 1.2. We will use the following theorem due to Nisan and Wigderson [24].

Theorem 5.1. *If there exists a function f computable in exponential time such that f cannot be weakly 2^{n^ϵ}-approximated by 2^{n^ϵ}-size circuits for some $\epsilon > 0$ then P = BPP.*

First we prove the following lemma.

Lemma 5.2. *If every unary language in the polynomial-time hierarchy is decidable in polynomial time then P = BPP.*

Proof. By counting arguments, for some $c > 0$, there exists a function on strings of $N = c \log n$ bits that cannot be weakly n-approximated by n-size circuits. We can encode one of these functions as a unary language decidable in Σ_4^P by choosing the lexicographically first function f from Σ^N to Σ that is not approximated by a circuit of size n. If we assume that every unary language in the polynomial-time hierarchy is decidable in polynomial time then we can compute f in polynomial time. The function $f : \Sigma^N \to \Sigma$ is computable in exponential time in N and cannot be weakly $2^{N^{1/c}}$-approximated by any $2^{N^{1/c}}$ size circuit. The lemma now follows from Theorem 5.1. □

Corollary 1.3 follows from this lemma by noticing that a simple padding argument (see [17]) shows that $EH = E$ if and only if all unary languages in the polynomial-time hierarchy are decidable in polynomial time.

Corollary 1.3 has the interesting property of showing that a collapse of two higher complexity classes imply a collapse at a lower level. Usually one sees the other direction, for example $P = NP$ implies $E = NE$ (see [17]).

Proof. [of Theorem 1.2] Suppose *BPP* does not have a weak subexponential time simulation. By Theorem 1.1 we have that $EXPTIME = MA$ and thus $PH = MA$ since $MA \subseteq \Sigma_2^P$. Also we have that $BPP \neq P$ so there exist unary languages in $PH - P$ and thus $MA - P$. □

6. Conclusions

Our proof utilizes a powerful new technique in complexity theory, namely, the use of multilinear functions in the simulation of certain complexity classes. The significance of this technique makes it worth studying its applications and limits (see [23, 25, 3, 4]).

If *BPP* has a weak subexponential simulation then *EXPTIME* properly contains *BPP* since one can easily create languages in *EXPTIME* that do not have a weak subexponential, or even weak $DTIME[2^{cn}]$, simulation. Ideally, we would like to prove that *EXPTIME* properly contains *BPP* without any assumptions. While an oracle making these two classes equal is known to exist (see [18]), the new methods indicated *do not relativize* and therefore a relativized collapse no longer seems as intimidating as it used to be.

Acknowledgments

The first author was partially supported by NSF Grant CCR-8710078. The second author was partially supported by NSF Grant CCR-9009936. The third author performed some of this work at MIT and was partially supported by NSF 865727-CCR and ARO DLL03-86-K-017. The third and fourth authors were partly supported by a grant from the Wolfson foundation, administered by the Israel Academy of Sciences and Humanities.

References

[1] L. Adleman, Two theorems on random polynomial time, in *Proceedings of the 19th IEEE Symposium on Foundations of Computer Science*, IEEE, New York, 1978, 75–83.

[2] L. Babai, Trading group theory for randomness, in *Proceedings of the 17th ACM Symposium on the Theory of Computing*, ACM, New York, 1985, 421–429.

[3] L. Babai and L. Fortnow, Arithmetization: A new method in structural complexity theory, *Computational Complexity*, **1**:1 (1991), 41–66.

[4] L. Babai, L. Fortnow, and C. Lund, Non-deterministic exponential time has two-prover interactive protocols, *Computational Complexity*, **1**:1 (1991), 3–40.

[5] L. Babai and S. Moran, Arthur-Merlin games: a randomized proof system, and a hierarchy of complexity classes, *Journal of Computer and System Sciences*, **36**:2 (1988), 254–276.

[6] D. Beaver and J. Feigenbaum, Hiding instances in multioracle queries, in *Proceedings of the 7th Symposium on Theoretical Aspects of Computer Science*, volume 415 of *Lecture Notes in Computer Science*, Springer, Berlin, 1990, 37–48.

[7] M. Ben-Or, S. Goldwasser, J. Kilian, and A. Wigderson, Multi-prover interactive proofs: How to remove intractability assumptions, in *Proceedings of the 20th ACM Symposium on the Theory of Computing*, ACM, New York, 1988, 113–131.

[8] C. Bennet and J. Gill, Relative to a random oracle, $P^A \neq NP^A \neq co-NP^A$ with probability one, *SIAM Journal on Computing*, **10** (1981), 96–113.

[9] M. Blum and S. Kannan, Designing programs that check their work, in *Proceedings of the 21st ACM Symposium on the Theory of Computing*, ACM, New York, 1989, 86–97.

[10] M. Blum, M. Luby, and R. Rubinfeld, Self-testing and self-correcting programs, with applications to numerical programs, in *Proceedings of the 22nd ACM Symposium on the Theory of Computing*, ACM, New York, 1990, 73–83.

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

[12] R. Boppana and R. Hirschfeld, Pseudorandom generators and complexity classes, in *Randomness and Computation*, volume 5 of *Advances in Computing Research*, S. Micali, ed., JAI Press, Greenwich, 1989, 1–26.

[13] O. Goldreich, H. Krawczyk, and M. Luby, On the existence of pseudo-random generators, in *Proceedings of the 29th IEEE Symposium on Foundations of Computer Science*, IEEE, New York, 1988, 12–24.

[14] O. Goldreich and L. Levin, A hard-core predicate for all one-way functions, in *Proceedings of the 21st ACM Symposium on the Theory of Computing*, ACM, New York, 1989, 25–32.

[15] S. Goldwasser, S. Micali, and C. Rackoff, The knowledge complexity of interactive proof-systems, *SIAM Journal on Computing*, **18**:1 (1989), 186–208.

[16] J. Håstad, Pseudo-random generators under uniform assumptions, in *Proceedings of the 22nd ACM Symposium on the Theory of Computing*, ACM, New York, 1990, 395–404.

[17] J. Hartmanis, N. Immerman, and V. Sewelson, Sparse sets in $NP-P$: *EXPTIME* versus *NEXPTIME*, *Information and Control*, **65** (1985), 158–181.

[18] H. Heller, On relativized exponential and probabilistic complexity classes, *Information and Computation*, **71** (1986), 231–243.

[19] R. Impagliazzo, L. Levin, and M. Luby, Pseudo-random number generation from one-way functions, in *Proceedings of the 21st ACM Symposium on the Theory of Computing*, ACM, New York, 1989, 12–24.

[20] R. Karp and R. Lipton, Some connections between nonuniform and uniform complexity classes, in *Proceedings of the 12th ACM Symposium on the Theory of Computing*, ACM, New York, 1980, 302–309.

[21] L. Levin, One-way functions and pseudo-random generators, *Combinatorica*, **7** (1987), 357–363.

[22] R. Lipton, New directions in testing, in *Distributed Computing and Cryptography*, volume 2 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, J. Feigenbaum and M. Merritt, eds., American Mathematical Society, Providence, 1991, 191 – 202.

[23] C. Lund, L. Fortnow, H. Karloff, and N. Nisan, Algebraic methods for interactive proof systems, *Journal of the ACM*, **39**:4 (1992), 859–868.

[24] N. Nisan and A. Wigderson, Hardness vs. randomness, in *Proceedings of the 29th IEEE Symposium on Foundations of Computer Science*, IEEE, New York, 1988, 2–11.

[25] A. Shamir, IP = PSPACE, *Journal of the ACM*, **39**:4 (1992), 869–877.

[26] A. Yao, Theory and applications of trapdoor functions, in *Proceedings of the 23rd IEEE Symposium on Foundations of Computer Science*, IEEE, New York, 1982, 80–91.