Pseudorandomness and Average-case Complexity via Uniform Reductions

Luca Trevisan [*] Salil Vadhan[†]

Abstract

Impagliazzo and Wigderson [17] recently gave the first construction of pseudorandom generators from a uniform *complexity assumption on* $\mathbf{EXP}$ *(namely* $\mathbf{EXP} \neq \mathbf{BPP}$*). Unlike results in the nonuniform setting, the result of [17] does not provide a continuous trade-off between worst-case hardness and pseudorandomness, nor does it explicitly establish an average-case hardness result.*

In this paper:

- *Our* main result *is a new, and* optimal, *worst-case to average-case complexity reduction for* $\mathbf{EXP}$*: if* $\mathbf{EXP} \not\subseteq \mathbf{BPTIME}(t(n))$*, then we show that* $\mathbf{EXP}$ *has problems that are very hard to solve on fraction* $1/2 + 1/t'(n)$ *of the inputs by* $\mathbf{BPTIME}(t'(n))$ *algorithms, for* $t' = \mathrm{poly}(t)$.

- *We also observe a generalization of the proof of [17] to arbitrary time bounds, yielding a pseudorandom generator generator that stretches* n *bits to* $\approx t(n)$ *bits under the assumption* $\mathbf{EXP} \not\subseteq \mathbf{BPTIME}(t(t(n))$ *(rather than the "optimal" assumption* $\mathbf{EXP} \not\subseteq \mathbf{BPTIME}(t(n))$*). The same generator can be constructed under the assumption that* $\#\mathbf{P} \not\subseteq \mathbf{BPTIME}(t(n))$*. Properties of the generator construction can be used to show that, under either assumption, there is a problem in* $\mathbf{EXP}$ *that is hard to solve on a fraction* $1/2 + 1/t'(n)$ *of the inputs by* $\mathbf{BPTIME}(t'(n))$ *algorithms for* $t' \approx t$*. (This is weaker than our main hardness amplification result, but it can be proved using only the techniques of [17].)*

- *We prove, directly, the existence of a* $\mathbf{PSPACE}$*-complete downward self-reducible and random self-reducible problem, thus slightly simplifying and strengthening the generalization of [17] to arbitrary time bounds (which used a* $\#\mathbf{P}$*-complete problem, namely the* PERMANENT*).*

- *We argue that the results in [17] and in this paper cannot be proved via "black-box" uniform reductions.*

1 Introduction

Pseudorandomness, Average-case Hardness, Worst-case Hardness

Over the past two decades, a rich body of work has investigated the relationship between three basic questions in complexity theory:

1. The existence of problems of high worst-case complexity in classes such a $\mathbf{EXP}$,
2. The existence of problems of high average-case complexity in such classes, and
3. The existence of good pseudorandom generators implying sub-exponential time or even polynomial time deterministic simulations of $\mathbf{BPP}$.

One of the exciting accomplishments of this body of work has been to show *equivalence* of the above three statements in the *nonuniform* setting. That is, $\mathbf{EXP}$ or $\mathbf{E} = \mathbf{DTIME}(2^{O(n)})$ contains problems of high worst-case *circuit complexity* iff it contains problems of high average-case circuit complexity iff there are strong pseudorandom generators against non-uniform distinguishers [22, 5]. This equivalence has become increasingly tight quantitatively, with weak (i.e. slightly superpolynomial) circuit lower bounds implying slight derandomization ($\mathbf{BPP} \subset \mathbf{SUBEXP}$), strong ($2^{\Omega(n)}$) circuit lower bounds implying complete derandomization ($\mathbf{BPP} = \mathbf{P}$), and a smooth tradeoff between these two extremes [13, 1, 16, 27, 15, 23].

Since proving unconditional circuit lower bounds seems out of reach, an important question is to what extent the *nonuniformity* is really necessary for such results? The results are proven by reductions showing how breaking the generators implies good average-case "algorithms" for $\mathbf{E}$,

[*] U.C. Berkeley, Computer Science Division. `luca@cs.berkeley.edu`. Work partly done while at Columbia University. Research supported by a Career Award and a Sloan Research Fellowship.

[†] Harvard University, Cambridge, MA. E-mail: `salil@eecs.harvard.edu`. Work begun at MIT and the Institute for Advanced Study, while supported by an NSF Mathematical Sciences Postdoctoral Research Fellowship.

and how this in turn implies good worst-case "algorithms" for $\mathbf{E}$. Almost all of these reductions are nonuniform and, as we discuss below, this is necessary for reductions that are "black box," that is, that work without making any assumptions on the hard problem being used and on the distinguisher being postulated.

Uniform Reductions

An exciting recent work of Impagliazzo and Wigderson [17] has broken out of this mould of nonuniformity, and their paper is the starting point of our investigation. They prove that under the *uniform assumption* $\mathbf{EXP} \neq \mathbf{BPP}$, it is possible to simulate $\mathbf{BPP}$ algorithms deterministically in subexponential time (on most inputs, for infinitely many input lengths). This result stands out as an isolated example of using of uniform hardness assumptions in derandomization,[1] and suggest that perhaps nonuniformity can be completely removed from this area. However, there is a contrasting result due to Impagliazzo, Kabanets and Wigderson [14], stating that $\mathbf{NEXP} \neq \mathbf{MA}$[2] if and only if $\mathbf{NEXP} \not\subseteq \mathbf{P}/\mathrm{poly}$. In words, the only way to prove a subexponential time derandomization of $\mathbf{MA}$ (or a subexponential time derandomization of promise-$\mathbf{BPP}$) is to prove (or assume) nonuniform lower bounds for $\mathbf{NEXP}$. Contrary to [17], this suggests that circuit lower bounds are *necessary* for derandomization. Where does the truth lie?

Hardness versus randomness results in the uniform setting thus stand at the border of what is provable without using circuit complexity lower bounds, and there is hope that ultimately they may lead to unconditional results (both in the form of derandomization and in the form of circuit lower bounds), or at least of interesting nonrelativizing results about classes that may not even have to do with randomness directly.[3] As we argue below, uniform reductions proving pseudorandomness and/or average-case complexity cannot be "black-box" reductions, and we believe that the pseudorandom generator construction of [17] and the results of this paper are not only impossible via black-box reductions, but also nonrelativizing.

The uniform result by Impagliazzo and Wigderson [17] uses many previous theorems in complexity theory, some of which do not appear related to derandomization. In addition, unlike previous (and subsequent) derandomization results in the nonuniform setting, it was not stated as giving a continuous tradeoff between hardness and randomness. It also was not proved by (explicitly) presenting a uniform reduction from worst-case to average-case hardness, which is typically the first step in previous derandomization results. Thus, their work leaves several intriguing open questions:

- What is the best tradeoff between hardness and derandomization in the uniform setting? In particular, can a sufficiently strong uniform lower bound on $\mathbf{E}$ yield a *polynomial-time* deterministic simulation of $\mathbf{BPP}$? By analogy with the nonuniform setting, we might hope to prove that if $\mathbf{E} \not\subset \mathbf{BPTIME}(t(n))$, then there is a pseudorandom generator that mapping $\approx n$ bits into roughly $\approx t(n)$ bits fooling uniform distinguishers running in time $\approx t(n)$ (which implies a time $2^{O(t^{-1}(n))}$ simulation of $\mathbf{BPP}$).
- Of the several previous results that are being used, how many are really necessary, and what are the properties of $\mathbf{EXP}$ that are essential to the proof?
- Is it possible to prove, along similar lines, that if $\mathbf{EXP} \neq \mathbf{BPP}$ then $\mathbf{EXP}$ contains problems that are hard on average? What is the best tradeoff for this worst-case vs. avg-case problem in the uniform setting?

Our Results

Generalizing [17]. We first revisit, in Section 3, the arguments of Impagliazzo and Wigderson [17], and make the following observations.

1. Generalizing [17], we can obtain a pseudorandom generator that stretches $\approx n$ bits into $\approx t(n)$ bits which fools distinguishers running in time $t(n)$, under the assumption that $\mathbf{EXP} \not\subseteq \mathbf{BPTIME}(t(t(n)))$ (roughly). (Recall that "ideal" result would have $t(n)$ instead of $t(t(n))$.)
2. The "ideal" derandomization can be obtained from uniform lower bounds on $\#\mathbf{P}$ (instead of $\mathbf{EXP}$). That is, the same generator as above can be obtained under the hypothesis that $\#\mathbf{P} \not\subset \mathbf{BPTIME}(t(n))$. The key property of $\#\mathbf{P}$ that is used in [17] is that it contains a complete problem which is both "random self-reducible" and "downward self-reducible", namely the PERMANENT [30].
3. Result 1 above is obtained by constructing two pseudorandom generators, one from an $\mathbf{EXP}$-complete problem and one from the PERMANENT. If there is a time $t(n)$ distinguisher for both generators, then they can be combined in a sophisticated way (not just by a uniform "reduction") to obtain a time $t(t(n))$ algorithm

[1]Here we do not include the works on "cryptographic" pseudorandom generators which are based on the stronger assumption that one-way functions exist [8, 31, 12], but are indeed uniform.

[2]$\mathbf{MA}$ is a probabilistic version of $\mathbf{NP}$; it is the class of languages for which membership has short proofs that can be checked in *probabilistic* polynomial time.

[3]Indeed, [14] use derandomization techniques to obtain the result $\mathbf{NEXP} \subseteq \mathbf{P}/\mathrm{poly} \Rightarrow \mathbf{NEXP} = \mathbf{EXP}$, which *a priori* has nothing to do with randomness; the stronger implication $\mathbf{NEXP} \subseteq \mathbf{P}/\mathrm{poly} \Rightarrow \mathbf{NEXP} = \mathbf{MA}$ is also true, and known to be non-relativizing.

for $\mathbf{EXP}$. This step makes crucial use of Toda's Theorem that $\mathbf{\Sigma_2} \subseteq \mathbf{P}^{\#\mathbf{P}}$ [28].

4. Using our restatement of these results, we can also obtain a uniform worst-case to average-case connection for $\mathbf{EXP}$: if every problem in $\mathbf{EXP}$ admits a probabilistic algorithm that runs in $t(n)$ time and solves the problem in a $1/2 + 1/t(n)$ fraction of inputs of length n, then (roughly) $\mathbf{EXP} \subseteq \mathbf{BPTIME}(t(t(n))$. This is another result that cannot be proved via black-box reduction.

These observations set the stage for our other results. One main goal is to remove the $t(t(n))$ lower bounds that are needed above, for they are far from matching the "ideal" tradeoff. They give nothing, for example, with $t(n) = 2^{n^\epsilon}$, and more generally limits t to being at most *half-exponential* [21]. In terms of derandomization, this means that we cannot get anything near a polynomial-time deterministic simulation of $\mathbf{BPP}$ from such results.

Derandomization from $\mathbf{PSPACE}$-hard problems. In Section 4, we give a direct construction of a $\mathbf{PSPACE}$-complete problem which is both random self-reducible and downward self-reducible. This simplifies the proof of the Impagliazzo–Wigderson result, eliminating the use of Valiant's Theorem and Toda's Theorem, and also strengthens Item 2 by replacing $\mathbf{P}^{\#\mathbf{P}}$ with $\mathbf{PSPACE}$. Our construction is based on the ideas using in proving $\mathbf{IP} = \mathbf{PSPACE}$ [20, 24].

Optimal Average-Case Hardness for $\mathbf{EXP}$. In Section 5 we present our main result: a new, uniform, worst-case to average-case reduction for $\mathbf{EXP}$ that whose parameters match the state-of-the-art in the nonuniform setting [27]. Specifically, we prove that if every problem in $\mathbf{E}$ can be solved in time $t(n)$ on a $1/2 + 1/t(n)$ fraction of inputs, then every problem in $\mathbf{E}$ can be solved in time roughly $t(n)$. Our result is based on combining the nonuniform version of the result from [27] with results about *instance checkers* for $\mathbf{EXP}$.

Black-box Reductions. In Appendix A, we argue that the uniform pseudorandom generator constructions and the uniform worst-case to average-case connections in [17] and here cannot be obtained by black-box reductions. The basic reason for this is that (nonuniform or uniform) black-box reductions can be interpreted *information theoretically*, and give rise to *randomness extractors* in the case of pseudorandom generators [29] and *error-correcting codes* in the case of worst-case-to-average-case reductions. We show that uniform black-box reductions yield such objects with absurd parameters.

This means that to achieve these uniform results, one must exploit special properties of either the hard function or the "adversary" in the reductions. For example, Impagliazzo and Wigderson [17] used the fact that the PERMANENT is downward and random self-reducible. Since only problems in $\mathbf{PSPACE}$ have downward self-reducible problems, this suggests that to obtain better results from the hardness of $\mathbf{EXP}$, we should try to identify special properties of $\mathbf{EXP}$-complete problems which can be exploited. Our optimal hardness amplification identifies one such property, namely *instance checkability*. We do not know whether this suffices for getting optimal derandomization. If not, hopefully it at least will point the direction to other properties of $\mathbf{EXP}$ that can be used.

2 Preliminaries

For notational convenience, all of the time bounds $t(\cdot)$ in this paper are assumed to satisfy several niceness conditions: $n \leq t(n) \leq 2^n$, $t(n)$ is monotone increasing, $t(n)$ is computable in time $\mathrm{poly}(n)$, and $t(O(n)) \leq \mathrm{poly}(t(n)) \leq t(\mathrm{poly}(n))$. All functions of the form n^c, $n^c \log n$, $n^{(\log n)^c}$, 2^{cn}, 2^{n^c} satisfy these conditions.

Now we define several of the classes of languages we will be examining throughout the paper. Sometimes we will abuse notation and refer to them as classes of functions, and we will sometimes identify a language with its characteristic function. $\mathbf{BPTIME}(t(n))$ denotes the class of languages solvable by probabilistic algorithms with 2-sided error running in time $t(n)$. $\mathbf{SIZE}t(n)$ denotes the class of languages which can be decided by (nonuniform) Boolean circuits of size $t(n)$. $\mathbf{\Sigma_k}t(n)$ denotes the class of problems that can be solved by time $t(n)$ alternating machines with k alternations (starting with an existential one).

We will also consider two notions of average-case complexity of functions. The first captures the idea that a function is "easy on average" for deterministic algorithms: $f \in \mathbf{avgDTIME}(t(n))$ if there is a time $t(n)$ deterministic algorithm A such that for every probabilistic sampling algorithm G which runs in time $t(n)$, $\Pr[A(G(1^n)) = f(G(1^n))] \geq 1 - 1/t(n)$. Thus, the statement $\mathbf{BPP} \subseteq \mathbf{avgDTIME}(t(n))$ for some small (e.g. quasipolynomial or even subexponential) time bound t means that we have a good deterministic simulation of $\mathbf{BPP}$ — it is even hard to generate instances on which the simulation will fail.

The other notion of average-case complexity weĺl look at is meant to capture strong average-case *hardness* rather than easiness: We say that $L \in \mathbf{advBPTIME}(t)$ if there is a probabilistic algorithm A running in $t(n)$ time such that for every n

$$\Pr[A(x) = L(x)] \geq 1/2 + 1/t(n)$$

where the probability is taken both over the coin tosses of A and over the uniform distribution of inputs x in $\{0,1\}^n$. Thus, if $L \notin \mathbf{advBPTIME}(t)$ it means that L is very hard-on-average. Indeed, it means that no probabilistic time t algorithm can decide membership in L with a significant *advantage* over random guessing (for infinitely many input lengths); hence the notation $\mathbf{advBPTIME}$. Constructing such hard-on-average languages (or, equivalently, Boolean functions) is typically the first step in a pseudorandom generator construction.

A prefix of **i.o.** to a complexity class means the class of languages which can be solved on *infinitely m2any input lengths* by algorithms within the stated resource bounds.

For $f : \{0,1\}^* \to \{0,1\}$, we denote the restriction of f to inputs of length n by f_n.

Definition 2.1 *$f : \{0,1\}^* \to \{0,1\}$ is* downward self-reducible *if there is a (deterministic) polynomial time algorithm A such that for all $x \in \{0,1\}^n$, $A^{f_{n-1}}(x) = f_n(x)$.*

Definition 2.2 *$f : \{0,1\}^* \to \{0,1\}$ is* random self-reducible *if there is a constant c and a probabilistic polynomial-time algorithm A such that if $g : \{0,1\}^n \to \{0,1\}$ is any function that agrees with f_n on at least a $1 - 1/n^c$ fraction of inputs, $\Pr[A^g(x) = f(x)] \geq 2/3$ for all $x \in \{0,1\}^n$. (The probability is just over the coin tosses of A.)*

Pseudorandom Generators. We define pseudorandom generators in a slightly nonstandard way to facilitate the presentation of our results. We say a function $G : \{0,1\}^* \to \{0,1\}^*$ has *stretch* $m(\cdot)$ if $|G(x)| = m(|x|)$ for all x. We say that D *distinguishes* G in *time* $t(\cdot)$ (resp., *size* $t(\cdot)$) if D runs in time $t(n)$ (resp., is computable by a circuit of size $t(n)$) on inputs of length $m(n)$ and

$$\Pr_{x \leftarrow U_n}[D(x, G(x)) = 1] - \Pr_{(x,y) \leftarrow U_{n+m(n)}}[D(x,y) = 1] > \frac{1}{t(n)},$$

for all sufficiently large n. Clearly, a "good" pseudorandom generator will have no efficient distinguisher. Note that, by this definition, if G cannot be distinguished in time t, it means that every time t algorithm fails to distinguish G *infinitely often*. Note also that we give the distinguisher the seed to the pseudorandom generator. This makes sense here because this line of work (starting from [22]) pseudorandom generators for derandomization (as opposed to, e.g., cryptography) works with generators whose running time is greater than that of the distinguisher. Indeed, all pseudorandom generators in this paper are computable in time $2^{O(n)}$ on seeds of length n.

The above convention about feeding the distinguisher the seed means that every pseudorandom generator gives rise to a hard-on-average function.

Lemma 2.3 *If there is an algorithm A which runs in time $t(\cdot)$ (resp., computed by a circuit of size $t(\cdot)$) such that*

$$\Pr_{x \leftarrow \{0,1\}^n}\left[A(x) = G(x)|_{m(n)}\right] > \frac{1}{2^{m(n)}} + \frac{1}{t(n)}$$

for some $m(\cdot)$ and all n, then there is an algorithm which distinguishes G in time $t(\cdot)$ (resp., in size $t(\cdot)$). Here $G(x)|_{m(n)}$ denotes the first $m(n)$ bits of $G(x)$.

In particular, if there is a generator G which cannot be distinguished in time $t(n)$, then the first bit of G defines a language which is not in $\mathbf{advBPTIME}(t(n))$.

3 Extensions of [17]

In this section, we describe the results of Impagliazzo and Wigderson [17], and, in the process, generalize them to larger time bounds.

The starting point for pseudorandom generation from Boolean functions of high circuit complexity was the construction of Nisan and Wigderson [22], whbich builds a pseudorandom generator from an average-case hard function.

Lemma 3.1 ([22]) *For every $t(\cdot)$ and every random self-reducible function f, there is a G with stretch $t(n)$ such that*

- *$G(x)$ can be computed in time $\mathrm{poly}(t(n))$ on inputs of length n, with oracle access to f.*
- *If G can be distinguished in size $t(n)$, then f is in $\mathbf{SIZE}(t(\mathrm{poly}(n)))$.*

Quantitatively better results that eliminate the $\mathrm{poly}(n)$ in $\mathbf{SIZE}(t(\mathrm{poly}(n)))$ are now known [16, 27, 15, 23], but we use the above for simplicity. The random self-reducible hard function f can be obtained from any hard function f by taking the low-degree extension:

Lemma 3.2 ([6, 5]) *For every function f, there is a random self-reducible function such that f reduces to f' in linear time, and f' can be computed in linear space with oracle access to f.*

The first new ingredient in [17] was the observation that the circuit complexity conclusion of Lemma 3.1 can be replaced with a uniform conclusion about *learnability*.

Lemma 3.3 ([17]) *For every $t(\cdot)$ and every random self-reducible f, there is a G with stretch $t(n)$ such that*

- *$G(x)$ can be computed in time $\mathrm{poly}(t(n))$ on inputs of length n, with oracle access to f.*

- *If G can be distinguished in time $t(n)$, then f is in $\mathbf{LEARN_{mem}}(t(\mathrm{poly}(n)))$, where $\mathbf{LEARN_{mem}}t(n)$ denotes the class of languages L which can be (exactly)* learned *with membership queries by a probabilistic algorithm A running in time $t(n)$.*[4]

The next new ingredient of [17] was showing that the learnability can be turned into standard uniform easiness if the function f is *downward self-reducible*.

Lemma 3.4 ([17]) *If f is downward self-reducible and $f \in \mathbf{LEARN_{mem}}(t(n))$, then $f \in \mathbf{BPTIME}(\mathrm{poly}(t(n)))$.*

The problem with this is that all downward self-reducible problems lie in $\mathbf{PSPACE}$, but we would like to start with a hard function in $\mathbf{EXP}$. The way this is overcome in [17] is to assume that $\mathbf{EXP}$ has polynomial-sized circuits (for otherwise we're already done by Lemma 3.1). Under this assumption, a version of the Karp–Lipton Theorem, attributed to Albert Meyer, collapses $\mathbf{EXP}$ to $\mathbf{\Sigma_2}$. Generalizing this to higher time bounds gives:

Lemma 3.5 (Meyer [18]) *If $\mathbf{EXP} \subset \mathbf{SIZE}(t(n))$, then $\mathbf{EXP} \subset \mathbf{\Sigma_2}(\mathrm{poly}(t(n)))$.*

Once $\mathbf{EXP}$ collapses to $\mathbf{\Sigma_2}$, we get a random self-reducible and downward self-reducible function from the following:

Lemma 3.6 ([30, 28, 19]) *There is a random self-reducible and downward self-reducible $\mathbf{\Sigma_2}$-hard problem, namely, the* Permanent.

Combining all these, we get the following generalization of the [17] Theorem.

Theorem 3.7 (generalizing [17]) *There is a generator G with stretch $t(\cdot)$ which cannot be distinguished in time $t(\cdot)$ unless $\mathbf{EXP} \subset \mathbf{BPTIME}(t(t(\mathrm{poly}(n))))$.*

Proof sketch: Let f_1 be a random self-reducible $\mathbf{EXP}$-complete problem (from Lemma 3.2) and let f_2 be the Permanent. Use Lemma 3.1 to construct a generator G_1 with stretch $t(\cdot)$ from f_1, and use Lemma 3.3 to construct a generator G_2 with stretch $t(\cdot)$ from f_2. If both G_1 and G_2 can be distinguished in time $t(\cdot)$, then $f_1 \in \mathbf{SIZE}(t(\mathrm{poly}(n)))$, $f_2 \in \mathbf{LEARN_{mem}}(t(\mathrm{poly}(n)))$. Since f_2 is downward self-reducible, Lemma 3.4 gives $f_2 \in \mathbf{BPTIME}(t(\mathrm{poly}(n)))$. Since f_1 is $\mathbf{EXP}$-complete, Lemma 3.5 gives $f_1 \in \mathbf{\Sigma_2}(t(\mathrm{poly}(n)))$. By Lemma 3.6, f_1 reduces to f_2 in time $t(\mathrm{poly}(n))$, from which we conclude $f_1 \in \mathbf{BPTIME}(t(t(\mathrm{poly}(n))))$. □

[4]That is, with high probability, $A^{L_n}(1^n)$ outputs a circuit which computes L_n within time $t(n)$, where L_n denotes the restriction of language L to inputs of length n.

A generator which fools uniform algorithms can be used to obtain an average-case derandomization of $\mathbf{BPP}$.

Corollary 3.8 (generalizing [17]) *If $\mathbf{EXP} \not\subset \mathbf{BPTIME}(t(t(\mathrm{poly}(n))))$, then $\mathbf{BPP} \subseteq \mathbf{i.o.\text{-}avgDTIME}(2^{t^{-1}(n)})$.*

Note that this only gives a deterministic simulation of $\mathbf{BPP}$ *infinitely often*. In most previous works on derandomization, it is also possible to obtain a simulation for all input lengths by assuming that $\mathbf{EXP}$ has a problem that is hard for all but finitely many input lengths, i.e. $\mathbf{EXP}$ is not in $\mathbf{i.o.\text{-}BPTIME}(t(n))$ for a small t'. However, one of the steps of the above proof, namely Lemma 3.4, breaks down if we try to work with an infinitely-often hypothesis.

We also observe that a uniform hardness amplification result now follows from Theorem 3.7 via Lemma 2.3.

Corollary 3.9 *If $\mathbf{EXP} \not\subset \mathbf{BPTIME}(t(t(\mathrm{poly}(n))))$, then $\mathbf{EXP} \not\subset \mathbf{advBPTIME}(t(\mathrm{poly}(n)))$.*

In Section 5, we improve this result in two ways. First, we eliminate the composition of t (along with other quantitative improvements) to obtain a result that matches best known nonuniform result. Second, we obtain an version which says that $\mathbf{EXP}$ has a problem that is worst-case hard for almost all input lengths, then it has a problem that is average-case hard for almost all input lengths (in contrast to the above, which is only implies hardness "infinitely often").

4 A Downward and Random Self-Reducible $\mathbf{PSPACE}$-complete Problem

The proof of Impagliazzo and Wigderson in Section 3 makes use of many previous results, and it is unclear how much of that machinery is really necessary for the result. By isolating the essential ingredients, we may ultimately succeed in removing the deficiencies described in the introduction. In this section, we show that Valiant's Theorem and Toda's Theorem, which were used in Lemma 3.6, are not necessary. Instead, we show that there is a random self-reducible and downward self-reducible complete problem for $\mathbf{PSPACE}$. At first, this seems easy. The canonical $\mathbf{PSPACE}$-complete problem QBF is downward self-reducible, and Lemma 3.2 says that $\mathbf{PSPACE}$ also has a random self-reducible complete problem. However, the Impagliazzo–Wigderson proof appears to need a *single* complete problem which has both properties simultaneously. In this section, we obtain such a problem by a careful arithmetization of QBF, using the ideas underlying the interactive proof system for $\mathbf{PSPACE}$ [20, 24].

In what follows, $\mathbb{F}_n$ is the finite field of size 2^n. It is known that a representation of this field (i.e. an irreducible polynomial of degree n over $\mathrm{GF}(2)$) can be found deterministically in time $\mathrm{poly}(n)$ [25].

Lemma 4.1 *For some polynomials t and m, there is a collection of functions $\{f_{n,i} : (\mathbb{F}_n)^{t(n,i)} \to \mathbb{F}_n\}_{i \le m(n)}$ with the following properties:*

1. *(Self-Reducibility) For $i < m(n)$, $f_{n,i}$ can be evaluated with oracle access to $f_{n,i+1}$ in time $\mathrm{poly}(n)$. $f_{n,m(n)}$ can be evaluated in time $\mathrm{poly}(n)$.*
2. *($\mathbf{PSPACE}$-hardness) For every language L in $\mathbf{PSPACE}$, there is a polynomial-time computable function g such that for all x, $g(x) = (1^n, y)$ with $y \in \mathbb{F}_n^{t(n,0)}$, and $f_{n,0}(y) = \chi_L(x)$.*
3. *(Low Degree) $f_{n,i}$ is a polynomial of total degree at most $\mathrm{poly}(n)$.*

Proof sketch: Consider the interactive proof system for $\mathbf{PSPACE}$-complete problem QBF, as presented in [26]. In the construction of the proof system, a QBF instance $\phi = \exists x_1 \forall x_2 \cdots \exists/\forall x_n \psi(x_1, \ldots, x_n)$ induces a sequence $f_0, f_1, \ldots, f_m$ ($m = \mathrm{poly}(n)$) of multivariate polynomials over any sufficiently large finite field, say $\mathbb{F}_n$. $f_m = f_m(x_1, \ldots, x_n)$ is an arithmetization of the propositional formula $\psi(x_1, \ldots, x_n)$, and for $j < n$, f_j is defined in terms of f_{j+1} using one of the rules:

$$\begin{aligned} f_j(x_1, \ldots, x_\ell) &= f_{j+1}(x_1, \ldots, x_\ell, 0) \cdot f_{j+1}(x_1, \ldots, x_\ell, 1) \\ f_j(x_1, \ldots, x_\ell) &= 1 - (1 - f_{j+1}(x_1, \ldots, x_\ell, 0)) \cdot (1 - f_{j+1}(x_1, \ldots, x_\ell, 1)) \\ f_j(x_1, \ldots, x_k, \ldots, x_\ell) &= x_k \cdot f_j(x_1, \ldots, 1, \ldots, x_\ell) + (1 - x_k) \cdot f_j(x_1, \ldots, 0, \ldots, x_\ell). \end{aligned}$$

(Which rule is used depends solely on i and n in an easily computable way). The construction provides the following guarantees:

- If f_j depends on ℓ variables, then when $x_1 \ldots, x_\ell$ take on Boolean values, $f_j(x_1, \ldots, x_\ell)$ equals the truth value of ϕ with the first ℓ quantifiers removed. f_0 is a constant polynomial, and thus equals the truth value of ϕ (with all quantifiers present).
- f_m can be evaluated in time $\mathrm{poly}(|\phi|)$.
- For $j < m$, f_j can be evaluated in time $\mathrm{poly}(|\phi|)$ given oracle access to f_{j+1}. (This follows from the three possible rules which define f_j in terms of f_{j+1}.)
- Each f_j is of total degree at most $\mathrm{poly}(|\phi|)$.

However, this does not yet accomplish what we want since these polynomials depend on ϕ, and not just its length. To solve this, we incorporate the formula ϕ into the arithmetization (as done for PCP's in, e.g. [4, 10] for different reasons). We do this by defining a single "universal" quantified formula Φ_n which has some *free* variables such that by setting these free variables appropriately, Φ_n can be specialized to any instance of QBF. Specifically, Φ_n has $2n^2$ free variables $\{y_{i,j}, z_{i,j} : 1 \le i, j \le n\}$, and is defined as follows:

$$\Phi_n(\overline{y}, \overline{z}) = \exists x_1 \forall x_2 \cdots \exists/\forall x_n \bigwedge_{i=1}^{n} \bigvee_{j=1}^{n} (y_{i,j} \wedge x_j) \vee (z_{i,j} \wedge \neg x_j)$$

Now let ϕ be any instance of QBF. Without loss of generality, we may assume ϕ is in the form $\phi = \exists x_1 \forall x_2 \cdots \exists/\forall x_n \psi(x_1, \ldots, x_n)$, where ψ is a CNF formula with at most n clauses. (These restrictions still preserve the fact that QBF is a $\mathbf{PSPACE}$-complete problem.) Define $\overline{y}(\phi)$ and $\overline{z}(\phi)$ as follows: $y_{i,j}(\phi) = 1$ iff the i'th clause of ψ contains x_j, and $z_{i,j}(\phi) = 1$ iff the i'th clause of ψ contains $\neg x_i$. Then, by inspection,

$$\Phi_n(\overline{y}(\phi), \overline{z}(\phi)) \equiv \phi \tag{1}$$

Now we define the polynomials $f_{n,0}, f_{n,1}, \ldots, f_{n,m}$ ($m = m(n)$) to be the sequence of polynomials obtained by applying the $\mathbf{IP} = \mathbf{PSPACE}$ arithmetization to Φ_n. Unlike a standard instance of QBF, Φ_n has some free variables $\overline{y}, \overline{z}$. However, the construction still applies, and each $f_{n,i}$ will have variables corresponding to these free variables in addition to x-variables corresponding to the quantifiers that have been "stripped off". The resulting sequence of polynomials has the following properties:

- If f_j depends on ℓ x-variables, then when $\overline{y}, \overline{z}$, and $x_1, \ldots x_\ell$ take on Boolean values, $f_{n,j}(\overline{y}, \overline{z}, x_1, \ldots, x_\ell)$ equals the truth value of Φ_n with the first ℓ quantifiers removed. $f_{n,0}$ depends on none of the x-variables, and thus $f_{n,0}(\overline{y}, \overline{z}) = \Phi_n(\overline{y}, \overline{z})$ on Boolean inputs.
- $f_{n,j}$ can be computed in time $\mathrm{poly}(n)$ given oracle access to $f_{n,j+1}$.
- Each $f_{n,j}$ is of total degree at most $\mathrm{poly}(n)$.
- $f_{n,m(n)}$ can be evaluated in time $\mathrm{poly}(n)$.

This establishes the self-reducibility and low degree properties. The $\mathbf{PSPACE}$-hardness, This completes the proof. □

Now, to deduce the final result, we simply combine the functions $f_{n,i}$ from Lemma 4.1 into a single function F, with a careful ordering of input length so as to turn the "upwards" reductions from $f_{n,i}$ to $f_{n,i+1}$ into a downward self-reduction for F. Details are in Appendix B.

Theorem 4.2 $\mathbf{PSPACE}$ *has a complete problem which is both random self-reducible and downward self-reducible.*

In addition to removing some steps from the Impagliazzo–Wigderson proof, Theorem 4.2 has the consequence that we can obtain the "right" derandomization of $\mathbf{BPP}$ from a uniform assumption about hard problems in $\mathbf{PSPACE}$ (as opposed to $\mathbf{P}^{\#\mathbf{P}}$, as follows from [17]) .

Corollary 4.3 *There is a generator G with stretch $t(\cdot)$ which cannot be distinguished in time $t(\cdot)$ unless $\mathbf{PSPACE} \subset \mathbf{BPTIME}(t(\mathrm{poly}(n)))$.*

Corollary 4.4 *If $\mathbf{PSPACE} \not\subset \mathbf{BPTIME}(t(\mathrm{poly}(n)))$, then $\mathbf{BPP} \subseteq \mathbf{i.o.\text{-}avgDTIME}(2^{t^{-1}(n)})$.*

5 Uniform Hardness Amplification

In this section we will prove that if every problem in $\mathbf{EXP}$ has a $\mathbf{BPTIME}(t(n))$ algorithm that solves the problem on a fraction $1/2 + 1/t(n)$ of the inputs of length n, then $\mathbf{EXP}$ is contained in $\mathbf{BPTIME}(t(\mathrm{poly}(n)))$. We will also prove a similar result for $\mathbf{PSPACE}$.

We will prove our result in a series of steps. First, we observe that the non-uniform worst-case to average-case reduction in [27] actually uses a "logarithmic amount of nonuniformity." More precisely, the reduction can be implemented by using a probabilistic algorithm that first picks its randomness, then receives a logarithmically long advice string (that depends only on the randomness), and finally receives and solves the input. We formalize this slightly nonstandard notion of nonuniform probabilistic computation as follows.

Definition 5.1 (nonuniform $\mathbf{BPP}$) *For functions t and a, we say that a language L is in $\mathbf{BPTIME}(t)//a$ if there is an algorithm A and a function f such that for every input x of length l, $\Pr_r[A(r, x, f(r)) = L(x)] \geq 3/4$, A runs in $t(n)$ time, and $|f(r)| \leq a(n)$.*

Using the above notation, we can restate the main result of Section 4 of Sudan, Trevisan, and Vadhan [27] in the following way:

Theorem 5.2 (implicit in [27]) *For every language L we can define a language L' such that*

- *L is reducible to L' (via a linear time Karp reduction);*
- *L' on inputs of length n is solvable in $2^{O(n)}$ time given oracle access to L (and all oracle queries are of size $\Theta(n)$);*
- *if L' is in $\mathbf{advBPTIME}(t(n))$, then L' is in $\mathbf{BPTIME}(\mathrm{poly}(t(n)))//O(\log t(n))$.*

Proof sketch: In [27], L' is based on a low-degree polynomial encoding f of L over a field of size $\mathrm{poly}(t(n))$. It is shown that the only nonuniformity required is the value of the polynomial at a single random point, which comes to $O(\log t(n))$ bits. $\square$

We note that earlier methods for achieving strong average-case hardness, namely, versions of Yao's XOR Lemma [31, 11, 13, 16], all appear to use much nonuniformity.

Finally, we show that if L (and so L') is $\mathbf{EXP}$-complete or $\mathbf{PSPACE}$-complete, then L' can be in $\mathbf{BPTIME}(\mathrm{poly}(t(n)))//O(\log(t(n)))$ only if it is also in $\mathbf{BPTIME}(t(\mathrm{poly}(n)))$. This will be a consequence of the fact that $\mathbf{EXP}$-complete and $\mathbf{PSPACE}$-complete problems have instance checkers.

Definition 5.3 (instance checker [7]) *An* instance checker *C for a language L is a polynomial time probabilistic oracle machine whose output is either* `pass` *or* `fail` *and such that*

- *for all inputs x, and all oracles L', if $L'(x) \neq L(x)$ then $\Pr[C^{L'}(x) = \texttt{fail}] \geq 3/4$;*
- *for all inputs x, $\Pr[C^{L}(x) = \texttt{fail}] = 0$.*

Intuitively, if L has an instance checker, then machine C, given an input x and an oracle that purports to decide L, with high probability will be able to verify the validity of the oracle on x.

As observed in [4], the proof of $\mathbf{MIP} = \mathbf{NEXP}$ in [4] implies the existence of instance checker for all $\mathbf{EXP}$-complete problems, and the proof of $\mathbf{IP} = \mathbf{PSPACE}$ in [20, 24] implies the existence of instance checkers for all $\mathbf{PSPACE}$-complete problems.

Theorem 5.4 ([4],[20, 24]) *Every $\mathbf{EXP}$-complete problem and every $\mathbf{PSPACE}$-complete problem has an instance checker.*

A result related to the existence of instance checkers for $\mathbf{EXP}$[5] has been used in complexity theory before, for example in [5] and [9]. Our use below, to eliminate limited non-uniform, seems, however, new.

Lemma 5.5 *Let $L \in \mathbf{BPTIME}(t)//a$ be a problem admitting an instance checker.*
Then $L \in \mathbf{BPTIME}(t(\mathrm{poly}n) \cdot 2^{a(\mathrm{poly}n)})$.

[5]Specifically, the fact that every problem in $\mathbf{EXP}$ has a PCP-type proof systems where the proof has exponential length and the verifier runs in polynomial time, with the additional property that valid proofs for YES-instances can be computed in exponential time.

Proof: Let C be the instance checker, and let n^c be an upper bound on the length of oracle queries made by C for inputs of length n, let $A(\cdot,\cdot,\cdot)$ be the algorithm for L and let f be the advice function. Let C' be the instance checker obtained by running C $O(a(n^c))$ times, and passing if and only if C always passes. Then C' also asks oracle queries of length at most n^c, the running time of C' is $\mathrm{poly}(n)\cdot a(n^c)$, and the error probability is, say, $2^{-a(n^c)-2}$.

On input x, we pick r at random, and consider the $2^{a(n^c)}$ possible advice strings s for the computation of inputs of size at most n^c, and run $C^{A(r,\cdot,s)}(x)$. Let s^* be the lexicographically smaller string for which $C^{A(r,\cdot,s)}(x)$ outputs `pass`; then we output $A(r,x,s^*)$. We output a random answer if there is no string s for which the output is `pass`. With high probability over r, and for the case $s = f(r)$, the oracle is correctly computing L, and C outputs `pass`; furthermore, except with probability $\leq 1/4$, in all cases in which C outputs `pass`, the answer is right. ■

We can now put together all the results, and prove our worst-case to average-case reduction in the uniform setting.

Theorem 5.6 *If* $\mathbf{EXP} \subseteq \mathbf{advBPTIME}(\mathrm{poly}(t(n)))$ *then* $\mathbf{EXP} \subseteq \mathbf{BPTIME}(t(\mathrm{poly}(n)))$.

Proof: Fix an $\mathbf{EXP}$-complete language L, and construct L' as in Theorem 5.2. Then L' is also $\mathbf{EXP}$-complete, and thus instance-checkable, and also we have $L' \in \mathbf{advBPTIME}(\mathrm{poly}(t(n)))$ and then $L' \in \mathbf{BPTIME}(\mathrm{poly}(t(O(n))))//O(\log t(O(n)))$. Using Lemma 5.5, we have $L' \in \mathbf{BPTIME}(t(\mathrm{poly}(n)))$, and, from the $\mathbf{EXP}$-completeness of L', we have $\mathbf{EXP} \subseteq \mathbf{BPTIME}(t(\mathrm{poly}(n)))$. ■

Notice that the above theorem is not as strong as it could be. For example, it would be nice to prove that $\mathbf{E} \not\subseteq \mathbf{BPTIME}(2^{o(n)})$ implies $\mathbf{E} \not\subseteq \mathbf{advBPTIME}(2^{o(n)})$, however Theorem 5.6 does not imply such a result. In order to do such finer worst-case to average-case reductions, we need to re-examine our argument very carefully. We state a stronger result that can be proved this way, and, in this extended abstract, only sketch the proof.

Theorem 5.7 *If* $\mathbf{E} \subseteq \mathbf{advBPTIME}(\mathrm{poly}(t(n)))$ *then* $\mathbf{E} \subseteq \mathbf{BPTIME}(\mathrm{poly}(t(n)))$.

Proof: [Sketch] Fix a language L that is $\mathbf{E}$-complete under linear-time reductions, for example L is the set of triples $(M, x, 1^n)$ such that machine M accepts input x in at most 2^n steps. Construct L' as in Theorem 5.2, then L' is also $\mathbf{E}$-complete under linear-time reductions. From the assumption of the theorem, we have that $L' \in \mathbf{advBPTIME}(t(n))$, and from Theorem 5.2 we have that also

$$L' \in \mathbf{BPTIME}(\mathrm{poly}(t(O(n))))//O(\log t(O(n))) \ . \quad (2)$$

From a stronger version of Theorem 5.4, it follows that L' has an instance checker that given an input of length n only makes oracle queries of size $O(n)$. (Such a result does not seem to follow from [4], however it can be derived from the proof of the PCP theorem [3, 2] and, in fact, the results of [3] are enough.) Using Equation (2), the strong instance checker of L', and a more careful version of the proof of Lemma 5.5, we conclude that L' is also in $\mathbf{BPTIME}(\mathrm{poly}(t(O(n))))$, and now the theorem follows from the niceness of t and the $\mathbf{E}$-completeness of L' under linear-time reductions. ■

Finally, we observe that the proof of Theorem 5.6 actually shows that given a language $L \in \mathbf{EXP}$ we can construct a language $L' \in \mathbf{EXP}$ such that in order to compute L on all inputs of length n, it is enough to compute L' well on average on inputs of length $n^{\Theta(1)}$. Standard padding arguments can then be used to show that if L' is easy on average for infinitely many input lengths, then L is easy on the worst case for infinitely many input lengths. Recall that the techniques of [17] do not provide this kind of result, the Lemma 3.4

Theorem 5.8 *If* $\mathbf{EXP} \subseteq \mathbf{i.o.}\text{-}\mathbf{advBPTIME}(t(\mathrm{poly} n))$, *then* $\mathbf{EXP} \subseteq \mathbf{i.o.}\text{-}\mathbf{BPTIME}(t(\mathrm{poly} n))$. *If* $\mathbf{E} \subseteq \mathbf{i.o.}\text{-}\mathbf{advBPTIME}(\mathrm{poly}(t(n))$ *then* $\mathbf{E} \subseteq \mathbf{i.o.}\text{-}\mathbf{BPTIME}(\mathrm{poly}(t(n))$.

Acknowledgments

We thank Avi Wigderson, Valentine Kabanets, and Lance Fortnow for helpful discussions.

References

[1] A. E. Andreev, A. E. F. Clementi, and J. D. P. Rolim. Worst-case hardness suffices for derandomization: A new method for hardness-randomness trade-offs. In P. Degano, R. Gorrieri, and A. Marchetti-Spaccamela, editors, *Automata, Languages and Programming, 24th International Colloquium*, volume 1256 of *Lecture Notes in Computer Science*, pages 177–187, Bologna, Italy, 7–11 July 1997. Springer-Verlag.

[2] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and hardness of approximation problems. *J. ACM*, 45(3):501–555, 1998. Preliminary version in *Proc. of FOCS'92*.

[3] S. Arora and S. Safra. Probabilistic checking of proofs: A new characterization of NP. *J. ACM*, 45(1):70–122, 1998. Preliminary version in *Proc. of FOCS'92*.

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

[5] 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.

[6] D. Beaver and J. Feigenbaum. Hiding instances in multioracle queries. In *7th Annual Symposium on Theoretical Aspects of Computer Science*, volume 415 of *Lecture Notes in Computer Science*, pages 37–48, Rouen, France, 22–24 Feb. 1990. Springer.

[7] M. Blum and S. Kannan. Designing programs that check their work. *Journal of the ACM*, 42(1):269–291, 1995.

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

[9] H. Buhrman, L. Fortnow, D. van Milkebeek, and L. Torenvliet. Using autoreducibility to separate complexity classes. *SIAM J. on Computing*, 29:1497–1520, 2000.

[10] U. Feige, S. Goldwasser, L. Lovász, S. Safra, and M. Szegedy. Interactive proofs and the hardness of approximating cliques. *J. ACM*, 43(2):268–292, 1996.

[11] O. Goldreich, N. Nisan, and A. Wigderson. On Yao's XOR lemma. Technical Report TR95–050, Electronic Colloquium on Computational Complexity, March 1995. `http://www.eccc.uni-trier.de/eccc`.

[12] J. Håstad, R. Impagliazzo, L. A. Levin, and M. Luby. A pseudorandom generator from any one-way function. *SIAM J. Comput.*, 28(4):1364–1396 (electronic), 1999.

[13] R. Impagliazzo. Hard-core distributions for somewhat hard problems. In *36th Annual Symposium on Foundations of Computer Science*, pages 538–545, Milwaukee, Wisconsin, 23–25 Oct. 1995. IEEE.

[14] R. Impagliazzo, V. Kabanets, and A. Wigderson. In search of an easy witness: Exponential time vs. probabilistic polynomial time. In *Proceedings of the Sixteenth Annual Conference on Computational Complexity*. IEEE, June 18–21 2001.

[15] R. Impagliazzo, R. Shaltiel, and A. Wigderson. Extractors and pseudo-random generators with optimal seed length. In *Proceedings of 32nd ACM Symposium on Theory of Computing*, 2000.

[16] R. Impagliazzo and A. Wigderson. $P = BPP$ if E requires exponential circuits: Derandomizing the XOR lemma. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing*, pages 220–229, El Paso, Texas, 4–6 May 1997.

[17] R. Impagliazzo and A. Wigderson. Randomness vs. time: De-randomization under a uniform assumption. In *36th Annual Symposium on Foundations of Computer Science*, Palo Alto, CA, November 8–11 1998. IEEE.

[18] R. M. Karp and R. J. Lipton. Turing machines that take advice. *L'Enseignement Mathématique. Revue Internationale. IIe Série*, 28(3-4):191–209, 1982.

[19] R. Lipton. New directions in testing. In *Proceedings of DIMACS Workshop on Distributed Computing and Cryptography*, 1989.

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

[21] P. B. Miltersen, N. V. Vinodchandran, and O. Watanabe. Super-polynomial versus half-exponential circuit size in the exponential hierarchy. In *Computing and combinatorics (Tokyo, 1999)*, pages 210–220. Springer, Berlin, 1999.

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

[23] R. Shaltiel and C. Umans. Simple extractors for all min-entropies and a new pseudo-random generator. In *42nd Annual Symposium on Foundations of Computer Science*. IEEE, 14–17 Oct. 2001.

[24] A. Shamir. IP = PSPACE. *J. ACM*, 39(4):869–877, Oct. 1992.

[25] V. Shoup. New algorithms for finding irreducible polynomials over finite fields. *Math. Comp.*, 54(189):435–447, 1990.

[26] M. Sipser. *Introduction to the Theory of Computation*. PWS Publishing, 1997.

[27] M. Sudan, L. Trevisan, and S. Vadhan. Pseudorandom generators without the XOR lemma. *Journal of Computer and System Sciences*, 62:236–266, 2001. Special issue on CCC '99. Extended abstract in *STOC–CCC '99* joint session.

[28] S. Toda. PP is as hard as the polynomial-time hierarchy. *SIAM J. Comput.*, 20(5):865–877, 1991.

[29] L. Trevisan. Constructions of near-optimal extractors using pseudo-random generators. In *Proceedings of the Thirty-First Annual ACM Symposium on the Theory of Computing*, pages 141–148, Atlanta, Georgia, 1–4 May 1999.

[30] L. G. Valiant. The complexity of computing the permanent. *Theoretical Computer Science*, 8(2):189–201, 1979.

[31] A. C. Yao. Theory and applications of trapdoor functions (extended abstract). In *23rd Annual Symposium on Foundations of Computer Science*, pages 80–91, Chicago, Illinois, 3–5 Nov. 1982. IEEE.

A Black-Box Reductions

In this section, we argue that *uniform, black-box* reductions cannot be used to prove the pseudorandom generator constructions and the worst-case-to-average-case reductions given in [17] and this paper. We conjecture that these negative results can be extended to actually show that the constructions are nonrelativizing. The fact that we are using reductions which cannot be black-box suggests that significant, and possibly unexpected results could come out of further studies of uniform reductions in this field.

Let us briefly explain what we mean by black-box reductions, and why uniform black-box reductions have very strong limitations. Suppose we want to construct a pseudorandom generator $G_f : \{0,1\}^n \to \{0,1\}^{t(n)}$ based on a hard function f; our approach (following [22] and all subsequent papers on the subject) could be to show that giving a distinguishing procedure D that distinguishes the output of G_f from the uniform distribution, it is possible to construct an oracle procedure P, which may be nonuniform (and indeed typically is), such that P^D computes f well on average. Now, if f is hard on average, and P is efficient, it cannot be the case that D is efficient. So no efficient procedure distinguishes the output of G_f from uniform, and G is a pseudorandom generator. The oracle procedure P implements the reduction from the task of breaking the generator to the task of computing the hard function f. Even though we are interested only in the case in which D is efficient, and f is in some bounded complexity class (such as $\mathbf{EXP}$), a "black-box" reduction will establish the existence of a predictor P for every function f and for every distinguisher D. As shown in [29], pseudorandom generator constructions having this type of black-box analysis also have very nice information-theoretic properties, and they yield *randomness extractors*. Now, if we follow the [29] argument in the case where the predictor P is a *uniform* oracle machine (or even a machine that uses limited advice), then we get randomness extractors with impossible parameters, and so we have to conclude that P has to be non-uniform. Basically, [29] proves that if (the truth table of) f is chosen randomly, from some arbitrary distribution having min-entropy at least k, where k is (roughly) the number of bits of advice used by P, then the output of the generator is close to uniform. For information-theoretic reasons, we must have, roughly, $n + k \geq t(n)$, or $k \geq t(n) - n$, and so P has to be highly non-uniform.

We can do a similar argument for worst-case to average-case reductions. A black-box reduction would involve a transformation H, such that given a function f that is hard in the worst case, the function H_f is hard on average. The latter means that for every procedure A that computes H_f on a fraction $1/2 + 1/t$ of the inputs, there is an oracle procedure R (implementing the reduction) such that R^A computes f everywhere. Now, we can think of (the truth table of) H_f as an error-correcting encoding of f, and of R as a way of uniquely specifying f (and hence H_f) given an oracle that may have a distance up to $1/2 - 1/t$ from H_f. This would imply that H_f is an error-correcting code with minimum distance $1 - 2/t$ which is impossible (codes with minimum distance larger than $1/2$ can only contain a finite amount of codewords). In fact, results from coding theory can be used to argue that R must use $\Omega(\log t)$ bits of advice, a bound that is met by [27].

B Proof of Theorem 4.2

We will combine the functions $f_{n,i}$ from Lemma 4.1 into a single function F, with a careful ordering of input lengths. Initially, we won't define F on all input lengths.

For $i \leq m(n)$, define $h(n,i)$ inductively as follows:

- $h(1, m(1)) = t(1, m(1))$.
- For $n > 1$, $h(n, m(n)) = \max\{h(n-1, 0) + 1, n \cdot t(n, m(n)) + \lceil \log n \rceil\}$.
- For $n > 1, i < m(n)$, $h(n,i) = \max\{h(n, i+1) + 1, n \cdot t(n,i) + \lceil \log n \rceil\}$.

Note that h is one-to-one, and $h(n,i) \leq \mathrm{poly}(n)$. For $i \leq m(n)$, we define $F_{h(n,i)}$ to encode the function $f_{n,i}$. Specifically, $F_{h(n,i)}(x,j)$ is the j'th bit of $f_{n,i}(x)$. Note that x takes $n \cdot t(n,i)$ bits to represent and j takes $\lceil \log n \rceil$ bits, so together they can indeed be represented by a string of length $h(n,i)$.

For lengths m not of the form $h(n,i)$, we define F_m to equal F_h where $h = \max\{h(n,i) : h(n,i) \leq m\}$. (Thus, F_m will ignore the last $m - h$ bits of its input.) It can be verified that h can be computed in time $\mathrm{poly}(m)$.

The downward self-reducibility and $\mathbf{PSPACE}$-hardness of F follow immediately from the corresponding properties in Lemma 4.1. The random self-reducibility follows from the well-known self-corrector for low-degree multivariate polynomials [19].