# Pseudorandomness and Average-case Complexity via Uniform Reductions*

Luca Trevisan †  Salil Vadhan‡

January 19, 2006

**Abstract**

Impagliazzo and Wigderson [IW2] gave the first construction of pseudorandom generators from a *uniform* complexity assumption on **EXP** (namely **EXP** $\neq$ **BPP**). Unlike results in the nonuniform setting, their result 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:

- We obtain an optimal worst-case to average-case connection for **EXP**: if **EXP** $\not\subseteq$ **BPTIME**$(t(n))$, then **EXP** has problems that cannot be solved on a fraction $1/2 + 1/t'(n)$ of the inputs by **BPTIME**$(t'(n))$ algorithms, for $t' = t^{\Omega(1)}$.
- We exhibit a **PSPACE**-complete self-correctible and downward self-reducible problem. This slightly simplifies and strengthens the proof of [IW2], which used a a #**P**-complete problem with these properties.
- We argue that the results in [IW2] and in this paper cannot be proved via "black-box" uniform reductions.

---

*A preliminary version of this paper appeared in *CCC '02* [TV].

†U.C. Berkeley, Computer Science Division. `luca@cs.berkeley.edu`. Work partly done while at Columbia University. Research supported by a Sloan Research Fellowship and the NSF Career award grants CCR-9984703, CCR-0406156.

‡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 completed at Harvard University, while supported by NSF grant CCF-0133096, US-Israel BSF grant 2002246, and ONR grant N00014-04-1-0478.

# 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 **EXP**,
2. The existence of problems of high average-case complexity in such classes, and
3. The existence of good pseudorandom generators implying subexponential time or even polynomial-time deterministic simulations of **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, **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 nonuniform distinguishers [NW, BFNW]. This equivalence has become increasingly tight quantitatively, with "low-end" results showing that weak (i.e. slightly superpolynomial) circuit lower bounds imply slight derandomization ($\mathbf{BPP} \subset \mathbf{SUBEXP}$), "high-end" results showing that strong ($2^{\Omega(n)}$) circuit lower bounds imply complete derandomization ($\mathbf{BPP} = \mathbf{P}$), and a smooth tradeoff between these two extremes [Imp1, ACR, IW1, STV, ISW, SU, Uma].

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 **E**, and how this in turn implies good worst-case "algorithms" for **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 [IW2] 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 **BPP** algorithms deterministically in subexponential time (on most inputs, for infinitely many input lengths). This result stands out as an isolated example of using uniform hardness assumptions in derandomization,[1] and suggests that perhaps nonuniformity can be completely removed from this area.

One approach to removing nonuniformity in derandomization results is to show that uniform lower bounds imply nonuniform lower bounds, in the spirit of Karp and Liption [KL]. In fact, one of the theorems in [KL], attributed to Albert Meyer, applies to **EXP** and shows that $\mathbf{EXP} \neq \mathbf{\Sigma_2}$ implies $\mathbf{EXP} \not\subseteq \mathbf{P/poly}$; the hypothesis was subsequently improved to $\mathbf{EXP} \neq \mathbf{MA}$ in [BFL].[2]

[1] Here we do not include the works on "cryptographic" pseudorandom generators, which are based on the stronger assumption that one-way functions exist [BM, Yao, HILL] but are indeed uniform. Also, subsequent to [IW2], there have been several other interesting results on derandomization in the uniform setting, e.g. [Kab, Lu, IKW], but none of these address the questions we enumerate below.

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

Thus, since we know how to get nontrivial derandomizations of $\mathbf{BPP}$ under the assumption $\mathbf{EXP} \not\subseteq \mathbf{P/poly}$, we also get the same derandomizations assuming $\mathbf{EXP} \neq \mathbf{MA}$ [BFNW]. However, this does not fully achieve the goal of removing nonuniformity, because the $\mathbf{MA}$ is not a uniform analogue of $\mathbf{P/poly}$; the two classes seem incomparable. Indeed, Impagliazzo and Wigderson [IW2] take a much more indirect approach to get a derandomization of $\mathbf{BPP}$ under only the assumption $\mathbf{EXP} \neq \mathbf{BPP}$.

Still, this leads one to wonder whether one can prove a "super Karp–Lipton" theorem for $\mathbf{EXP}$, showing that $\mathbf{EXP} \not\subseteq \mathbf{P/poly}$ if and only if $\mathbf{EXP} \neq \mathbf{BPP}$, or more generally, that if $\mathbf{EXP}$ is not contained in circuit size $t(n)$ then $\mathbf{EXP} \not\subseteq \mathbf{BPTIME}(t(n)^{\Omega(1)})$. Such a result would be extremely important beyond the application to derandomization, and it requires a non-relativizing proof [For]. Interestingly, it is plausible that research in derandomization could lead to such a result. Indeed, Impagliazzo, Kabanets, and Wigderson [IKW] use derandomization techniques to prove that $\mathbf{NEXP} \neq \mathbf{MA}$ if and only if $\mathbf{NEXP} \not\subseteq \mathbf{P/poly}$. Already, this is major progress towards a "super Karp–Lipton" theorem, because it establishes an *equivalence* between a uniform lower bound and a nonuniform lower bound. However, it still considers a uniform class ($\mathbf{MA}$) that seems incomparable to the the nonuniform class ($\mathbf{P/poly}$).

Another interpretation of the Impagliazzo–Kabanets–Wigderson result is that any nontrivial derandomization of $\mathbf{MA}$ implies circuit lower bounds for $\mathbf{NEXP}$, because a derandomization of $\mathbf{MA}$ (such as $\mathbf{MA} \subseteq \mathbf{NSUBEXP}$) would presumably separate $\mathbf{MA}$ from $\mathbf{NEXP}$ by the Hierarchy Theorem for Nondeterministic Time (or a variant). It also implies circuit lower bounds from derandomizations of **promise-BPP**, because these imply derandomizations of $\mathbf{MA}$ via the observation that $\mathbf{MA} = \mathbf{NP}^{\mathbf{promise\text{-}BPP}}$ (cf. [GZ]). Considering that Impagliazzo and Wigderson [IW2] prove that $\mathbf{EXP} \neq \mathbf{BPP}$ implies a sub-exponential time derandomization of **promise-BPP** (for most inputs, on infinitely many input lengths), the two results together come close to proving that if $\mathbf{EXP} \neq \mathbf{BPP}$ then $\mathbf{NEXP} \not\subseteq \mathbf{P/poly}$. Unfortunately, the result of [IW2] only implies a derandomization that works on many inputs, which is not enough to derive a derandomization of $\mathbf{MA} = \mathbf{NP}^{\mathbf{promise\text{-}BPP}}$ and apply [IKW]. Still, the combination of the results of [IW2, IKW], come tantalizingly close to a "super Karp-Lipton" result, and provide motivation to further investigate uniform versus non-uniform results in derandomization and related topics.

In this paper, our goal is to provide uniform versions of the known nonuniform trade-offs between worst-case complexity of problems in $\mathbf{EXP}$, average-case complexity of problems in $\mathbf{EXP}$, and derandomization.

The uniform result by Impagliazzo and Wigderson [IW2] 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 $\mathbf{BPP}$). Below, we refer to this as the *ideal tradeoff*.

- Of the several previous results that are being used, how many are really necessary, and what are the properties of **EXP** that are essential to the proof?
- Is it possible to prove, along similar lines, that if $\mathbf{EXP} \neq \mathbf{BPP}$ then **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?

### The Impagliazzo–Wigderson Proof

We first revisit, in Section 3, the arguments of Impagliazzo and Wigderson [IW2], and make the following observations.

1. Analyzing [IW2] for general time bounds, we can obtain a pseudorandom generator that stretches $\approx n$ bits into $\approx t(n)$ bits that fools distinguishers running in time $\approx t(n)$, under the assumption that $\mathbf{EXP} \not\subseteq \mathbf{BPTIME}(t(t(n)))$.

   Recall that the ideal tradeoff, matching what is known in the nonuniform setting, has $t(n)$ instead of $t(t(n))$.

2. As observed by Cai, Nerukar, and Sivakumar [CNS], the ideal tradeoff for derandomization can be obtained from uniform lower bounds on $\#\mathbf{P}$ (instead of **EXP**). That is, the same generator as above can be obtained under the hypothesis that $\#\mathbf{P} \not\subseteq \mathbf{BPTIME}(t(n))$. The key property of $\#\mathbf{P}$ that is used in [IW2, CNS] is that it has a complete problem that is both "random self-reducible" and "downward self-reducible", namely the Permanent [Val].

3. Result 1 above is obtained by constructing two pseudorandom generators, one from an **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 for **EXP**. This step makes crucial use of Toda's Theorem that $\mathbf{\Sigma_2} \subseteq \mathbf{P}^{\#\mathbf{P}}$ [Tod].

4. Using the techniques of [IW2], we can also obtain a uniform worst-case to average-case connection for **EXP**: if every problem in **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. (Previously, a uniform worst-case/average-case connection was also known for $\#\mathbf{P}$, again using special properties of the Permanent [CPS].)

### Our Results

The observations above set the stage for our 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, which incurs only a polynomial loss in $t$ rather than a composition. They give nothing, for example, with $t(n) = 2^{n^\epsilon}$, and more generally limits $t$ to being at most *half-exponential* [MVW]. In terms of derandomization, this means that we cannot get anything near a polynomial-time deterministic simulation of **BPP** from such results.

**Derandomization from PSPACE-hard problems.** In Section 4, we give a direct construction of a **PSPACE**-complete problem that is both random self-reducible and downward self-reducible. (This result was apparently known to some other researchers, but we do not know of a published proof.) 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 giving an ideal derandomization from a uniform lower bound on **PSPACE** rather than #**P**. (It does not, however, provide noticeable improvements when starting from an assumption on **EXP**.) Our construction is based on the ideas used in proving **IP** = **PSPACE** [LFKN, Sha].

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

Recall that worst-case to average-case reductions are only the first step in derandomization. We do not know how to remove the $t(t(n))$ loss incurred in the remaining part of [IW2], namely going from an average-case hard problem to a pseudorandom generator, and leave this as an open problem for future work.

**Black-box Reductions.** In Section 6, we argue that the uniform pseudorandom generator constructions and the uniform worst-case to average-case connections in [IW2] and here cannot be obtained by black-box reductions. The basic reason for this is that black-box reductions can be interpreted information theoretically, and give rise to *randomness extractors* in the case of pseudorandom generators [Tre1] 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 [IW2] used the fact that the Permanent is self-correctible and downward self-reducible. Since only problems in **PSPACE** have downward self-reducible problems, this suggests that to obtain better results from the hardness of **EXP**, we should try to identify special properties of **EXP**-complete problems that can be exploited. Our optimal hardness amplification identifies one such property, namely *instance checkability.* We do not know whether this property suffices for getting optimal derandomization. Even if not, it might help point the direction to other properties of **EXP** that can be used.

## 2 Preliminaries

We call a function $t : \mathbb{N} \to \mathbb{R}^+$ a *nice time bound* if $n \le t(n) \le 2^n$, $t(n)$ is non-decreasing, $t(n)$ is computable in time $\mathrm{poly}(n)$, and $t(O(n)) \le \mathrm{poly}(t(n)) \le t(\mathrm{poly}(n))$. All functions of the form $n^c$, $n^c \log n$, $2^{(\log n)^c}$, $2^{cn}$ 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 often identify a language with its characteristic function. $\mathbf{BPTIME}(t(n))$ denotes the class of languages decidable by probabilistic algorithms with 2-sided error running in time $t(n)$. $\mathbf{SIZE}(t(n))$ denotes the class of languages that 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). A prefix of **i.o.** to a complexity class means the class of languages that can be solved on *infinitely many input lengths* by algorithms within the stated resource bounds. For example a language $L$ with characteristic function $f$ is in **i.o.-BPTIME**$(t)$ if there is a probabilistic algorithm $A$ running in time $t$ such that for infinitely many $n$ we have that $\Pr[A(x) = f(x)] \geq 2/3$ for all inputs $x$ of length $n$.[3]

### 2.1 Average-case Complexity.

Let $L$ be a language and $\mathcal{D} = \{D_n\}_{n\geq 1}$ be an ensemble of distributions, where $D_n$ is a distribution over inputs of length $n$. Then we call the pair $(L, \mathcal{D})$ a *distributional problem*.

We say that an ensemble $\mathcal{D}$ is *samplable in time* $t(n)$ if there is a $t(n)$-time probabilistic algorithm $S$ such that $\Pr[S(1^n) = \perp] \leq 1/2$ and the output of $S(1^n)$, conditioned on $S(1^n) \neq \perp$, is distributed identically to $D_n$ for every $n$. **PSamp** denotes the class of ensembles that are samplable in polynomial time. We denote by $\{U_n\}_{n\geq 1}$ the ensemble where each distribution $U_n$ is uniform over $\{0,1\}^n$.

We will give definitions of average-case tractability and intractability of distributional problems. The conference version of this paper [TV] used different terminology, but here we switch to more commonly used terminology.

**Definition 2.1 (Heuristic Time for Distributional Problems)** *Let $(L, \mathcal{D})$ be a distributional problem and $f : \{0,1\}^* \to \{0,1\}$ be the characteristic function of $L$. Let $t(n)$ be a nice time bound and $\delta(n)$ be a* poly$(n)$*-time computable non-increasing function. We say that $(L, \mathcal{D}) \in$ **HeurTIME**$(t(n), \delta(n))$ if there is a time $t(n)$ deterministic algorithm $A$ such that*

$$\Pr_{x \leftarrow D_n}[A(x) = f(x)] \geq 1 - \delta(n).$$

If $L$ is a paddable language and $(L, \mathcal{D}) \in$ **HeurTIME**$(n^{O(1)}, 1/n^{\Omega(1)})$, then it is possible to show that $(L, \mathcal{D})$ is also in the class **HP**, for *Heuristic Polynomial Time* defined by Impagliazzo [Imp2].

The class **HeurBPTIME**$(t(n), \delta(n))$ is defined similarly, except that the algorithm $A$ is allowed to be probabilistic, and the probability of outputting a correct answer is computed over the random choices of the algorithm as well as over the distribution of inputs.

We think of a language $L$ as being very hard on average when $(L, U) \notin$ **HeurBPTIME**$(t(n), 1/2 - \delta(n))$ for large $t(n)$ and small $\delta(n)$ (or even better, $(L, U) \notin$ **i.o.-HeurBPTIME**$(t(n), 1/2 - \delta(n))$). This means that for every fixed algorithm of running time $t(n)$, if we pick an input $x$ at random, the algorithm does not have a much better chance of correctly solving the problem on input $x$ as an algorithm that simply makes a random guess.

A more attractive notion for defining average-case 'easiness' is the following, which requires that there is one algorithm that works well for all efficiently samplable distributions.

[3]This differs from the traditional definition of **i.o.** classes, which would require that there is language $L' \in$ **BPTIME**$(t)$ such that $L \cap \Sigma^n = L' \cap \Sigma^n$ for infinitely many $n$. The reason is that our definition allows $A$ to accept with probability between 1/3 and 2/3 outside the infinitely many 'good' $n$, so it may not correspond to a language $L' \in$ **BPTIME**$(t)$. Another way around this difficulty is to work with classes of *promise problems*, in which case our definition and the traditional definition coincide [Gol].

**Definition 2.2 (Pseudo-Time for Languages [Kab])** *Let $L$ be a language, $f$ be its characteristic function, and let $t(n)$ be a nice time bound. Then $L \in$ **PseudoTIME**$(t(n))$ (resp., $L \in$ **i.o.-PseudoTIME**$(t(n))$ if there is a deterministic algorithm $A$ that runs in time $t(n)$ such that for every ensemble $\mathcal{D} \in$ **PSamp** and every constant $c$, we have*

$$\Pr_{x \leftarrow D_n}[A(x) = f(x)] \geq 1 - 1/n^c,$$

*for all sufficiently large $n$ (resp., for infinitely many $n$).*

Note that if $L \in$ **PseudoTIME**$(t(n))$, then $(L, \mathcal{D}) \in$ **HeurTIME**$(t(n), 1/n^c)$ for every $\mathcal{D} \in$ **PSamp** and every constant $c$. However, the converse is not clear, because saying that $(L, \mathcal{D}) \in$ **HeurTIME**$(t(n), 1/n^c)$ allows the time-$t(n)$ algorithm to depend on the samplable ensemble $\mathcal{D}$, whereas **PseudoTIME**$(t(n))$ requires a single algorithm that is good for all samplable $\mathcal{D}$. Kabanets [Kab] observed that the results of Impagliazzo and Wigderson [IW2] achieve the stronger notion of **PseudoTIME**, even though they were originally stated in terms of **HeurTIME**.

## 2.2 Self-Reducibility.

For $f : \{0,1\}^* \to \{0,1\}$, we denote the restriction of $f$ to inputs of length $n$ by $f_n$. We say that $f : \{0,1\}^* \to \{0,1\}$ is *downward self-reducible* if there is a (deterministic) polynomial time algorithm $A$ such that for all $x \in \{0,1\}^n$, $A^{f_{n-1}}(x) = f_n(x)$.

We call $f : \{0,1\}^* \to \{0,1\}$ *self-correctible* [BK, Lip, BLR] if there is a constant $c$ and a probabilistic polynomial-time algorithm $A$ (a *self-corrector* for $f$) 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 $x \in \{0,1\}^n$. Note that if $f$ is self-correctible and $f \notin$ **BPTIME**$(t(n))$, then $(f, U) \notin$ **HeurBPTIME**$(t(n)/n^c, 1/n^c)$ for some constant $c$. [4] (The probability is just over the coin tosses of $A$.)

## 2.3 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 $T$ $\epsilon(\cdot)$-*distinguishes* $G$ in *time* $t(\cdot)$ (resp., *size* $t(\cdot)$) if $T$ is a probabilistic algorithm that runs in time $t(n)$ (resp., is computable by a circuit of size $t(n)$) on inputs of length $m(n)$ and

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

for all sufficiently large $n$. Intuitively, a "good" pseudorandom generator will have no efficient distinguisher. Note that, by our definition, if $G$ cannot be distinguished in time $t$, it only means that every time $t$ algorithm fails to distinguish $G$ *infinitely often*. Note also that we give the distinguisher the seed $x$ to the pseudorandom generator; sometimes such generators are referred to as "strong" in analogy with strong randomness extractors (cf. [Sha]). This makes sense here

[4] A similar definition is that of $f$ being *random self-reducible* [AFK, FKN, FF], in which we allow $A$ oracle access to $f$ itself (rather than a corrupted version of $f$), but put restrictions on the distribution of the oracle queries. This implies that $f$ is as hard in the worst case as it is hard on average with respect to some samplable distribution. For our purposes, it is more convenient that the definition more directly enforces hardness on average with respect to the *uniform* distribution.

because we, following [NW], work with pseudorandom generators whose running time is greater than that of the distinguisher. (In constrast, "cryptographic" pseudorandom generators, in the sense of [BM, Yao], allow the distinguisher more time than the generator, and thus rely on the secrecy of the seed.) However, we will require our pseudorandom generators to be computable in time $2^{O(n)}$ on seeds of length $n$; we call such a generator *quick*.

As usual, pseudorandom generators imply derandomization. With the above definitions, the standard fact that pseudorandom generators imply derandomization becomes:

**Lemma 2.3** *Suppose that there exists a quick generator $G$ with stretch $m(\cdot)$ that cannot be $1/m(\cdot)^c$-distinguished in time $m(\cdot)^c$ for any constant $c$. Then,*

$$\mathbf{BPTIME}(n) \subseteq \mathbf{i.o.\text{-}PseudoTIME}(n^2 \cdot 2^{m^{-1}(n^2)}).$$

**Proof:** Let $M(x;r)$ be a $\mathbf{BPTIME}(n)$ algorithm for a language $L$, where $x$ denotes the input and $r$ the coin tosses. By standard error reduction, we can obtain an algorithm $M'$ with error probability $2^{-\Omega(n)}$ and running time $n^2$. Let $\ell(n) = m^{-1}(n^2)$.

Consider the deterministic algorithm $A$ that, on any input $x$ of length $n$, computes

$$A(x) = \operatorname*{maj}_{y \in \{0,1\}^{\ell(n)}} M'(x; G(y)).$$

We argue that $A$ decides $L$ correctly with high probability on every distribution $D$ samplable in polynomial time, for infinitely many input lengths. Let $f$ be the characteristic function of $L$, and suppose that

$$\Pr_{x \leftarrow D_n}[A(x) \neq f(x)] \geq 1/n^c$$

for infinitely many $n$.

Consider a distinguisher $T$ that does the following on input $(y, r)$, where $|y| = \ell(n)$ and $|r| = n^2$: Choose $x \leftarrow D_n$ and $s \leftarrow \{0,1\}^{n^2}$ uniformly at random. Let $b_r = M(x;r)$ and $b_s = M(x;s)$. If $b_r \neq b_s$, output 1, else output 0.

Notice that $b_s$ equals $f(x)$ except with probability $2^{-\Omega(n)}$. Now if $r \leftarrow \{0,1\}^{t'(n)}$ uniformly at random, then $b_r$ also equals $f(x)$, except with probability $2^{-\Omega(n)}$, so $T$ outputs 1 with probability at most $2 \cdot 2^{-\Omega(n)}$. On the other hand, if $r$ comes from $G(U_{\ell(n)})$, then by the definition of $A$ and the fact that $A$ has error probability at least $1/n^c$, $T$ outputs 1 with probability at least $(1/2) \cdot (1/n^c)$.

Thus $T$ distinguishes the output of $G(U_{\ell(n)})$ from $U_{n^2}$ with advantage $1/(2n^c) - 2^{-\Omega(n)} \geq 1/4n^c$, for infinitely many $n$. Moreover, $T$ has running time $\mathrm{poly}(n) + O(n^2) = \mathrm{poly}(n)$. This violates the pseudorandomness of $G$. ■

We note that the above lemma also implies derandomization of all of **BPP** by a standard padding argument.

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

**Lemma 2.4** *If there is an algorithm $A$ that runs in time $t(\cdot)$ (resp., computed by a circuit of size $t(\cdot)$) such that*

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

*for some $m(\cdot)$ and all $n$, then there is an algorithm that $\epsilon(\cdot)$-distinguishes $G$ in time $t(\cdot) + m(\cdot)$ (resp., in size $t(\cdot)$). Here $G(x)|_k$ denotes the first $k$ bits of $G(x)$.*

*In particular, if there is a generator $G$ that cannot be $\epsilon(\cdot)$-distinguished in time $t(\cdot)$, then the first bit of $G$ defines a language $L$ such that $(L, U)$ is not in* $\mathbf{HeurBPTIME}(t(n) - O(1), 1/2 - \epsilon(\cdot))$.

## 3 The Impagliazzo–Wigderson Proof

The main result of Impagliazzo and Wigderson [IW2] is the following.

**Theorem 3.1 ([IW2])** *If* $\mathbf{EXP} \neq \mathbf{BPP}$, *then* $\mathbf{BPP} \subseteq \mathbf{i.o.\text{-}PseudoTIME}\left(2^{n^\epsilon}\right)$ *for every* $\epsilon > 0$.

In this section, we recall the proof of Theorem 3.1 and, in the process, analyze it for larger time bounds.

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

**Lemma 3.2 ([NW])** *For every nice time bound $m(\cdot)$ and every self-correctible function $f$, there is a generator $G$ with stretch $m(\cdot)$ and a constant $d$ such that*

- *$G(x)$ can be computed in time $m(n)^d$ on inputs $x$ of length $n$, given oracle access to $f$ on inputs of length at most $n$.*
- *If $G$ can be $(1/t(n))$-distinguished in size $t(n)$ for some nice time bound $t(n) \geq m(n)$, then $f$ is in* $\mathbf{SIZE}(t(n^d))$.

Quantitatively better results that improve the $\mathbf{SIZE}(t(n^d))$ to $\mathbf{SIZE}(t(n)^d)$ are now known [IW1, STV, ISW, SU, Uma], but we use the above for simplicity. The self-correctible hard function $f$ can be obtained from any hard function $f$ by taking a multilinear extension:

**Lemma 3.3 ([BF, BFNW])** *For every function $f$, there is a self-correctible function $f'$ 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 [IW2] was the observation that the circuit complexity conclusion of Lemma 3.2 can be replaced with a uniform conclusion about *learnability*.

**Definition 3.4** *A function $f$ is in* $\mathbf{LEARN_{mem}}(t(n))$ *if there is a probabilistic $t(n)$-time algorithm $A$ such that for every $n$, $A^{f_n}(1^n)$ outputs a circuit that computes $f_n$ with probability at least $2/3$.*

**Lemma 3.5 ([IW2])** *For every nice time bound $m(\cdot)$ and every self-correctible $f$, there is a generator $G$ with stretch $m(\cdot)$ and a constant $d$ such that*

- *$G(x)$ can be computed in time $m(n)^d$ on inputs $x$ of length $n$, given oracle access to $f$ on inputs of length at most $n$.*
- *If $G$ can be $(1/t(n))$-distinguished in time $t(n)$ for some nice time bound $t(n) \geq m(n)$, then $f$ is in* $\mathbf{LEARN_{mem}}(t(n^d))$.

The next new ingredient of [IW2] was showing that the learnability can be turned into standard uniform easiness if the function $f$ is *downward self-reducible*.

**Lemma 3.6 ([IW2])** *For every downward self-reducible $f$ and nice time bound $t(n)$, there is a constant $d$ such that if $f \in \mathbf{LEARN_{mem}}(t(n))$, then $f \in \mathbf{BPTIME}(t(n)^d)$.*

The problem with this is that all downward self-reducible problems lie in **PSPACE**, but we would like to start with a hard function in **EXP**. The way this is overcome in [IW2] is to assume that **EXP** has polynomial-sized circuits (for otherwise we're already done by Lemma 3.2). Under this assumption, a version of the Karp–Lipton Theorem, attributed to Albert Meyer, collapses **EXP** to $\mathbf{\Sigma_2}$. Generalizing this to higher time bounds gives:

**Lemma 3.7 (Meyer [KL])** *For every* **EXP***-complete function $f$, there is a constant $d$ such that if $f \in \mathbf{SIZE}(t(n))$ for a nice time bound $t$, then $f \subset \mathbf{\Sigma_2}(t(n^d))$.*

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

**Lemma 3.8 ([Val, Tod, Lip, IW2])** *There is a self-correctible and downward self-reducible $\mathbf{\Sigma_2}$-hard problem, namely, the* Permanent.

Combining all of the above, we get the pseudorandom generator construction.

**Theorem 3.9 (implicit in [IW2])** *For every function $f \in \mathbf{EXP}$, there is a constant $d$ such that if $f \notin \bigcup_c \mathbf{BPTIME}(t(t(n^d)^c))$, then there is a generator $G$ with stretch $t(\cdot)$ that cannot be $1/t(\cdot)^c$-distinguished in time $t(\cdot)^c$ for any constant $c$.*

**Proof sketch:** Let $f_1$ be a self-correctible **EXP**-complete problem (given by Lemma 3.3), and let $f_2$ be the Permanent. Use Lemma 3.2 to construct a generator $G_1$ with stretch $t(\cdot)$ from $f_1$, and use Lemma 3.5 to construct a generator $G_2$ with stretch $t(\cdot)$ from $f_2$. Suppose for sake of contradiction that both $G_1$ and $G_2$ can be $1/t(\cdot)^c$-distinguished in time $t(\cdot)^c$. Then $f_1 \in \mathbf{SIZE}(t(\mathrm{poly}(n))^c)$ and $f_2 \in \mathbf{LEARN_{mem}}(t(\mathrm{poly}(n))^c)$. (Here and throughout this proof, the $\mathrm{poly}(\cdot)$ refer to fixed polynomials that depend only on the function $f$, whereas by definition $c$ depends on the distinguishers to $G_1$ and $G_2$.) Since $f_2$ is downward self-reducible, Lemma 3.6 gives $f_2 \in \mathbf{BPTIME}(t(\mathrm{poly}(n))^c)$. Since $f_1$ is **EXP**-complete, Lemma 3.7 gives $f_1 \in \mathbf{\Sigma_2}(t(\mathrm{poly}(n))^c)$. By Lemma 3.8, $f_1$ reduces to $f_2$ in time $t(\mathrm{poly}(n))^c$, from which we conclude $f_1 \in \mathbf{BPTIME}(t(t(\mathrm{poly}(n))^c)^c) \subseteq \mathbf{BPTIME}(t(t(\mathrm{poly}(n))^{c'}))$, where the latter inclusion uses the fact that $t$ is a nice time bound. Since $f_1$ is **EXP**-complete, we deduce that $f \in \mathbf{BPTIME}(t(t(\mathrm{poly}(n))^{c'}))$, contradicting the hypothesis. □

Combining this with Lemma 2.3, we get the following generalization of Theorem 3.1.

**Corollary 3.10 (implicit in [IW2])** *If* $\mathbf{EXP} \not\subset \bigcup_c \mathbf{BPTIME}(t(t(n^c)))$, *then*

$$\mathbf{BPP} \subseteq \bigcup_c \textbf{i.o.-PseudoTIME}(n^c \cdot 2^{t^{-1}(n)}).$$

**Proof:** We will show that $\mathbf{BPTIME}(n^k) \subseteq \mathbf{i.o.}\text{-}\mathbf{PseudoTIME}(n^{2k} \cdot 2^{t^{-1}(n)})$ for every constant $k$. Let $t'(n) = t(n)^{2k}$. Then for any constant $c$, $t'(t'(n^c)) \leq t(t(n^{c'}))$ for a constant $c'$, because $t$ is a nice time bound. So $\mathbf{EXP} \not\subset \bigcup_c \mathbf{BPTIME}(t'(t'(n^c)))$. By Theorem 3.9 and Lemma 2.3, $\mathbf{BPTIME}(n) \subseteq \mathbf{i.o.}\text{-}\mathbf{PseudoTIME}(n^2 \cdot 2^{(t')^{-1}(n^2)})$. By padding, $\mathbf{BPTIME}(n^k) \subseteq \mathbf{i.o.}\text{-}\mathbf{PseudoTIME}(n^{2k} \cdot 2^{(t')^{-1}(n^{2k})})$. Noting that $(t')^{-1}(n^{2k}) \leq t^{-1}(n)$, by the definition of $t'$, the proof is complete. ■

Note that this only gives a deterministic simulation of **BPP** *infinitely often*. In most previous works on derandomization, it is also possible to obtain a simulation for all input lengths by assuming that **EXP** has a problem that is hard for almost all input lengths, i.e. **EXP** is not in $\mathbf{i.o.}\text{-}\mathbf{BPTIME}(t(n))$ for some $t(\cdot)$. However, one of the steps of the above proof, namely Lemma 3.6, breaks down if we try to work with an infinitely-often hypothesis.

We also observe that a worst-case vs. average-case connection now follows from Theorem 3.9 via Lemma 2.4.

**Corollary 3.11** *If* $\mathbf{EXP} \not\subset \bigcup_c \mathbf{BPTIME}(t(t(n^c)))$, *then*

$$\mathbf{EXP} \times \{U\} \not\subset \bigcup_c \mathbf{HeurBPTIME}(t(n^c), 1/2 - 1/t(n^c)).$$

In Section 5, we improve this corollary 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 a version that says that if **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 only implies hardness "infinitely often").

# 4 A Self-Correctible and Downward Self-Reducible PSPACE-complete Problem

The proof of Impagliazzo and Wigderson described 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 and the previous section. In this section, we show that Valiant's Theorem and Toda's Theorem, which were used in Lemma 3.8, are not necessary. Instead, we show that there is a self-correctible and downward self-reducible complete problem for **PSPACE**. At first, this seems easy. The canonical **PSPACE**-complete problem QBF is downward self-reducible, and Lemma 3.3 impliess that **PSPACE** also has a self-correctible complete problem. However, the Impagliazzo–Wigderson proof appears to need a *single* complete problem that 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 **PSPACE** [LFKN, Sha].

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 GF(2)) can be found deterministically in time poly($n$) [Sho].

**Lemma 4.1** *For some polynomials $t$ and $m$, there is a collection of functions* $\{f_{n,i} : (\mathbb{F}_n)^{t(n,i)} \to \mathbb{F}_n\}_{n \in \mathbb{N}, i \leq m(n)}$ *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. *(**PSPACE**-hardness) For every language $L$ in **PSPACE** ,with characteristic function $\chi_L$, 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 **PSPACE**-complete problem QBF, as presented in [Sip]. 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 $\mathbb{F}_n$. Each $f_i$ has variables $(x_1, \ldots, x_\ell)$ for some $\ell = \ell(i) \leq m$. $f_m = f_m(x_1, \ldots, x_n)$ is an arithmetization of the propositional formula $\psi(x_1, \ldots, x_n)$, and for $j < n$, $f_j(x_1, \ldots, x_\ell)$ is defined in terms of $f_{j+1}$ using one of the following 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 first corresponds to universal quantifiers, the second to existential quantifiers, and the third is used to reduce the degree in variable $x_k$.) The construction provides the following guarantees:

- If $f_j$ depends on $\ell$ variables, then when $x_1 \ldots, x_\ell$ take on Boolean values, $f_j(x_1, \ldots, x_\ell)$ equals the truth value of $\phi$ with the first $\ell$ quantifiers removed. $f_0$ is a constant polynomial, and thus equals the truth value of $\phi$ (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 that 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 $\phi$, and not just its length. To solve this, we incorporate the formula $\phi$ into the arithmetization (as done for PCP's in, e.g. [BFL, FGL+] for different reasons). We do this by defining a single "universal" quantified formula $\Phi_n$ which has some *free* variables such that by setting these free variables appropriately, $\Phi_n$ can be specialized to any instance of QBF. Specifically, $\Phi_n$ has $2n^2$ free variables $\{y_{i,j}, z_{i,j} : 1 \leq i, j \leq n\}$, 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 $\phi$ be any instance of QBF. Without loss of generality, we may assume $\phi$ is in the form $\phi = \exists x_1 \forall x_2 \cdots \exists/\forall x_n \psi(x_1, \ldots, x_n)$, where $\psi$ is a CNF formula with at most $n$ clauses. (These restrictions still preserve the fact that QBF is a **PSPACE**-complete problem.) Define $\overline{y}(\phi)$ and

$\overline{z}(\phi)$ as follows: $y_{i,j}(\phi) = 1$ iff the $i$'th clause of $\psi$ contains $x_j$, and $z_{i,j}(\phi) = 1$ iff the $i$'th clause of $\psi$ contains $\neg x_j$. 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 above-described **IP** = **PSPACE** construction to $\Phi_n$. One difference is that, unlike a standard instance of QBF, $\Phi_n$ has the free variables $\overline{y} = (y_{i,j})$, $\overline{z} = (z_{i,j})$. The effect of this is that each $f_{n,i}$ will have variables $(\overline{y}, \overline{z}, x_1, \ldots, x_\ell)$ for some $\ell \leq n$ (rather than just $(x_1, \ldots, x_\ell)$ as in the original construction.) Analogous to the original construction, the resulting sequence of polynomials has the following properties:

- If $f_j$ depends on $\ell$ of the $x$-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 $\Phi_n$ with the first $\ell$ 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.
- $f_{n,m(n)}$ can be evaluated in time poly$(n)$.
- $f_{n,j}$ can be computed in time poly$(n)$ given oracle access to $f_{n,j+1}$.
- Each $f_{n,j}$ is of total degree at most poly$(n)$.

This establishes the self-reducibility and low degree properties. The **PSPACE**-hardness follows from the fact that $f_{n,0}(\overline{y}, \overline{z}) = \Phi_n(\overline{y}, \overline{z})$. □

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 lengths so as to turn the "upwards" reductions from $f_{n,i}$ to $f_{n,i+1}$ into a downward self-reduction for $F$.

**Theorem 4.2** **PSPACE** *has a complete problem that is both self-correctible and downward self-reducible.*

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

The downward self-reducibility and **PSPACE**-hardness of $F$ follow immediately from the corresponding properties in Lemma 4.1. The self-correctibility follows from the fact that each $f_{n,i}$ is a multivariate polynomial of total degree at most poly($n$) over a field of size $2^n$. Specifically, the well-known self-corrector for multivariate polynomials [Lip, BF] can be used to correctly compute such a polynomial everywhere (with high probability) given oracle access to a function that agrees with it in a $1 - 1/n^c$ fraction of positions for some constant $c$. (In fact, there are now results that require much less agreement.) ■

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 **BPP** from a uniform assumption about hard problems in **PSPACE** (as opposed to $\mathbf{P}^{\#\mathbf{P}}$, as in [IW2, CNS]).

**Corollary 4.3** *For every function $f \in \mathbf{PSPACE}$, there is a constant $d$ such that if $f \notin \bigcup_c \mathbf{BPTIME}(t(n^d)^c)$, then there is a generator $G$ with stretch $t(\cdot)$ that cannot be $1/t(\cdot)^c$-distinguished in time $t(\cdot)^c$ for any constant $c$, and thus $\mathbf{BPP} \subseteq \bigcup_c \textbf{i.o.-PseudoTIME}(n^c \cdot 2^{t^{-1}(n)})$.*

**Corollary 4.4** *If $\mathbf{PSPACE} \not\subseteq \bigcap_{\epsilon>0} \mathbf{BPTIME}(2^{n^\epsilon})$, then $\mathbf{BPP} \subseteq \bigcup_c \textbf{i.o.-PseudoTIME}(2^{\log^c n})$.*

# 5 Uniform Hardness Amplification

In this section we will prove that if every problem in **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 **EXP** is contained in $\mathbf{BPTIME}(t(\mathrm{poly}(n)))$.

We will prove our result in a series of steps. First, we observe that the nonuniform worst-case to average-case reduction in [STV] actually uses only a "logarithmic amount of nonuniformity." More precisely, the reduction can be implemented by 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 BPP)** *For functions $t$ and $a$, we say that a language $L$ with characteristic function $f$ is in $\mathbf{BPTIME}(t)//a$ if there is a $t(n)$-time algorithm $A$ and a function $\alpha$ such that for every $n$,*

$$\Pr_{r \in \{0,1\}^{t(n)}} [\forall x \in \{0,1\}^n A(x, r, \alpha(r)) = f(x)] \geq \frac{3}{4},$$

*and $|\alpha(r)| = a(n)$ for $|r| = t(n)$.*

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

**Theorem 5.2 ([STV])** *There is a constant $d$ such that for every boolean function $f$ and nice time bound $t$, there is a boolean function $f'$ such that*

- *$f$ is reducible to $f'$ (via a linear-time Karp reduction);*
- *$f'$ is computable in linear space (and hence in $\mathbf{E}$) given oracle access to $f$ (and all oracle queries are of size $\Theta(n)$);*

- *if* $(f', U)$ *is in* $\mathbf{HeurBPTIME}(t(n), 1/2{-}1/t(n))$*, then* $f'$ *and* $f$ *are in* $\mathbf{BPTIME}(t(n)^d)//d\log t(n)$.

**Proof sketch:** The truth table of the function $f'$ is an encoding of the truth table of the function $f$ using the error-correcting code of Lemma 28 in [STV]. The code is computable in time polynomial in (and space logarithmic in) the length of the truth table of $f$, and so if $f$ has inputs of length $n$, then $f'$ is computable in time $2^{O(n)}$ and space $O(n)$ given the truth-table of $f$. (Or, equivalently, given oracle access to $f$.) The input length of $f'$ is $O(n)$, and the construction of the code is such that for every $x$ there is a linear-time constructible bit string $x'$ such that $f(x) = f'(x')$. The decoding algorithm of Lemma 28 of [STV] is such that a $\mathbf{HeurBPTIME}(t(n), 1/2 - 1/t(n))$ algorithm for $(f', U)$ implies the existence of a probabilistic algorithm that runs in time $\text{poly}(t(n))$ and outputs a list of $\text{poly}(t(n))$ circuits such that with high probability one of them computes $f'$ (and hence, by the simple reduction, $f$) on all inputs. $O(\log t(n))$ bits of advice can then be used to choose a correct circuit from the list, thus showing that $f'$ (and $f$) are in $\mathbf{BPTIME}(\text{poly}(t(n)))//O(\log t(n))$. □

We note that some other methods for achieving strong average-case hardness, such as derandomized versions of Yao's XOR Lemma [Imp1, IW1, Tre2], appear to require significantly more nonuniformity.

Finally, we show that for certain **EXP**-complete or **PSPACE**-complete $f$, then $f'$ can be in $\mathbf{BPTIME}(\text{poly}(t(n)))//O(\log(t(n)))$ only if it is also in $\mathbf{BPTIME}(t(\text{poly}(n)))$. This will be a consequence of the fact that **EXP**-complete and **PSPACE**-complete problems have instance checkers in the sense of Blum and Kannan [BK].

**Definition 5.3 (instance checker)** *An* instance checker *$C$ for a boolean function $f$ is a polynomial-time probabilistic oracle machine whose output is in* $\{0, 1, \texttt{fail}\}$ *such that*

- *for all inputs $x$,* $\Pr[C^f(x) = f(x)] = 1$.
- *for all inputs $x$, and all oracles $f'$, then* $\Pr[C^{f'}(x) \notin \{f(x), \texttt{fail}\}] \leq 1/4$*;*

Intuitively, if $f$ has an instance checker, then machine $C$, given an input $x$ and an oracle $f'$ that purports to compute $f$, with high probability will be able to verify the validity of the oracle on $x$ by comparing $f'(x)$ to $C^{f'}(x)$. This definition is slightly different from the original definition of [BK], but is easily seen to be equivalent and is more convenient for our purposes.

As observed in [BFL], the proof of $\mathbf{MIP} = \mathbf{NEXP}$ in [BFL] implies the existence of instance checkers for all **EXP**-complete problems, and the proof of $\mathbf{IP} = \mathbf{PSPACE}$ in [LFKN, Sha] implies the existence of instance checkers for all **PSPACE**-complete and $\mathbf{P}^{\#\mathbf{P}}$-complete problems.

**Theorem 5.4 ([BFL],[LFKN, Sha])** *Every* **EXP***-complete problem, every* **PSPACE***-complete problem, and every* $\mathbf{P}^{\#\mathbf{P}}$*-complete problem has an instance checker. Moreover, there are* **EXP***-complete problems,* **PSPACE***-complete problems, and* $\mathbf{P}^{\#\mathbf{P}}$*-complete problems for which the instance checker $C$ only makes oracle queries of length exactly $\ell(n)$ on inputs of length $n$ for some polynomial $\ell$.*

A result[5] essentially equivalent to the existence of instance checkers for **EXP** has been in complexity theory before, e.g. in [BFL, BFNW, BFvT, Vin]. Our application of it to worst-case/average-case connections, however, seems new.

[5]Specifically, the fact that every problem in **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.

**Lemma 5.5** *Let $f \in \mathbf{BPTIME}(t)//a$ be a problem admitting an instance checker that makes queries of length exactly $\ell(n)$ on inputs of length $n$. Then $f \in \mathbf{BPTIME}(\mathrm{poly}(t(\ell(n))) \cdot 2^{a(\ell(n))})$.*

**Proof:** Let $C$ be the instance checker, let $A(\cdot,\cdot,\cdot)$ be the $\mathbf{BPTIME}(t)//a$ algorithm for $f$ and let $\alpha$ be the advice function. We reduce the error probability of the instance checker $C$ to $2^{-a(\ell(n))-3}$ by taking independent repetitions, at the cost of increasing its running time to $\mathrm{poly}(t(n)) \cdot a(\ell(n))$.

We now describe a $\mathbf{BPTIME}(\mathrm{poly}(t(\ell(n))) \cdot 2^{a(\ell(n))})$ algorithm for $f$. On input $x$ of length $n$, we pick $r$ at random, and run $C^{A(\cdot,r,s)}(x)$ for all $2^{a(\ell(n))}$ possible advice strings $s$ for the computation of $A$ on inputs of length $\ell(n)$. The first time $C^{A(\cdot,r,s)}(x)$ outputs a value $\sigma$ other than `fail`, we output $\sigma$. If $C^{A(\cdot,r,s)}(x)$ outputs `fail` for all $s$, we output `fail`.

We now bound the probability that the above algorithm outputs either `fail` or $\sigma \neq f(x)$. By the error reduction of our instance checker for every fixed $r$ and $s$, the probability that $C^{A(\cdot,r,s)}(x) \notin \{\mathtt{fail}, f(x)\}$ is at most $2^{-a(\ell(n))-3}$. Thus the probability that the above algorithm's output is not in $\{\mathtt{fail}, f(x)\}$ is at most $2^{a(\ell(n))} \cdot 2^{-a(\ell(n))-3} = 1/8$.

By the definition of $\mathbf{BPTIME}(t)//a$, $A(\cdot, r, \alpha(r))$ correctly computes $f$ on inputs of length $\ell(n)$ with probability at least $3/4$ over the choice of $r$. If this happens, then $C^{A(\cdot,r,\alpha(r))}(x) = f(x)$ by the definition of instance checker. Thus, the above algorithm outputs `fail` with probability at most $1/4$.

Therefore the probability that the above algorithm doesn't output $f(x)$ is at most $1/4 + 1/8 = 3/8$, so we indeed have a bounded-error algorithm. The running time can be verified by inspection. ■

Combining Theorem 5.4 with Lemma 5.5, we get:

**Proposition 5.6** *There is an* **EXP***-complete function $f$ and a constant $d$ such that if $f \in \mathbf{BPTIME}(t(n))// \log t(n)$, then $f \in \mathbf{BPTIME}(t(n^d))$.*

This is analogous to the well-known fact that $\mathbf{NP} \subseteq \mathbf{P}/\mathbf{log} \Rightarrow \mathbf{NP} = \mathbf{P}$, which makes use of the equivalence of search and decision for **NP**-complete problems. For our result, we instead used the instance-checkability of **EXP**-complete problems.

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

**Theorem 5.7** *For every function $f \in \mathbf{EXP}$, there is a constant $d$ such that if $f \notin \mathbf{BPTIME}(t(n^d))$ for a nice time bound $t$, then there is a function $f' \in \mathbf{EXP}$ such that*

$$(f', U) \notin \mathbf{HeurBPTIME}(t(n), 1/2 - 1/t(n)).$$

**Proof:** Let $g$ be the **EXP**-complete problem from Proposition 5.6. Let $f'$ be the function obtained by applying Theorem 5.2 to $g$. If $(f', U) \in \mathbf{HeurBPTIME}(t(n), 1/2 - 1/t(n))$, then $g \in \mathbf{BPTIME}(\mathrm{poly}(t(n)))//O(\log t(n))$, by Theorem 5.2. Proposition 5.6 then implies that $g \in \mathbf{BPTIME}(\mathrm{poly}(t(n)))$. By the **EXP**-completeness of $g$, we deduce that $f \in \mathbf{BPTIME}(\mathrm{poly}(t(\mathrm{poly}(n)))) = \mathbf{BPTIME}(t(\mathrm{poly}(n)))$. ■

Theorem 5.7 improves on what we were able to obtain using the techniques of [IW2], namely Corollary 3.11, in that we no longer incur a composition $t(t(\mathrm{poly}(n)))$ in the conclusion. Still, the above theorem does not match what is known in the nonuniform setting. For example, we should be able to prove that $\mathbf{E} \not\subseteq \mathbf{BPTIME}(2^{o(n)})$ implies $\mathbf{E} \not\subseteq \mathbf{HeurBPTIME}(2^{o(n)}, 1/2 -$

$1/2^{o(n)}$). Obtaining such finer worst-case to average-case connections in the nonuniform setting received significant attention in the past few years [Imp1, IW1, STV] and was essential in obtaining $\mathbf{P} = \mathbf{BPP}$ under worst-case assumptions [IW1]. To obtain such a result for the uniform setting, we observe that results on probabilistically checkable proofs imply the following strengthening of Theorem 5.4 for $\mathbf{E}$.

**Theorem 5.8 ([BFLS])** *There is a problem complete for* $\mathbf{E}$ *under linear-time reductions that has an instance checker that only makes oracle queries of length exactly* $\ell(n) = O(n)$ *on inputs of length* $n$.

**Proof:** We begin a paddable language $L$ that is complete for $\mathbf{E}$ under linear-time reductions, e.g. $L = \{x$ : the universal TM accepts $x$ within $2^{|x|}$ steps$\}$, and let $f$ be its characteristic function. The work of Babai, Fortnow, Levin, and Szegedy [BFLS] gives a probabilistic polynomial-time 'verifier' $V$ and a (deterministic) prover $P$ running in time $2^{cn}$ for a constant $c$ such that for every $x$ of length $n$ and every $b \in \{0, 1\}$,

1. If $f(x) = b$ and we set $\pi = P(x, b)$, then $\Pr\left[V^{\pi}(x, b) = 1\right] = 1$.
2. If $f(x) \neq b$, then for every $\pi^*$, $\Pr\left[V^{\pi^*}(x, b) = 1\right] \leq 1/4$.

The actual statements of the results in [BFLS] refer to a verifier that checks 'theorem-proof candidates' $(t, p)$ once they are encoded to allow efficient probabilistic verification. In their formulation, the verifier is given $t$ and runs in time $\mathrm{polylog}(|t|, |p|)$ and the prover (constructing the encoded proof) is given $t$ and $p$ and runs in time $\mathrm{poly}(t, p)$. Above, we apply this to the theorem $t = [f(x) = b]$ of length $n + 1$ together with its corresponding proof $p_0$ of length $2^{O(n)}$ (namely the computation of the universal TM). The construction of [BFLS] provides $t$ to the verifier as an oracle in an error-correcting format, but here our verifier has enough time to read $t$ in its entirety and so can simulate the theorem-oracle on its own. Our prover $P$ above will compute the original proof $p_0$ and then encode it via the construction of [BFLS].

Now we consider the function $f'(x, b, i) = P(x, b)_i$. By definition, the oracle queries of the verifier $V$ to the proof $P(x, b)$ can be replaced with oracle queries to the function $f'(x, b, i)$, where $|i| = O(|x|)$ (since $P(x, b)$ only has enough time to construct a proof of length $2^{O(|x|)}$). In addition, $f'$ is in $\mathbf{E}$, so $f'$ can be reduced to $f$ in linear time, so $V$'s queries to $P(x, b)$ can actually be replaced with queries to $f$ itself. This gives us the checker $C$ desired: $C^g(x)$ runs $V^g(x, g(x))$ and outputs $g(x)$ if this accepts, and `fail` otherwise. The queries of $C$ are all of length $O(|x|)$ and can be made the exactly the same length by padding. ■

Using this instance-checker in the proof of Theorem 5.7, we obtain:

**Theorem 5.9** *For every function* $f \in \mathbf{E}$*, there is a constant* $d$ *such that if* $f \notin \mathbf{BPTIME}(t(n)^d)$ *for a nice time bound* $t$*, then there is a function* $f' \in \mathbf{E}$ *such that*

$$(f', U) \notin \mathbf{HeurBPTIME}(t(n), 1/2 - 1/t(n)).$$

**Corollary 5.10** *If* $\mathbf{E} \not\subset \bigcap_{\epsilon>0} \mathbf{BPTIME}(2^{\epsilon n})$*, then*

$$\mathbf{E} \times \{U\} \not\subset \cap_{\epsilon>0} \mathbf{HeurBPTIME}(2^{\epsilon n}, 1/2 - 1/2^{\epsilon n}).$$

Finally, we observe that our reductions work on an input-length by input-length basis. That is, if every function in $\mathbf{E}$ can be computed on average for infinitely many input lengths, then every function in $\mathbf{E}$ can be computed in the worst-case for infinitely many input lengths. Equivalently, given a function in $\mathbf{E}$ that is worst-case hard for all but finitely many $n$, we can obtain a function in $\mathbf{E}$ that is average-case hard for all but finitely many $n$.

**Theorem 5.11** *For every function $f \in \mathbf{E}$, there is a constant $d$ such that if $f \notin$ **i.o.-BPTIME**$(t(n)^d)$ for a nice time bound $t$, then there is a function $f' \in \mathbf{E}$ such that*

$$(f', U) \notin \textbf{i.o.-HeurBPTIME}(t(n), 1/2 - 1/t(n)).$$

Recall that the techniques of [IW2] did not provide this kind of result (and instead only gave us Corollary 3.11), because the proof of Lemma 3.6 does not work on an input-length by input-length basis.

# 6 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 [IW2] and this paper. We suspect that these negative results can be extended to actually show that the constructions are nonrelativizing. The fact that we are using reductions that 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 [NW] and most subsequent papers on the subject) could be to show that given 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 $f$. Now, if $f$ is hard to compute 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 $f$. More formally, we would have the following notion of black-box construction of a pseudorandom generator from a hard predicate.

**Definition 6.1 (Black-Box Pseudorandom Generator Construction)** *Let $G^{()} : \{0,1\}^d \to \{0,1\}^m$ be an oracle algorithm that expects an oracle of the form $f : \{0,1\}^\ell \to \{0,1\}$.*

*We say that $G$ is a* black-box generator construction *with* distinguishing parameter $\epsilon$, reduction time $t$, *and* reduction advice $a$ *if there is an oracle algorithm $R$ (the* reduction*) that runs in time at most $t$ such that for every function $f : \{0,1\}^\ell \to \{0,1\}$ and every function $D : \{0,1\}^m \to \mathcal{D}$, if*

$$\Pr\left[D(G^f(U_n)) = 1\right] - \Pr\left[D(U_m) = 1\right] \geq \epsilon,$$

*then there exists an advice function $A : \{0,1\}^* \to \{0,1\}^a$ such that for every $x$,*

$$\Pr\left[R^D(x, U_r, A(U_r)) = f(x)\right] \geq \frac{3}{4}$$

*We say that $G$ is* uniform *if $a = 0$.*

In the taxonomy of [RTV], this definition captures a "fully black-box reduction," because both the construction ($f \mapsto G^f$) and the proof of correctness ($D \mapsto R^D$) are "black box" in their use of the hard function $f$ and adversary $D$, respectively. (Unlike the notion discussed in [RTV], we do not give $R$ oracle access to $f$, since we are in a setting where it is infeasible for the adversary to compute the function $f$.)

To make sense of the definition, note that if such a construction exists, and $f$ is not solvable in time $T$ using $a$ bits of advice, then the output of the generator fools every distinguisher that uses time at most $T/t$, up to an error of $\epsilon$.

As shown in [Tre1], pseudorandom generator constructions having this type of black-box analysis also have very nice information-theoretic properties. Specifically, they yield *randomness extractors* [NZ]. In particular, Lemma 3 of [Tre1] states that construction of [IW1] is a pseudorandom generator that meets the above definition, and Section 2.3 of [Tre1] shows how to view such a construction as an extractor.

In order to state this connection, we first recall the definition of extractors. A random variable $X$ ranging over $\{0,1\}^n$ is said to be a $k$-source if, for every $a \in \{0,1\}^n$ we have $\Pr[X = a] \leq 1/2^k$. Two random variables $X, Y$ ranging over $\{0,1\}^m$ are said to be *$\epsilon$-close* if for every function $T : \{0,1\}^m \to \{0,1\}$ we have

$$|\Pr[T(X) = 1] - \Pr[T(Y) = 1]| \leq \epsilon$$

A function $E : \{0,1\}^n \times \{0,1\}^d \to B^m$ is a $(k,\epsilon)$-*extractor* if, for every $k$-source $X$, $E(X, U_d)$ is $\epsilon$-close to $U_m$.

Suppose that we have black-box construction as in Definition 6.1. Define the function $E : \{0,1\}^{2^\ell} \times \{0,1\}^d \to \{0,1\}^m$ as follows: $E(x,z) = G^{f_x}(z)$, where $f_x : \{0,1\}^\ell \to \{0,1\}$ is the function whose truth-table is the string $x \in \{0,1\}^{2^\ell}$. Section 2.3 in [Tre1] shows that such a function is necessarily a $(k, 2\epsilon)$-extractor, where $k = a + \log 1/\epsilon$. (See also [Vio, Sha].)

It follows immediately from the definition that a $(k, 2\epsilon)$-extractor must satisfy $2^{k+d} \geq (1-2\epsilon) \cdot 2^m$. Taking $\epsilon = 1/4$, we see that the advice must satisfy $a \geq m - d - 3$. In particular, we cannot have a uniform (that is, $a = 0$) construction of a PRG that stretches by more 3 bits, and in fact the advice must grow linearly with the stretch. In fact, Radhakrishnan and Ta-Shma [RTS] have proven a stronger bound on extractors, showing that that $k + d \geq m + 2\log 1/\epsilon - O(1)$ (provided $k \leq n - O(1)$, $d \leq m-2$), which implies that we need advice that also grows as the error goes to zero. Specifically, we have $a \geq m - d + \log(1/\epsilon) - O(1)$ (provided $a + \log(1/\epsilon) < 2^\ell - 1$ and $d \leq m - 2$). Summarizing, we have:

**Proposition 6.2** *Let $G^{()} : \{0,1\}^d \to \{0,1\}^m$ be a black-box generator construction from functions $f : \{0,1\}^\ell \to \{0,1\}$, with distinguishing parameter $\epsilon$, reduction advice $a$. Then*

- *$a \geq m - d - 3$, provided $\epsilon \leq 1/4$, and*
- *$a \geq m - d + \log(1/\epsilon) - O(1)$, provided $d \leq m - 2$ and $a + \log(1/\epsilon) < 2^\ell - c$ for a universal constant $c$.*

We can do a similar argument for worst-case to average-case reductions. A black-box worst-case to average-case reduction is essentially what is called a "nice code" in [STV]. We give a precise definition below.

**Definition 6.3 (Black-box Worst-Case to Average-Case Reduction)** *A* black-box worst-case to average-case reduction *with* advantage parameter $\epsilon$, reduction time $t$, *and* reduction advice $a$, *is transformation $H$ that maps functions $f : \{0,1\}^n \to \{0,1\}$ into functions $H(f) : \{0,1\}^{n'} \to \{0,1\}$ and a probabilistic oracle randomized algorithm $R$ with advice such that for every function $P : \{0,1\}^{n'} \to \{0,1\}^n$, if*

$$\Pr[P(U_{n'}) = f'(U_{n'})] \geq \frac{1}{2} + \epsilon,$$

*then there is an advice function $A : \{0,1\}^* \to \{0,1\}^a$ such that for every $x$,*

$$\Pr[R^P(x, U_r, A(U_r)) = f(x)] \geq \frac{3}{4}$$

It is easy to see (cf. [Vio]) that $H$ can be thought of as an error-correcting code $C : \{0,1\}^N \to \{0,1\}^{2^{N'}}$, where $N = 2^n$ and $N' = 2^{n'}$, that is $(1/2 - \epsilon, 2^a)$-list-decodable, meaning that for every string $z \in \{0,1\}^{N'}$ there are at most $2^a$ strings $w \in \{0,1\}^N$ for which $C(w)$ and $z$ differ in at most $(1/2 - \epsilon) \cdot N'$ coordinates. If $a = 0$, then $H$ is uniquely decodable from a $1/2 - \epsilon$ fraction of error, something that is known to be impossible if $1/2 - \epsilon > 1/4$ and $n$ is large enough. Specifically, the Plotkin bound says that a code that is uniquely decodable from $(1+\delta)/4$ errors has message length $N \leq 1 + 1/\delta$. In addition, it is known that for a code to be $(1/2 - \epsilon, L)$-list decodable, we must have $|L| = \Omega(1/\epsilon^2)$ [Bli, GV] (provided $N \geq c \cdot (1/\epsilon^2) \cdot \log(1/\epsilon)$ for a certain constant $c$) and so $a \geq 2\log 1/\epsilon - O(1)$. Note that the black-box worst-case to average-case reduction of [STV] essentially matches this bound, achieving $a = O(\log 1/\epsilon)$. Summarizing, we have:

**Proposition 6.4** *Let $H$ be a black-box worst-case to average-case reduction mapping functions $f : \{0,1\}^n \to \{0,1\}$ into functions $H(f) : \{0,1\}^{n'} \to \{0,1\}$ advantage parameter $\epsilon$ and reduction advice $a$. Then,*

- $a > 0$, *provided that* $\epsilon < (1 - 1/(2^n - 1))/4$.
- $a \geq 2\log(1/\epsilon) - O(1)$, *provided that* $n \geq c\log(1/\epsilon)$, *where $c$ is a universal constant.*

# 7 Conclusions and Subsequent Work

The most immediate open problem is whether there is a "high-end" analogue of the Impagliazzo–Wigderson Theorem (Thm. 3.1). That is, can we get a polynomial-time or quasi-polynomial-time deterministic simulation of **BPP** from a uniform assumption on **E** or **EXP**, such as $\mathbf{E} \not\subseteq \cap_{\epsilon>0}\mathbf{BPTIME}(2^{\epsilon n})$ or $\mathbf{EXP} \not\subseteq \bigcap_{\epsilon>0}\mathbf{BPTIME}(2^{n^\epsilon})$? More generally, can we remove the $t(t(n))$ composition from Corollary 3.10? Recall that we have shown how to do this if **EXP** is replaced by **PSPACE** (Corollary 4.3), or if we are interested in only average-case hardness instead of derandomization (Thms. 5.7 and 5.9). By Lemma 3.5, it would suffice to show that there is an **EXP**-complete function $f$ such that if $f \in \mathbf{LEARN_{mem}}(t(n))$ then $f \in \mathbf{BPTIME}(t(\mathrm{poly}(n)))$. Perhaps instance-checkability and other properties relating to the PCP characterization of **EXP** will help here.

Subsequent to our work, Gutfreund, Shaltiel, and Ta-Shma [GST] obtained a "high-end," uniform derandomization of **AM**: if $\mathbf{E} \not\subseteq \cap_{\epsilon>0}\mathbf{AMTIME}(2^{\epsilon n})$, then $\mathbf{AM} \subseteq \mathbf{i.o.\text{-}PseudoNP}$. They also utilize the instance-checkability of **E**, together with a special property of the Miltersen–Vinodchandran [MV] hitting-set generator construction. Interestingly, no "low-end" version of their result (i.e. one based on $\mathbf{EXP} \neq \mathbf{AM}$) is known.

Instance checkability of **EXP** was also recently used by Barak [Bar] to establish a hierarchy theorem for probabilistic machines with a small amount of advice. The advice requirement in Barak's result was subsequently improved by Fortnow and Santhanam [FS] by making use of our construction of a **PSPACE**-complete self-correctible and downward self-reducible problem. (More recent improvements [GST, FST, vMP] have eliminated the need for our complete problem and for instance checking.) Our **PSPACE**-complete problem was also used by Allender et al. [ABK$^+$] to prove that the set of strings with high space-bounded Kolmogrov complexity is complete for **PSPACE** under zero-error randomized Cook reductions.

Our observation in Section 6 that the connections to extractors and list-decodable codes could be used to prove negative results about black-box pseudorandom generator constructions and hardness amplification was taken further by Viola [Vio] and Lu, Tsai, and Wu [LTW], who used these connections to prove negative results about doing such constructions in low complexity classes, such as the polynomial-time hierarchy.


## Acknowledgments

We thank Avi Wigderson, Valentine Kabanets, Lance Fortnow, Russell Impagliazzo, Madhu Sudan, and the anonymous CCC reviewers for helpful comments and discussions.


## References


[AFK] M. Abadi, J. Feigenbaum, and J. Kilian. On hiding information from an oracle. *Journal of Computer and System Sciences*, 39:21–50, 1989.

[ABK$^+$] E. Allender, H. Buhrman, M. Kouck'y, D. van Melkebeek, and D. Ronneburger. Power from Random Strings. In *Proceedings of the 43rd IEEE Symposium on Foundations of Computer Science*, pages 669–678, 2002.

[ACR] 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.

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

[BFNW] 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.

[BFLS] L. Babi, L. Fortnow, L. Levin, and M. Szegedy. Checking Computations in Polylogarithmic Time. In *Proceedings of the Twenty Third Annual ACM Symposium on Theory of Computing*, pages 21–31, New Orleans, Louisiana, 6–8 May 1991.

[Bar] B. Barak. A probabilistic-time hierarchy theorem for "slightly non-uniform" algorithms. In *Randomization and approximation techniques in computer science*, volume 2483 of *Lecture Notes in Comput. Sci.*, pages 194–208. Springer, Berlin, 2002.

[BF] 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.

[Bli] V. M. Blinovsky. Bounds for codes in the case of list decoding of finite volume. *Problems of Information Transmission*, 22(1):7–19, 1986.

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

[BLR] M. Blum, M. Luby, and R. Rubinfeld. Self-Testing/Correcting with Applications to Numerical Problems. *Journal of Computer and System Sciences*, 47(3):549–595, 1993.

[BM] M. Blum and S. Micali. How to Generate Cryptographically Strong Sequences of Pseudo-Random Bits. *SIAM Journal on Computing*, 13(4):850–864, Nov. 1984.

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

[CNS] J.-Y. Cai, A. Nerurkar, and D. Sivakumar. Hardness and hierarchy theorems for probabilistic quasi-polynomial time. In *Annual ACM Symposium on Theory of Computing (Atlanta, GA, 1999)*, pages 726–735 (electronic). ACM, New York, 1999.

[CPS] J.-Y. Cai, A. Pavan, and D. Sivakumar. On the Hardness of the Permanent. In *16th International Symposium on Theoretical Aspects of Computer Science*, Lecture Notes in Computer Science, Trier, Germany, March 4–6 1999. Springer-Verlag.

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

[FF] J. Feigenbaum and L. Fortnow. Random-Self-Reducibility of Complete Sets. *SIAM Journal on Computing*, 22(5):994–1005, Oct. 1993.

[FKN] J. Feigenbaum, S. Kannan, and N. Nisan. Lower bounds on random-self-reducibility. In *Proceedings of the 5th IEEE Conference on Structure in Complexity Theory*, pages 100–109, 1990.

[For] L. Fortnow. Comparing Notions of Full Derandomization. In *Proceedings of the Sixteenth Annual Conference on Computational Complexity*, pages 28–34. IEEE, June 18–21 2001.

[FS] L. Fortnow and R. Santhanam. Hierarchy theorems for probabilistic polynomial time. In *Proceedings of the 45th IEEE Symposium on Foundations of Computer Science*, pages 316–324, 2004.

[FST] L. Fortnow, R. Santhanam, and L. Trevisan. Hierarchies for semantic classes. In *Proceedings of the 37th ACM Symposium on Theory of Computing*, pages 348–355, 2005.

[Gol] O. Goldreich. On Promise Problems (in memory of Shimon Even, 1935–2004). Technical Report TR05-018, Electronic Colloquium on Computational Complexity, 2005. See June 2005 revision, available from author's homepage.

[GST] O. Goldreich, M. Sudan, and L. Trevisan. From logarithmic advice to single-bit advice. Technical Report TR04-093, ECCC, 2004.

[GZ] O. Goldreich and D. Zuckerman. Another proof that BPP $\subseteq$ PH (and more). *Electronic Colloquium on Computational Complexity* Technical Report TR97-045, September 1997. `http://www.eccc.uni-trier.de/eccc`.

[GV] V. Guruswami and S. Vadhan. A Lower Bound on List Size for List Decoding. In C. Chekuri, K. Jansen, J. D. P. Rolim, and L. Trevisan, editors, *Proceedings of the 8th International Workshop on Randomization and Computation (RANDOM '05)*, number 3624 in Lecture Notes in Computer Science, pages 318–329, Berkeley, CA, August 2005. Springer.

[GST] D. Gutfreund, R. Shaltiel, and A. Ta-Shma. Uniform hardness versus randomness tradeoffs for Arthur-Merlin games. *Computational Complexity*, 12(3-4):85–130, 2003.

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

[Imp1] 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.

[Imp2] R. Impagliazzo. A Personal View of Average-Case Complexity. In *Proceedings of the 10th IEEE Conference on Structure in Complexity Theory*, pages 134–147, 1995.

[IKW] R. Impagliazzo, V. Kabanets, and A. Wigderson. In search of an easy witness: exponential time vs. probabilistic polynomial time. *Journal of Computer and System Sciences*, 65(4):672–694, 2002.

[ISW] R. Impagliazzo, R. Shaltiel, and A. Wigderson. Extractors and Pseudo-Random Generators with Optimal Seed Length. In *Proceedings of the Thirty-second Annual ACM Symposium on Theory of Computing*, pages 1–10, Portland, Oregon, May 2000. See also ECCC TR00-009.

[IW1] 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.

[IW2] 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.

[Kab] V. Kabanets. Easiness assumptions and hardness tests: trading time for zero error. *Journal of Computer and System Sciences*, 63(2):236–252, 2001.

[KL] 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.

[Lip] R. Lipton. New Directions in Testing. In *Proceedings of DIMACS Workshop on Distributed Computing and Cryptography*, 1989.

[Lu] C.-J. Lu. Derandomizing Arthur-Merlin games under uniform assumptions. In *Algorithms and computation (Taipei, 2000)*, pages 302–312. Springer, Berlin, 2000.

[LTW] C.-J. Lu, S.-C. Tsai, and H.-L. Wu. On the Complexity of Hardness Amplification. In *Proceedings of the 20th Annual IEEE Conference on Computational Complexity (CCC '05)*, pages 170–182, San Jose, 11–15 June 2005. IEEE.

[LFKN] C. Lund, L. Fortnow, H. Karloff, and N. Nisan. Algebraic Methods for Interactive Proof Systems. *Journal of the ACM*, 39(4):859–868, Oct. 1992.

[MV] P. B. Miltersen and N. V. Vinodchandran. Derandomizing Arthur-Merlin games using hitting sets. In *40th Annual Symposium on Foundations of Computer Science (New York, 1999)*, pages 71–80. IEEE Computer Soc., Los Alamitos, CA, 1999.

[MVW] 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.

[NW] N. Nisan and A. Wigderson. Hardness vs Randomness. *Journal of Computer and System Sciences*, 49(2):149–167, Oct. 1994.

[NZ] N. Nisan and D. Zuckerman. Randomness is Linear in Space. *Journal of Computer and System Sciences*, 52(1):43–52, Feb. 1996.

[RTS] J. Radhakrishnan and A. Ta-Shma. Bounds for dispersers, extractors, and depth-two superconcentrators. *SIAM Journal on Discrete Mathematics*, 13(1):2–24 (electronic), 2000.

[RTV] O. Reingold, L. Trevisan, and S. Vadhan. Notions of Reducibility between Cryptographic Primitives. In M. Naor, editor, *Proceedings of the First Theory of Cryptography Conference (TCC '04)*, volume 2951 of *Lecture Notes in Computer Science*, pages 1–20. Springer-Verlag, 19–21 February 2004.

[Sha] R. Shaltiel. Recent Developments in Extractors. In G. Paun, G. Rozenberg, and A. Salomaa, editors, *Current Trends in Theoretical Computer Science*, volume 1: Algorithms and Complexity. World Scientific, 2004.

[SU] R. Shaltiel and C. Umans. Simple Extractors for All Min-Entropies and a New Pseudo-Random Generator. *Journal of the ACM*, 52(2):172–216, 2005.

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

[Sho] V. Shoup. New algorithms for finding irreducible polynomials over finite fields. *Mathematics of Computation*, 54(189):435–447, 1990.

[Sip] M. Sipser. *Introduction to the Theory of Computation.* Course Technology, 2nd edition, 2005.

[STV] M. Sudan, L. Trevisan, and S. Vadhan. Pseudorandom Generators without the XOR Lemma. *Journal of Computer and System Sciences*, 62:236–266, 2001.

[Tod] S. Toda. PP is as hard as the polynomial-time hierarchy. *SIAM Journal on Computing*, 20(5):865–877, 1991.

[Tre1] L. Trevisan. Extractors and pseudorandom generators. *Journal of the ACM*, 48(4):860–879 (electronic), 2001.

[Tre2] L. Trevisan. List Decoding Using the XOR Lemma. In *Proceedings of the 44th IEEE Symposium on Foundations of Computer Science*, pages 126–135, Cambridge, MA, October 2003.

[TV] L. Trevisan and S. Vadhan. Pseudorandomness and Average-Case Complexity via Uniform Reductions. In *Proceedings of the 17th Annual IEEE Conference on Computational Complexity (CCC '02)*, pages 129–138, Montréal, CA, May 2002. IEEE.

[Uma] C. Umans. Pseudo-random generators for all hardnesses. *Journal of Computer and System Sciences*, 67(2):419–440, 2003.

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

[vMP] D. van Melkebeek and K. Pervyshev. A Generic Time Hierarchy for Semantic Models With One Bit of Advice. Manuscript, 2005.

[Vin] N. V. Vinodchandran. $\mathrm{AM}_{\mathrm{exp}} \not\subseteq (\mathrm{NP} \cap \mathrm{coNP})/\mathrm{poly}$. *Information Processing Letters*, 89(1):43–47, 2004.

[Vio] E. Viola. The Complexity of Constructing Pseudorandom Generators from Hard Functions. *Computational Complexity*, 13(3-4):147–188, 2004.

[Yao] 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.