Electronic Colloquium on Computational Complexity, Report No. 7 (2008)

Limitations of Hardness vs. Randomness under Uniform Reductions

Dan Gutfreund *
Math Department and CSAIL
Massachusetts Institute of Technology

Salil Vadhan†
School of Engineering and Applied Sciences
Harvard University

February 5, 2008

Abstract

We consider (uniform) reductions from computing a function f to the task of distinguishing the output of some pseudorandom generator G from uniform. Impagliazzo and Wigderson [IW] and Trevisan and Vadhan [TV] exhibited such reductions for every function f in PSPACE. Moreover, their reductions are "black box," showing how to use *any* distinguisher T, given as oracle, in order to compute f (regardless of the complexity of T). The reductions are also adaptive, but only *mildly* (queries of the same length do not occur in different levels of adaptivity). Impagliazzo and Wigderson [IW] also exhibited such reductions for every function f in EXP, but those reductions are not black-box, because they only work when the oracle T is computable by small circuits.

Our main results are that:

- *Nonadaptive* black-box reductions as above can only exist for functions f in $\mathrm{BPP}^{\mathrm{NP}}$ (and thus are unlikely to exist for all of PSPACE).
- *Mildly adaptive* black-box reductions as above can only exist for functions f in PSPACE (and thus are unlikely to exist for all of EXP).

Keywords: pseudorandom generators, derandomization, black-box reductions

*Research was partially supported by NSF grant CCF-0514167. Part of this research was done while the author was at Harvard University and supported by ONR grant N00014-04-1-0478 and NSF grant CNS-0430336.

†Work done in part while visiting UC Berkeley, supported by the Miller Institute for Basic Research in Science and the Guggenheim Foundation. Also supported by ONR grant N00014-04-1-0478 and US-Israel BSF grant 2006060.

ISSN 1433-8092

1 Introduction

The hardness versus randomness paradigm, first developed by Blum, Micali, Yao, Nisan, and Wigderson [BM, Yao, NW], is one of the most exciting achievements of the field of computational complexity. It shows how to use the hardness of a function f (computable in exponential time) to construct a pseudorandom generator G, which can then be used to derandomize probabilistic algorithms. By now there are many varieties of such results, trading off different assumptions on the function f, different types of probabilistic algorithms (e.g. BPP algorithms or AM proof systems), and different levels of derandomization.

For many years, all of the results of this type (based on the hardness of an arbitrary exponential-time computable function) required the function f to be hard for even nonuniform algorithms, e.g. $f \notin \mathrm{P/poly}$. Nearly a decade ago, Impagliazzo and Wigderson [IW] overcame this barrier, showing how to construct pseudorandom generators assuming only the existence of an exponential-time computable function f that is hard for uniform probabilistic algorithms, i.e. assuming $\mathrm{EXP} \neq \mathrm{BPP}$. This result and subsequent work [Kab, IKW, TV, KI, GST1, SU2] have raised the hope that we may be able to prove an equivalence between uniform and nonuniform hardness assumptions, or even obtain unconditional derandomization and new lower bounds.

The work of Impagliazzo and Wigderson [IW], as well as the subsequent ones on derandomization from uniform assumptions, have used a number of ingredients that were not present in earlier works on hardness vs. randomness. In this paper, following [TV], we explore the extent to which these new ingredients are really necessary. The hope is that such an understanding will help point the way to even stronger results,[1] and also highlight techniques that might be used to overcome barriers in other parts of complexity theory. We now describe the new ingredients introduced by Impagliazzo and Wigderson [IW].

Black-box reductions. Classic results on hardness vs. randomness can be formulated as "black box" constructions. That is, they are obtained by providing two efficient oracle algorithms G and R. The *construction* G uses oracle access to a (supposedly hard) function f to compute a *generator* G^f, which stretches a short seed to a long sequence of bits. The *reduction* R is meant to show that the output of G^f is pseudorandom if the function f is hard. Specifically, we require that for every statistical test T that distinguishes the output of G^f from uniform, there exists an "advice string" z such that $R^T(z, \cdot)$ computes the function f. Note that if T is efficient, then by hardwiring z, we obtain a small circuit computing f. Put in the contrapositive, this says that if f cannot be computed by small circuits, then there cannot exist an efficient test T distinguishing the output of G^f from uniform.

Note that the above notion requires both the construction G and the reduction R to be black box, and requires that they work for every function f and statistical test T, regardless of the complexity of f and T. In the taxonomy of [RTV], these are referred to as *fully black-box* constructions. The advice string z that we provide to the reduction R is what makes the reduction *nonuniform*, and thereby require a nonuniform hardness assumption on the function f to deduce that G^f is pseudorandom. If the advice string could be eliminated, then then we would immediately get results based on uniform assumptions, like those of [IW]. Unfortunately, as shown in [TV], it is impossible to have a fully black-box construction of

[1] A seemingly modest but still elusive goal is a "high-end" version of [IW], whereby one can construct a pseudorandom generator with exponential stretch from the assumption that EXP does not have subexponential-time probabilistic algorithms.

a pseudorandom generator without a significant amount of advice. Thus the Impagliazzo–Wigderson construction necessarily deviates from the fully black-box framework.

The most obvious way in which the Impagliazzo–Wigderson [IW] construction is not fully black box is that the construction is not proven to work for every function f, and rather it makes use of the fact that f is in EXP or some other complexity class such as $\mathrm{P}^{\#\mathrm{P}}$ or PSPACE [TV]. For example, in the case of $\mathrm{P}^{\#\mathrm{P}}$ or PSPACE, it uses the fact that f can be reduced to a function f' that is both downward self-reducible and self-correctible (e.g. f' is the PERMANENT), which is then used to construct the pseudorandom generator. That is, the construction algorithm G is not black box. Whether the Impagliazzo–Wigderson reduction algorithm R is or is not black box (i.e. works for every test T given as oracle) depends on which class f is taken from. For functions in $\mathrm{P}^{\#\mathrm{P}}$ or PSPACE, R is black box. But if we are only given a function in EXP, then the reduction relies on the fact that the test T is efficiently computable. Another interesting aspect of the reduction R is that it makes *adaptive* queries to the statistical test T, whereas earlier reductions in this area were nonadaptive. (There are subsequent reductions, due to Shaltiel and Umans [SU1, Uma], that are also adaptive.)

Our results. Our main results provide evidence that some of these new ingredients are necessary. Specifically, we consider arbitrary (non-black-box) constructions of a pseudorandom generator G from a function f, and uniform reductions R (i.e. with no advice) from computing f to distinguishing the output of G from uniform. For simplicity, we also assume that the generator G is computable in time exponential in its seed length and that it stretches by a factor of at least 4. (More general statements are given in the body of the paper.)

Our first result shows that adaptivity is likely to be necessary unless we assume the function is in PH (rather than PSPACE or EXP).

Theorem 1.1 (informal)**.** *If there is a nonadaptive, uniform, black-box reduction R from distinguishing a generator G to computing a function f, then f is in* $\mathrm{BPP}^{\mathrm{NP}}$*.*

Next, we consider reductions R that are only *mildly adaptive* in the sense that all the queries of a particular length can be made simultaneously, but they may depend on answers of the statistical test on queries of other lengths. (We call this *1-adaptive* later in the paper, as a special case of a more general notion.) The Impagliazzo–Wigderson reduction for functions f in $\mathrm{P}^{\#\mathrm{P}}$ or PSPACE is mildly adaptive. We show that this property is unlikely to extend to EXP.

Theorem 1.2 (informal)**.** *If there is a mildly adaptive, uniform, black-box reduction R from distinguishing a generator G to computing a function f, then f is in* PSPACE*.*

Thus, to obtain a result for arbitrary functions f in EXP, the reduction must either be non-black-box or "more adaptive." Impagliazzo and Wigderson exploit the former possibility, giving a non-black-box reduction, and their method for doing so turns out to have a substantial price — a statistical test running in time $t(n)$ yields an algorithm computing f that runs in time roughly $t(t(n))$, rather than something polynomially related to t, which is what is needed for a "high end" result. (See [TV].) Theorem 1.2 suggests that their result might be improved by employing reductions with greater adaptivity, such as [SU1, Uma]. Alternatively, it would be interesting to rule out such an improvement by strengthening Theorem 1.2 to hold for arbitrary adaptive reductions.

Finally, we consider "how non-black-box" the Impagliazzo–Wigderson reduction is for EXP. Specifically, we observe that even though the analysis of the reduction R relies on the fact that T is efficient (i.e. computable by small size circuits), the reduction itself only needs oracle access to T (i.e., it does not need the description of the circuits). We call such reductions *size-restricted black-box reductions*. Reductions of this type were recently studied by Gutfreund and Ta-Shma [GT].[2] They exhibited such a reduction (based on [GST2]) for a worst-case/average-case connection that cannot be established via a standard black-box reduction. Theorem 1.2, together with Theorem 1.3 below (which is implicit in [IW]), provides another example of a size-restricted black-box reduction that bypasses black-box limitations. For technical reasons, we state the [IW] result in terms of *hitting-set generators*, which are a natural weakening of pseudorandom generators that suffice for derandomizing probabilistic algorithms with 1-sided error (i.e. RP rather than BPP). Theorems 1.1 and 1.2 above can be strengthened to apply also to hitting-set generators.

Theorem 1.3 (implicit in [IW], informal)**.** *For every function f in* EXP*, there is a generator G and a mildly adaptive, uniform, size-restricted black-box reduction from distinguishing G as a hitting set to computing f.*

A final result of ours is an "infinitely-often" version of the Impagliazzo–Wigderson reduction [IW]. The original versions of their reductions are guaranteed to compute f correctly on *all* input lengths assuming that the statistical test T successfully distinguishes the generator on *all* input lengths. Unlike most other results in the area, it is not known how to obtain reductions that compute f correctly on infinitely many input lengths when the test T is only guaranteed to succeed on infinitely many input lengths. We observe that such a result can be obtained for constructing *hitting-set generators* (and derandomizing RP) from hard problems in PSPACE rather than constructing pseudorandom generators (and derandomizing BPP) from hard problems in EXP as done in [IW].

Perspective. As discussed above, one motivation for studying the limitations of black-box reductions is to help identify potential approaches to overcoming apparent barriers. Another motivation is that black-box reductions sometimes have advantages over non-black-box reductions, and thus it is informative to know when these advantages cannot be achieved. For example, Trevisan's realization that fully black-box constructions of pseudorandom generators yield *randomness extractors* [Tre] yielded substantial benefits for both the study of pseudorandom generators and extractors. Similarly, Klivans and van Melkebeek [KvM] observed that black-box constructions of pseudorandom generators extend naturally to derandomize classes other than BPP, such as AM.

Unfortunately, results showing the limitations of black-box reductions are often interpreted as saying that proving certain results (e.g. such as worst-case/average-case connections for NP, for which nonadaptive black-box reductions were ruled out in [BT]) are outside the reach of "current techniques". We strongly disagree with these kinds of interpretations, and indeed hope that our results together with [IW] will serve as another reminder that such limitations can be overcome.

[2] There are subtle differences between the reductions that we consider and the ones in [GT], see Remark 2.5.

2 Preliminaries

For $n \in \mathbb{N}$, we denote by U_n the uniform distribution over $\{0,1\}^n$. For a distribution D, we denote by $x \leftarrow D$ that x is a sample drawn from D. For two distributions D_1, D_2 on a discrete space Γ, we denote by $\Delta(D_1, D_2)$ there statistical difference. I.e.,

$$\Delta(D_1, D_2) = \max_{S \subset \Gamma} |\Pr_{x \leftarrow D_1}[x \in S] - \Pr_{y \leftarrow D_2}[y \in S]|$$

2.1 Complexity Classes

We assume that the reader is familiar with standard complexity classes such as EXP, BPP, the polynomial-time hierarchy etc., as well as with standard models of computation such as probabilistic Turing Machines, and Boolean circuits.

For class $\mathcal{C}$ of *algorithms,* we let $io-\mathcal{C}$ be the class of languages such that an algorithm from $\mathcal{C}$ correctly decides L for infinitely many input lengths.

For a a time bound $T = T(n)$ and $\epsilon = \epsilon(n)$, we define the complexity class $HeurTIME_\epsilon(T)$ to be the class of languages L for which there is a deterministic algorithm A that runs in time $T(n)$ on instances of size n and for every large enough n,

$$\Pr_{x \leftarrow U_n}[A(x) \neq L(x)] \leq \epsilon(|x|)$$

2.2 Pseudorandom Generators and Hardness vs. Randomness

Definition 2.1. Let $b : \mathbb{N} \to \mathbb{N}$ be such that for every a, $b(a) > a$. Let $\mathcal{G} = \{G_a : \{0,1\}^a \to \{0,1\}^{b(a)}\}_{a \in \mathbb{N}}$ be a sequence of functions, and let $\mathcal{T} = \{T : \{0,1\}^* \to \{0,1\}\}$ be a family of Boolean functions (which we call *statistical tests*). For $\delta > 0$ we say that,

1. $\mathcal{G}$ is a sequence of *pseudorandom generators (PRGs for short) that δ-fools $\mathcal{T}$* i.o. (infinitely often), if for every $T \in \mathcal{T}$, there are infinitely many $a \in \mathbb{N}$ such that

$$|\Pr_{y \leftarrow U_a}[T(G_a(y)) = 1] - \Pr_{x \leftarrow U_{b(a)}}[T(x) = 1]| < \delta \tag{1}$$

2. $\mathcal{G}$ is a sequence of *hitting-set generators (HSGs for short) that δ-hits $\mathcal{T}$* i.o., if for every $T \in \mathcal{T}$, there are infinitely many $a \in \mathbb{N}$ such that

$$\Pr_{x \leftarrow U_{b(a)}}[T(x) = 0] \geq \delta \Rightarrow \Pr_{y \leftarrow U_a}[T(G_a(y)) = 0] > 0 \tag{2}$$

If a function $T : \{0,1\}^* \to \{0,1\}$ violates (1) (respectively (2)), we say that it δ-*distinguishes* $\mathcal{G}$ from uniform a.e. (almost everywhere).

δ-fooling (respectively δ-hitting) a.e. and δ-distinguishing i.o. are defined analogously with the appropriate changes in the quantification over input lengths.

Note that if G is a PRG that δ-fools $\mathcal{T}$ i.o. (respectively a.e.) then it is also a HSG that δ-hits $\mathcal{T}$ i.o. (respectively a.e.).

Definition 2.2. A (uniform) *black-box reduction* from deciding a language L to δ-distinguishing a.e. a family of (either pseudorandom or hitting-set) generators $\mathcal{G} = \{G_a : \{0,1\}^a \to \{0,1\}^{b(a)}\}_{a\in\mathbb{N}}$, is a probabilistic polynomial-time oracle Turing Machine (TM) R, such that for every statistical test T that δ-distinguishes $\mathcal{G}$ a.e., for every large enough $n \in \mathbb{N}$ and for every $x \in \{0,1\}^n$,

$$\Pr[R^T(x) = L(x)] > 2/3$$

where the probability is over the random coins of R, and $R^T(x)$ denotes the execution of R on input x and with oracle access to T.

We say that such a reduction asks *single-length* queries if for every n, there exist $a = a(n)$ such that on every execution of R on instances of length n, all the queries that R makes are of length exactly $b(a)$.

We say that the reduction has $k = k(n)$ *levels of adaptivity* if on every execution of R on inputs of length n and every statistical test T, the queries to T can be partitioned to $k+1$ subsets (which are called the levels of adaptivity), such that each query in the i'th set is a function of the input x, the randomness of R, the index of the query within the i'th set (as well as i itself), and the answers that T gives on queries in the sets $1, \ldots, i-1$. We say that a reduction is nonadaptive if it has zero levels of adaptivity.

Finally, we say that the reduction is $k(a,n)$-*adaptive* if for every statistical test T, every instance of length n and every a, there are at most $k(a,n)$ levels of adaptivity in which queries of length $b(a)$ appear with positive probability (over the randomness of R when it is given oracle access to T).

We now define a different notion of reductions that still only have oracle access to the distinguishers, however the correctness of the reduction is only required to hold when the distinguisher is restricted to be a function that is computable by polynomial-size circuits.

Definition 2.3. A (uniform) *size-restricted black-box reduction* from deciding a language L to δ-distinguishing a.e. a family of (pseudorandom or hitting-set) generators $\mathcal{G} = \{G_a : \{0,1\}^a \to \{0,1\}^{b(a)}\}_{a\in\mathbb{N}}$, is a probabilistic polynomial-time oracle TM R, such that for every statistical test T that δ-distinguishes $\mathcal{G}$ a.e., *and is computable by a sequence of quadratic-size circuits*[3], for every large enough $n \in \mathbb{N}$ and for every $x \in \{0,1\}^n$,

$$\Pr[R^T(x) = L(x)] > 2/3$$

where the probability is over the random coins of R.

Quantifiers over query length and adaptivity are defined as in the black-box case.

Remark 2.4. The quadratic bound on the circuit size of the distinguishers is arbitrary and can be any (fixed) polynomial. The reason for our quadratic choice is that restricting the attention to distinguishers of this size is enough for derandomization. In Section 5, we explain why the reduction of [IW] is size-restricted black-box reduction according to Definition 2.3. We stress though that it is in fact a size-restricted black-box reduction with respect to any (fixed) polynomial bound on the size of the distinguishers.

Remark 2.5. The restricted black-box reductions that we consider here run in any arbitrary polynomial time bound, which in particular can be larger than the fixed (quadratic) polynomial bound on the size of the distinguishers. In contrast, the notion of *class-specific*

[3]Recall that the distinguishers' circuit size is measured with respect to their input length, which is the output length of the generator.

black-box reductions defined in [GT], considers reductions that run in a fixed polynomial-time that is independent of the running time (or the circuit size) of the oracle (i.e. the oracle function can be computed by algorithms that run in arbitrary polynomial time).

We need the following hardness vs. randomness result.

Theorem 2.6. *[BFNW] For every constant $\delta > 0$, there exist constants $\epsilon < \delta$ and $d \in \mathbb{N}$ such that for every $1 \leq k \leq n^\epsilon$, for any functions $f : \{0,1\}^k \to \{0,1\}$ and $T : \{0,1\}^n \to \{0,1\}$, there is a function $G^f : \{0,1\}^{n^\delta} \to \{0,1\}^n$ that is computable in polynomial-time with oracle access to f, as well as a function $R^T : \{0,1\}^{n^d} \times \{0,1\}^k \to \{0,1\}$ that is computable in probabilistic polynomial-time with oracle access to T, such that if T $\frac{1}{n}$-distinguishes G^f from uniform then there is $a \in \{0,1\}^{n^d}$ such that $\Pr[R^T(a,x) = f(x)] > 1 - 1/k^2$, where the probability is over the randomness of R and x chosen uniformly from $\{0,1\}^k$.*

Note that the above theorem applies to every $1 \leq k \leq n^\epsilon$. Typically, only $k = n^\epsilon$ is of interest because it implies the weakest hardness assumption. We, however, use the fact that it also works for $k < n^\epsilon$ (see Section 6). The fact that the theorem is true for all $k < n^\epsilon$, follows trivially from the case $k = n^\epsilon$. This is because we can view any function on k bits as a function on n^ϵ bits simply by ignoring the last $n^\epsilon - k$ bits of the input.

Observe that if f is self-correctible (in the sense defined below), we can modify the reduction in Theorem 2.6 to compute f on *every* input correctly with high probability.

Definition 2.7. A function f is *self-correctible* if there is a probabilistic polynomial-time oracle TM C such that for every large enough n, for every $g_n : \{0,1\}^n \to \{0,1\}$ for which $\Pr_{x \leftarrow U_n}[g_n(x) = f(x)] > 1 - 1/n$, and for every $x \in \{0,1\}^n$:

$$\Pr[C^{g_n}(x) = f(x)] > 2/3$$

where the probability is over the randomness of C.

2.3 Approximate counting

We will need the following variant of the *approximate counting* problem: given a circuit C with input length $n \leq |C|$ and a parameter $0 < \epsilon < 1$, output r such that,

$$(1-\epsilon) \Pr_{x \leftarrow U_n}[C(x) = 1] \leq r \leq (1+\epsilon) \Pr_{x \leftarrow U_n}[C(x) = 1]$$

Theorem 2.8. *[Sip, Sto, JVV] There is a probabilistic algorithm that uses an* NP*-oracle, runs in time poly$(|C|, \epsilon^{-1}, \log(\delta^{-1}))$, and solves the approximate counting problem with success probability $1 - \delta$.*

3 Nonadaptive Reductions

In this section we show that any black-box nonadaptive reduction from deciding a language L to distinguishing a generator implies that L is in the polynomial-time hierarchy.

Theorem 3.1. *Let $L \subseteq \{0,1\}^*$ be a language, and let $\mathcal{G} = \{G_a : \{0,1\}^a \to \{0,1\}^{b(a)}\}_{a \in \mathbb{N}}$ be a family of hitting-set generators such that G_a is computable in time $2^{O(a)}$, and $b(a) > 4a$. If there is a nonadaptive black-box reduction R from L to $\frac{1}{2}$-distinguishing $\mathcal{G}$ a.e., then L is in* $\mathrm{BPP}^{\mathrm{NP}}$*. If we remove the time bound condition on computing G_a then L is in* $\mathrm{P}^{\mathrm{NP}}/poly$*.*

Proof outline: Let us concentrate on the single-length case. We describe a $\mathrm{BPP}^{\mathrm{NP}}$ algorithm that decides L. Fix an input $x \in \{0,1\}^n$, and let $a \in \mathbb{N}$ be such that R queries its oracle on instances of length $b = b(a)$ when given inputs of length n.

The basic idea is to define, based on x, a statistical test T (that may not be efficiently computable) that has the following properties:

1. T $\frac{1}{2}$-distinguishes G_a. This means that R^T decides L correctly on every instance of length n.

2. There is a function T' that can be computed in $\mathrm{BPP}^{\mathrm{NP}}$, such that R^T and $R^{T'}$ behave almost the same on the input x. This means that $R^{T'}$ decides correctly the membership of x in L (since so does R^T), but now the procedure *together* with the oracle computations can be implemented in $\mathrm{BPP}^{\mathrm{NP}}$.

To construct such a T, we classify queries that R makes on input x, according to the probability that they come up in the reduction (where the probability is over R's randomness).[4] We call a query *heavy* if the probability it comes up is at least 2^{-t} and *light* otherwise, where t is the average of a and $b = b(a)$. Note that the classification to heavy/light is well defined and is *independent* of any oracle that R may query, because the reduction is nonadaptive. We define T to be 1 on heavy queries as well as on elements in the image of G_a, and 0 otherwise.

First, we argue that T $\frac{1}{2}$-distinguishes G_a. This is because clearly it is always 1 on a sample taken by G_a. On the other hand, the number of elements for which T is 1 is small relative to the universe $\{0,1\}^b$. This is because there are only 2^a elements in the image of G_a, and at most 2^t heavy elements. Recall that both a and t are smaller than b.

We define T' to be 1 on heavy elements and 0 otherwise. Note that the only difference between T and T' is on light elements in the image set of G_a (T gives them the value 1, while T' gives them the value 0). When we run R on input x, the probability that such elements appear is small because their number is small (at most 2^a) and each one appears with small probability (because it is light). So R, on input x, behaves roughly the same when it has oracle access to either T or T'. We therefore conclude that $R^{T'}$ decides correctly the membership of x in L.

Now to show that T' is computable in $\mathrm{BPP}^{\mathrm{NP}}$, we use Theorem 2.8 to approximate the weight of queries made by R and thus simulate its run with the oracle T'. Since for every query we only get an approximation of its weight, we cannot handle a sharp threshold between heavy and light queries. To that end, instead of defining the threshold t to be the average of a and b, we define two thresholds (both are a weighted average of a and b), such that light queries are those below the low threshold, heavy are those above the high threshold, and undetermined queries in between. We now need more subtle definitions of T and T', but still the outline described above works.

Proof. [Of Theorem 3.1] For simplicity, we prove the theorem for the case that R makes single-length queries. At the end we explain how the proof can be modified to handle different lengths.

Fix an input length n and fix an instance $x \in \{0,1\}^n$. Let $a \in \mathbb{N}$ be such that R queries its oracle on instances of length $b = b(a)$ when given inputs of length n. Let $k = k(n) = \mathrm{poly}(n)$ be the number of queries that R makes on instances of length n. We may assume that $10 \log k < a$. Otherwise, since $a = O(\log k) = O(\log n)$, we can hardwire

[4] A similar idea appears in [BT].

all the outputs of G_a into R, and we get a polynomial-size circuit that does not need any oracle and decides the language L. If G_a is computable in time $2^{O(a)}$, we can compute the outputs of the generator in time poly(n), and hence place L in BPP. Let

$$\mathrm{Im}(G_a) \stackrel{def}{=} \{q \in \{0,1\}^b : \exists y \in \{0,1\}^a \text{ s.t. } G_a(y) = q\}$$

We call $q \in \{0,1\}^b$ *heavy* if

$$\Pr[R(x) \text{ asks the query } q] > 2^{-(\frac{b}{3}+\frac{2a}{3})},$$

where the probability is over the random coins of R. We call q *light* if

$$\Pr[R(x) \text{ asks the query } q] < 2^{-(\frac{2b}{3}+\frac{a}{3})}$$

If neither of the two cases above hold we say that q is *undetermined*. (Recall that R is nonadaptive, so light/heavy/undetermined does not depend on the oracle.) Define the sets $H \stackrel{def}{=} \{q \in \{0,1\}^b : q \text{ is heavy}\}$, $F \stackrel{def}{=} \{q \in \{0,1\}^b : q \text{ is light}\}$ and $E \stackrel{def}{=} \{q \in \{0,1\}^b : q \text{ is undetermined}\}$.

Consider the family of statistical tests $\mathcal{T} = \{T : \{0,1\}^b \to \{0,1\}\}$, consisting of statistical tests $T : \{0,1\}^b \to \{0,1\}$ satisfying:

$$T(q) \quad = \quad \begin{cases} 1 & \text{if } q \in H \cup \mathrm{Im}(G_a) \\ 0 & \text{if } q \in F \setminus \mathrm{Im}(G_a) \end{cases}$$

Note that if $q \in E$, $T(q)$ may take an arbitrary 0/1 value (hence we get a family of tests and not just one). Also note that since we now only handle the single-length case, it is enough to define the function T only on the domain $\{0,1\}^b$.

Claim 3.2. *For every $T \in \mathcal{T}$, T $\frac{1}{2}$-distinguishes G_a.*

Proof. By the definition of $\mathcal{T}$, $\Pr_{y \leftarrow U_a}[T(G_a(y)) = 1] = 1$. On the other hand,

$$\begin{aligned} \Pr_{y' \leftarrow U_b}[T(y') = 1] &\leq \Pr_{y' \leftarrow U_b}[y' \in H \cup E \cup \mathrm{Im}(G_a)] \\ &\leq k \cdot 2^{(\frac{2b}{3}+\frac{a}{3})} \cdot 2^{-b} + 2^a \cdot 2^{-b} \\ &\leq 2k \cdot 2^{-(\frac{b-a}{3})} \\ &< 1/2 \end{aligned}$$

The second inequality follows from the fact that there are at most $2^{(\frac{2b}{3}+\frac{a}{3})}$ non-light elements, and there are at most k queries in which such an element can appear. In the last inequality we use the fact that $\log k < a/10$, and the fact that $b > 4a$. □

Corollary 3.3. *For every $T \in \mathcal{T}$, $\Pr[R^T(x) = L(x)] > 2/3$, where the probability is over the random coins of R.*

Define a family of functions $\mathcal{T}' = \{T' : \{0,1\}^b \to \{0,1\}\}$ each taking the following form:

$$T'(q) \quad = \quad \begin{cases} 1 & \text{if } q \in H \\ 0 & \text{if } q \in F \end{cases}$$

Claim 3.4. *For every $T' \in \mathcal{T}'$, exists $T \in \mathcal{T}$, such that $\Delta(R^T(x), R^{T'}(x)) \leq 2^{-\frac{b-a}{3}}$*

Proof. We take T to be such that on every $q \in H \cup \mathrm{Im}(G_a)$, $T(q) = 1$, on every $q \in F$, $T(q) = 0$, and otherwise T agrees with T'. Note that the answers of T and T' only differ on light or undetermined elements in $\mathrm{Im}(G_a)$. So,

$$\begin{aligned}\Delta(R^T(x), R^{T'}(x)) &\leq \Pr[R \text{ queries an element in } (F \cup E) \cap \mathrm{Im}(G_a)] \\ &\leq 2^a \cdot 2^{-(\frac{b}{3}+\frac{2a}{3})} \\ &= 2^{-(\frac{b-a}{3})}\end{aligned}$$

□

From Corollary 3.3 and Claim 3.4 we obtain:

Corollary 3.5. $\Pr[R^{T'}(x) = L(x)] > 3/5$*, where the probability is over the random coins of R.*

We are now ready to describe a $\mathrm{BPP}^{\mathrm{NP}}$ algorithm that decides L on every input. On input $x \in \{0,1\}^n$, we run R to generate queries $q_1, \ldots, q_k \in \{0,1\}^b$. We now simulate the oracle used by R as follows: For every $1 \leq i \leq k$, estimate the probability that q_i is generated by $R(x)$, using the approximate counting procedure from Theorem 2.8 with $\delta = 2^{-n}$. If the estimate it at most $2^{-\frac{b+a}{2}}$ we let the answer of the oracle on q_i to be 0, and otherwise 1.

By Theorem 2.8 the procedure above can be implemented in $\mathrm{BPP}^{\mathrm{NP}}$. We now argue correctness. For every i, if q_i is heavy then with probability at least $1-2^{-n}$ its estimate will be at least $2^{-\frac{b+a}{2}}$ and the "oracle" will give it the value 1. If q_i is light then with probability at least $1-2^{-n}$ the estimate will be at most $2^{-\frac{b+a}{2}}$, and the "oracle" will give it the value 0. Thus with probability at least $1-k2^{-n}$, R runs as if it has access to some oracle $T' \in \mathcal{T}'$. By Corollary 3.5, this implies that the procedure decides correctly the membership of x in L with probability at least $3/5 - k2^{-n}$ (which can be amplified in the standard way). This conlcludes the proof for the single-length case.

In case R makes queries of different lengths, we modify the proof as follows. After fixing $x \in \{0,1\}^n$, we say that a is relevant if R makes queries of length $b(a)$ with positive probability. We set a_0 to be $10 \log k$, and $b_0 = b(a_0)$. For every relevant $a < a_0$ we hardwire the entire image of G_a into R as described above. So from now on we may assume that R makes only queries of length $b(a)$ for every relevant $a \geq a_0$. We then define queries of length $b(a)$ to be heavy/light/undetermined as before (with respect to a and $b(a)$). We then define for each such a a family $\mathcal{T}_a$ similar to the way we defined $\mathcal{T}$. And we take $\mathcal{T}$ to be the union of these families. I.e. $\mathcal{T}$ contains all the functions that agree with one of the functions in the family $\mathcal{T}_a$ for every relevant a (now $\mathcal{T}$ is defined on the domain $\bigcup_{\text{relevant } a}\{0,1\}^{b(a)}$). Claim 3.2 is proven as above (arguing seperately for each relevant a). $\mathcal{T}'$ is defined as above but now over all strings of length $b(a)$ for all the relevant a. Claim 3.4 is proven seperately for each relevant a. Then to bound $\Delta(R^T(x), R^{T'}(x))$ we take the union bound over all relevant a. Since there are at most k of them, this can be bounded by $k2^{-\frac{b_0-a_0}{3}} \leq 2^{-10}$ (by the facts that $b_0 > 4a_0$ and $a_0 > 10 \log k$) which is all that we need for Corollary 3.5. The rest of the proof continues as before. □

4 Adaptive Reductions

In this section we show that any black-box reduction, from a language L to distinguishing a generator, that is mildly adaptive (in the sense that queries of the same length do not appear in too many different levels), implies that L is in PSPACE.

Theorem 4.1. *Let $L \subseteq \{0,1\}^*$ be a language, and let $\mathcal{G} = \{G_a : \{0,1\}^a \to \{0,1\}^{b(a)}\}_{a\in\mathbb{N}}$ be a family of hitting-set generators such that G_a is computable in time $2^{O(a)}$, and $b(a) > 4a$. If there is a $\ell(a,n)$-adaptive black-box reduction R from L to $\frac{1}{2}$-distinguishing $\mathcal{G}$ a.e., where $\ell(a,n) \leq \frac{b(a)-a}{40\log n}$ for $a \geq 15\log n$, then L is in* PSPACE. *If we remove the time bound condition on computing G_a then L is in* PSPACE/*poly*.

Proof outline: Our starting point is the proof of Theorem 3.1 (see proof outline in Section 3). Our aim is to construct, based on an input x, the functions T and T' as before. The problem that we face when trying to implement the same ideas is that now, because the reduction is adaptive, the property of a query being light or heavy depends on the oracle that R queries (this is because queries above the first level depend on answers of the oracle). We therefore cannot define T in the same manner (such a definition will be circular). Instead, we classify queries to light and heavy separately for each level of adaptivity (i.e. a query can be light for one level and heavy for the other). We do that inductively as follows. For the first level we set a threshold 2^{-t_1} (where t_1 is a weighted average of a and $b = b(a)$). We then define light and heavy with respect to this threshold. The distribution over queries at the first level is independent of any oracle, so the classification is well defined. We then define a function T_1 to be 1 on queries that are heavy for the first level and 0 otherwise. We can now proceed to define light and heavy for the second level when considering the distribution over queries at the second level when running $R(x)$ with oracle access to T_1 at the first level. We continue with this process inductively to define light/heavy at level i, with respect to the distribution obtained by running $R(x)$ with oracles $T_1, \ldots, T_{i-1}$ (each at the corresponding level). Here T_j is defined to be 1 on queries that are heavy for at least one of levels from the j'th down (and 0 otherwise). For each level i we define a different threshold 2^{-t_i}, with the property that the thresholds gradually increase with the levels (the reason for this will soon be clear).

We now define the statistical test T to be 1 on elements that are heavy for at least one of the levels as well as on elements in the image set of G_a (and 0 otherwise). The argument showing that T $\frac{1}{2}$-distinguishes G_a, is similar to the one in the proof of Theorem 3.1.

In the next step, instead of defining a T' as in the proof of Theorem 3.1, we directly compare the outcomes of running $R(x)$ with T as an oracle and running $R(x)$ with oracles $T_1, \ldots, T_\ell$ (where ℓ is the number of adaptivity levels), each at the corresponding level. We argue that the two runs should be roughly the same (in the sense that the distributions over the outputs will be close). To do that, we observe that at each level i, the answer of T on a query q differs from the answer of T_i on this query if one of the following occurs:

1. q is in the image set of G_a and it is light for level $1, \ldots, i$.
2. q is light for all levels $1, \ldots, i$ but heavy for at least one of the levels $i+1, \ldots, \ell$.

In both cases T will give q the value 1, while T_i the value 0. We bound the probability that queries as above are generated by $R(x)$ when it is given the oracles $T_1, \ldots, T_\ell$. The argument that bounds the probability that queries of the first type are generated is similar

to the argument in the proof of Theorem 3.1. The probability that queries of the second type are generated at the i'th level is bounded as follows: the total number of heavy elements for levels above the i'th is small (it is the reciprocal of their weight which is high). Of these elements, those that are light at level i have small probability to be generated at level i by virtue of them being light for that level. When we take the union bound over all such queries we still get a small probability of at least one of them being generated. The point is that the number of elements in the union bound is computed according to thresholds of levels above the i'th, while their probability is taken according to the threshold of the i'th level. By the fact that thresholds increase with the levels, we get that the number of elements in the union bound is much smaller than the reciprocal of their probabilities, and therefore the overall probability of such an event is small. We conclude that the output distributions of running R with oracle access to T and running $R(x)$ with oracle access to $T_1, \ldots, T_\ell$ are very close, and therefore the latter decides correctly the membership of x in L.

Finally we show that $T_1, \ldots, T_\ell$ can be implemented in PSPACE and thus the whole procedure of running $R(x)$ and computing these oracles is in PSPACE. To compute the answers of the oracle T_i (at level i) we compute the exact weight of the query. We do that by a recursive procedure that computes the exact weights of all the queries (at levels below the i'th) that appear along the way. The fact that T_i only depends on T_j for $1 \leq j < i$ allows this procedure to run in polynomial-space. We proceed with the formal proof.

Proof. [of Theorem 4.1] As in the proof of Theorem 3.1 we will first prove the theorem for the case that the reduction makes single-length queries, and towards the end of the proof explain how it can be generalized.

Fix an input length n and fix an instance $x \in \{0,1\}^n$. Let $a \in \mathbb{N}$ be such that R queries its oracle on instances of length $b = b(a)$ when given inputs of length n. Let $k = k(n) = \text{poly}(n) \geq n^2$ be an upper bound on the number of queries that R makes at each one of the $\ell \leq \frac{b(a)-a}{40 \log n}$ levels. As in the proof of Theorem 3.1, we may assume that $a > 10 \log k > 15 \log n$ (otherwise we can hardwire/compute the outputs of the generator). We can also assume that $\frac{\log(b-a)}{10} < a$ (i.e. the stretch of the generator is at most exponential). Otherwise, since $b \leq \text{poly}(n)$ (b is the length of the queries that the poly(n)-time reduction is making), we will get that $a = O(\log n)$ and again we can hardwire/compute the outputs of the generator. Let,

$$\text{Im}(G_a) \stackrel{def}{=} \{q \in \{0,1\}^b : \exists y \in \{0,1\}^a \text{ s.t. } G_a(y) = q\}$$

For a sequence of functions $f_1, \ldots, f_\ell$ (where $f_i : \{0,1\}^b \to \{0,1\}$), we denote by $R^{f_1,\ldots,f_\ell}(x;r)$ the outcome of running R on input x using a random string r, where at level i, R has access to the oracle f_i. We denote by $R^{f_1,\ldots,f_{i-1}}(x;r)$ the queries that R generates at the i'th level when given access to the oracles $f_1, \ldots, f_{i-1}$ (each at the corresponding level).

Let $0 < \epsilon_1 < 1$ be a value to be determined later. We call $q \in \{0,1\}^b$ *heavy for level 1* if,

$$\Pr[R(x) \text{ asks the query } q \text{ in the first level}] > 2^{-(\epsilon_1 a + (1-\epsilon_1)b)}$$

(where the probability is over the random coins of R). Otherwise we say that q is *light for level 1*. Define the function $T_1 : \{0,1\}^b \to \{0,1\}$:

$$T_1(q) = \begin{cases} 1 & \text{if } q \text{ is heavy for the level 1} \\ 0 & \text{otherwise} \end{cases}$$

We continue inductively and say that $q \in \{0,1\}^b$ is heavy for the i'th level if,

$$\Pr[R^{T_1,\ldots,T_{i-1}}(x) \text{ asks the query } q \text{ in level } i] > 2^{-(\epsilon_i a+(1-\epsilon_i)b)}$$

and it is light for the i'th level otherwise.

And define $T_i : \{0,1\}^b \to \{0,1\}$:

$$T_i(q) \quad = \quad \begin{cases} 1 & \text{if } q \text{ is heavy for at least one} \\ & \text{of the levels } 1,\ldots,i \\ 0 & \text{otherwise} \end{cases}$$

We set $\epsilon_1 = 20 \log n \cdot (b-a)^{-1}$, and $\epsilon_i = \epsilon_{i-1} + \epsilon_1$.

Define the statistical test T as follows:

$$T(q) \quad = \quad \begin{cases} 1 & \text{if } q \text{ is heavy for at least one} \\ & \text{of the levels } 1,\ldots,\ell \text{ or } q \in \mathrm{Im}(G_a) \\ 0 & \text{otherwise} \end{cases}$$

Claim 4.2. *T $\frac{1}{2}$-distinguishes G_a.*

Proof. By the definition of T, $\Pr_{y \leftarrow U_a}[T(G_a(y)) = 1] = 1$. On the other hand,

$$\begin{aligned} \Pr_{y' \leftarrow U_b}[T(y') = 1] &\leq \Pr_{y' \leftarrow U_b}[y' \text{ is heavy for at least one of the levels or } y' \in \mathrm{Im}(G_a)] \\ &\leq \left((\sum_{i=1}^{\ell} k \cdot 2^{\epsilon_i a+(1-\epsilon_i)b}) + 2^a \right) \cdot 2^{-b} \\ &\leq (k \cdot \ell \cdot 2^{\epsilon_1 a+(1-\epsilon_1)b} + 2^a) \cdot 2^{-b} \\ &\leq 1/2 \end{aligned}$$

The second inequality follows from the fact that there are at most $2^{\epsilon_i a+(1-\epsilon_i)b}$ heavy elements for level i, and there are at most k queries (in level i) in which such an element can appear. The third inequality follows from the fact that $\epsilon_1 < \epsilon_2 < \ldots < \epsilon_\ell$. In the last inequality we use the facts that $\epsilon_1 = 20 \log n \cdot (b-a)^{-1}$, $\log k < a/10$, $\ell \leq \frac{b-a}{40 \log n}$, and $b > 4a$. □

Corollary 4.3. $\Pr[R^T(x) = L(x)] > 2/3$, *where the probability is over the random coins of R.*

Claim 4.4. $\Delta(R^{T_1,\ldots,T_\ell}(x), R^T(x)) \leq 1/10$

Proof. We bound the probability of the event that at least one query in the execution of $R^{T_1,\ldots,T_\ell}(x)$ receives a different answer than it would receive if run with the oracle T. We consider each level at a time.

A query q that is asked at level i will possibly be answered differently by T_i and T if one of the following occurs:

1. $q \in \mathrm{Im}(G)$ and it is light for all levels $1,\ldots,i$. Or,
2. q is light for all levels $1,\ldots,i$ but heavy for at least one of the levels $i+1,\ldots,\ell$.

We bound the probabilities that queries of these two types are generated by the reduction, starting with the first type. The probability that a query of the first type is generated at level i is bounded by the probability of the event $E_1^i \stackrel{def}{=} \{R^{T_1,\ldots,T_{i-1}}(x)$ asks a query that is light for level i and is in $\mathrm{Im}(G)\}$.

$$\begin{aligned}\Pr[E_1^i] &\leq 2^a \cdot 2^{-(\epsilon_i a+(1-\epsilon_i)b)} \\ &= 2^{-(1-\epsilon_i)(b-a)}\end{aligned}$$

We now bound the probability that a query of the second type is generated at level i. The probability that this happens is bounded by the probability of the event $E_2^i \stackrel{def}{=} \{R^{T_1,\ldots,T_{i-1}}(x)$ asks a light query for level i that is heavy for at least one of the levels $i+1,\ldots,\ell$.

$$\begin{aligned}\Pr[E_2^i] &\leq \left(\sum_{j=i+1}^{\ell} k \cdot 2^{\epsilon_j a+(1-\epsilon_j)b}\right) \cdot 2^{-(\epsilon_i a+(1-\epsilon_i)b)} \\ &\leq k\ell \cdot 2^{\epsilon_{i+1} a+(1-\epsilon_{i+1})b} \cdot 2^{-(\epsilon_i a+(1-\epsilon_i)b)} \\ &= k\ell \cdot 2^{-(\epsilon_{i+1}-\epsilon_i)(b-a)}\end{aligned}$$

In the second inequality we use the fact that $\epsilon_{i+1} < \ldots < \epsilon_\ell$.

Taking the union bound over all levels, we get that the probability that $T_1,\ldots,T_\ell$ give different answers than T (in at least one query) is bounded from above by

$$\ell \cdot 2^{-(1-\epsilon_\ell)(b-a)} + k\ell^2 \cdot 2^{-\epsilon_1 \cdot (b-a)} \leq 1/10 \tag{3}$$

Here we use our choice of parameters and in particular the facts that $\epsilon_1 < \ldots < \epsilon_\ell$, and that $\epsilon_1 = \epsilon_{i+1} - \epsilon_i$ for every $1 \leq i \leq \ell - 1$, and also recalling that $10 \log k < a$, and $\frac{\log(b-a)}{10} < a$. □

From Corollary 4.3 and Claim 4.4 we get,

Corollary 4.5. $\Pr[R^{T_1,\ldots,T_\ell}(x) = L(x)] \geq 19/30$*, where the probability is over the random coins of* R*.*

We show how to implement $R^{T_1,\ldots,T_\ell}(x)$ in probabilistic polynomial space, which in turn can be done deterministically by taking the majority vote over all possible random strings. On randomness r for the reduction R, we simulate an execution of $R^{T_1,\ldots,T_\ell}(x;r)$. For every query q that is generated at some level $1 \leq i \leq \ell$ we take the oracle answer to be the value returned from running the procedure Simulate-Oracle(q,i) given in Figure 1. This procedure checks whether q is heavy for at least one of the levels up to the i'th, by computing its exact weight at each level. It returns 1 or 0 accordingly. Thus it simulates at the i'th level precisely the oracle T_i. Hence by Corollary 4.5 the algorithm decides correctly the membership of x in L.

We proceed to analyze the space complexity of the algorithm. R runs in polynomial-time, and hence in polynomial space. Let us now analyze the space complexity of procedure Simulate-Oracle. The depth of the recursion of Simulate-Oracle on input (q,i), is $i \leq \mathrm{poly}(n)$. Each level requires space that is polynomial in the running-time and space used by R (to store *counter*, r, j, h and the state of R). Note that at each point in time and for every $1 \leq i \leq \ell$, there is at most one execution of Simulate-Oracle$(\cdot,i)$ that is running.

SIMULATE-ORACLE(q, i)

1. For every $1 \leq j \leq i$:
   (a) *counter* $\leftarrow 0$.
   (b) For every $r \in \{0,1\}^m$ (where m is the number of random coins that R uses on inputs of length n):
      i. Run $R(x;r)$ up to level j. For every oracle query $q' \in \{0,1\}^b$ that is generated during this execution at some level $h < j$, answer via a recursive call Simulate-Oracle(q', h).
      ii. If q is one of the queries that $R(x;r)$ generates at level j then: *counter* $\leftarrow$ *counter* $+ 1$.
   (c) If *counter* $> 2^{-(\epsilon_j a + (1-\epsilon_j)b)} \cdot 2^m$ output 1 and halt. Otherwise proceed to the next iteration.
2. Output 0.

Figure 1: The procedure Simulate-Oracle

Hence by re-using space between different calls we conclude that the whole procedure can be done in poly(n) space.

We now explain how to generalize the proof to handle queries of different lengths. We follow the outline above, with the only difference that we define light/heavy seperately for each possible query length. That is, we have a different sequence of ϵ's for each length (and as before we need only consider statistical tests for generators with seed length more than $10 \log n$), and different sequence of T_i's as well as a different T. Claim 4.2 is proven for each length separately, and conclude with Corollary 4.3 (where now T is defined over several lengths). Finally, in Claim 4.4 we consider different events E_1^i and E_2^i for different query lengths. And for each length a, we bound the probability that these events happen by taking the union bound over all levels in which the query length has positive probability to appear (which is at most $\ell(a,n) = \frac{b(a)-a}{40 \log n}$). Thus for each length we have a bound similar to the left hand side of inequality (3). We now observe that since R only use $a > 10 \log n$ (images of G_a for smaller a's are hardwired), each one of these expressions is bounded by $\frac{1}{10n}$, and then by taking a union bound over all possible lengths, we get that the probability that one of the "bad" events happens is at most $1/10$. The rest of the proof is as above. □

5 Reductions from complete languages

In this section we revisit known reductions from deciding languages in high classes (such as EXP and PSPACE) to distinguishing pseudorandom (and hitting-set) generators, in light of our negative results. Impagliazzo and Wigderson [IW] showed that for any language L that is both downwards self-reducible and self-correctible, there is a pseudorandom generator $\{G_L : \{0,1\}^a \to \{0,1\}^{poly(a)}\}_{a \in \mathbb{N}}$, and a black-box reduction from deciding L to $\frac{1}{2}$-distinguishing G_L. The reduction is 1-adaptive and has n levels of adaptivity (each query length upto n, where n is the length of the L-instance, appears in at most one level of adaptivity). Specifically, G_L is the generator from Theorem 2.6 where the function f is the characteristic

function of L. The reduction uses the self-correctibility of L to compute it on every instance given an approximator that computes it on most instances. It uses the downwards self-reducibility property of L to compute (in a recursive way) the values of certain instances of L that originally (in [BFNW]) were given as a non-uniform advice (see Theorem 2.6). The use of downwards self-reducibility is what makes this reduction adaptive.

It is well known that the permanent function is complete for the class $P^{\sharp P}$ and is both downwards self-reducible and self-correctible [Lip]. Another function having these properties and is PSPACE-complete, was presented in [TV]. Thus we obtain,

Theorem 5.1. *[IW, TV] For every language L in PSPACE there exists a polynomial function $k(\cdot)$ such that for every polynomial function $b(\cdot)$, there is a uniform black-box reduction from deciding L to distinguishing a certain family of pseudorandom generators $\mathcal{G} = \{G : \{0,1\}^a \to \{0,1\}^{b(a)}\}_{a\in\mathbb{N}}$ a.e.. The reduction is 1-adaptive and has $k(n)$ levels of adaptivity.*

We conclude by Theorem 3.1 that the black-box reduction from the theorem above is *inherently* adaptive, unless $\mathrm{PSPACE} = \mathrm{BPP}^{\mathrm{NP}}$.

The reduction of [IW] from deciding languages in EXP (rather than PSPACE) is more involved and requires an additional idea. The reason is that (unless EXP = PSPACE) there are no EXP-complete languages that are downwards self-reducible. I.e. one cannot use directly their reduction from deciding languages that have this property to distinguishing PRGs. To get around this, [IW] argue as follows: if there is an *efficient* statistical test that distinguishes the generator (i.e. one that is computable by poly-size circuits), then by the non-uniform reduction of [BFNW] (Theorem 2.6), f, the EXP-complete function, is computable by polynomial-size circuits. This implies by [KL] that EXP = PSPACE (in fact EXP = PH), which means that there is (another) EXP-complete function that is both downwards self-reducible and self-correctible, and we can continue with the reduction described above. Note that this part makes the reduction non-black-box; the argument only works when considering efficient statistical tests (inefficient ones do not imply that $f \in \mathrm{P}/poly$ and thus EXP = PSPACE).[5] We do note however, that the reduction itself (i.e. the TM that implements the reduction) still only has oracle access to the statistical test that distinguishes the generator. It is only the proof of correctness that makes a non-black-box use of the adversary by restricting the argument only to efficient statistical tests (rather than arbitrary ones). More formally, the reduction is size-restricted black-box (see Definition 2.3).

Note that the above argument does not give a single reduction from computing f to distinguishing a generator. Rather it shows that if there is an efficient adversary that distinguishes both the generator from Theorem 2.6 based on f and the generator from Theorem 5.1 (which amounts to the generator from Theorem 2.6 constructed from a downwards self-reducible and self-correctible function), then f can be decided efficiently (via a uniform reduction). To make this a reduction to distinguishing a single generator, we combine the two to get a hitting-set generator.

Theorem 5.2. *(implicit in [IW]) For every language L in EXP and polynomial function $b(\cdot)$, there is a polynomial function $k(\cdot)$ and a uniform size-restricted black-box reduc-*

[5] The reduction of [BFNW] gives a polynomial-size circuit with oracle to the distiguisher T that computes f (see Theorem 2.6). Only if T is itself computable by polynomial-size circuits, we can conclude that there is a polynomial-size circuit (without an oracle) that computes f.

tion from deciding L to distinguishing a certain family of hitting-set generators $\mathcal{G} = \{G : \{0,1\}^a \to \{0,1\}^{b(a)}\}_{a\in\mathbb{N}}$ a.e.. The reduction is 1-adaptive and has $k(n)$ levels of adaptivity.[6]

Proof sketch. Let f be a self-correctible EXP-complete function (e.g. f can be the multilinear extension [BFL] of some fixed EXP-complete function). Let g be a downwards self-reducible and self-correctible PSPACE-complete function (as in [TV]). Let $\mathcal{G}_0$ be the pseudorandom generator from Theorem 2.6 that is based on f, setting δ to be such that $b(a)^\delta \leq a$ for all sufficiently large a. (On seed length a, $\mathcal{G}_0$ uses f on inputs of length $m = \lfloor b(a)^\epsilon \rfloor$ for some constant $\epsilon > 0$ that depends on δ.) Let $\mathcal{G}_1$ be the pseudorandom generator from Theorem 2.6 that is based on g, with the same δ. By Theorem 2.6, if there is a statistical test T that is computable by quadratic-size circuits and distinguishes $\mathcal{G}_0$ a.e. then $f \in$ P/poly and therefore EXP $\subseteq$ P/poly. By Karp and Lipton [KL] this implies that EXP = PH = PSPACE and hence f reduces to g. Specifically, f has circuits of size poly$(b(a))$ on instances of length $m = \lfloor b(a)^\epsilon \rfloor$, and thus reduces to g-instances of length poly$(b(a))$ = poly(m), where the latter polynomial depends on ϵ and hence on the degree of b. Composing this with the reduction from L to f, we get a reduction from deciding L on instances of length n to computing g on instances of length $\ell(n) =$ poly(n), where this polynomial depends on both L and b.

By Theorem 5.1, there is a uniform black-box reduction from computing g on instances of length $\ell(n)$ to distinguishing a certain generator from the family $\mathcal{G}_1$. This reduction is 1-adaptive, has $k(n) =$ poly$(\ell(n))$ levels of adaptivity. Thus if such a distinguisher for $\mathcal{G}_0$ exists, we get a black-box poly(n)-time reduction from distinguishing L on instances of length n, to distinguishing a generator in the family $\mathcal{G}_1$.

Now, consider the family of hitting-set generators $\mathcal{G} = \{G : \{0,1\}^a \times \{0,1\} \to \{0,1\}^{b(a)}\}_{n\in\mathbb{N}}$ defined as: $G(s,b) = G_b(s)$ (for every $a \in \mathbb{N}$, $s \in \{0,1\}^a$ and $b \in \{0,1\}$). If there is a statistical test T that is computable by quadratic-size circuits and $\frac{1}{2}$-distinguishes $\mathcal{G}$ a.e., then it must also $\frac{1}{2}$-distinguish both $\mathcal{G}_0$ and $\mathcal{G}_1$ a.e. as hitting-set generators (and thus distinguish them from uniform also as pseudorandom generators). We can then plug this statistical test into the black-box reduction (described above) from deciding L to distinguishing $\mathcal{G}_1$. This reduction indeed decides L correctly a.e. since the statistical tests, which are computable by quadratic-size circuits, also distinguish $\mathcal{G}_0$ a.e.. Thus we get a size-restricted black-box reduction with the specified parameters from deciding L to $\frac{1}{2}$-distinguishing $\mathcal{G}$ a.e.. Note that indeed the reduction only makes oracle queries to the distinguisher (since it runs the black-box reduction from deciding L to distinguishing $\mathcal{G}_1$). However, the correctness of this reduction relies on the fact that the distinguisher is of polynomial-size (since only in this case there is a reduction from f to g, which is one of the steps in the black-box reduction from deciding L to distinguishing $\mathcal{G}_1$). □

The reduction in Theorem 5.2 should be contrasted with Theorem 4.1, which says that any reduction that is 1-adaptive cannot be black box in the sense of Definition 2.2 (unless EXP = PSPACE). That is, the 'size-restricted' aspect of Theorem 5.2 cannot be removed.

6 An "Almost Everywhere" version of [IW] for RP

The reduction from Theorem 5.2 gives the main result of [IW]:

[6] Note that the order of quantifiers over the polynomials $b(\cdot)$ and $k(\cdot)$ is swapped, as compared to Theorem 5.1. This is the result of the two reductions, both using the same distinguisher for two different generators each on a different output length (see the details in the proof.)

Theorem 6.1. *At least one of the following is true:*

1. EXP = BPP
2. BPP $\subseteq io - HeurTIME_{n^{-c}}(2^{n^\delta})$ *for every* $c, \delta > 0$.

The proof of [IW], when applied to a single HSG, as outlined in the previous section, shows a reduction that works for every input length whenever we have a family of distinguishers for every seed length. In this section we observe that for derandomizing the class RP (of languages decidable in probabilistic polynomial-time with 1-sided error), a theorem similar to Theorem 6.1 is true when we switch the i.o. quantifier from (2) to (1), and replace EXP with PSPACE in item (1). In terms of reductions, this shows that when considering HSGs (rather than PRGs), we can obtain a reduction that succeeds infinitely often, when given access to a family of distinguishers that only succeeds (to distinguish) infinitely often.

Theorem 6.2. *At least one of the following is true:*

1. PSPACE $\subseteq io -$ BPP
2. RP $\subseteq HeurTIME_{n^{-c}}(2^{n^\delta})$ *for every* $c, \delta > 0$.

Proof sketch. Let $L \in$ RP and A be a probabilistic polynomial-time algorithm with 1-sided error that observes that. Suppose that A runs in time $m = \text{poly}(n)$ on instances of length n and hence uses at most m random bits. Let $f : \{0,1\}^* \to \{0,1\}$ be a PSPACE-complete function that is both downwards self-reducible and self-correctible (as in [TV]). Let $\delta > 0$ be an arbitrary constant and let ϵ be the constant from Theorem 2.6 that depends on δ.

For $1 \leq k \leq m^\epsilon$, let $G^{f^k}_{\delta,m} : \{0,1\}^{m^\delta} \to \{0,1\}^m$ be the generator from Theorem 2.6, where f^k is the restriction of f to instances of length k. Define the hitting-set generator $G_{\delta,m} : \{0,1\}^{m^\delta} \times [\lfloor m^\epsilon \rfloor] \to \{0,1\}^m$ to be $G_{\delta,m}(x,k) = G^{f^k}_{\delta,m}(x)$. Note that since the f^k's are computable in exponential-time (in their input length), $G_{\delta,m}$ is computable in time at most $2^{m^{\delta'}}$, where $\delta' = \kappa \cdot \delta$ for some universal constant $\kappa > 0$.

To deterministically decide L, on instance $x \in \{0,1\}^n$, we run the $(\text{poly}(m) \cdot 2^{m^{\delta'}})$-time deterministic simulation of A using the outputs of $G_{\delta,m}$ as random strings. If at least one of the executions $A(x,r)$ (where r is an output of the generator) accepts, then we accept x.

If for every $L \in$ RP, for every δ, c and every large enough n, this deterministic algorithm decides L correctly on a $1 - n^{-c}$ fraction of the instances of length n then we are done (RP $\subseteq HeurTIME_{n^{-c}}(2^{n^\delta})$ for every $\delta > 0$ and $c > 0$). Otherwise, there exist a language $L \in$ RP and there are $\delta, c > 0$ and infinitely many n's such that $G_{\delta,m}$ fails on at least n^{-c} fraction of the L-instances of length n. Each one of them gives rise to a Boolean circuit of size m that $\frac{1}{2}$-distinguishes all the generators $G^{f^1}_{\delta,m}, \ldots, G^{f^k}_{\delta_m}$ (as HSGs and therefore also as PRGs). Furthermore, we can generate, for every $1 \leq k \leq \lfloor m^\epsilon \rfloor$, in poly$(m)$-time with oracle access to f^k, a size poly(m) circuit D_k that $\frac{1}{2}$-distinguishes $G^{f^k}_{\delta,m}$. This is done as in [IW]. That is, we sample n^{2c} L-instances of length n. From each one we create the circuit that simulates the algorithm A, with the instance hardwired into it, and its input is the randomness of A. For each such circuit, we check whether it distinguishes $G^{f^k}_{\delta,m}$ from uniform, by estimating its acceptance probabilities when it is fed the uniform distribution and when it is fed the generated one. This is done by taking enough samples from each distribution. Note that by Theorem 2.6, this procedure runs in poly(m) time given oracle access to f^k. With very high probability we indeed sample an instance on which the deterministic simulation fails (and

thus this instance gives rise to a distinguisher). Furthermore, by taking enough samples for our acceptance estimates, we can ensure that with very high probability we determine correctly for each circuit D_k whether it is a distinguisher or not.

The rest of the proof now follows the outline of [IW] which we now briefly sketch. We will use Theorem 2.6 to obtain from these distinguishers D_k a sequence of $poly(m)$-size circuits C_k that compute f^k (for every $1 \leq k \leq m^\epsilon$). The realization of [IW] is that C_k can actually be constructed in probabilistic polynomial time given oracle access to f^k. (That is, an advice string a satisfying the conclusion of Theorem 2.6 can be constructed in probabilistic polynomial time given oracle access to $T = D_k$ and $f = f^k$.) Finally, the oracle calls to f^k needed in the construction of D_k and C_k are eliminated using the downwards self-reducibility property of f. We conclude that there as a probabilistic polynomial-time algorithm B such that for infinitely many m and every input x of length at most $\lfloor m^\epsilon \rfloor$, $\Pr[B(1^m, x) = f(x)] \geq 1 - 2^{-m}$ (after amplifying the success probability).[7] Moreover, by inspecting the construction (or using the fact that PSPACE has 'instance checkers'), we can also ensure that $\Pr[B(1^m, x) \in \{f(x), \perp\}] \geq 1 - 2^{-m}$. That is, whp B recognizes when it fails and outputs $\perp$ rather than $\neg f(x)$. We now argue that this implies PSPACE $\subseteq io - \text{BPP}$. Let K be an arbitrary language in PSPACE, and let ρ be a polynomial-time reduction from K to f. Then the algorithm $C(x)$ that runs $B(1^m, \rho(x))$ for all integers m in the interval $[|\rho(x)|^{1/\epsilon}, (|\rho(x)| + 1)^{1/\epsilon}]$ and outputs the first answer that is not $\perp$ will decide K correctly for infinitely many input lengths. □

Acknowledgments

We thank Ronen Shaltiel for helpful comments on the write-up.

References

[BFL] L. Babai, L. Fortnow, and C. Lund. Non-deterministic exponential time has two-prover interactive protocols. In *Proceedings of the 31st Annual IEEE Symposium on Foundations of Computer Science*, pages 16–25, 1990.

[BFNW] L. Babai, L. Fortnow, N. Nisan, and A. Wigderson. BPP has subexponential simulation unless Exptime has publishable proofs. *Computational Complexity*, 3:307–318, 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, November 1984.

[BT] A. Bogdanov and L. Trevisan. On worst-case to average-case reductions for NP problems. In *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science*, pages 308–317, 2003.

[GST1] D. Gutfreund, R. Shaltiel, and A. Ta-Shma. Uniform hardness vs. randoness tradeoffs for Arthur-Merlin games. *Computational Complexity*, 12:85–130, 2003.

[7] The only difference between our argument and [IW] is that our distinguishers are of size m (even for very small k's), while their distinguishers are of size $\text{poly}(k)$, this however does not change the fact that the final algorithm runs in time $\text{poly}(m) = \text{poly}(m^\epsilon)$.

[GST2] D. Gutfreund, R. Shaltiel, and A. Ta-Shma. if NP languages are hard in the worst-case then it is easy to find their hard instances. *To appear in Computational Complexity*, 2007.

[GT] D. Gutfreund and A. Ta-Shma. Worst-case to average-case reductions revisited. In *11th International Workshop on Randomization and Computation, RANDOM*, pages 569–583, 2007.

[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.

[IW] R. Impagliazzo and A. Wigderson. Randomness vs. time: de-randomization under a uniform assumption. *Journal of Computer and System Sciences*, 63(4):672–688, 2001.

[JVV] M. Jerrum, L. Valiant, and V. Vazirani. Random generation of combinatorial structures from a uniform distribution. *Theor. Comput. Sci.*, 43:169–188, 1986.

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

[KI] V. Kabanets and R. Impagliazzo. Derandomizing polynomial identity tests means proving circuit lower bounds. *Computational Complexity*, 13(1-2):1–46, 2004.

[KL] R. M. Karp and R. J. Lipton. Some connections between nonuniform and uniform complexity classes. In *Proceedings of the 12th Annual ACM Symposium on Theory of Computing*, pages 302–309, 1980.

[KvM] A. R. Klivans and D. van Melkebeek. Graph nonisomorphism has subexponential size proofs unless the polynomial-time hierarchy collapses. *SIAM Journal on Computing*, 31(5):1501–1526, 2002.

[Lip] R. Lipton. New directions in testing. *Proceedings of DIMACS workshop on distributed computing and cryptography*, 2:191–202, 1991.

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

[RTV] O. Reingold, L. Trevisan, and S. Vadhan. Notions of reducibility between cryptographic primitives. In *Theory of Cryptography, First Theory of Cryptography Conference, TCC 2004*, pages 1–20, 2004.

[Sip] M. Sipser. A complexity theoretic approach to randomness. In *Proceedings of the 15th Annual ACM Symposium on Theory of Computing*, pages 330–335, 1983.

[Sto] L. Stockmeyer. On approximation algorithms for ♯P. *SIAM Journal on Computing*, 14(4):849–861, 1985.

[SU1] R. Shaltiel and C. Umans. Simple extractors for all min-entropies and a new pseudorandom generator. *Journal of the ACM*, 52(2):172–216, 2005.

[SU2] R. Shaltiel and C. Umans. Low-end uniform hardness vs. randomness tradeoffs for AM. In *Proceedings of the 39th Annual ACM Symposium on Theory of Computing*, pages 430–439, 2007.

[Tre] L. Trevisan. Construction of extractors using pseudo-random generators. *Journal of the ACM*, 48(4):860–879, 2001.

[TV] L. Trevisan and S. Vadhan. Pseudorandomness and average-case complexity via uniform reductions. *To appear in Computational Complexity*, 2007.

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

[Yao] A. C. Yao. Theory and applications of trapdoor functions. In *Proceedings of the 23rd Annual IEEE Symposium on Foundations of Computer Science*, pages 80–91, 1982.

ECCC ISSN 1433-8092

http://eccc.hpi-web.de/