Limitations of Hardness vs. Randomness under Uniform Reductions

Dan Gutfreund[1,⋆] and Salil Vadhan[2,⋆⋆]

[1] Department of Mathematics and CSAIL
MIT
danny@math.mit.edu
[2] School of Engineering and Applied Sciences
Harvard University
salil@eecs.harvard.edu

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 [10] and Trevisan and Vadhan [24] 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 with the restriction that queries of the same length do not occur in different levels of adaptivity. Impagliazzo and Wigderson [10] 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).
- *Adaptive* black-box reductions, with the same restriction on the adaptivity as above, can only exist for functions f in PSPACE (and thus are unlikely to exist for all of EXP).

Beyond shedding light on proof techniques in the area of hardness vs. randomness, our results (together with [10,24]) can be viewed in a more general context as identifying techniques that overcome limitations of black-box reductions, which may be useful elsewhere in complexity theory (and the foundations of cryptography).

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 2002246.

A. Goel et al. (Eds.): APPROX and RANDOM 2008, LNCS 5171, pp. 469–482, 2008.

1 Introduction

A central goal in the theory of computation is to identify relations between the complexities of different computational tasks. Indeed, some of the greatest discoveries in computational complexity (and the foundations of cryptography) are surprising connections between the complexities of tasks that seem very different in nature. The traditional way of showing such relations is by various forms of "black-box" reductions. That is, the proofs exhibit an efficient (oracle) algorithm R that when given oracle access to any function f that solves task T_1, R^f solves task T_2. Such an algorithm R proves that if T_1 has an efficient solution then so does task T_2. In complexity theory, such algorithms R are often referred to as just "reductions" (or "Cook reductions"), but here we use "black box" to emphasize that R only has oracle access to the function f solving task T_1 and is required to work regardless of the complexity of this oracle.

There are also a number of negative results about black-box reductions, showing that for certain pairs of tasks T_1, T_2, it is unlikely that such an algorithm R exists. The natural interpretation of such negative results is that proving the desired relation between T_1 and T_2 may be a difficult task that is beyond the reach of "current techniques." However, black-box reductions are not the only way computational relations can be obtained, and by now there are several proofs of relations that are unlikely to be provable using black-box reductions. A classic example is the result that an efficient algorithm for SAT implies an efficient algorithm for every language in the polynomial-time hierarchy; this is proven via a non-black-box argument and indeed a black-box reduction showing this relation seems unlikely (as it would imply the collapse of the polynomial-time hierarchy). It is useful to isolate and identify techniques like this, which overcome limitations of black-box reductions, because they may enable overcoming barriers elsewhere in complexity theory.

In this paper, we study black-box reductions in the area of hardness vs. randomness (i.e. constructing pseudorandom generators from hard functions). Specifically, we take T_1 to be the task of distinguishing the output distribution of a pseudorandom generator G from the uniform distribution, and T_2 to be the task of computing some supposedly 'hard' function g, and we are interested in reductions R such that R^f computes g if f is any function distinguishing G from uniform. We show limitations of such black-box reductions, and together with [10,24] (which do show that such connections exist), point to two ways in which black-box limitations can be overcome:

- Allowing the reduction R to make *adaptive* queries to its oracle f. We show that a relation established in [10,24] using an adaptive reduction is unlikely to be provable using a nonadaptive reduction. This may be interpreted as a hope to overcome other barriers known for nonadaptive reductions, such as worst-case/average-case connections for NP [4] and strong hardness amplification for constant-depth circuits [22]. (We mention though that for the latter connection, even adaptive black-box reductions are ruled out, unless they are highly non-uniform [5].)

– Using the efficiency of the oracle f in the *analysis* of the reduction R. Namely, we consider redcutions that use f as an oracle, but their analysis relies on the fact that f can be computed efficiently.[1] We observe that a reduction of this type is implicit in [10], and we show that it is unlikely to have an analogous reduction that works regardless of the complexity of the oracle. (A similar separation was given in the context of worst-case/average-case reductions for NP [6,7].)

We hope that these realizations will prove useful in obtaining stronger results in hardness vs. randomness and in overcoming limitations elsewhere in complexity theory.

Hardness vs. Randomness. We start with some background on hardness vs. randomness and the use of reductions in this area. The hardness versus randomness paradigm, first developed by Blum, Micali, Yao, Nisan, and Wigderson [3,26,16], 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 [10] 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 $\mathrm{EXP} \neq \mathrm{BPP}$.[2] This result and some work that followed it have raised the hope that we may be able to prove an equivalence between uniform and nonuniform hardness assumptions (since in some cases derandomization implies non-uniform lower bounds [9,12,18]), or even obtain unconditional derandomization and new lower bounds.

The work of Impagliazzo and Wigderson [10], 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 [24], 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

[1] Here the reduction does not need to use the *code* of the algorithm for f, but just the fact that an efficient algorithm exists. This is in contrast to the example of SAT vs. PH mentioned above.

[2] The generator of [10], as well as other generators that are based on uniform hardness assumptions, are weaker than those that are based on nonuniform assumptions, in the sense that they only fool (uniform) Turing machines and hence only imply an average-case derandomization of probabilistic classes.

even stronger results,[3] and also, as we mentioned above, 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 [10].

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 require 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 [17], 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 we would immediately get results based on uniform assumptions, like those of [10]. Unfortunately, as shown in [24], it is impossible to have a fully black-box construction of 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 [10] construction is not fully black box is that it is not proven to work for every function f, and rather the construction (and its proof of correctness) makes use of the fact that f is in EXP or some other complexity class such as $\mathrm{P}^{\#\mathrm{P}}$ or PSPACE [24]. 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 this

[3] A seemingly modest but still elusive goal is a "high-end" version of [10], whereby one can construct a pseudorandom generator with exponential stretch from the assumption that EXP does not have subexponential-time probabilistic algorithms.

area were nonadaptive. (There are subsequent reductions, due to Shaltiel and Umans [21,25], 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 (See Theorems 7 and 8).

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 (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 *adaptive*, but with the restriction that all the queries of a particular length must be made simultaneously (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 (see Definition 5).) The Impagliazzo–Wigderson reduction for functions f in $\mathrm{P}^{\#\mathrm{P}}$ or PSPACE is 1-adaptive. We show that this property is unlikely to extend to EXP.

Theorem 2 (informal). *If there is a 1-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 [24]). Theorem 2 suggests that their result might be improved by employing reductions with greater adaptivity, such as [21,25]. Alternatively, it would be interesting to rule out such an improvement by strengthening Theorem 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 [7].[4]

[4] There are subtle differences between the reductions that we consider and the ones in [7], see the remark following Definition 6.

They exhibited such a reduction (based on [6]) for a worst-case/average-case connection that cannot be established via standard black-box reductions. Theorem 2, together with Theorem 3 below (which is implicit in [10]), provides another example of a size-restricted black-box reduction that bypasses black-box limitations. For technical reasons, we state the [10] 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 and 2 above can be strengthened to apply also to hitting-set generators.

Theorem 3 (implicit in [10], informal). *For every function f in* EXP*, there is a generator G and a 1-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 [10]. 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 [10]. Due to space limitations, the statement and proof of this result is deferred to the full version of this paper [8].

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* [23] yielded substantial benefits for both the study of pseudorandom generators and extractors. Similarly, Klivans and van Melkebeek [14] observed that black-box constructions of pseudorandom generators extend naturally to derandomize classes other than BPP, such as AM.

Unfortunately, as we have mentioned, results showing the limitations of black-box reductions are often interpreted as saying that proving certain results are outside the reach of "current techniques". We strongly disagree with these kinds of interpretations, and indeed hope that our results together with [10] will serve as another reminder that such limitations can be overcome.

2 Preliminaries

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

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

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.

2.1 Pseudorandom Generators and Hardness vs. Randomness

Definition 4. *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 5. *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 6. *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[5]*, 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.

A remark about the quadratic size bound. 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.

A comparison to the definition of [7]. 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 black-box reductions* defined in [7], 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).

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.

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

Theorem 7. *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. We give here the main ideas in the proof. For the formal details refer to the full version of this paper [8]. 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) with 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 $\mathrm{BPP}^{\mathrm{NP}}$.

Before we explain how to construct T and T', we want to stress that these functions depend on the specific input x, and the fact that R^T and $R^{T'}$ behave almost the same is only guaranteed when we run them on that x. I.e. every instance determines different functions T and T' (we avoid using the notation T_x and T'_x because in the proof there are other parameters involved and the notations become cumbersome). The point is that given any instance x, the answers of the oracle T', that is determined by x, can be computed (from scratch) in $\mathrm{BPP}^{\mathrm{NP}}$.

Now, if G_a were computable in time $\mathrm{poly}(b(a))$, we could simply take $T = T' = \mathrm{Im}(G_a)$. Indeed, $\mathrm{Im}(G_a)$ is the optimal distinguisher for G_a, and membership in $\mathrm{Im}(G_a)$ can be decided in nondeterministic polynomial time if G_a is efficiently computable (by guessing a corresponding seed). However, as in [16,10], we allow the generator to run in time $2^{O(a)} \gg b(a)$, since this suffices when pseudorandom generators are used for derandomization. In such a case, deciding membership in $\mathrm{Im}(G_a)$ may not be feasible in the polynomial hierarchy. So instead we will take $T = \mathrm{Im}(G_a) \cup H$ and $T' = H$ where H is a "small" set defined so that R^T and $R^{T'}$ behave almost the same.

To construct such a set H, 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). (A similar idea appears in [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 H to be the set of heavy queries.

First, we argue that $T = H \cup \mathrm{Im}(G_a)$ $\frac{1}{2}$-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.

Next, we argue that the behavior of $R^T(x)$ is roughly the same as $R^{T'}(x)$, where $T' = H$. 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.

Finally, to show that $T' = H$ is computable in $\mathrm{BPP}^{\mathrm{NP}}$, we use the fact that approximate counting can be done in $\mathrm{BPP}^{\mathrm{NP}}$ [19,20,11], which allows us 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 of which are a weighted average of a and b), such that those queries with weight below the low threshold are considered light, those with weight above the high threshold are considered heavy, and those in between can be classified arbitrarily. We now need more subtle definitions of T and T', but still the outline described above works.

4 Adaptive Reductions

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

Theorem 8. *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. We give here the main ideas in the proof. For the formal details refer to the full version of this paper [8]. Our starting point is the proof of Theorem 7 (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 would 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 another). 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 7.

In the next step, instead of defining a T' as in the proof of Theorem 7, 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 levels $1, \ldots, i$.
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 7. 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 at most 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.

5 Comparison to Known Reductions

In this section we contrast our negative results regarding black-box reductions to known relations between deciding languages and distinguishing pseudorandom (and hitting-set) generators. Impagliazzo and Wigderson [10] showed such a reduction from every language in the class $\mathrm{P}^{\#\mathrm{P}}$. This was extended by Trevisan and Vadhan [24] to languages in PSPACE. These reductions are black-box and adaptive (1-adaptive to be precise, see Definition 5).

Theorem 9. *[10,24] 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 $\mathcal{G} = \{G_a : \{0,1\}^a \to \{0,1\}^{b(a)}\}_{a \in \mathbb{N}}$ a.e., where G_a is computable in time $2^{O(a)}$. The reduction is 1-adaptive and has $k(n)$ levels of adaptivity.*

We conclude by Theorem 7 that the black-box reduction from the theorem above is *inherently* adaptive, unless $\mathrm{PSPACE} = \mathrm{BPP}^{\mathrm{NP}}$.

Next we turn our attention to reductions from languages in the class EXP. Such a reduction was given by Impagliazzo and Wigderson [10]. Their reduction is not black-box but rather *size-restricted* black-box (see Definition 6). We refer the reader to the full version of this paper [8] where we explain how the fact that the distinguisher can be computed by small-size circuits plays a role in this reduction.

Theorem 10. *(implicit in [10]) 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 reduction from deciding L to distinguishing a certain family of hitting-set generators $\mathcal{G} = \{G_a : \{0,1\}^a \to \{0,1\}^{b(a)}\}_{a \in \mathbb{N}}$ a.e., where G_a is computable in time $2^{O(a)}$. The reduction is 1-adaptive and has $k(n)$ levels of adaptivity.*

Theorem 10 should be contrasted with Theorem 8, which says that any reduction that is 1-adaptive cannot be black box (unless EXP = PSPACE). That is, the 'size-restricted' aspect of Theorem 10 cannot be removed.

A remark about reductions from computing a function on the average. Typically, constructions of pseudorandom (resp. hitting-set) generators from hard

functions (both against uniform and non-uniform classes) combine two reductions: the first reduces the task of computing f (the supposedly hard function) on every instance to computing some related function $\hat{f}$ on the average. The second reduces computing $\hat{f}$ on the average to distinguishing the generator. In particular, the proof of [10] takes this form. We mention that our negative results about black-box reductions can be strengthened to show the same limitations for reductions from computing a function on the average to distinguishing a generator from the uniform distribution. In other words, it is really the fact that we reduce to distinguishing a generator that makes it impossible to do with black-box reductions, and not the fact that we start from a worst-case hardness assumption. In fact, *nonadaptive* (and uniform, black-box) worst-case to average-case reductions for PSPACE-complete and EXP-complete functions are known [1,2,24].

Acknowledgements

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

References

1. Babai, L., Fortnow, L., Lund, C.: Non-deterministic exponential time has two-prover interactive protocols. In: Proceedings of the 31st Annual IEEE Symposium on Foundations of Computer Science, pp. 16–25 (1990)
2. Babai, L., Fortnow, L., Nisan, N., Wigderson, A.: BPP has subexponential simulation unless EXPTIME has publishable proofs. Computational Complexity 3, 307–318 (1993)
3. Blum, M., Micali, S.: How to generate cryptographically strong sequences of pseudo-random bits. SIAM Journal on Computing 13(4), 850–864 (1984)
4. Bogdanov, A., Trevisan, L.: On worst-case to average-case reductions for NP problems. In: Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science, pp. 308–317 (2003)
5. Gutfreund, D., Rothblum, G.: The complexity of local list decoding. Technical Report TR08-034, Electronic Colloquium on Computational Complexity (2008)
6. Gutfreund, D., Shaltiel, R., Ta-Shma, A.: If NP languages are hard in the worst-case then it is easy to find their hard instances. Computational Complexity 16(4), 412–441 (2007)
7. Gutfreund, D., Ta-Shma, A.: Worst-case to average-case reductions revisited. In: Charikar, M., Jansen, K., Reingold, O., Rolim, J. (eds.) RANDOM 2007. LNCS, vol. 4627, pp. 569–583. Springer, Heidelberg (2007)
8. Gutfreund, D., Vadhan, S.: Limitations of hardness vs. randomness under uniform reductions. Technical Report TR08-007, Electronic Colloquium on Computational Complexity (2008)
9. Impagliazzo, R., Kabanets, V., Wigderson, A.: In search of an easy witness: Exponential time vs. probabilistic polynomial time. Journal of Computer and System Sciences 65(4), 672–694 (2002)

10. Impagliazzo, R., Wigderson, A.: Randomness vs. time: de-randomization under a uniform assumption. Journal of Computer and System Sciences 63(4), 672–688 (2001)
11. Jerrum, M., Valiant, L., Vazirani, V.: Random generation of combinatorial structures from a uniform distribution. Theor. Comput. Sci. 43, 169–188 (1986)
12. Kabanets, V., Impagliazzo, R.: Derandomizing polynomial identity tests means proving circuit lower bounds. Computational Complexity 13(1-2), 1–46 (2004)
13. Karp, R.M., Lipton, R.J.: Some connections between nonuniform and uniform complexity classes. In: Proceedings of the 12th Annual ACM Symposium on Theory of Computing, pp. 302–309 (1980)
14. Klivans, A.R., van Melkebeek, D.: Graph nonisomorphism has subexponential size proofs unless the polynomial-time hierarchy collapses. SIAM Journal on Computing 31(5), 1501–1526 (2002)
15. Lipton, R.: New directions in testing. In: Proceedings of DIMACS workshop on distributed computing and cryptography, vol. 2, pp. 191–202 (1991)
16. Nisan, N., Wigderson, A.: Hardness vs. randomness. Journal of Computer and System Sciences 49, 149–167 (1994)
17. Reingold, O., Trevisan, L., Vadhan, S.: Notions of reducibility between cryptographic primitives. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 1–20. Springer, Heidelberg (2004)
18. Santhanam, R.: Circuit lower bounds for arthur–merlin classes. In: Proceedings of the 39th Annual ACM Symposium on Theory of Computing, pp. 275–283 (2007)
19. Sipser, M.: A complexity theoretic approach to randomness. In: Proceedings of the 15th Annual ACM Symposium on Theory of Computing, pp. 330–335 (1983)
20. Stockmeyer, L.: On approximation algorithms for ♯P. SIAM Journal on Computing 14(4), 849–861 (1985)
21. Shaltiel, R., Umans, C.: Simple extractors for all min-entropies and a new pseudorandom generator. Journal of the ACM 52(2), 172–216 (2005)
22. Shaltiel, R., Viola, E.: Hardness amplification proofs require majority. In: Proceedings of the 40th Annual ACM Symposium on Theory of Computing, pp. 589–598 (2008)
23. Trevisan, L.: Construction of extractors using pseudo-random generators. Journal of the ACM 48(4), 860–879 (2001)
24. Trevisan, L., Vadhan, S.: Pseudorandomness and average-case complexity via uniform reductions. Computational Complexity 16(4), 331–364 (2007)
25. Umans, C.: Pseudo-random generators for all hardnesses. Journal of Computer and System Sciences 67(2), 419–440 (2003)
26. Yao, A.C.: Theory and applications of trapdoor functions. In: Proceedings of the 23rd Annual IEEE Symposium on Foundations of Computer Science, pp. 80–91 (1982)