Worst-Case to Average-Case Reductions Revisited

Dan Gutfreund[1] and Amnon Ta-Shma[2]

[1] SEAS, Harvard University,
Cambridge, MA 02138
danny@eecs.harvard.edu
[2] Computer Science Department,
Tel-Aviv University, Israel, 69978
amnon@post.tau.ac.il

Abstract. A fundamental goal of computational complexity (and foundations of cryptography) is to find a polynomial-time samplable distribution (e.g., the uniform distribution) and a language in NTIME($f(n)$) for some polynomial function f, such that the language is hard on the average with respect to this distribution, given that NP is worst-case hard (i.e. NP $\neq$ P, or NP $\not\subseteq$ BPP). Currently, no such result is known even if we relax the language to be in nondeterministic sub-exponential time.
There has been a long line of research trying to explain our failure in proving such worst-case/average-case connections [FF93,Vio03,BT03,AGGM06]. The bottom line of this research is essentially that (under plausible assumptions) non-adaptive Turing reductions cannot prove such results.
In this paper we revisit the problem. Our first observation is that the above mentioned negative arguments extend to a non-standard notion of average-case complexity, in which the distribution on the inputs with respect to which we measure the average-case complexity of the language, is only samplable in super-polynomial time. The significance of this result stems from the fact that in this non-standard setting, [GSTS05] did show a worst-case/average-case connection. In other words, their techniques give a way to bypass the impossibility arguments. By taking a closer look at the proof of [GSTS05], we discover that the worst-case/average-case connection is proven by a reduction that "almost" falls under the category ruled out by the negative result. This gives rise to an intriguing new notion of (almost black-box) reductions.
After extending the negative results to the non-standard average-case setting of [GSTS05], we ask whether their positive result can be extended to the standard setting, to prove some new worst-case/average-case connections. While we can not do that unconditionally, we are able to show that under a mild derandomization assumption, the worst-case hardness of NP implies the average-case hardness of NTIME($f(n)$) (under the uniform distribution) where f is computable in quasi-polynomial time.

1 Introduction

Proving that the worst-case hardness of NP implies the average-case hardness of NP, is a fundamental open problem in the fields of computational complex-

ity and foundations of cryptography (as it is a necessary step towards basing the existence of one-way functions on worst-case NP hardness). Bogdanov and Trevisan [BT03] (building on Feigenbaum and Fortnow [FF93]), show that "it is impossible (using non-adaptive reductions) to base the average-case hardness of a problem in NP or the security of a one-way function on the worst-case complexity of an NP complete problem (unless the polynomial hierarchy collapses)". This result is taken as demonstrating a major obstacle for showing a worst-case/average-case equivalence within NP.

Our first observation is that the arguments of [BT03] can be extended to a non-standard notion of average-case complexity, in which hardness is measured with respect to distributions that are samplable in super-polynomial time (rather than in fixed polynomial time):

Theorem 1. *Suppose that there is a language $L \in NP$ and a distribution $\mathcal{D}$ samplable in time $n^{\log n}$ such that there is a non-adaptive reduction from solving SAT on the worst-case to solving L on the average with respect to $\mathcal{D}$. Then every language in coNP can be computed by a family of nondeterministic Boolean circuits of size $n^{polylog(n)}$.*

Similar to the original result of [BT03], this should be taken as demonstrating a major obstacle for showing a worst-case/average-case equivalence within NP, even for this non-standard notion of average-case complexity. Nevertheless, Gutfreund, Shaltiel and Ta-Shma [GSTS05] do prove exactly the worst-case/average-case connection ruled out in Theorem 1 (by virtue of non-adaptive reductions).

Theorem 2. *[GSTS05] There exists a distribution $\mathcal{D}$ samplable in time $n^{\log n}$, such that if there exists a BPP algorithm solving SAT on the average with respect to $\mathcal{D}$, then there exists a BPP algorithm solving SAT on the worst-case.*

This surprising state of affairs gives rise to two questions. First, what can we learn from the proof of [GSTS05] about worst-case to average-case reductions, given that their technique bypasses the limitations imposed by Theorem 1? Second, after showing that the negative arguments can be extended to the non-standard notion of [GSTS05], can we turn the argument around and show that their positive result be extended and serve as a basis to prove new worst-case/average-case connections under the standard notion of average-case complexity?

Let us start with the first question. Looking at the proof of [GSTS05], we observe that it follows by constructing a distribution $\mathcal{D}$ as in the statement, and a fixed polynomial time machine R, s.t. for every probabilistic polynomial-time machine A solving SAT well on the average with respect to $\mathcal{D}$, the machine R^A (i.e. R with oracle access to A) solves SAT well in the worst case. In fact, the machine R is easy to describe, and unexpectedly turns out to be the familiar search-to-decision reduction for SAT.[1] I.e., given a SAT formula ϕ, R^A runs a

[1] A search to decision reduction for SAT uses an oracle that decides SAT to find a satisfying assignment for a given formula if such an assignment exist. The known

search to decision reduction on ϕ, where for each SAT query in the reduction it queries A. At the end R^A holds an assignment, and it accepts ϕ iff the assignment satisfies ϕ. Since the proof works with any search to decision reduction, we can use the non-adaptive search-to-decision reduction of [BDCGL90], and obtain what seems to be a non-adaptive worst-case to average-case reduction that by Theorem 1 implies that every language in coNP can be computed by a family of nondeterministic Boolean circuits of size $n^{polylog(n)}$. So did we prove an unexpected collapse?

The answer is of course no. Instead we showed a way to bypass the limitation imposed by Theorem 1. But to understand how, we need to give a careful look at the seemingly innocent term "reduction".

1.1 What is a Reduction?

The term "reduction" (or more precisely "Turing reduction") used in [BT03], and many other papers, is defined as follows. Suppose P and P' are two computational tasks. E.g., P might be solving SAT on the worst case, and P' might be solving SAT on the average with respect to the distribution $\mathcal{D}$. We say that P reduces to P' if there exists a probabilistic polynomial-time oracle machine R such that for *every* oracle A that solves P', R^A solves P. So in our example, P reduces to P' if there exists one fixed polynomial time machine R s.t. for every A solving SAT well on the average with respect to $\mathcal{D}$, the machine R^A solves SAT well in the worst case. From such a reduction, one in particular can deduce that if P' is easy (e.g., for BPP) then so does P. Reversing it, one may conclude that if P is hard then so does P'.

So does the proof of [GSTS05] uses a reduction? On the one hand, R is indeed a probabilistic, polynomial time oracle machine. However, the proof shows something weaker than a reduction regarding R: for *every probabilistic, polynomial-time* oracle machine A that solves P', R^A solves P. Namely, instead of showing that for every A that solves P', R^A solves P, it is only shown that for every *efficient* A this is the case. Thus, the argument of [GSTS05] is *not* a reduction. Nevertheless, it is still *useful*. That is, we can still conclude that if efficient machines cannot solve SAT on the worst-case then efficient machines cannot solve SAT on the average with respect to $\mathcal{D}$. The fact that we restrict the proof of correctness only to apply to efficient machines, does not make a difference since efficient machines is all that we care about!

We believe that this state of affairs calls for some new notation. First, to make matters more precise, we believe what we simply called "a reduction" should be called "a black-box reduction". This is because the key property captured in the definition is the black-box use of A by R (i.e., the reduction is oblivious to the actual solution of P') and the black-box use of A in the correctness proof (i.e., R^A is correct whenever A is correct, regardless of what A is). We then suggest a new kind of reduction:

reductions are either sequential and deterministic or parallel, non-adaptive and randomized [BDCGL90].

Definition 1 (Class-specific black-box reductions). *Let P, P' be two computational problems, and $\mathcal{C}$ a class of functions. We say that R is a $\mathcal{C}$-black-box reduction from P to P', if R is a probabilistic polynomial-time oracle machine such that for every oracle $A \in \mathcal{C}$ that solves P', R^A solves P.*

If R queries A non-adaptively, we say that the reduction is non-adaptive. If $\mathcal{C}$ is the class of all *functions, we simply say that R is a black-box reduction.*

Later we will also consider reductions that run in super-polynomial time, and we will state the running time explicitly when this is the case. If we do not state the running time then it is polynomial. Also, unless stated otherwise, whenever we say reduction, we mean black-box reduction.

Note that Definition 1 is only meaningful when the class $\mathcal{C}$ is more powerful than R. Otherwise R can run the oracle by itself. This is indeed the case in [GSTS05] where R runs in linear time and A runs in arbitrary polynomial time.

In Definition 1 the reduction R is still black-box in A, but now the correctness proof is not black-box in A, and works only when A comes from a bounded class. As we said before, normally a reduction is used to prove that if some task P' is easy then so does P (or the contra-positive: if P is hard then so does P'). Notice that for this purpose class black-box reductions are as useful as general reductions, because the argument is that if an *efficient* A solves P', then an efficient algorithm R^A solves P. The big advantage is that class black-box reductions are more flexible, and as Theorems 1 and 2 show, we can construct a class black-box reduction that is impossible to achieve with a general reduction (assuming the hierarchy does not collapse).

1.2 Back to Average-Case Complexity

We now turn to the second question we raised, whether we can leverage the result (or the proof) of [GSTS05] to prove a new worst-case/average-case connection under the standard notion of average-case complexity. Indeed we are able to show such a result but only assuming an unproven derandomization assumption. Before we state and discuss our result, some background is needed.

Most cryptographic primitives require at least the existence of One-Way Functions (OWFs) [IL89]. A function $f : \{0,1\}^* \to \{0,1\}^*$ is one-way if f is computable in a fixed polynomial time, and, for every constant $c > 0$, for every polynomial time algorithm A, for all large enough input lengths, $\Pr_{y \in f(U_n)} [\, A(y) \in f^{-1}(y) \,] < \frac{1}{n^c}$. In words, f can be computed by an algorithm that runs in some fixed polynomial time, while f^{-1} is hard (on the average) for *all* polynomial-time algorithms.

It is a common belief that OWFs exist, and a standard assumption in cryptography. The question whether it can be based on the worst-case hardness of NP is a major open problem. There are several relevant notions of hardness involved, and we identify the following hierarchy of conjectures:

- (Worst-case) Some function in NP is worst-case hard for a class of adversaries (usually BPP).

– (Average-case) Some function in NP is average-case hard for a class of adversaries (usually BPP).
– (One-way) Some function in P is hard to invert on average by a class of adversaries (usually BPP).
– (Pseudo-randomness) Some function in P generates distributions of low entropy that are indistinguishable from uniform by a class of adversaries (usually BPP).

The only non-trivial relation among these assumptions that is known today is that the one-wayness assumption is equivalent to the pseudo-randomness assumption when the class of adversaries is BPP [HILL99].

In all the above assumptions we assume that some function "fools" some class of adversaries, where the meaning of "fooling" varies (being worst-case hardness, average-case hardness, one-wayness or pseudo-randomness). The usual choice in cryptography is that the function lies in P or NP (according to the assumption we work with), while the class of adversaries is BPP (or sometimes even BPTIME($t(n)$) for some super-polynomial $t(n)$). Thus, while the function is computable in a fixed polynomial time, the adversary may run in unbounded polynomial time, and so *has more resources than the algorithm for the function itself*. We therefore call this setting the "weak-fooling-strong" setting.

Another setting that often arises in complexity, is almost identical to the above except that the class of adversaries is *weaker* than the class it tries to fool. E.g., a key observation of Nisan and Wigderson [NW94] is that generators that are used to derandomize probabilistic complexity classes, can run in time n^c while fooling algorithms running in time n^b for some $b \ll c$. We call this the "strong-fooling-weak" setting.

This difference is a crucial (though subtle) dividing line. The canonic example for this distinction is the Blum-Micali-Yao PRG v.s. the Nisan-Wigderson PRG. This distinction applies not only to PRGs, and in particular we believe it is a central issue when discussing worst-case to average-case reductions for NP. Indeed the techniques that we use, apply to the strong-fooling-weak setting, but not to the weak-fooling-strong setting.

So now we focus on the strong-fooling-weak setting, and review previous work on the problem. We begin by listing the exponential-time analogues of the assumptions we listed above:

– (Worst-case hardness in EXP) Some function in EXP is worst-case hard for BPP.
– (Average-case hardness in EXP) Some function in EXP is average-case hard for BPP.
– (Exponential-time pseudo-random generators) For every constant $\epsilon > 0$, there exists a pseudorandom generator $G : n^\epsilon \to n$ fooling BPP, and the generator is computable in time 2^{n^ϵ} (i.e. exponential in the seed length).

Note that each of the above statements is implied by the corresponding statement in the weak-fooling-strong setting. Impagliazzo and Wigderson [IW98] (see

also [TV02]), building on a long line of works such as [NW94,BFNW93,IW97], show that all three assumptions are equivalent. Their work was done in the context of understanding the power of randomness in computation, and indeed the equivalence above easily extends to include the following statement about the ability to reduce the amount of randomness used by efficient probabilistic algorithms.

– (Subexponential-time derandomization) For every probabilistic polynomial-time TM A (that outputs one bit), and a constant $\epsilon > 0$, there exists another probabilistic TM A^ϵ, that runs in time 2^{n^ϵ}, uses at most n^ϵ random coins, and behaves essentially the same as A.[2]

These beautiful and clean connections shed light on some of the fundamental questions in complexity theory regarding randomness and computational hardness of functions. Unfortunately, no such connections are known below the exponential level. As an example consider the following question that lies in the strong-fooling-weak setting, yet in the sub-exponential regime.

Open Problem 1. *Does NP $\not\subseteq$ BPP imply the existence of a language L in $\widetilde{NP} = NTIME(n^{O(\log n)})$ such that L is hard on average for BPP with respect to the uniform distribution?*

We believe that solving Open Problem 1 will be a major breakthrough. Here we give an affirmative answer under a weak derandomization assumption. We begin with the derandomization assumption. Following Kabanets [Kab01], we say that two probabilistic TM's are δ-indistinguishable if no samplable distribution can output with δ probability an instance on which the answers of the machines differ significantly (the acceptance probabilities, averaging over their randomness, differ by at least δ). See Definition 3 for a formal definition. We can now formalize the derandomization hypothesis:

Hypothesis 1. *For every probabilistic polynomial-time decision algorithm A (that outputs one bit), and every constant $\epsilon > 0$, there exists another probabilistic polynomial-time algorithm A^ϵ that on inputs of length n, tosses at most n^ϵ coins, and A, A^ϵ are $\frac{1}{100}$-indistinguishable.*

We then prove:

Theorem 3. *(Informal) If NP is worst-case hard and weak derandomization of BPP is possible (i.e. Hypothesis 1 is true) then there exists a language in $\widetilde{NP}$ that is hard on average for BPP.*[3]

[2] Here we mean that A^ϵ maintains the functionality of A *on the average* in the sense of Kabanets [Kab01] (see Definition 3 and Hypothesis 1). This notion of derandomization is standard when working with hardness against uniform TM's (rather than circuits) and with generators that fool TM's (rather than circuits).

[3] Our result is actually slightly stronger than the one stated here. It gives a hard on average language in the class NTIME($n^{\omega(1)}$) with the additional constraint that

The formal statement is given in Section 4 (Theorem 7). The weak derandomization assumption is the precise (polynomial-time) analogue of the subexponential-time derandomization assumption stated above. That is, in both assumptions, A^ϵ reduces the randomness of A by a polynomial factor, and the result is indistinguishable to polynomial time adversaries (see Definition 3). However, in the latter we let A^ϵ run in subexponential-time, while in Theorem 3 we demand it runs in polynomial time.

We remark that there is an almost trivial proof of the theorem if we replace the weak derandomization assumption by a strong derandomization assumption, namely that BPP = P. In our notation, this is like assuming that A^ϵ can replace the $poly(n)$ random coins of A by logarithmically many random coins. This assumption is much stronger than the assumption in Theorem 3, where we assume A^ϵ reduces the number of coins just by a polynomial factor. Indeed, under the strong assumption, we can apply standard hierarchy theorems to separate $\widetilde{\text{NP}}$ from BPP (which is now P) even under average-case complexity measure. Note however, that our weak derandomization does not imply that BPP is (strictly) contained in $\widetilde{\text{NP}}$ and therefore we cannot apply hierarchy theorems.

Another strong assumption that implies the average-case hardness in our conclusion, is the existence of ceratin pseudo-random generators. Again, our assumption is much weaker than that since it only states that derandomization is possible (not necessarily via a construction of pseudo-random generators).

We therefore believe that the relationship that we obtain between worst-case hardness, average-case hardness and derandomization, is intriguing as it shows that highly non-trivial connections between these notions do exist below the exponential level.

2 Preliminaries

BPTIME$(t(n), c(n))$ is the class of languages that can be decided by randomized Turing machines that run in time $t(n)$ and use $c(n)$ random coins. NTIME$(t(n), w(n))$ is the class of languages that can be decided by nondeterministic Turing machines that run in time $t(n)$, and take witnesses of length $w(n)$. BPTIME$(t(n))$ and NTIME$(t(n))$ stand for BPTIME$(t(n), t(n))$ and NTIME$(t(n), t(n))$ respectively. PPM denotes the class of probabilistic polynomial-time TM's.

An ensemble of distributions $\mathcal{D}$ is an infinite set of distributions $\{\mathcal{D}_n\}_{n\in N}$, where $\mathcal{D}_n$ is a distribution over $\{0,1\}^n$. We denote by $\mathcal{U}$ the uniform distribution. Let $A(\cdot;\cdot)$ be a probabilistic TM, using $m(n)$ bits of randomness on inputs of length n. We say that A is a sampler for the distribution $\mathcal{D} = \{\mathcal{D}_n\}_{n\in N}$, if for every n, the random variable $A(1^n, y)$ is distributed identically to $\mathcal{D}_n$, where the distribution is over the random string $y \in_R \{0,1\}^{m(n)}$. In particular, A

membership witnesses are of *polynomial* length, i.e. only the verification takes super-polynomial time, making it closer to NP. In particular, this class is contained in EXP. Also, standard separation techniques (such as the nondeterministic time hierarchy) can not separate this class from NP.

always outputs strings of length n on input 1^n. If A runs in polynomial time we simply say $\mathcal{D}$ is *samplable*. A *distributional problem* is a pair $(L, \mathcal{D})$, where L is a language and $\mathcal{D}$ is an ensemble of distributions.

Definition 2 (Average BPP). *Let $(L, \mathcal{D})$ be a distributional problem, and $s(n)$ a function from N to $[0, 1]$. We say that $(L, \mathcal{D})$ can be* efficiently decided on average *with success $s(n)$, and denote it by $(L, \mathcal{D}) \in Avg_{s(n)}BPP$, if there is an algorithm $A \in PPM$ such that for every large enough n, $\Pr[A(x) = L(x)] \geq s(n)$, where the probability is over an instance $x \in \{0, 1\}^n$ sampled from $\mathcal{D}_n$ and the internal coin tosses of A.*

We mention that the average-case definition that we give here is from [Imp95] (there it is denoted Heur-BPP). It differs from the original definition of Levin [Lev86]. Generally speaking, all previous works about hardness amplification and worst-case to average-case reductions, and in particular those that we use here (e.g. [BT03,IL90,GSTS05]), hold under Definition 2.

We denote the computational problem of deciding $(L, \mathcal{D})$ on average with success s by $(L, \mathcal{D})_s$. For a language L defined by a binary relation $R \subseteq \{0, 1\}^* \times \{0, 1\}^*$ (i.e. $L = \{x : \exists y \text{ s.t. } (x, y) \in R\}$), the *search problem* associated with L is given x find y such $(x, y) \in R$ if such a y exist (i.e. if $x \in L$) and output 'no' otherwise. The average-case analogue of solving the search problem of $(L, \mathcal{D})$ with success s, is to solve the search problem of L with probability at least s, over an instance of L drawn from $\mathcal{D}$ (and the internal coins of the search process). We denote this computational problem by $(L, \mathcal{D})_{search,s}$.

We need to define a non-standard (and weaker) solution to search problems, by letting the searching procedure output a list of candidate witnesses rather than one, and not requiring that the algorithm recognize 'no' instances.[4] For a langauge L defined by a binary relation R, the *list-search problem* associated with L is given x find a list $y_1, \ldots, y_m$ (where $m = \text{poly}(|x|)$) such that $\exists i \in [m]$ for which $(x, y_i) \in R$, if $x \in L$. Note that for $x \notin L$ we are not required to answer 'no'. We denote by $(L, \mathcal{D})_{list-search,s}$ the average-case analogue.

Indistinguishability. Following Kabanets [Kab01], we say that two probabilistic TM's are indistinguishable if no samplable distribution can output with high probability an instance on which the answers of the machines differ significantly (averaging over their randomness). Below is the formal definition.

Definition 3. *Let A_1 and A_2 be two probabilistic TM's outputting 0/1, such that on inputs of length n A_1 uses $m_1(n)$ random coins and A_2 uses $m_2(n)$ random coins. For $\epsilon, \delta > 0$, we say that A_1 and A_2 are (ϵ, δ)-indistinguishable,*

[4] The reason we need this is that we are going to apply search procedures on languages in $\text{NTIME}(n^{\omega(1)}, \text{poly}(n))$. In this case an efficient procedure cannot check whether a candidate witness is a satisfying one. We therefore cannot amplify the success probability of such procedures. On the other hand, when we only require a list that contains a witness we can apply standard amplification techniques.

if for every samplable distribution $\mathcal{D} = \{\mathcal{D}_n\}_{n\in N}$ *and every* $n \in N$,

$$\Pr_{x\in\mathcal{D}_n}\left[\left|\Pr_{r\in_R\{0,1\}^{m_1(n)}}[A_1(x,r)=1] - \Pr_{r'\in_R\{0,1\}^{m_2(n)}}[A_2(x,r')=1]\right| > \epsilon\right] \leq \delta$$

To save on parameters, we will sometimes take ϵ *to be equal to* δ *and then we will say that* A_1, A_2 *are* δ*-indistinguishable (meaning* (δ,δ)*-indistinguishable).*

3 Impossibility Results and How to Bypass Them

We begin with showing that under standard assumptions, the [GSTS05] result cannot be proven via black-box and non-adaptive reductions. We then take a closer look at the reduction of [GSTS05], in order to understand what enables it to bypass the impossibility result.

The following statement can be obtained by generalizing the proof of [BT03] to arbitrary time bounds.

Theorem 4 (Implicit in [BT03]). *Suppose that there is a language* $L \in NTIME(n^{O(\log n)})$ *and a reduction* R *from solving SAT on the worst-case, to solving* $(L,\mathcal{U})$ *on the average with success* $1 - 1/n^{O(\log n)}$*. Further suppose that* R *is non-adaptive and black-box, and is computable in time* $n^{polylog(n)}$*. Then every language in coNP can be computed by a family of nondeterministic Boolean circuits of size* $n^{polylog(n)}$*.*

From that we can conclude Theorem 1, which we now re-state more formally (the proof is omitted due to space limitation).

Theorem 5. *Suppose there is a language* $L \in NP$ *and a distribution* $\mathcal{D}$ *samplable in time* $n^{\log n}$*, such that there is a black-box and non-adaptive reduction from solving SAT on the worst-case, to solving* $(L,\mathcal{D})$ *on the average with success* $1-1/n$*. Then every language in coNP can be computed by nondeterministic Boolean circuits of size* $n^{polylog(n)}$*.*

3.1 A Closer Look at the Reduction of [GSTS05]

There are two steps in the argument of [GSTS05]. First it is shown that assuming NP $\not\subseteq$ BPP, any probabilistic, polynomial-time algorithm BSAT for SAT has a hard polynomial-time samplable distribution $\mathcal{D}_{\mathrm{BSAT}}$, and then they show one quasi-polynomial time distribution $\mathcal{D}$ that is hard for all polynomial-time, probabilistic algorithms.

We now recall how the first step is achieved. Given BSAT we define the probabilistic polynomial time algorithm SSAT that tries to solve the search problem of SAT using oracle calls to BSAT (via the downwards self-reducibility property of SAT), and answers "yes" if and only if it finds a satisfying assignment. We then define the SAT formula:

$$\exists_{x\in\{0,1\}^n} \quad [\ \mathrm{SAT}(x) = 1 \ \text{ and } \ \mathrm{SSAT}(x) \neq \text{'yes'}\ ] \tag{1}$$

The assumption NP $\not\subseteq$ BPP implies that SSAT does not solve SAT and the sentence is true. We now ask SSAT to find a satisfying assignment to it. If it fails doing so, then BSAT is wrong on one of the queries made along the way. Otherwise, the search algorithm finds a SAT sentence x on which SSAT is wrong. This means that BSAT is wrong on one of the queries SSAT makes on input x. Things are somewhat more complicated because SSAT is a probabilistic algorithm and not a deterministic one, and so the above sentence is not really a SAT formula, but we avoid these technical details and refer the interested reader to [GSTS05]. In any case, we produce a small set of queries such that on at least one of the sentences in the set, BSAT is wrong, i.e., we get a polynomial-time samplable distribution on which BSAT has a non-negligible error probability.

To implement the second step, [GSTS05] define the distribution $\mathcal{D}$ that on input 1^n picks at random a machine from the set of probabilistic machines with description size at most, say, $\log n$ and runs it up to, say, $n^{\log n}$ time. We know that for any probabilistic polynomial-time algorithm BSAT there is a hard distribution $\mathcal{D}_{\mathrm{BSAT}}$, and this distribution is sampled by some polynomial-time algorithm with a fixed description size. Thus, for n large enough, we pick this machine with probability at least $1/n$ (because its description size is smaller than $\log n$) and then we output a bad input for BSAT with probability $1/n$. The sampling time for $\mathcal{D}$ is $n^{\log n}$ (or, in fact, any super-polynomial function).

We now ask: Can we interpret the [GSTS05] result as a worst-case to average-case reduction?

Indeed, given an algorithm BSAT for solving $(\mathrm{SAT}, \mathcal{D})$, we define the algorithm $R^{\mathrm{BSAT}} = \mathrm{SSAT}$, where R is a search-to-decision reduction. The analysis shows that if BSAT indeed solves SAT well on the average with respect to $\mathcal{D}$, then it also solves it well on the average with respect to $\mathcal{D}_{\mathrm{BSAT}}$. This implies that necessarily the sentence in Eq (1) is true, i.e., SSAT solves SAT in the worst-case. In other words, the standard search to decision reduction for SAT is also a worst-case to average-case reduction!

Another question is now in place: is the reduction black-box?

Looking at Definition 1 we see that a reduction is *black-box* if it has the following *two* properties:

1. (Property 1) R makes a black-box use of the adversary A (in our case BSAT). I.e., R may call A on inputs but is not allowed to look into the code of A.
2. (Property 2) The reduction is correct for any A that solves the problem (in our case SAT), putting no limitations on the nature of A. E.g. A may even be undecidable.

We see that in the reduction of [GSTS05], the first condition is satisfied. R is merely the standard search-to-decision reduction for SAT which queries the decision oracle on formulas along a path of the search tree. We can replace the standard search-to-decision reduction with the one by Ben-David et. al. [BDCGL90]. The latter makes only non-adaptive queries to the decision oracle. Thus we get a non-adaptive reduction. However, the second condition is violated. Indeed, the key point in the *analysis* of [GSTS05] is that it works only for *efficient*

oracles BSAT. This is so because the analysis encodes the failure of R^{BSAT} as an NP statement. Here the analysis crucially uses the fact that BSAT (and therefore R^{BSAT}) is in BPP, and therefore its computation has a short description as a Boolean formula.

So let us now summarize this surprising situation: from the reduction R's point of view, it is black-box, i.e. Property 1 holds (R does not rely on the inner working of the oracle BSAT or its complexity), but for the *analysis* to work, the oracle BSAT has to be efficient, i.e. Property 2 is violated.[5] This motivates the definition of class-specific black-box reductions that we gave in the introduction.

Given this definition and the discussion about non-adaptivity above, we can restate the result of [GSTS05] as follows:

Theorem 6. *There exists a distribution $\mathcal{D}$ samplable in time $n^{\log n}$, such that there is a BPP-black-box and non-adaptive reduction from solving SAT on the worst-case to solving $(L, \mathcal{D})$ on average with success $1 - 1/n$.*

Theorem 6 is in sharp contrast to Theorem 5. Theorem 5, as well as [FF93,BT03,AGGM06], say that the requirement in black-box reductions that they succeed whenever they are given a "good" oracle, *regardless of its complexity*, is simply too strong, i.e. such reductions are unlikely to exist. Theorem 6, on the other hand, says that weakening the requirement to work only for efficient oracles (i.e. allowing to violate Property 2, but not property 1) is enough to bypass the limitation.

We mention that there are other cases of non-black-box reductions in complexity theory that bypass black-box limitations. For example, the fact that the polynomial-time hierarchy collapses under the assumption NP = P is proven via a non-black-box reduction from solving SAT efficiently to solving $\mathrm{Q}SAT_2$ efficiently ($\mathrm{Q}SAT_2$ is the canonic Σ_2-complete problem of deciding the validity of a Boolean first-order formula with two quantifier alternations). Indeed if a black-box reduction between these tasks exists then the hierarchy collapses unconditionally. It is interesting, however, that in this argument both the reduction and the proof of correctness are non-black-box, because the reduction queries the NP-oracle on statements that are encodings of the computation of the oracle itself (assuming that this oracle can be realized efficiently). I.e. both Properties 1 and 2 are violated. Examples from the field of cryptography can be found in the work of Barak [Bar01] (and some following papers that use similar ideas). Again his proof (when considered as a reduction) violates both properties of black-boxness.

The only other BPP-black-box reduction that we are aware of appears in the work of Imapgliazzo and Wigderson [IW98].[6] Their proof shows that given an

[5] We mention that recently, Atserias [Ats06] gave an alternative proof to [GSTS05] where he shows that even the analysis can be done almost black-box. That is, it does not need to use the description of BSAT, it only needs to know the *running time* of BSAT. In contrast, the analysis in [GSTS05] does use the description of BSAT.

[6] Although some proofs of security in cryptography appear to use the fact that the adversary is efficient (e.g. in zero-knowledge proofs with black-box simulation), when written as reductions they are in fact black-box in the standard sense. That is, it is

efficient oracle that breaks a ceratin pseudo-random generator, one can use it to compute an EXP-complete language. We do not know if this reduction can be done in a black-box way, nor do we have evidence that it cannot.

Finally, we want to mention that one should not confuse black-box limitations with non-relativizing arguments. The proof of [GSTS05] (as well as the collapse of the polynomial-time hierarchy) *can* be done in a relativizing way.

4 Top-Down Overview of Theorem 3

We re-state Theorem 3 in a formal way.

Theorem 7. *If Hypothesis 1 is true and SAT $\notin$ RP then there exists a language $L \in NTIME(t(n), poly(n))$, such that, $(L,\mathcal{U}) \notin Avg_{1/2+1/\log^{\alpha} n}BPP$, where $t(n) = n^{\omega(1)}$ is an arbitrary time-constructible super-polynomial function, and $\alpha > 0$ is a universal constant.*

We now explain the intuition and give a top-down overview of the proof. The full proof is omitted due to space limitations and will appear in the final version of the paper.

Our starting point is Theorem 2, which says that if SAT is worst-case hard, then there exist a single distribution on SAT instances, $\mathcal{D}_{hard}$, on which every probabilistic polynomial-time algorithm errs with relatively high probability, but the distribution is only samplable in quasi-polynomial-time (for this reason we denote it by $\mathcal{D}_{hard}$). Our goal is to somehow extract from $\mathcal{D}_{hard}$ a *simple* distribution on which the same holds. Ideally, the uniform distribution will be good enough. The key tool that we use is a reduction given by Impagliazzo and Levin [IL90] that shows that if there exists a *polynomial-time* samplable distribution $\mathcal{D}$ that is hard on average for some language $L \in$ NP, then there exists another language $L' \in$ NP for which the *uniform distribution* $\mathcal{U}$ is hard on average. We would like to apply this reduction on SAT and the distribution $\mathcal{D}_{hard}$.

However we immediately run into a problem because $\mathcal{D}_{hard}$ is samplable in super-polynomial time, while the argument of [IL90] only applies to distributions that are samplable in polynomial time. To understand this, let us elaborate on how the complexity of the distribution influences the reduction of [IL90]. There are several different entities to consider in this reduction: The language L, The distribution $\mathcal{D}_{hard}$, The language L' we reduce to, and the reduction R itself that solves $(L, \mathcal{D}_{hard})$ on average given an oracle that solves $(L',\mathcal{U})$ on average.

We can expect that both the complexity of L' as well as the complexity of the reduction R depend on $\mathcal{D}_{hard}$. Indeed, using the [IL90] reduction, the

shown that *any* adversary that breaks the cryptographic primitive, implies breaking the security assumption (e.g. bit-commitments in the case of zero-knowledge). Of course, the contradiction with the security assumption is only true when the adversary is efficient. However, this is an artifact of the security assumption, not the way the proof is derived (or in other words, if we change the security assumption to hold against say, sub-exponential-time adversaries rather than polynomial-time adversaries, the same proof of security holds).

nondeterministic procedure for the language L' involves checking membership in the language L as well as running the sampler for $\mathcal{D}_{hard}$. In our case, since $\mathcal{D}_{hard}$ is samplable in super-polynomial-time, this results in L' having a super-polynomial non-deterministic complexity (and so puts us in the strong-fooling-weak setting).

It seems that the same should hold for the reduction R. Indeed the reduction of [IL90] is from search problems to search problems, which means that R must handle the membership witnesses for the language L'. As we said above, this involves computing $\mathcal{D}_{hard}$ and in particular, the complexity of R is at least that of $\mathcal{D}_{hard}$. This however means that we can not deduce from the hardness on average of $(L, \mathcal{D}_{hard})$ the hardness on average of $(L', \mathcal{U})$, because the hardness of $(L, \mathcal{D}_{hard})$ is only against algorithms with complexity smaller than that of $\mathcal{D}_{hard}$, and the reduction's complexity is at least that of $\mathcal{D}_{hard}$. This makes the reduction of [IL90] useless for us (at least a direct use of it).

The surprising thing, and the main observation here, is that in the Impagliazzo-Levin argument the running time of the reduction does not depend on the time complexity of $\mathcal{D}_{hard}$! It only depends on the number of random coins the sampler for $\mathcal{D}_{hard}$ uses: while the reduction R does look at the membership witnesses for L', the size of these witnesses is only a function of the number of random coins the sampler uses. Furthermore, during this process, R is never required to verify the witnesses and therefore does not need to run the sampler. We formalize this observation in the following lemma.

Lemma 1. *For every Distribution $\mathcal{D}$ samplable in $BPTIME(t(n), c(n))$, a language $L \in NTIME(t_L(n), c_L(n))$ and $0 < \delta(n) < 1$, there exists $L' \in NTIME(t(n) + t_L(n) + poly(n, c(n)), c(n) + c_L(n))$ such that there is a probabilistic non-adaptive reduction, R, from $(L, \mathcal{D})_{search, 1-O(\delta(n)\cdot c^2(n))}$ to $(L', \mathcal{U})_{list-search, 1-\delta(n)}$. Furthermore, the running time of R is $poly(n, c(n), t_L(n))$, and note that it is independent of $t(n)$.*

The conclusion of the Lemma is that in order to keep the reduction efficient, we need to reduce the randomness complexity of the sampler for $\mathcal{D}_{hard}$ to a fixed polynomial, but we do not need to reduce its time complexity. This is fortunate because while in general there is no reason to believe that we can reduce the running time of algorithms, it is widely believed that randomness can be reduced without paying much penalty in running time. To that end we use Hypothesis 1, and prove:

Lemma 2. *Assume Hypothesis 1 is true. Let $t(n)$ be an arbitrary super-polynomial function. There is a distribution $\mathcal{D}$ samplable in $BPTIME(O(t(n), O(n^3))$ such that, $(SAT, \mathcal{D}) \in Avg_{1-1/n}BPP \Rightarrow SAT \in RP$.*

Note that Hypothesis 1 does not seem to derandomize general distributions that are samplable in super-polynomial-time. First, Hypothesis 1 only derandomizes polynomial-time algorithms and only by polynomial factors (while here we want to derandomize a sampler that runs in super-polynomial time and uses a super-polynomial number of coins). And second, Hypothesis 1 only applies

to decision algorithms.[7] We show, however, that the specific distribution $\mathcal{D}_{hard}$ from [GSTS05] can be derandomized under Hypothesis 1. The proof is quite technical and involves getting into the details of [GSTS05].

The above two lemmas give us $1 - 1/\text{poly}(n)$ hardness on the average for the list-search version of the problem (given the worst-case hardness and the derandomization assumptions). To get $1/2 + 1/\log^{\alpha} n$ hardness on the average for the decision problem, we use generalizations of known techniques in average-case complexity [BDCGL90,Tre05]. The tricky part is doing the hardness amplification using a reduction whose running time is $poly(n, c(n))$ and, in particular, independent of $t(n)$. By using careful generalizations of [BDCGL90], Trevisan's amplification technique [Tre05] goes through, and we obtain Theorem 7.

Acknowledgements

We thank Ronen Shaltiel and Salil Vadhan for many helpful discussions. Andrej Bogdanov for pointing out to us that Theorem 5 follows directly from a generalization of [BT03] (i.e. Theorem 4). Previously we had a more complicated and weaker statement. Charlie Rackoff for suggesting the relevance of the argument that the polynomial-time hierarchy collapses under P = NP, Alex Healy for commenting on the manuscript, and Oded Goldreich for a helpful conversation.

The first author is supported by ONR grant N00014-04-1-0478 and NSF grant CNS-0430336. The second author is supported by the Israel Science Foundation grant 217/05, the Binational Science Foundation, and the EU Integrated Project QAP.

References

[AGGM06] A. Akavia, O. Goldreich, S. Goldwasser, and D. Moshkovitz. On basing one-way functions on NP-hardness. In *Proceedings of the 38th Annual ACM Symposium on Theory of Computing*, pages 701–710, 2006.

[Ats06] A. Atserias. Distinguishing SAT from polynomial-size circuits through black-box queries. In *Proceedings of the 21th Annual IEEE Conference on Computational Complexity*, pages 88–95, 2006.

[Bar01] B. Barak. How to go beyond black-box simulation barrier. In *Proceedings of the 43rd Annual IEEE Symposium on Foundations of Computer Science*, pages 106–115, 2001.

[BDCGL90] S. Ben-David, B. Chor, O. Goldreich, and M. Luby. On the theory of average case complexity. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing*, pages 379–386, 1990.

[BFNW93] L. Babai, L. Fortnow, N. Nisan, and A. Wigderson. BPP has subexponential simulation unless Exptime has publishable proofs. *Computational Complexity*, 3:307–318, 1993.

[7] In general, standard derandomization results do not apply to probabilistic procedures that output many bits. We refer the reader to [DI06] for a discussion about derandomizing procedures that output many bits versus decision procedures (that output a single bit).

[BT03] 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.

[DI06] B. Dubrov and Y. Ishai. On the randomness complexity of efficient sampling. In *Proceedings of the 38th Annual ACM Symposium on Theory of Computing*, pages 711–720, 2006.

[FF93] J. Feigenbaum and L. Fortnow. Random-self-reducibility of complete sets. *SIAM Journal on Computing*, 22:994–1005, 1993.

[GSTS05] 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. In *Proceedings of the 20th Annual IEEE Conference on Computational Complexity*, pages 243–257, 2005.

[HILL99] J. Håstad, R. Impagliazzo, L. Levin, and M. Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28(4):1364–1396, 1999.

[IL89] R. Impagliazzo and M. Luby. One-way functions are essential for complexity based cryptography. In *Proceedings of the 30th Annual IEEE Symposium on Foundations of Computer Science*, pages 230–235, 1989.

[IL90] R. Impagliazzo and L. Levin. No better ways of finding hard NP-problems than picking uniformly at random. In *Proceedings of the 31st Annual IEEE Symposium on Foundations of Computer Science*, pages 812–821, 1990.

[Imp95] R. Impagliazzo. A personal view of average-case complexity. In *Proceedings of the 10th Annual Conference on Structure in Complexity Theory*, pages 134–147, 1995.

[IW97] R. Impagliazzo and A. Wigderson. P = BPP if E requires exponential circuits: Derandomizing the XOR lemma. In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing*, pages 220–229, 1997.

[IW98] R. Impagliazzo and A. Wigderson. Randomness vs. time: de-randomization under a uniform assumption. In *Proceedings of the 39th Annual IEEE Symposium on Foundations of Computer Science*, pages 734–743, 1998.

[Kab01] V. Kabanets. Easiness assumptions and hardness tests: Trading time for zero error. *Journal of Computer and System Sciences*, 63 (2):236–252, 2001.

[Lev86] L. Levin. Average case complete problems. *SIAM Journal on Computing*, 15 (1):285–286, 1986.

[NW94] N. Nisan and A. Wigderson. Hardness vs. randomness. *Journal of Computer and System Sciences*, 49:149–167, 1994.

[Tre05] L. Trevisan. On uniform amplification of hardness in NP. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing*, pages 31–38, 2005.

[TV02] L. Trevisan and S. Vadhan. Pseudorandomness and average-case complexity via uniform reductions. In *Proceedings of the 17th Annual IEEE Conference on Computational Complexity*, pages 129–138, 2002.

[Vio03] E. Viola. Hardness vs. randomness within alternating time. In *Proceedings of the 18th Annual IEEE Conference on Computational Complexity*, pages 53–62, 2003.