Using Nondeterminism to Amplify Hardness

Alexander Healy *   Salil Vadhan †   Emanuele Viola ‡

Harvard University

ABSTRACT

We revisit the problem of hardness amplification in $\mathcal{NP}$, as recently studied by O'Donnell (STOC '02). We prove that if $\mathcal{NP}$ has a balanced function f such that any circuit of size $s(n)$ fails to compute f on a $1/\operatorname{poly}(n)$ fraction of inputs, then $\mathcal{NP}$ has a function f' such that any circuit of size $s'(n) = s(\sqrt{n})^{\Omega(1)}$ fails to compute f' on a $1/2 - 1/s'(n)$ fraction of inputs. In particular,

1. If $s(n) = n^{\omega(1)}$, we amplify to hardness $1/2 - 1/n^{\omega(1)}$.
2. If $s(n) = 2^{n^{\Omega(1)}}$, we amplify to hardness $1/2 - 1/2^{n^{\Omega(1)}}$.
3. If $s(n) = 2^{\Omega(n)}$, we amplify to hardness $1/2 - 1/2^{\Omega(\sqrt{n})}$.

These improve the results of O'Donnell, which only amplified to $1/2 - 1/\sqrt{n}$. O'Donnell also proved that no construction of a certain general form could amplify beyond $1/2 - 1/n$. We bypass this barrier by using both *derandomization* and *nondeterminism* in the construction of f'.

We also prove impossibility results demonstrating that both our use of nondeterminism and the hypothesis that f is balanced are necessary for "black-box" hardness amplification procedures (such as ours).

Keywords. average-case complexity, hardness amplification, pseudorandom generators for space-bounded computation, noise stability

*Email: ahealy@fas.harvard.edu Research supported in part by NSF grant CCR-0205423.

†Email: salil@eecs.harvard.edu Research supported by NSF grant CCR-0133096, a Sloan Research Fellowship, and US-Israel BSF grant 2002246. Work done in part while at the Radcliffe Institute for Advanced Study at Harvard University.

‡Email: viola@eecs.harvard.edu Research supported by NSF grant CCR-0133096 and US-Israel BSF grant 2002246.

STOC'04, June 13–15, 2004, Chicago, Illinois, USA.

1. INTRODUCTION

Average-case complexity is a fundamental topic in complexity theory, whose study has at least two distinct motivations. On one hand, it may provide more meaningful explanations than worst-case complexity about the intractability of problem instances actually encountered in practice. On the other hand, it provides us with methods to generate hard instances, allowing us to harness intractability for useful ends such as cryptography and derandomization.

One of the goals of this area is to establish connections between average-case complexity and worst-case complexity, since the latter is much better-understood. While this has been accomplished for high complexity classes such $\#\mathcal{P}$ and $\mathcal{EXP}$ (e.g. [17, 3, 2, 7, 6, 23, 25, 26]), it remains a major open question for $\mathcal{NP}$. In fact, there are results showing that such connections for $\mathcal{NP}$ are unlikely to be provable using the same kinds of techniques used for the high complexity classes [8, 26, 5].

A more modest goal is "hardness amplification", where we seek to establish connections between "mild" average-case complexity and "strong" average-case complexity. That is, given a problem for which a nonnegligible fraction of inputs are "hard", can we obtain a problem for which almost all inputs are hard? To make this precise, let's define "hard".

Definition 1.1. *For $\alpha \in [0, 1/2]$, a function $f : \{0,1\}^n \to \{0,1\}$ is* α-hard for size s *if every circuit of size s fails to compute f on at least an α fraction of inputs.*

Note that the maximum value of the hardness parameter α is $1/2$ because f is boolean (so can trivially be computed with error probability at most $1/2$.)

The *hardness amplification problem* is to convert a function f that is δ-hard for size s to a function f' that is $(1/2 - \epsilon)$-hard for size polynomially related to s. Typically, $\delta = 1/\operatorname{poly}(n)$ and the aim is to make $\epsilon = \epsilon(n)$ vanish as quickly as possible.

The standard approach to hardness amplification employs Yao's XOR Lemma [10]: Given a mildly hard-on-average function $f : \{0,1\}^n \to \{0,1\}$, we define

$$f'(x_1, \ldots, x_k) \stackrel{\text{def}}{=} f(x_1) \oplus f(x_2) \oplus \cdots \oplus f(x_k)$$

for some $k = \operatorname{poly}(n)$. The XOR Lemma says that the hardness of f' approaches $1/2$ exponentially fast with k. More precisely:

Yao's XOR Lemma. *If f is δ-hard for size s, then f' is $(1/2 - 1/2^{\Omega(\delta k)} - 1/s')$-hard for size $s' = s^{\Omega(1)}$.*

In particular, taking k to be a sufficiently large polynomial in n, the amplified hardness is dominated by the $1/s'$ term. That is, we can amplify to $(1/2-\epsilon)$, where ϵ is polynomially related to the circuit size for which f was hard.

However, if we are interested in hardness amplification within $\mathcal{NP}$, we cannot use the XOR lemma; it does not ensure that f' is in $\mathcal{NP}$ when f is in $\mathcal{NP}$. Hardness amplification within $\mathcal{NP}$ was first addressed in a recent paper of O'Donnell [22], which is the starting point for our work.

1.1 O'Donnell's Hardness Amplification

To ensure that the new function f' is in $\mathcal{NP}$ when f is in $\mathcal{NP}$, O'Donnell [22] was led to study constructions of the form

$$f'(x_1,\ldots,x_k) \stackrel{\text{def}}{=} C(f(x_1), f(x_2),\ldots,f(x_k)), \qquad (1)$$

where C is an efficiently computable *monotone* function. The monotonicity of C ensures that f' is in $\mathcal{NP}$ when f is in $\mathcal{NP}$. But we are left with the task of choosing such a function C and proving that it indeed amplifies hardness.

Remarkably, O'Donnell was able to precisely characterize the amplification properties of Construction 1 in terms of a combinatorial property of the combining function C, called its *expected bias*. (The actual definition is not needed for this discussion, but can be found in Section 3.) By finding a monotone combining function in which this expected bias is small, he obtained the first positive result on hardness amplification in $\mathcal{NP}$:

O'Donnell's Theorem [22]. *If $\mathcal{NP}$ has a balanced function that is $1/\operatorname{poly}(n)$-hard for polynomial-size circuits, then $\mathcal{NP}$ has a function that is $(1/2-1/n^{1/2-\alpha})$-hard for polynomial-size circuits (where α is an arbitrarily small positive constant).*

However, the amplification provided by O'Donnell's theorem is not as strong as what the XOR Lemma gives. It is limited to $1/2-1/\sqrt{n}$, regardless of the circuit size s for which the original function is hard, even if s is exponentially large. The XOR Lemma, on the other hand, amplifies to $1/2-1/s^{\Omega(1)}$. O'Donnell showed that this difference is inherent — no construction of the form (1) with a monotone combining function C can always amplify hardness to better than $1/2-1/n$.[1]

1.2 Our Result

In this paper, we manage to amplify hardness within $\mathcal{NP}$ beyond the $1/2-1/n$ barrier:

Main Theorem. *If $\mathcal{NP}$ has a balanced function that is $1/\operatorname{poly}(n)$-hard for circuits of size $s(n)$, then $\mathcal{NP}$ has a function that is $(1/2-1/s'(n))$-hard for circuits of size $s'(n)=s(\sqrt{n})^{\Omega(1)}$. In particular,*

1. *If $s(n)=n^{\omega(1)}$, we amplify to hardness $1/2-1/n^{\omega(1)}$.*
2. *If $s(n)=2^{n^{\Omega(1)}}$, we amplify to hardness $1/2-1/2^{n^{\Omega(1)}}$.*
3. *If $s(n)=2^{\Omega(n)}$, we amplify to hardness $1/2-1/2^{\Omega(\sqrt{n})}$.*

Note that Parts 1 and 2 match the parameters of Yao's XOR Lemma. Part 3, however, is a bit worse, amplifying to $1/2-1/2^{\Omega(\sqrt{n})}$ rather than $1/2-1/2^{\Omega(n)}$. This gap is *not* inherent in our approach and, as mentioned below, would be eliminated given a corresponding improvement in one of the tools we employ.

Of course, our construction cannot be of the form in Construction (1). Below we describe our two main points of departure.

1.3 Techniques

To explain how we bypass it, we first look more closely at the source of the $1/2-1/n$ barrier. The actual lower bound is $1/2-1/k$, where k is the input length of the monotone combining function C. (This is based on the [15] lower bound on the noise stability of monotone functions.) Since in Construction (1), f' has input length $n'=n\cdot k\geq k$, it follows that we cannot amplify beyond $1/2-1/n'$.

Derandomization. Given the above, our first idea is to break the link between the input length of f' and the input length of the combining function C. We do this by *derandomizing* O'Donnell's construction. That is, the inputs $x_1,\ldots,x_k$ are no longer taken independently (as in Construction (1), but are generated pseudorandomly from a short seed of length $n'\ll k$, which becomes the actual input to f'. Our method for generating the x_i's is based on combinatorial designs (as in the Nisan–Wigderson generator [21]) and Nisan's pseudorandom generator for space-bounded computation [19], and reduces the input length of f' from $n\cdot k$ to $n'=O(n^2+\log^2 k)$. We stress that this derandomization is unconditional, i.e. requires no additional complexity assumption. We remark that it is the quadratic seed length of Nisan's generator that limits our amplification to $1/2-1/2^{\Omega(\sqrt{n})}$ rather than $1/2-1/2^{\Omega(n)}$ in Part 3 of our Main Theorem, and thus any improvement in Nisan's generator would yield a corresponding improvement in our result.

Similar derandomizations have previously been achieved for Yao's XOR Lemma by Impagliazzo [11] and Impagliazzo and Wigderson [13]. The analysis of such derandomizations is typically tailored to a particular proof, and indeed both [11, 13] gave new proofs of the XOR Lemma for that purpose. In our case, we do not know how to derandomize O'Donnell's original proof, but instead manage to derandomize a different proof due to Trevisan [24].

Our derandomization allows for k to be larger than the input length of f', and hence we can go beyond the $1/2-1/n'$ barrier. Indeed, by taking k to be a sufficiently large polynomial, we amplify to $1/2-1/(n')^c$ for any constant c.

Using Nondeterminism. To amplify further, it is tempting to take k superpolynomial in the input length of f'. But then we run into a different problem: how do we ensure that f' is in $\mathcal{NP}$? The natural algorithm for f' requires running the algorithm for f on k inputs.

To overcome this difficulty, we observe that we need only give an efficient *nondeterministic* algorithm for f'. Each nondeterministic path may involve only polynomially many evaluations of f while the global outcome $f'(x)$ depends on

[1] The gap between O'Donnell's positive result of $1/2-1/\sqrt{n}$ and his negative result of $1/2-1/n$ is not significant for what follows, and in particular, it will be subsumed by our improvements.

exponentially many evaluations. To implement this idea, we exploit the specific structure of the combining function C. Namely, we (like O'Donnell) use the TRIBES function of Ben-Or and Linial [4], which is a monotone DNF with clauses of size $O(\log k)$. Thus, the nondeterministic algorithm for f' can simply guess a satisfied clause and evaluate f on the $O(\log k)$ corresponding inputs.

1.4 Other Results

We also present some complementary negative results:

- We show that the assumption that the original hard function is balanced is necessary, in the sense that no monotone black-box hardness amplification can amplify unbalanced functions of unknown bias (or even improve their bias).
- We show that our use of nondeterminism is necessary, in the sense that any black-box hardness amplification in which each evaluation of f' is a monotone function of at most k evaluations of f can amplify hardness to at most $1/2 - 1/k$.

Note that most results on hardness amplification against circuits are black-box. (For a non-black-box hardness amplification against uniform machines, see [14, 25]).

Our framework also gives a new proof of the hardness amplification by Impagliazzo and Wigderson [13]. Our proof is simpler and in particular its analysis does not employ the Goldreich–Levin [9] step.

The rest of the paper is organized as follows. In Section 2, we discuss some preliminaries; in Section 3 we review existing results on hardness amplification in $\mathcal{NP}$; in Section 4 we present our main results and new techniques, and Sections 5 through 9 treat the details of the proof of our main theorem; hardness amplification of unbalanced functions is discussed in Section 10, and finally in Section 11 we show that the use of nondeterminism in our main result is likely to be necessary.

2. PRELIMINARIES

We denote the uniform distribution on $\{0,1\}^n$ by U_n. If U_n occurs more than once in the same expression, it is understood that these all represent the same random variable; for example, $U_n \cdot f(U_n)$ denotes the random variable obtained by choosing $X \stackrel{\mathrm{R}}{\leftarrow} \{0,1\}^n$ and outputting $X \cdot f(X)$.

Definition 2.1. *Let X and Y be two random variables taking values over the same set S. Then the* statistical difference *between X and Y, is*

$$\Delta(X,Y) \stackrel{\text{def}}{=} \max_{T \subseteq S} \left| \Pr[X \in T] - \Pr[Y \in T] \right|.$$

We view probabilistic functions as functions of two inputs, e.g. $\mathrm{h}(x;r)$, the first being the input to the function and the second being the randomness. (Deterministic functions may be thought of as probabilistic functions that ignore the randomness.) For notational convenience, we will often omit the second input to a probabilistic function, e.g. writing $\mathrm{h}(x)$ instead of $\mathrm{h}(x;r)$, in which case we view $\mathrm{h}(x)$ as the random variable $\mathrm{h}(x;U_{|r|})$.

Definition 2.2. *The* bias *of a* 0-1 *random variable X is*

$$\mathrm{Bias}\,[X] \stackrel{\text{def}}{=} \left| \Pr[X=0] - \Pr[X=1] \right| = 2 \cdot \Delta(X, U_1).$$

Analogously, the bias *of a probabilistic function $f : \{0,1\}^n \to \{0,1\}$ is*

$$\mathrm{Bias}\,[f] \stackrel{\text{def}}{=} \left| \Pr[f(U_n)=0] - \Pr[f(U_n)=1] \right|,$$

where the probabilities are taken over both the input chosen according to U_n and the coin tosses of f. We say that f is balanced *when* $\mathrm{Bias}\,[f] = 0$.

We say that the random variables X and Y are *ϵ-indistinguishable for size s* if for every circuit C of size s,

$$\left| \Pr_X[C(X)=1] - \Pr_Y[C(Y)=1] \right| \le \epsilon.$$

We will routinely use the following connection between hardness and indistinguishability.

Lemma 2.3 ([27]). *Let $h : \{0,1\}^n \to \{0,1\}$ be any probabilistic function. Then the distributions $U_n \cdot h(U_n)$ and $U_n \cdot U_1$ are ϵ-indistinguishable for size s if, and only if, h is $(1/2 - \epsilon/2)$-hard for size $s + \Theta(1)$.*

Finally, whenever we amplify the hardness of a function $f : \{0,1\}^n \to \{0,1\}$ that is hard for circuits of size $s(n)$, we assume that $s(n)$ is well-behaved in the sense that it is computable in time $\mathrm{poly}(n)$ and $s(cn) = s(n)^{O(1)}$, for all constants $c > 0$. Most natural functions smaller than 2^n, such as $n^k, 2^{\log^k n}, 2^{n^\epsilon}, 2^{\epsilon n}$ are well-behaved in this sense.

3. OVERVIEW OF HARDNESS AMPLIFICATION IN $\mathcal{NP}$

In this section we review the essential components of existing results on hardness amplification in $\mathcal{NP}$. We then discuss the limitations of these techniques. By the end of this section, we will have sketched the main result of O'Donnell [22], following the approach of Trevisan [24]. We outline this result in a way that will facilitate the presentation of our results.

Let $f : \{0,1\}^n \to \{0,1\}$ be an average-case hard function, and let $\mathrm{C} : \{0,1\}^k \to \{0,1\}$ be any function. In [22], O'Donnell studies the hardness of functions of the form

$$\mathrm{C} \circ f^{\otimes k} : (\{0,1\}^n)^k \to \{0,1\}$$

where $f^{\otimes k}(x_1, \ldots, x_k) \stackrel{\text{def}}{=} (f(x_1), \ldots, f(x_k))$, and $\circ$ denotes composition. That is,

$$(\mathrm{C} \circ f^{\otimes k})(x_1, \ldots, x_k) \stackrel{\text{def}}{=} \mathrm{C}(f(x_1), \ldots, f(x_k)).$$

In order to ensure that $\mathrm{C} \circ f^{\otimes k} \in \mathcal{NP}$ whenever $f \in \mathcal{NP}$, O'Donnell chooses C to be a polynomial-time computable *monotone* function. (Indeed, it is not hard to see that a monotone combination of $\mathcal{NP}$ functions is itself in $\mathcal{NP}$.)

O'Donnell characterizes the hardness of $\mathrm{C} \circ f^{\otimes k}$ in terms of a combinatorial property of the combining function C, called its *expected bias* (which we define later).

We will now review the key steps in establishing this characterization and O'Donnell's final amplification theorem.

Step 1: Impagliazzo's hardcore set. An important tool for establishing this connection is the so-called hardcore set lemma of Impagliazzo [11], which allows us to pass from computational hardness to information-theoretic hardness.

Definition 3.1. *We say that a (probabilistic) function* $\mathrm{g} : \{0,1\}^n \to \{0,1\}$ *is* δ-random *if* g *is balanced and there exists a subset* $H \subseteq \{0,1\}^n$ *with* $|H| = 2\delta 2^n$ *such that* $\mathrm{g}(x) = U_1$ *(i.e. a coin flip) for* $x \in H$ *and* $\mathrm{g}(x)$ *is deterministic for* $x \notin H$.

Thus, a δ-random function has a set of relative size 2δ on which it is information-theoretically unpredictable. The Impagliazzo hardcore set lemma says that any δ-hard function $f : \{0,1\}^n \to \{0,1\}$ has a hardcore set $H \subseteq \{0,1\}^n$ of density $\approx 2\delta$ such that f is very hard-on-average on H. Thus, f looks like a δ-random function to small circuits (cf., Lemma 2.3). (Following subsequent works, our formulation of Impagliazzo's lemma differs from the original one in several respects.)

Lemma 3.2 ([11], [16], [23], [22]). *For any function* $f : \{0,1\}^n \to \{0,1\}$ *that is balanced and* δ*-hard for size* s*, there exists a* δ'*-random function* $\mathrm{g} : \{0,1\}^n \to \{0,1\}$ *such that* $X \cdot f(X)$ *and* $X \cdot g(X)$ *are* ϵ*-indistinguishable for size* $\Omega(s\epsilon^2/\log(1/\delta))$*, with* $\delta \le \delta' \le 2\delta$*, where* $X \equiv U_n$.

In particular, by a standard hybrid argument,

$$X_1 \cdots X_k \cdot f(X_1) \cdots f(X_k) \text{ and } X_1 \cdots X_k \cdot g(X_1) \cdots g(X_k)$$

are $k\epsilon$*-indistinguishable for size* $\Omega(s\epsilon^2/\log(1/\delta))$*, where the* X_i*'s are uniform and independent.*

Step 2: Expected Bias. By the above, proving the computational hardness of $\mathrm{C} \circ f^{\otimes k}$ reduces to calculating the information-theoretic hardness of $\mathrm{C} \circ g^{\otimes k}$ for some δ'-random g. It turns out that information-theoretic hardness can be characterized by the following quantity.

Definition 3.3. *Let* $\mathrm{h} : \{0,1\}^n \to \{0,1\}$ *be any probabilistic function. We define the* expected bias *of* h *by*

$$\mathrm{ExpBias}\,[\mathrm{h}] \stackrel{\mathrm{def}}{=} \mathop{\mathrm{E}}_{x \leftarrow U_n} \left[\, \mathrm{Bias}\,[\mathrm{h}(x)]\,\right],$$

where $\mathrm{Bias}\,[\mathrm{h}(x)]$ *is taken over the coin tosses of* h.

The next lemma shows that information-theoretic hardness is equivalent to expected bias.

Lemma 3.4. *For any probabilistic* $\mathrm{h} : \{0,1\}^n \to \{0,1\}$,

$$\Delta(U_n \cdot h(U_n), U_n \cdot U_1) = \frac{1}{2}\,\mathrm{ExpBias}\,[h]\,.$$

Proof. $\Delta(U_n \cdot h(U_n), U_n \cdot U_1) = \mathop{\mathrm{E}}_{x \in U_n} [\Delta(h(x), U_1)] = \mathop{\mathrm{E}}_{x \leftarrow U_n} [\mathrm{Bias}\,[h(x)]\,/2] = \mathrm{ExpBias}\,[h]\,/2$. □

In particular, no circuit (regardless of its size) can distinguish between $U_n \cdot h(U_n)$ and $U_n \cdot U_1$ with advantage greater than $\mathrm{ExpBias}\,[\mathrm{h}]\,/2$.

Now we characterize the hardness of $\mathrm{C} \circ f^{\otimes k}$ in terms of expected bias. Specifically, by taking $\epsilon = 1/s^{1/3}$ in Lemma 3.2 and using Lemmas 2.3 and 3.4, one can show the following.

Lemma 3.5 ([22]). *Let* $f : \{0,1\}^n \to \{0,1\}$ *be balanced and* δ*-hard for size* s*, and let* $\mathrm{C} : \{0,1\}^k \to \{0,1\}$ *be any function. Then there exists a* δ*-random function* $\mathrm{g} : \{0,1\}^n \to \{0,1\}$ *such that* $\mathrm{C} \circ f^{\otimes k} : (\{0,1\}^n)^k \to \{0,1\}$ *has hardness*

$$\frac{1}{2} - \frac{\mathrm{ExpBias}\left[\mathrm{C} \circ g^{\otimes k}\right]}{2} - \frac{k}{s^{1/3}}$$

for circuits of size $\Omega\left(s^{1/3}/\log(1/\delta)\right) - \mathrm{size(C)}$*, where* $\mathrm{size(C)}$ *denotes the size of a smallest circuit computing* C.

What makes this lemma so useful is that the quantity $\mathrm{ExpBias}\left[\mathrm{C} \circ g^{\otimes k}\right]$ turns out to be independent of the choice of the δ-random function g and hence also of the particular hard function f. (Specifically, it equals the expectation of the bias of C after a random restriction that leaves each input bit unrestricted with probability δ.) Thus we are left with the task of understanding a purely combinatorial property of the combining function C.

Step 3: Noise Stability. Unfortunately, it is often difficult to analyze the expected bias directly. Nonetheless, the expected bias is closely related to the *noise stability*, a quantity that is more amenable to analysis and well-studied (e.g., [15], [22], [18]).

Definition 3.6. *The noise stability of* C *with respect to noise* δ*, denoted* $\mathrm{NoiseStab}_\delta[\mathrm{C}]$*, is defined by*

$$\mathrm{NoiseStab}_\delta[\mathrm{C}] \stackrel{\mathrm{def}}{=} 2 \cdot \Pr_{x,\eta}[\mathrm{C}(x) = \mathrm{C}(x \oplus \eta)] - 1,$$

where x *is random,* η *is a vector whose bits are independently one with probability* δ *and* $\oplus$ *denotes bitwise XOR.*

The following lemma bounds the expected bias of $\mathrm{C} \circ g^{\otimes k}$ (and hence the hardness in Lemma 3.5) in terms of the noise stability of C.

Lemma 3.7. *Let* $\mathrm{g} : \{0,1\}^n \to \{0,1\}$ *be* δ*-random. Then*

$$\mathrm{ExpBias}\left[\mathrm{C} \circ \mathrm{g}^{\otimes k}\right] \le \sqrt{\mathrm{NoiseStab}_\delta[\mathrm{C}]}.$$

Combining this with Lemma 3.5, we find that the hardness of $\mathrm{C} \circ f^{\otimes k}$ is roughly $1/2 - \sqrt{\mathrm{NoiseStab}_\delta[\mathrm{C}]}/2$. The next step is to exhibit a combining function C with a small noise stability (to ensure that the hardness of $\mathrm{C} \circ f^{\otimes k}$ is as close to $1/2$ as possible). The following is shown in [22].

Lemma 3.8 ([22]). *For all* $\delta > 0$*, there exists a* $k = \mathrm{poly}(1/\delta)$ *and a polynomial-time computable monotone function* $\mathrm{C} : \{0,1\}^k \to \{0,1\}$ *with* $\mathrm{NoiseStab}_\delta[\mathrm{C}] \le 1/k^{\Omega(1)}$.

Finally, by combining Lemmas 3.5, 3.7 and 3.8, we obtain the following weaker version of O'Donnell's hardness amplification within $\mathcal{NP}$. (While a stronger version of O'Donnell's result was mentioned in the introduction, the following version will suffice as a starting point for our work.)

Theorem 3.9 ([22]). *If there is a balanced* $f \in \mathcal{NP}$*,* $f : \{0,1\}^n \to \{0,1\}$ *that is* $1/\mathrm{poly}(n)$*-hard for size* $s(n)$*, then there is* $f' \in \mathcal{NP}$*,* $f' : \{0,1\}^m \to \{0,1\}$ *that is* $(1/2 - 1/m^{\Omega(1)})$*-hard for size* $s(m^{\Omega(1)})^{\Omega(1)}$.

Limitations of Direct Product Constructions. O'Donnell also showed that Theorem 3.9 is essentially the best result that one can obtain using the techniques that we have described thus far. He showed that for all monotone combining functions C there is a δ-hard f such that the hardness of $\mathrm{C} \circ f^{\otimes k}$ is no better than $1/2 - \mathrm{NoiseStab}_\delta[\mathrm{C}]/2$. This is problematic because the noise stability of monotone functions cannot become too small.

THEOREM 3.10 ([15]). *For any monotone function* $\mathrm{C}: \{0,1\}^k \to \{0,1\}$, $\mathrm{NoiseStab}_\delta[\mathrm{C}] \geq (1-2\delta)\Omega(\log^2 k/k)$.

Therefore, for any monotone $\mathrm{C}: \{0,1\}^k \to \{0,1\}$ there is a δ-hard f such that $\mathrm{C} \circ f^{\otimes k}$ does *not* have hardness $1/2 - \mathrm{NoiseStab}_\delta[\mathrm{C}]/2 \leq 1/2 - \Omega(1/k)$. Since $\mathrm{C} \circ f^{\otimes k}$ takes inputs of length $m = n \cdot k \geq k$, this implies that *we must employ a new technique to amplify beyond hardness* $1/2 - \Omega(1/m)$.

4. OUR RESULTS

In this paper, we obtain the following improvement upon Theorem 3.9.

THEOREM 4.1 (MAIN THEOREM). *If there is a balanced* $f \in \mathcal{NP}$, $f: \{0,1\}^n \to \{0,1\}$ *that is* $1/\mathrm{poly}(n)$*-hard for size* $s(n)$, *then there is* $f' \in \mathcal{NP}$, $f': \{0,1\}^m \to \{0,1\}$ *that is* $(1/2 - 1/s(\sqrt{m})^{\Omega(1)})$*-hard for size* $s(\sqrt{m})^{\Omega(1)}$.

We also show that the assumption that we start with a *balanced* function f is essential. Specifically, we show (Section 10) that no monotone black-box hardness amplification can amplify the hardness of functions whose bias is unknown. Most hardness amplifications, including the one in this paper, are black-box. However, the assumption that f is balanced can be dispensed with when amplifying within $\mathcal{NP}/\mathrm{poly}$ (i.e., nondeterministic polynomial size circuits).

We now describe our two main techniques that allow us to prove Theorem 4.1. As explained in the introduction, these two techniques are derandomization and nondeterminism.

4.1 Derandomization

As in the previous section, let $f: \{0,1\}^n \to \{0,1\}$ be our hard function and let $C: \{0,1\}^k \to \{0,1\}$ be a (monotone) combining function.

We will derandomize O'Donnell's construction using an "appropriately pseudorandom" generator.

DEFINITION 4.2. *A* generator $G: \{0,1\}^l \to (\{0,1\}^n)^k$ *is any function. We call* l *the* seed length *of* G, *and we often write* $G(\sigma) = X_1 \cdots X_k$, *with each* $X_i \in \{0,1\}^n$.

G *is* explicitly computable *if given* $\sigma, 1 \leq i \leq k$, *we can compute* X_i *in time* $\mathrm{poly}(l, \log k)$, *where* $G(\sigma) = X_1 \cdots X_k$.

Instead of using the function $\mathrm{C} \circ f^{\otimes k}: (\{0,1\}^n)^k \to \{0,1\}$, we take a generator $G: \{0,1\}^l \to (\{0,1\}^n)^k$ (where $l \ll nk$) and use $(\mathrm{C} \circ f^{\otimes k}) \circ G: \{0,1\}^l \to \{0,1\}$, i.e.,

$$(\mathrm{C} \circ f^{\otimes k}) \circ G(\sigma) = \mathrm{C}\left(f(X_1), \ldots, f(X_k)\right),$$

where $(X_1, \ldots, X_k) \in (\{0,1\}^n)^k$ is the output of $G(\sigma)$. This reduces the input length of the function to l. Therefore, if $l \ll nk$ we would expect $(\mathrm{C} \circ f^{\otimes k}) \circ G$ to be harder (with respect to its input length) than $\mathrm{C} \circ f^{\otimes k}$. We will show that this is indeed the case, provided the generator G satisfies the following requirements:

1. **G is indistinguishability-preserving:** Analogously to Lemma 3.5, the generator G should be such that the computational hardness of $(\mathrm{C} \circ f^{\otimes k}) \circ G$ is at least the information-theoretic hardness of $(\mathrm{C} \circ \mathrm{g}^{\otimes k}) \circ G$ for some δ-random function g, that is, at least $1/2 - \mathrm{ExpBias}\left[(\mathrm{C} \circ \mathrm{g}^{\otimes k}) \circ G\right]$. We will see that this can be achieved provided that G is *indistinguishability-preserving*; that is (analogously to the last part of Lemma 3.2),
$$\sigma \cdot f(X_1) \cdots f(X_k) \text{ and } \sigma \cdot g(X_1) \cdots g(X_k)$$
should be indistinguishable, for any δ-random g, when $\sigma \xleftarrow{\mathrm{R}} \{0,1\}^l$ and $X_1, \ldots, X_k \in \{0,1\}^n$ are the outputs of G on input σ.

2. **G fools the expected bias:** G should be such that for any δ-random g, $\mathrm{ExpBias}\left[(\mathrm{C} \circ \mathrm{g}^{\otimes k}) \circ G\right]$ is approximately $\mathrm{ExpBias}\left[\mathrm{C} \circ \mathrm{g}^{\otimes k}\right]$, and thus, by Lemma 3.7:
$$\mathrm{ExpBias}\left[(\mathrm{C} \circ \mathrm{g}^{\otimes k}) \circ G\right] \leq \sqrt{\mathrm{NoiseStab}_\delta[\mathrm{C}] + \epsilon}, \quad (2)$$
for a suitably small ϵ. Actually, we will not show that G fools the expected bias directly and instead will work with a related quantity (the expected collision probability), which will still suffice to show Inequality (2).

The first requirement is achieved through a generator that outputs *combinatorial designs*. This construction is essentially from Nisan and Wigderson [20, 21] and has been used in many places, e.g. [13, 23].

The second requirement is achieved as follows. We show that if G is pseudorandom against space-bounded algorithms and the combining function C is computable in small space, then Inequality (2) holds. We then use Nisan's *unconditional* pseudorandom generator against space-bounded algorithms [19], and show that combining functions with low noise stability can in fact be computed in small space.[2] Note that we only use the pseudorandomness of the generator G to relate the expected bias with respect to G to a combinatorial property of the combining function C. In particular, it is *not* used to fool the circuits trying to compute the hard function. This is what allows us to use an unconditional generator against a relatively weak model of computation.

Our final generator, Γ, is the generator obtained by XORing a generator that is indistinguishability-preserving and a generator that fools the expected bias, obtaining a generator that has both properties. The approach of XORing two generators in this way appeared in [13], and was subsequently used in [23].

The net effect of the two above requirements of the generator Γ, is that the hardness of $(\mathrm{C} \circ f^{\otimes k}) \circ \Gamma$ is roughly the hardness of $\mathrm{C} \circ f^{\otimes k}$, while dramatically reducing the input length from nk to l (the seed length of Γ).

4.2 Using Nondeterminism

The derandomization described above gives hardness amplification up to $1/2 - 1/n^c$ *for every* c. This already improves upon the best previous result, namely Theorem 3.9. However, to go beyond that new techniques are required. The problem is that if C is a function on k bits, then by Theorem 3.10 its noise stability will always be at least $1/k^{\Omega(1)}$, and hence the hardness of the amplified function will be at most $1/2 - 1/k^{\Omega(1)}$.

Therefore, if we want C to be computable in polynomial time we will always have $k = \mathrm{poly}(n)$ and we will never amplify beyond $1/2 - 1/\mathrm{poly}$.

We solve this problem taking full advantage of the power of $\mathcal{NP}$, namely nondeterminism. This allows us to use a

[2]The same approach also works using the unconditional pseudorandom generator against constant-depth circuits of [20] and showing that the combining function is computable by a constant-depth circuit; however, the space generator gives us slightly better parameters.

function $\mathrm{C} : \{0,1\}^k \to \{0,1\}$ which is computable in *nondeterministic* time $\mathrm{poly}(n, \log(k))$; thus, the amplified function will still be in $\mathcal{NP}$ *for* k *as large as* 2^n.

Conversely, in Section 11 we show that any non-adaptive monotone black-box hardness amplification that amplifies to hardness $1/2 - 1/n^{\omega(1)}$ cannot be computed in $\mathcal{P}$, i.e. the use of nondeterminism is essential.

We will now proceed by discussing the details of the derandomization (Sections 5, 6 and 7) and the use of nondeterminism (Section 8). For clarity of exposition, we focus on the case where the original hard function f is balanced and is 1/3-hard. Hardness amplification from hardness $1/\mathrm{poly}(n)$ is discussed in Section 9, and hardness amplification of unbalanced functions is discussed in Section 10.

5. PRESERVING INDISTINGUISHABILITY

The main result in this section is that if G is pseudorandom in an appropriate sense, then the hardness of $(\mathrm{C} \circ f^{\otimes k}) \circ G$ is roughly

$$1/2 - \mathrm{ExpBias}\left[(\mathrm{C} \circ \mathrm{g}^{\otimes k}) \circ G\right]$$

for some δ-random function g. As we noted in the previous section, it will be sufficient for G to be *indistinguishability-preserving* . We give the definition of indistinguishability-preserving and then our main result.

Definition 5.1. *A generator* $G : \{0,1\}^l \to (\{0,1\}^n)^k$ *is said to be* indistinguishability-preserving for size t *if for all (possibly probabilistic) functions* $f_1, \ldots, f_k, g_1, \ldots, g_k$ *the following holds:*

If for every $i, 1 \le i \le k$ *the distributions*

$$U_n \cdot f_i(U_n) \text{ and } U_n \cdot \mathrm{g}_i(U_n)$$

are ϵ*-indistinguishable for size* s*, then*

$$\sigma \cdot f_1(X_1) \cdots f_k(X_k) \text{ and } \sigma \cdot \mathrm{g}_1(X_1) \cdots \mathrm{g}_x(X_k)$$

are $k\epsilon$*-indistinguishable for size* $s - t$*, where* σ *is a random seed of length* l *and* $X_1 \cdots X_k$ *is the output of* $G(\sigma)$*.*

Lemma 5.2. *Let* $f : \{0,1\}^n \to \{0,1\}$ *be* δ*-hard for size* s*, let* $G : \{0,1\}^l \to (\{0,1\}^n)^k$ *be a generator that is indistinguishability-preserving for size* t *and let* $C : \{0,1\}^k \to \{0,1\}$ *be any function. Then there exists a* δ'*-random* g*, with* $\delta \le \delta' \le 2\delta$ *such that the function* $(\mathrm{C} \circ f^{\otimes k}) \circ G : \{0,1\}^l \to \{0,1\}$ *has hardness*

$$\frac{1}{2} - \frac{\mathrm{ExpBias}\left[(\mathrm{C} \circ \mathrm{g}^{\otimes k}) \circ G\right]}{2} - \frac{k}{s^{1/3}}$$

for circuits of size $\Omega\left(s^{1/3}/\log(1/\delta)\right) - t - \mathrm{size}(\mathrm{C})$ *where* $\mathrm{size}(\mathrm{C})$ *denotes the size of a smallest circuit computing* C*.*

Proof. By Lemma 3.2, there exists a δ'-random function g with $\delta \le \delta' \le 2\delta$, such that $U_n \cdot f(U_n)$ and $U_n \cdot \mathrm{g}(U_n)$ are ϵ-indistinguishable for size $\Omega(s\epsilon^2, \log(1/\delta))$.

For the remainder of this proof, σ will denote a uniform random seed in $\{0,1\}^l$, and $X_1 \cdots X_k$ will denote the output of $G(\sigma)$.

Since G is a indistinguishability-preserving for size t, by assumption, this implies that

$$\sigma \cdot f(X_1) \cdots f(X_k) \text{ and } \sigma \cdot \mathrm{g}(X_1) \cdots \mathrm{g}(X_k)$$

are $k\epsilon$-indistinguishable for size $\Omega(s\epsilon^2 \log(1/\delta)) - t$.

This in turn implies that

$$\sigma \cdot \mathrm{C}(f(X_1) \cdots f(X_k)) \text{ and } \sigma \cdot \mathrm{C}(\mathrm{g}(X_1) \cdots \mathrm{g}(X_k))$$

$$(\text{ i.e., } \sigma \cdot (\mathrm{C} \circ f^{\otimes k}) \circ G(\sigma) \text{ and } \sigma \cdot (\mathrm{C} \circ \mathrm{g}^{\otimes k}) \circ G(\sigma))$$

are $k\epsilon$-indistinguishable for size $\Omega(s\epsilon^2/\log(1/\delta)) - t - \mathrm{size}(\mathrm{C})$.

By Claim 3.4,

$$\sigma \cdot (\mathrm{C} \circ \mathrm{g}^{\otimes k}) \circ G \text{ and } \sigma \cdot U_1$$

are $(\mathrm{ExpBias}\left[(\mathrm{C} \circ \mathrm{g}^{\otimes k}) \circ G\right]/2)$-indistinguishable for any size; therefore we have that

$$\sigma \cdot (\mathrm{C} \circ f^{\otimes k}) \circ G \text{ and } \sigma \cdot U_1$$

are $(\mathrm{ExpBias}\left[(\mathrm{C} \circ \mathrm{g}^{\otimes k}) \circ G\right]/2 + k\epsilon)$-indistinguishable for size $\Omega(s\epsilon^2/\log(1/\delta)) - t - \mathrm{size}(\mathrm{C})$. The result follows by setting $\epsilon = 1/s^{1/3}$ and applying Lemma 2.3. $\square$

In particular, we note that the *identity generator* $G : \{0,1\}^{nk} \to (\{0,1\}^n)^k$, i.e. $G(x) = x$, is indistinguishability-preserving for size 0 (by a hybrid argument), and thus Lemma 3.5 is a corollary of Lemma 5.2. However, the identity generator has seed-length nk and is therefore a very poor pseudorandom generator. Fortunately, there are indistinguishability-preserving pseudorandom generators with much shorter seeds which will allow us to use Lemma 5.2 to obtain much stronger hardness amplifications.

Lemma 5.3. *For every* $n \ge 2$ *and every* k *there is an explicitly computable generator* $IP_k : \{0,1\}^l \to (\{0,1\}^n)^k$ *with seed length* $l = O(n^2)$ *that is indistinguishability-preserving for size* k^2*.*

Proof. The generator is due to Nisan and Wigderson [20, 21], and is based on combinatorial designs. Specifically, we let $S_1, \ldots, S_k \subseteq [l]$ be an explicit family of sets such that $|S_i| = n$ for all i, and $|S_i \cap S_j| \le \log k$ for all $i \ne j$. Nisan [20] gives a construction of such sets with $l = O(n^2)$. Then the generator $IP_k : \{0,1\}^l \to (\{0,1\}^n)^k$ is defined by $IP_k(\sigma) = (\sigma|_{S_1}, \ldots, \sigma|_{S_k})$, where $\sigma|_{S_i} \in \{0,1\}^n$ denotes the projection of σ onto the coordinates indexed by the set S_i. The proof that this generator is indistinguishability preserving for size k^2 follows the arguments in [21, 23]. $\square$

6. FOOLING THE EXPECTED BIAS

In this section we prove a derandomized version of Lemma 3.7. Informally, we show that if C is computable in a restricted model of computation and G "fools" that restricted model of computation, then for any δ-random function g:

$$\mathrm{ExpBias}\left[(\mathrm{C} \circ \mathrm{g}^{\otimes k}) \circ G\right] \le \sqrt{\mathrm{NoiseStab}_\delta[\mathrm{C}] + \epsilon}.$$

The restricted model of computation we consider is that of nonuniform space-bounded algorithms which make one pass through the input, reading it in blocks of length n. These are formally modelled by branching programs of the following type:

Definition 6.1. *A (probabilistic, read-once, oblivious)* branching program *of* size s *with* block-size n *is a finite state machine with* s *states, over the alphabet* $\{0,1\}^n$ *(with a fixed start state, and an arbitrary number of accepting states). Each edge is labelled with a symbol in* $\{0,1\}^n$*. For every state* a *and symbol* $\alpha \in \{0,1\}^n$*, the edges leaving* a *and labelled with* α *are assigned a probability distribution. Then* *computation proceeds as follows. The input is read sequentially, one block of n bits at a time. If the machine is in state a and it reads α, then it chooses an edge leaving a and labelled with α according to its probability, and moves along it. The* width *of a branching program is the maximum, over i, of the number of states that are reachable after reading i symbols.*

Intuitively, the space of the algorithm is the logarithm of the width.

In [19], Nisan builds an unconditional generator against branching programs. (Improved PRGs were obtained in [12] but for our purposes Nisan's original PRG is essentially as good, and we use it because of its simplicity.)

Now we formally define pseudorandom generators against branching programs and then we give Nisan's result.

Definition 6.2. *A generator $G : \{0,1\}^l \to (\{0,1\}^n)^k$ is* ϵ-pseudorandom against branching programs of size s and block-size n *if for every branching program* B *of size s and block-size n:*

$$\left|\Pr[\mathrm{B}(G(U_l)) = 1] - \Pr[\mathrm{B}(U_{nk}) = 1]\right| \leq \epsilon.$$

Theorem 6.3 ([19]). *There exists a generator*

$$N_k : \{0,1\}^l \to (\{0,1\}^n)^k$$

such that, for every $k \leq 2^n$,

- *N_k is 2^{-n}-pseudorandom against branching programs of size 2^n and block-size n.*
- *N_k has seed length $O(n \log k)$.*
- *N_k is explicitly computable.*

We now state the derandomized version of Lemma 3.7.

Lemma 6.4. *Let*

- *$\mathrm{g} : \{0,1\}^n \to \{0,1\}$ be a δ-random function,*
- *$\mathrm{C} : \{0,1\}^k \to \{0,1\}$ be computable by a branching program of width w and block-size 1,*
- *$G : \{0,1\}^l \to (\{0,1\}^n)^k$ be $\epsilon/2$-pseudorandom against branching programs of size $k \cdot w^2$ and block-size n.*

Then $\mathrm{ExpBias}\left[(\mathrm{C} \circ \mathrm{g}^{\otimes k}) \circ G\right] \leq \sqrt{\mathrm{NoiseStab}_\delta[\mathrm{C}] + \epsilon}$.

Proof. We will not show that G fools the expected bias, but rather the following related quantity. For a probabilistic boolean function $\mathrm{h}(x; r)$ we define its (normalized) *expected collision probability* as

$$\mathrm{ExpCP}[\mathrm{h}] \stackrel{\mathrm{def}}{=} \mathop{\mathrm{E}}_x\left[2 \cdot \Pr_{r,r'}[\mathrm{h}(x;r) = \mathrm{h}(x;r')] - 1\right].$$

The same reasoning that shows Lemma 3.7, shows that for every probabilistic boolean function h:

$$\mathrm{ExpBias}[\mathrm{h}] \leq \sqrt{\mathrm{ExpCP}[\mathrm{h}]}. \tag{3}$$

Let $\mathrm{h}(x; r) : (\{0,1\}^n)^k \to \{0,1\}$ be the probabilistic function $\mathrm{C} \circ \mathrm{g}^{\otimes k}$. Even though h is defined in terms of g, it turns out that its expected collision probability is the same for all δ-random functions g, and simply equals the noise stability of C:

$$\mathrm{ExpCP}[\mathrm{h}] = \mathrm{NoiseStab}_\delta[\mathrm{C}]. \tag{4}$$

Now we construct a probabilistic branching program $M : (\{0,1\}^n)^k \to \{0,1\}$ of width w^2, size kw^2 and block-size n such that for every $x \in (\{0,1\}^n)^k$:

$$\Pr[M(x) = 1] = \Pr_{r,r'}[\mathrm{h}(x;r) = \mathrm{h}(x;r')].$$

To do this, we first note that, using the branching program for C, we can build a probabilistic branching program with block-size n and width w which computes $\mathrm{C} \circ \mathrm{g}^{\otimes k}$: The states of the branching program are the same as those of the branching program for C, and we define the transitions as follows. Upon reading symbol $\alpha \in \{0,1\}^n$ in state s, if $g(\alpha) = 0$ (resp. $g(\alpha) = 1$), we deterministically go to the state given by the 0-transition (resp., 1-transition) of C from state s, and if $g(\alpha)$ is a coin flip, then we put equal probability on these two transitions.

Then, to obtain M, run two *independent* copies of this branching program (i.e., using independent choices for the probabilistic state transitions) and accept if and only if exactly one of the two copies accepts. Now,

$$\begin{aligned}
&\left|\mathrm{ExpCP}[(\mathrm{C} \circ \mathrm{g}^{\otimes k}) \circ G] - \mathrm{NoiseStab}_\delta[\mathrm{C}]\right| \\
&= \left|\mathrm{ExpCP}[(\mathrm{C} \circ \mathrm{g}^{\otimes k}) \circ G] - \mathrm{ExpCP}[\mathrm{C} \circ \mathrm{g}^{\otimes k}]\right| \quad \text{(by (4))} \\
&= 2 \cdot \left|\Pr[M \circ G(U_l) = 1] - \Pr[M(U_{n \cdot k}) = 1]\right| \\
&\leq \epsilon. \quad \text{(by pseudorandomness of } G\text{)}
\end{aligned}$$

The lemma follows combining this with Equation (3). □

7. AMPLIFICATION UP TO $1/2 - 1/\mathrm{poly}$

In this section we sketch our hardness amplification up to $1/2 + 1/n^c$, for every c:

Theorem 7.1. *If there is a balanced $f : \{0,1\}^n \to \{0,1\}$ in $\mathcal{NP}$ that is $(1/3)$-hard for size $s(n) \geq n^{\omega(1)}$, then for every $c > 0$ there is a function $f' : \{0,1\}^m \to \{0,1\}$ in $\mathcal{NP}$ that is $(1/2 - 1/m^c)$-hard for size $(s(\sqrt{m}))^{\Omega(1)}$.*

To amplify we use the TRIBES function, a monotone read-once DNF.

Definition 7.2. *The TRIBES function on k bits is:*

$$\mathrm{TRIBES}_k(x_1, \ldots, x_k) \stackrel{\mathrm{def}}{=}$$
$$(x_1 \wedge \ldots \wedge x_b) \vee (x_{b+1} \wedge \ldots \wedge x_{2b}) \vee \ldots \vee (x_{k-b+1} \wedge \ldots \wedge x_k)$$

where there are k/b clauses each of size b, and b is the largest integer such that $(1 - 2^{-b})^{k/b} \geq 1/2$. Note that this makes $b = O(\log k)$.

The TRIBES DNF has very low noise stability when perturbed with constant noise.

Lemma 7.3 ([22, 18]). *For every constant $\delta > 0$,*

$$\mathrm{NoiseStab}_\delta[\mathrm{TRIBES}_k] \leq \frac{1}{k^{\Omega(1)}}.$$

A key step in our result is that TRIBES_k is (trivially) computable by a branching program of width 3, and therefore we can use Lemma 6.4 to fool its expected bias.

We now define the generator we will use in our derandomized direct product construction.

DEFINITION 7.4. *The generator Γ_k is defined as follows:*

$$\Gamma_k(x,y) := IP_k(x) \oplus N_k(y).$$

We recall the properties of Γ we are interested in:

LEMMA 7.5. *The following hold:*

1. *Γ_k is indistinguishability-preserving for size k^2.*
2. *Γ_k is 2^{-n}-pseudorandom against branching programs of size 2^n and block-size n.*
3. *Γ_k has seed length $O(n^2)$.*
4. *Γ_k is explicitly computable (see Definition 4.2 for the definition of explicit).*

PROOF. (1) By Lemma 5.3 and the fact that an indistinguishability-preserving generator XORed with any fixed string (in particular, $N_k(y)$ for any y) is still indistinguishability-preserving . (2) By Theorem 6.3 and the fact that XORing with any fixed string (in particular, $IP_k(x)$ for any x) preserves pseudorandomness against branching programs. (3) By the seed lengths of IP_k (Lemma 5.3) and N_k (Theorem 6.3). (4) Because IP_k is explicit (Lemma 5.3) and N_k is explicit (Theorem 6.3). $\square$

PROOF OF THEOREM 7.1. Given $f : \{0,1\}^n \to \{0,1\}$ that is δ-hard for size $s(n)$ (for $\delta = 1/3$) and a constant c, let $k = n^{c'}$ for $c' = O(c)$ to be determined later. Consider the function $f' : \{0,1\}^m \to \{0,1\}$ defined by

$$f' \stackrel{\text{def}}{=} (\text{TRIBES}_k \circ f^{\otimes k}) \circ \Gamma_k.$$

Note that $f' \in \mathcal{NP}$ since $f \in \mathcal{NP}$, TRIBES is monotone and both Γ and TRIBES are efficiently computable.

We now analyze the hardness of f'. Since Γ_k is indistinguishability-preserving for size k^2 by Lemma 7.5, Lemma 5.2 implies that there is a δ'-random function g (for $\delta \le \delta' \le 2\delta$) such that f' has hardness

$$\frac{1}{2} - \frac{\text{ExpBias}\left[(\text{TRIBES}_k \circ \text{g}^{\otimes k}) \circ \Gamma_k\right]}{2} - \frac{k}{s(n)^{1/3}} \qquad (5)$$

for circuits of size $\Omega\left(s(n)^{1/3}\right) - k^2 - \text{size}(\text{TRIBES}_k)$. Next we bound the hardness. By Lemma 7.5, we know that Γ_k is 2^{-n}-pseudorandom against branching programs of size 2^n and block-size n. In particular, since $k = \text{poly}(n)$, Γ_k is $1/k$-pseudorandom against branching programs of size $9k$ and block-size n. Since TRIBES_k is trivially computable by a branching program of width 3, we can apply Lemma 6.4 in order to bound $\text{ExpBias}\left[(\text{TRIBES}_k \circ \text{g}^{\otimes k}) \circ \Gamma_k\right]$ by $\sqrt{\text{NoiseStab}_{\delta'}[\text{TRIBES}_k] + 2/k}$. And this noise stability is at most $1/k^{\Omega(1)}$ by Lemma 7.3. Since $k = \text{poly}(n)$ and $s(n) = n^{\omega(1)}$, the $k/s^{1/3}$ term in the hardness (5) is negligible and we obtain hardness at least $1/2 - 1/k^{\Omega(1)}$.

We now bound the circuit size: Since TRIBES_k is computable by circuits of size $O(k)$, and $s(n) = n^{\omega(1)}$, the size is at least $s(n)^{\Omega(1)}$.

To conclude, note that f' has input length $m = n^2$ by Lemma 7.5. The result then follows by an appropriate choice of $c' = O(c)$. $\square$

8. USING NONDETERMINISM

In this section we discuss how to use nondeterminism to get the following theorem.

THEOREM 8.1. *If there is a balanced $f : \{0,1\}^n \to \{0,1\}$ in $\mathcal{NP}$ that is $(1/3)$-hard for size $s(n)$, then there is an $f' : \{0,1\}^m \to \{0,1\}$ in $\mathcal{NP}$ that is $(1/2 - 1/(s(\sqrt{m}))^{\Omega(1)})$-hard for size $(s(\sqrt{m}))^{\Omega(1)}$.*

Our main observation is that TRIBES_k is a DNF with clause size $O(\log k)$, and therefore it is computable in $\mathcal{NP}$ *even for superpolynomial k*:

LEMMA 8.2. *Let $f : \{0,1\}^n \to \{0,1\}$ be in $\mathcal{NP}$, and let $G_k : \{0,1\}^l \to (\{0,1\}^n)^k$ be any explicitly computable generator (see Definition 4.2) with $l \ge n$. Then the function $f' \stackrel{\text{def}}{=} (\text{TRIBES}_k \circ f^{\otimes k}) \circ G_k$ is computable in $\mathcal{NP}$ for every $k \le 2^{\text{poly}(n)}$.*

PROOF. We compute $f'(\sigma)$ nondeterministically as follows: Guess a clause $v_i \wedge \ldots \wedge v_j$ in TRIBES_k. Accept if for every h s.t. $i \le h \le j$ we have $f(X_h) = 1$, where $G(\sigma) = (X_1, \ldots, X_k)$ and the values $f(X_h)$ are computed using the $\mathcal{NP}$ algorithm for f.

It can be verified that this algorithm has an accepting computation path on input σ iff $f'(\sigma) = 1$. Note that the clauses have size logarithmic in k, which is polynomial in n. Moreover, G is explicitly computable. The result follows. $\square$

Now the proof of Theorem 8.1 proceeds along the same lines as the proof of Theorem 7.1, setting $k \stackrel{\text{def}}{=} s(n)^{\Omega(1)}$.

9. AMPLIFYING FROM HARDNESS $1/\text{poly}$

Our amplification from hardness $\Omega(1)$ to $1/2 - \epsilon$ (Theorem 7.1) can be combined with O'Donnell's amplification from hardness $1/\text{poly}$ to hardness $\Omega(1)$ to obtain an amplification from $1/\text{poly}$ to $1/2 - \epsilon$. However, since O'Donnell's construction blows up the input length polynomially, we will only obtain $\epsilon = 1/s(n^{\Omega(1)})$ (where the hidden constant depends on the initial polynomial hardness) rather than $\epsilon = 1/s(\sqrt{n})^{\Omega(1)}$ (as in Theorem 7.1). Thus we show here how to amplify directly from $1/\text{poly}$ to $1/2 - \epsilon$ using our approach.

Amplification from hardness $\Omega(1)$ (Theorem 7.1) relies on the fact that the TRIBES DNF has very low noise stability with respect to noise parameter $\delta = \Omega(1)$ (i.e., Lemma 7.3.). Similarly, to amplify from hardness $1/\text{poly}(n)$ we need to employ a combining function that has low noise stability with respect to noise $1/\text{poly}(n)$. To this end, following [22], we also employ the recursive-majorities function, RMAJ_r. Let MAJ denote the majority function.

DEFINITION 9.1. *The RMAJ_r function on 3^r bits is defined recursively by:*

$$\text{RMAJ}_1(x_1,x_2,x_3) \stackrel{\text{def}}{=} \text{MAJ}(x_1,x_2,x_3)$$
$$\text{RMAJ}_r(x_1,\ldots,x_{3^r}) \stackrel{\text{def}}{=}$$
$$\text{RMAJ}_{r-1}\left(\text{MAJ}(x_1,x_2,x_3),\ldots,\text{MAJ}(x_{3^n-2},x_{3^n-1},x_{3^n})\right)$$

Unfortunately, RMAJ_r does not have sufficiently low noise stability to be used on its own; for this reason, we will consider the function $\text{TRIBES}_k \circ \text{RMAJ}_r^{\otimes k}$. Using a similar analysis to [22] one can prove the following.

PROPOSITION 9.2. *For $r = \Omega(\log(1/\delta))$,*

$$\mathrm{NoiseStab}_\delta[\mathrm{TRIBES}_k \circ \mathrm{RMAJ}_r^{\otimes k}] \le \frac{1}{k^{\Omega(1)}}$$

We set $r \stackrel{\text{def}}{=} O(\log n)$, where the constant in the choice of $O(\log n)$ will depend on the hardness of the original mildly hard function.

The next step is to show that the derandomization of Section 6 still goes through. To achieve this, it suffices to show that $\mathrm{TRIBES}_k \circ \mathrm{RMAJ}_r^{\otimes k}$ can be computed by a small branching program. Indeed, a simple argument shows that RMAJ_r can be computed by branching program of width $2^{O(r)}$, and combining this with the observation that TRIBES_k can be computed by a branching program of constant width, we obtain the following.

LEMMA 9.3. $\mathrm{TRIBES}_k \circ \mathrm{RMAJ}_r^{\otimes k}$ *can be computed by a (read-once, oblivious) branching program of width $2^{O(r)}$.*

The remaining details for proving Theorem 4.1 are essentially the same as in the proof of Theorem 8.1. All that is left to observe is that $(\mathrm{TRIBES}_k \circ \mathrm{RMAJ}_{O(\log n)}^{\otimes k} \circ f^{\otimes k}) \circ \Gamma \in \mathcal{NP}$ whenever $f \in \mathcal{NP}$ because the recursive majorities each only depend on $2^{O(\log n)} = \mathrm{poly}(n)$ bits of the input, and hence we only incur a (deterministic) polynomial slow-down.

10. ON THE BALANCING HYPOTHESIS

The hardness amplification results in the previous sections start from balanced functions. In this section we study this hypothesis. Our main finding is that, while this hypothesis is not necessary for hardness amplification within $\mathcal{NP}/\mathrm{poly}$ (i.e., non-deterministic polynomial size circuits), it is likely to be necessary for hardness amplification within $\mathcal{NP}$.

To see that this hypothesis is not necessary for amplification within $\mathcal{NP}/\mathrm{poly}$, note that if the bias $B(n)$ of the original hard function $f : \{0,1\}^n \to \{0,1\}$ is known, then we can easily pad f to obtain a balanced function $\bar{f} : \{0,1\}^{n+1} \to \{0,1\}$:

$$\bar{f}(x,p) \stackrel{\text{def}}{=} \begin{cases} f(x) & \text{if } p = 0 \\ 1 & \text{if } p = 1 \text{ and } x \le B(n)2^n \\ 0 & \text{otherwise} \end{cases}$$

It is easy to see that $\bar{f}$ is $1/\mathrm{poly}(n)$-hard if f is. Since a circuit can know (non-uniformly) the bias $B(n)$ of f, the following hardness amplification within $\mathcal{NP}/\mathrm{poly}$ is a corollary to the proof of Theorem 4.1.

COROLLARY 10.1. *If there is $f \in \mathcal{NP}/\mathrm{poly}$, $f : \{0,1\}^n \to \{0,1\}$ that is $1/\mathrm{poly}(n)$-hard for size $s(n)$, then there is $f' \in \mathcal{NP}/\mathrm{poly}$, $f' : \{0,1\}^m \to \{0,1\}$ that is $(1/2 - 1/s(\sqrt{m})^{\Omega(1)})$-hard for size $s(\sqrt{m})^{\Omega(1)}$.*

Now we return to hardness amplification within $\mathcal{NP}$. One should note that, in our results, to amplify the hardness of $f : \{0,1\}^n \to \{0,1\}$ up to $1/2 - \epsilon$ it is only necessary that $\mathrm{Bias}[f] \le \epsilon^c$ for some universal constant c. The argument is standard and can be found, for example, in [24].

Combining this observation with the above padding technique, O'Donnell constructs several candidate hard functions, one for each 'guess' of the bias of the original hard function. He then combines them in a single function which is very hard infinitely often. However, it seems that this approach, even in conjunction with derandomization and nondeterminism, can not give better hardness than $1/2 - 1/n$.

To what extent can we amplifiy the hardness of functions whose bias is *unknown*? Non-monotone hardness amplifications, such as Yao's XOR Lemma, work regardless of the bias of the original hard function. However, in the rest of this section we show that, for hardness amplifications that are *monotone* and *black-box*, this is impossible. In particular, we show that black-box monotone hardness amplifications cannot amplify the hardness beyond the bias of the original function.

We now formalize the notion of black-box monotone hardness amplification and then state our lower bound.

DEFINITION 10.2. *An oracle algorithm $Amp : \{0,1\}^l \to \{0,1\}$ is a* black-box β-bias $[\delta \mapsto (1/2-\epsilon)]$-hardness amplification for length n and size s *if for every $f : \{0,1\}^n \to \{0,1\}$ such that $\mathrm{Bias}[f] \le \beta$ and for every $A : \{0,1\}^l \to \{0,1\}$ such that*

$$\Pr[A(U_l) \ne Amp^f(U_l)] \le 1/2 - \epsilon,$$

there is an oracle circuit C of size at most s such that

$$\Pr[C^A(U_n) \ne f(U_n)] \le \delta.$$

Amp is monotone *if for every x, $Amp^F(x)$ is a monotone function of the truth table of F.*

Note that if *Amp* is as in Definition 10.2 and if f is δ-hard for size s' and $\mathrm{Bias}[f] \le \beta$, then Amp^f is $(1/2 - \epsilon)$-hard for size s'/s.

THEOREM 10.3. *For any constant $\gamma > 0$, if Amp is a* monotone *black-box β-bias $[\delta \mapsto (1/2-\epsilon)]$-hardness amplification for length n and size $s \le 2^{n/3}$ such that $1/2 - 4\epsilon > \delta + \gamma$, then $\beta \le O(\epsilon + 2^{-n})$.*

The main ideas for proving this bound are the same as in the lower bounds for black-box hardness amplification in [26]: First we show that the above kind of hardness amplification satisfies certain coding-like properties. (Roughly, *Amp* can be seen as a list-decodable code where the distance property is guaranteed only for δ-distant messages with bias at most β (cf., [24]).) Then we show that monotone functions fail to satisfy these properties. The limitation we prove of monotone functions relies on the Kruskal-Katona theorem (see [1]).

11. NONDETERMINISM IS NECESSARY

In this section we show that *deterministic*, monotone, non-adaptive black-box hardness amplifications cannot amplify hardness beyond $1/2 - 1/\mathrm{poly}(n)$. Thus, the use of *nondeterminism* in our results (Section 8) is likely to be necessary. Note that most hardness amplifications, including the one in this paper, are black-box and non-adaptive.

O'Donnell [22] proves that any monotone direct product construction cannot amplify to hardness better than $1/2 - 1/n$, if the amplification works *for every function*. Our result is orthogonal: we relax the assumption that the hardness amplification is a direct product construction, but on the other hand we require the proof of correctness to be black-box.

We prove our bound even for hardness amplifications that amplify only balanced functions (i.e. $\beta = 0$ in Def. 10.2).

THEOREM 11.1. *For every constant $\delta < 1/2$, if Amp is a black-box 0-bias $[\delta \mapsto (1/2 - \epsilon)]$-hardness amplification for* *length n and size $s \leq 2^{n/3}$ such that $Amp^f(x)$ is a monotone function of $k \leq 2^{n/3}$ values of f, then*

$$\epsilon \geq \Omega\left(\frac{\log^2 k}{k}\right).$$

The proof of this bound follows closely the proof of the lower bound on hardness amplification in [26]. The main difference is here we use bounds on the noise stability of monotone functions rather than constant depth circuits.

12. ACKNOWLEDGMENTS

We thank Ryan O'Donnell and Rocco Servedio for an email exchange about noise stability. We thank Richard Stanley for pointing out the Kruskal-Katona theorem. We also thank Oded Goldreich, Luca Trevisan, Avi Wigderson, and the anonymous reviewers for helpful suggestions.

13. REFERENCES

[1] Ian Anderson. *Combinatorics of finite sets*. Dover Publications Inc., Mineola, NY, 2002. Corrected reprint of the 1989 edition.

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

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

[4] Michael Ben-Or and Nathan Linial. Collective coin-flipping. In Silvio Micali, editor, *Randomness and Computation*, pages 91–115. Academic Press, New York, 1990.

[5] Andrej Bogdanov and Luca Trevisan. On worst-case to average-case reductions for NP problems. In *Proceedings of 44th FOCS*, Cambridge, Massachusetts, 11–14 October 2003.

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

[7] Uriel Feige and Carsten Lund. On the hardness of computing the permanent of random matrices. *Computational Complexity*, 6(2):101–132, 1996.

[8] Joan Feigenbaum and Lance Fortnow. Random-self-reducibility of complete sets. *SIAM J. on Computing*, 22(5):994–1005, October 1993.

[9] Oded Goldreich and Leonid A. Levin. A hard-core predicate for all one-way functions. In *Proceedings of 21st STOC*, pages 25–32, Seattle, Washington, 15–17 May 1989.

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

[11] Russell Impagliazzo. Hard-core distributions for somewhat hard problems. In *Proceedings of 36th FOCS*, pages 538–545, Milwaukee, Wisconsin, 23–25 October 1995. IEEE.

[12] Russell Impagliazzo, Noam Nisan, and Avi Wigderson. Pseudorandomness for network algorithms. In *Proceedings of 26th STOC*, pages 356–364, Montréal, Québec, Canada, 23–25 May 1994.

[13] Russell Impagliazzo and Avi Wigderson. $P = BPP$ if E requires exponential circuits: Derandomizing the XOR lemma. In *Proceedings of 29th STOC*, pages 220–229, El Paso, Texas, 4–6 May 1997.

[14] Russell Impagliazzo and Avi Wigderson. Randomness vs. time: De-randomization under a uniform assumption. In *Proceedings of 36th FOCS*, Palo Alto, CA, November 8–11 1998. IEEE.

[15] Jeff Kahn, Gil Kalai, and Nathan Linial. The influence of variables on Boolean functions (extended abstract). In *Proceedings of 29th FOCS*, pages 68–80, White Plains, New York, 24–26 October 1988. IEEE.

[16] Adam Klivans and Rocco A. Servedio. Boosting and hard-core sets. *Machine Learning*, 53(3):217–238, 2003.

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

[18] E. Mossel and R. O'Donnell. On the noise sensitivity of monotone functions. *Random Structures & Algorithms*.

[19] Nisan. Pseudorandom generators for space-bounded computation. *Combinatorica*, 12, 1992.

[20] Noam Nisan. Pseudorandom bits for constant depth circuits. *Combinatorica*, 11(1):63–70, 1991.

[21] Noam Nisan and Avi Wigderson. Hardness vs randomness. *Journal of Computer and System Sciences*, 49(2):149–167, October 1994.

[22] Ryan O'Donnell. Hardness amplification within NP. In *Proceedings of 34th STOC*, pages 751–760. ACM, May 2002.

[23] Madhu Sudan, Luca Trevisan, and Salil Vadhan. Pseudorandom generators without the XOR lemma. *J. Comput. System Sci.*, 62(2):236–266, 2001. Special issue on the Fourteenth Annual IEEE Conference on Computational Complexity (Atlanta, GA, 1999).

[24] Luca Trevisan. List decoding using the XOR lemma. In *Proceedings of 44th FOCS*, Cambridge, Massachusetts, 11–14 October 2003.

[25] Luca Trevisan and Salil Vadhan. Pseudorandomness and average-case complexity via uniform reductions. In *Proceedings of the 17th Annual IEEE Conference on Computational Complexity*, pages 129–138, Montréal, CA, May 2002. IEEE.

[26] Emanuele Viola. The complexity of constructing pseudorandom generators from hard functions. Technical report, Electronic Colloquium on Computational Complexity, 2004. `http://www.eccc.uni-trier.de/eccc`. Preliminary version titled 'Hardness vs. Randomness within Alternating Time', in Eighteenth Annual IEEE Conference on Computational Complexity.

[27] Andrew C. Yao. Theory and applications of trapdoor functions (extended abstract). In *Proceedings of 23rd FOCS*, pages 80–91, Chicago, Illinois, 3–5 November 1982. IEEE.