Hardness Amplification for Errorless Heuristics

Andrej Bogdanov[*]   Muli Safra[†]

October 9, 2007

Abstract

An errorless heuristic is an algorithm that on all inputs returns either the correct answer or the special symbol $\perp$, which means "I don't know." A central question in average-case complexity is whether every distributional decision problem in NP has an errorless heuristic scheme: This is an algorithm that, for every $\delta > 0$, runs in time polynomial in the instance size and $1/\delta$ and answers $\perp$ only on a δ fraction of instances.

We study the question from the standpoint of hardness amplification and show that

- If every problem in $(\mathrm{NP}, \mathcal{U})$ has errorless heuristic circuits that output the correct answer on $n^{-2/9+o(1)}$-fraction of inputs, then $(\mathrm{NP}, \mathcal{U})$ has non-uniform errorless heuristic schemes.

- If every problem in $(\mathrm{NP}, \mathcal{U})$ has randomized errorless heuristic algorithms that output the correct answer on $(\log n)^{-1/10+o(1)}$-fraction of inputs, then $(\mathrm{NP}, \mathcal{U})$ has randomized errorless heuristic schemes.

In both cases, the low-end amplification is achieved by analyzing a new sensitivity property of monotone boolean functions in NP. In the non-uniform setting we use a "holographic function" introduced by Benjamini, Schramm, and Wilson (STOC 2005). For the uniform setting we introduce a new function that can be viewed as an efficient version of Talagrand's "random DNF".

1 Introduction

The notion of NP-hardness has led to many advances and insights both within and outside the theory of computation. From its beginnings in the works of Cook, Karp and Levin in the '70s, this project has never lost its steam and continuous to produce ingenious algorithmic techniques, as well as strong intractability results.

One objection to NP-hardness is that what it really measures is the difficulty of contrived instances as opposed to "real-life" ones. This begs the question of how real-life instances should be modeled. Average-case complexity views instances of a problem as samples from some distribution and postulates that efficiency is measured not with respect to all inputs, but instead with respect to "typical" inputs from this distribution. Formalizing this intuition into a natural set of definitions is a delicate matter which has been carried out beginning with Levin [Lev86], and following work by several other researchers [BCGL92, IL90, Imp95b].

---

[*]andrejb@tsinghua.edu.cn. ITCS, Tsinghua University, FIT Building 4-608-7, Beijing 100084, China. Work done while at DIMACS, Rutgers Univesity.

[†]safra@post.tau.ac.il. Tel Aviv University, Israel

1.1 Average-case complexity

The objects of study of average-case complexity are *distributional problems*: These are pairs $(L, \mathcal{D})$, where L is a language and $\mathcal{D}$ is a distribution from which one chooses instances for L. Roughly, problems are considered "tractable on average" if there is a decision algorithm which determines membership in L for "typical" instances chosen from $\mathcal{D}$.

The class distributional NP consists of all distributional problems $(L, \mathcal{D})$, where L is in NP and $\mathcal{D}$ is a samplable distribution. The main question in average-case complexity is whether all problems in distributional NP are tractable. This is the average-case version of the P versus NP question.

A central result in average-case complexity is the existence of complete problems for distributional NP [Lev86, IL90]: There exists a single problem $(L, \mathcal{U})$, where $\mathcal{U}$ is the uniform distribution, for which it is known that if $(L, \mathcal{U})$ were tractable on average, then all problems in distributional NP would be tractable on average.

We wish to stress the central role of the uniform distribution in average-case complexity; in some sense, it is the "hardest" distribution and will be the focus of our study here. We will use the notation $(\mathrm{NP}, \mathcal{U})$ for the subclass of distributional NP where the distribution is uniform.

It is widely believed that there are problems in distributional NP that are intractable on average. However proving this would in particular establish that $\mathrm{P} \neq \mathrm{NP}$. A more reasonable question to ask is whether $\mathrm{P} \neq \mathrm{NP}$ implies the existence of hard problems in distributional NP, but even this appears out of reach for current techniques [BT03, Vio05]. Below we discuss an easier variant of this question which is closer to the topic of our work.

We now explain in more detail what it means for a distributional problem to be tractable on average.

Three definitions Intuitively, the term "average-case tractability" can have three reasonable interpretations. Suppose we are given a distributional problem $(L, \mathcal{D})$. Consider the following meanings of "solving the problem efficiently on average":

1. We have an efficient algorithm that gives the correct answer for most instances of L, but gives the wrong answer on a few instances.

2. We have an efficient algorithm that gives the correct answer for most instances of L, and says "I don't know" on the remaining ones.

3. We have an algorithm that solves all instances of L, but is efficient only for most instances of L.

It turns out that (in the proper formalization) the second and third definitions are equivalent (cf. [Imp95b]), while the first one is strictly weaker.

To illustrate the difference between 1 and 2, consider the following widely studied problem (e.g., [Fei02]): Given a random 3CNF φ that has n variables and $50n$ clauses, is φ satisfiable? Under definition 1, the problem is trivial as most φ are unsatisfiable; under definition 2, it is equivalent to certifying unsatisfiability of boolean formulas, an interesting subject of study which has not yet delivered the answer to this problem.

Algorithms that satisfy definition 1 are called *heuristics*; algorithms that satisfy definitions 2 and 3 are called *errorless heuristics*.

Degrees of hardness and our results Another natural question to ask is what it means to solve an algorithm on "most instances": Are 1% of the instances an adequate notion, or is it 51%, or 99%, or $1 - 1/n$, or $1 - n^{-100}$? We cannot push this too far because at the end of the spectrum we obtain algorithm that solve all instances, thus we are back to worst-case complexity.

On the high end of the spectrum, it is irrelevant whether we choose $1 - n^{-0.01}$, $1 - 1/n$, or $1 - n^{-100}$ as our notion of hardness: If all of $(\mathrm{NP}, \mathcal{U})$ is tractable under one of these notions, then it is tractable under the other notions (cf. [Imp95b]). In average-case complexity, this is the gold standard of tractability; languages that satisfy it are said to have *heuristic schemes*.

In the lower end, the situation has been studied for the case of heuristics. Here the least we can require is that the heuristic solves the problem on $1/2 + \varepsilon$ fraction of instances, where ε is some small constant or a shrinking function of the instance size n. The following containments are known:

- O'Donnell [O'D02] showed that if every language in $(\mathrm{NP}, \mathcal{U})$ can be solved on $1/2 + n^{-1/3 + o(1)}$ fraction of instances by polynomial-size circuits, then every language in $(\mathrm{NP}, \mathcal{U})$ has non-uniform heuristic schemes.

- Healy, Vadhan, and Viola [HVV04] showed, for arbitrary $c > 0$, that if every language in $(\mathrm{NP}, \mathcal{U})$ can be solved on $1/2 + n^{-c}$ fraction of instances by polynomial-size circuits, then all *balanced* languages in $(\mathrm{NP}, \mathcal{U})$ (languages that have the same number of "yes" and "no" instances on every input length) have non-uniform heuristic schemes.

- Trevisan [Tre03, Tre05] showed, for some universal $\alpha > 0$, that if every language in $(\mathrm{NP}, \mathcal{U})$ can be solved on $1/2 + (\log n)^{-\alpha}$ fraction of instances by randomized heuristics, then every language in $(\mathrm{NP}, \mathcal{U})$ has randomized heuristic schemes.

The subject of this paper is the study of the same question in the setting of errorless heuristics. Errorless heuristics are meaningful even if they solve the problem only on an ε-fraction of instances, where ε is some small constant or even a shrinking function of n. Our results are the following:

- If every language in $(\mathrm{NP}, \mathcal{U})$ can be solved on a $n^{-2/9 + o(1)}$ fraction of instances by polynomial size circuits, then every language in $(\mathrm{NP}, \mathcal{U})$ has non-uniform errorless heuristic schemes;

- If every language in $(\mathrm{NP}, \mathcal{U})$ can be solved on a $(\log n)^{-1/10 + o(1)}$ fraction of instances by randomized errorless heuristic algorithms, then every language in $(\mathrm{NP}, \mathcal{U})$ has randomized errorless heuristic schemes.

Our first result should be compared to O'Donnell's result for errorless heuristics. O'Donnell [O'D02] argues that the exponent $1/3$ is a barrier in his analysis. Similarly (but for a different analytical reason) our analysis cannot go beyond exponent $1/3$, though we currently don't know how to match this exponent. Healy, Vadhan and Viola [HVV04] circumvent this barrier for balanced languages by amplifying over dependent inputs. However we were unable to apply their ideas to our proof. We do not know if the constant $1/10$ in the uniform result is optimal. (Trevisan does not give a specific constant in his result.)

Errorless heuristics that work on an ε fraction of inputs can be turned into heuristics that work on $\varepsilon/2$ fraction of inputs: If the errorless algorithm answers, output the answer; if it says "I

don't know" flip a coin[1] (or wire a bit of advice). In general there is no reverse transformation. However Trevisan observed that a reverse transformation trivially holds in settings with worst-case to average-case equivalence: For instance, if (PSPACE,$\mathcal{U}$) had errorless heuristics that worked even on only $1/\mathrm{poly}(n)$ fraction of instances of size n, then the errorless heuristic can be turned into a heuristic that works on $1/2 + 1/\mathrm{poly}(n)$ fraction of inputs. By worst-case to average-case equivalence [STV01, TV02] this implies PSPACE $\subseteq$ BPP, so the heuristic, errorless, and worst-case model are then all equivalent.

1.2 Heuristic versus errorless amplification

Most known proofs of hardness amplification for NP in the heuristic setting take as their model Impagliazzo's proof of the XOR lemma [Imp95a]. (The only exception we know of is a recent proof of Buresh-Oppenheim et al. [BKS06] for uniform amplification.) Impagliazzo's proof consists of two steps, that we briefly sketch from the viewpoint of [STV01, Tre03]. We start with a function f that is computationally hard on an ε-fraction of inputs.

1. First Impagliazzo shows the existence of a *hard-core set H* for f of size $\Omega(\varepsilon)$: This is a set of inputs for f where f is very hard, in particular conditioned on an input falling in H, $f(x)$ is computationally indistinguishable from a random bit.

2. Now we consider the function $f(x_1) \oplus \ldots \oplus f(x_k)$ on a random input. As long as one of the inputs $x_1, \ldots, x_k$ falls into the hardcore set H — for $k \gg 1/\varepsilon$ this happens with high probability — the bit $f(x_i)$ becomes computationally indistinguishable from random, so the output of $f(x_1) \oplus \ldots \oplus f(x_k)$ also appears random.

O'Donnell [O'D02] shows that the second step carries through even if instead of XOR, another function that is sufficiently sensitive to noise is used. He then shows that there exist monotone functions in NP with the appropriate noise sensitivity, which allows him to carry out the construction in NP.

What happens when we try to imitate Impagliazzo's and O'Donnell's argument in the errorless setting? The step of constructing a hard-core set carries over (for a different notion of "hard-core"), and in fact is easier to do in the errorless setting. However, there does not seem to be a useful analogue of "pseudorandomness" in the errorless setting that allows for a natural extension of the second step.

Instead we follow a different approach. We begin with an "errorless XOR lemma" for circuits, which is considerably easier than the heuristic version and is much closer in spirit to Yao's lemma on hardness amplification for one-way functions [Yao82, Gol01]. We then follow O'Donnell's basic approach of abstracting the properties used by XOR in the proof of our lemma. In our case we will not be interested noise-sensitivity but in a different property of boolean functions. Say a coordinate is *pivotal* for an input if flipping its value causes the output value of the function to change. For us, it turns out that the relevant property of XOR is the following one: For XOR on k variables, all inputs of XOR have n pivotal coordinates. It turns out that it is sufficient (with some loss in parameters) that *most* inputs of the amplifying function have n^α pivotal coordinates for some $\alpha > 0$.

---

[1]Technically this won't be a heuristic algorithm since some of the answers may be ambiguous, but Trevisan's result applies even to such algorithms with ambiguities.

This leads us to the study of monotone boolean functions that not only have large *total influence* (i.e., the expected number of pivotal coordinates of a random vertex is large), but whose *vertex boundary* is also large — that is *most* inputs have many pivotal coordinates. This distinction is not merely technical: For instance, the majority function has the largest possible total influence among boolean functions, but its vertex boundary is very small.

1.3   Hardness amplification

The general approach to hardness amplification can be explained as follows. Suppose we know we can compute every function in $(\mathrm{NP}, \mathcal{U})$ on some small fraction of inputs. Now we want to compute some $f \colon \{0,1\}^n \to \{0,1\}$ on most inputs. For simplicity we will assume that f is a balanced function. This assumption can be made without loss of generality at some cost in the parameters using a technique of O'Donnell [O'D02].

We are given some function $C \colon \{0,1\}^k \to \{0,1\}$ that is highly sensitive to changes of its input coordinates (the precise sense will be discussed shortly), and we consider the combined function $C \circ f \colon \{0,1\}^{n \times k} \to \{0,1\}$, which is a function that takes k sets of n inputs, applies f to each set, and applies C to the outcome.

By assumption we have an oracle that computes $C \circ f$ on some small fraction of inputs. Now assuming C is highly sensitive, we proceed as follows. On input x for f, choose randomly k strings $x_1, \ldots, x_k$ of length n, replace the ith string by x, then call the oracle that computes $C \circ f$ on input $(x_1, \ldots, x_n)$.

It is possible that we were unlucky and that the oracle for $C \circ f$ has answered $\bot$. In this case, we attempt the reduction again and again. We show that for most x, after a reasonable number of attempts we obtain an answer from the oracle.

Once we have obtained an answer for $C \circ f$, how do we interpret it? Suppose we were lucky, and it happened that the i-th coordinate of the point $(f(x_1), \ldots, f(x_n))$ was pivotal for C. Then assuming we had access to the values $f(x_1), \ldots, f(x_n)$ except for $f(x_i)$, we could determine the value $f(x)$ from the output of the oracle.

This leads us to the requirement that the function C has to have many pivotal coordinates on many inputs. Observe that if the oracle can compute $C \circ f$ on an ε-fraction of inputs, then it better be the case that the fraction of inputs that do not have pivotal coordinates be at most ε. Moreover, to hope for any sort of amplification, at least intuitively, we want most vertices of f to have many pivotal neighbors.

The most extreme example of a function that has the largest possible number of pivotal coordinates for the maximum possible number of inputs is the XOR function. However to carry out the amplification in NP we want a monotone function, and to obtain good parameters we want a monotone function in NP (this issue is implicit in [O'D02] and explicit in [HVV04]).

Our goal now is to find a highly "sensitive" monotone Boolean function C under this definition.

Let us point out a specific difficulty that arises in the setting of *errorless* heuristics. Observe that if on input x the reduction algorithm outputs $b \neq \bot$, it must be the case that the reduction has received b as an answer to at least one of its oracle queries, namely $C(f(x_1), \ldots, f(x_n)) = b$ for at least one query $(x_1, \ldots, x_n)$ made by the reduction. Indeed, suppose that $f(x) = b$ but the reduction has only received answers of the form $\overline{b}$ or $\bot$ from the oracle. Then all the answers

provided by the oracle are consistent with both the possibilities $f(x) = b$ (by assumption) and $f(x) = \overline{b}$ (by the monotonicity of C). Hence the reduction has found no certificate that $f(x) = b$ and it is forced to output $\perp$.

Thus if the rate of $\perp$ answers is at least 50%, an adversarial oracle can force the reduction to either never output "yes", or never output "no". For instance, if $\Pr[(C \circ f)(x_1, \ldots, x_n) = 0] \leq 1/2$, then an adversarial oracle may answer $\perp$ on all "no" instances of $C \circ f$, thereby forcing the reduction to answer $\perp$ on all "no" instances of f. Similarly if $\Pr[(C \circ f)(x_1, \ldots, x_n) = 1] \leq 1/2$ then the reduction can be forced to answer $\perp$ on all "yes" instances of f. So if the function f is balanced, this approach can never get us past a 50% rate of "I don't know" answers.

To overcome this problem we use separate combining functions C and $C^\dagger$ to amplify zeros and ones. The function C will be highly unbalanced towards zero, while $C^\dagger$ will be highly unbalanced towards one. It is easier to explain why this makes sense from the perspective of hardness. If NP has a language L that is mildly hard for errorless heuristics, then either the language is mildly hard to certify (hard on "yes" instances) or mildly hard to refute (hard on "no" instances). If L is hard to refute, we use C to construct a new language that is very hard to refute; most of the instances in this language will be "no" instances. If L is hard to certify, we use $C^\dagger$ to construct a new language that is very hard to certify; most of the instances in this language will be "yes" instances.

1.4 Amplifier functions

To summarize from the above discussion, we need two monotone functions C and $C^\dagger$ that satisfy the following requirements. The function C has to evaluate to zero almost everywhere, while $C^\dagger$ has to evaluate to one almost everywhere. Both C and $C^\dagger$ must have many pivotal coordinates (some constant power of n) almost everywhere; and C and $C^\dagger$ should be both in NP.

We say a monotone function C is a $(t, \delta)^\uparrow$-*amplifier* if on all but a δ fraction on inputs C evaluates to zero but has at least t pivotal coordinates, namely

$$\Pr\big[f(x) = 1 \text{ or } |\{i : f(x + e_i) = 1\}| < t\big] \leq \delta.$$

here $x + e_i$ is the input obtained by flipping the ith coordinate of x. Similarly, $C^\dagger$ is a $(t, \delta)^\downarrow$-*amplifier* if on all but a δ fraction on inputs $C^\dagger$ evaluates to one but has at least t pivotal coordinates.

In the discussion we will focus on the construction of C, and we will define $C^\dagger$ as the function

$$C^\dagger(x_1, \ldots, x_n) = \overline{C(\overline{x}_1, \ldots, \overline{x}_n)}.$$

All required properties of $C^\dagger$ then follow directly from the corresponding properties of C, except for membership in NP. In particular, if C is a $(t, \delta)^\uparrow$-amplifier, then $C^\dagger$ is a $(t, \delta)^\downarrow$-amplifier.

We show that if a monotone function on n inputs is an $(n^\alpha, n^{-\beta})^\uparrow$-amplifier, where $\alpha, \beta > 0$ are constant and n is sufficiently large, then $\alpha + \beta \leq 1/2$. We are interested in matching these parameters as closely as possible, especially getting β as large as possible. Roughly, β determines the strength of the assumption, and α determines the strength of the conclusion in the result. Once α is constant, no matter how small, simple padding techniques can be used to improve the conclusion, so for our application we are interested in $(n^\alpha, n^{-1/2+\varepsilon})^\uparrow$ amplifiers where α, ε are small constants.

Let us consider some known monotone functions. One candidate is an unbalanced version of the tribes function — the OR of ANDs over pairwise disjoint variables. Let us look at a tribes on n variables where each clause has size t, so the number of clauses is n/t.

The probability that a particular clause is satisfied is 2^{-t}, while the probability that the clause is false but contains a pivotal bit is $t2^{-t}$ (each bit is pivotal with probability 2^{-t}). If we choose t so that all clauses are false with probability ρ, then almost all the inputs will have about $t\rho$ pivotal coordinates. Setting ρ appropriately shows that tribes is roughly a $(\log n, 1/\log n)^{\uparrow}$-amplifier.

The pivotality of the tribes function can in fact be improved by replacing each input with a recursive majority over indepedent bits (this was pointed out to us by Elchanan Mossel.) Using this construction, one can show that tribes concatenated with recursive majorities of 3 inputs is in fact a $(n^{\alpha}, 1/\log n)^{\uparrow}$-amplifier for some constant $\alpha > 0$. However, we do not know how to improve the second parameter in this construction. In particular it appears that as long as the first parameter is nontrivial, it must be the case that this function evaluates to 1 on at least a $\Omega(1/\log n)$ fraction of inputs.

It appears that the main limitation of tribes comes from the fact that its clauses are independent. This leads us to consider monotone functions that when viewed in DNF have dependencies among the clauses.

One such candidate is the "random DNF" of Talagrand. The reason that random DNF should be better amplifier than tribes is that by introducing dependencies among the clauses, we can increase the size of each clause, and thereby increase the probability that the function evaluates to zero in terms of the number of variables n. Once this obstacle is surmounted we can obtain much better amplification parameters.

However, random DNF is very nonconstructive — finding a DNF with the prescribed amplification parameters by enumeration may take time doubly exponential in n.

The DDNF function   Towards satisfying the efficiency requirement, we introduce a new function that achieves the same parameters as Talagrand's but is not random. This function can be viewed as a derandomization of Talagrand's construction; we call it DDNF, which stands for "designed DNF".

The amplification properties of Talagrand's function are determined by the intersection sizes among pairs of clauses. These decrease exponentially: About a λ-fraction of pairs has intersection 1, about a λ^2 fraction has intersection 2, and so on, where λ is a small constant (in fact slightly subconstant). The DDNF function meets this behavior of intersection sizes constructively.

The DDNF function (for a typical setting of parameters) is a function over $|\mathbb{F}| \times |\mathbb{F}|$ variables $z_{i,j}$, where $\mathbb{F}$ is a finite field of size $\sqrt{n}$. Each clause of DDNF (when viewed in DNF) is indexed by a polynomial of degree $d \approx \sqrt{n}/\log n$ and contains the variables $z_{i,p(i)}$ for all $i \in \mathbb{F}$:[2]

$$\mathrm{DDNF}(z_{1,1}, \ldots, z_{t,q}) = \bigvee_{p} \bigwedge_{i=1}^{t} z_{i,p(i)}$$

We choose the degree of the polynomials, which in turn determines the number of clauses, so that the vanishing probability of DDNF is about $1 - n^{-1/2+\alpha}$. Then by the same argument as for tribes,

---

[2]It is possible to view the construction described above as a specific implementation of combinatorial designs [Nis91, NW94] using polynomials. Hartman and Raz [HR03] in fact analyze this construction.

DDNF will have about n^α pivotal coordinates in expectation for a random vertex. However, it does not immediately follow that almost all the inputs will have many pivotal coordinates. Using the fact that the pairwise intersection sizes decrease exponentially (which is related to basic properties of polynomials), a second moment argument allows us to deduce that all but $n^{-\alpha}$ fraction of inputs have about n^α pivotal coordinates, thus DDNF is an $(n^\alpha, n^{-\alpha})$-amplifier.

We can improve the second parameter by taking the OR of a constant number independent copies of DDNF: This increases the nonvanishing probability by at most a constant factor, but improves the vertex-boundary exponentially. Using appropriate parameters we obtain that the OR of independent DDNFs is a $(n^\alpha, O(\sqrt{\log n} \cdot n^{-1/2+\alpha}))$ amplifier — optimal up to logarithmic factors for every α.

The DDNF function is in NP. However, we do not know if the function DDNF† is also in NP. Therefore we don't know if it is possible to use these two function families for nonuniform errorless amplification in NP (for the range of parameters considered here). Nevertheless, as we discuss in the next section, DDNF and DDNF† can be used for uniform amplification (where much weaker computability requirements than membership in NP suffice).

Percolation on the butterfly graph The "holographic function" of Benjamini, Schramm, and Wilson [BSW05] is an alternative to DDNF that was introduced in a different context. The variables of this function correspond to edges in a certain directed graph H that we describe below, and their value indicates the presence or absence of the edge. The function indicates the presence of a cycle in H of a given length.

The vertices of the graph H are indexed by pairs (x,i), where $0 \le x < h$ and $0 \le i < w$, and h is a power of two. Each vertex (x,i) has two outgoing edges to the vertices $(2x \bmod h, i+1 \bmod w)$ and $(2x+1 \bmod h, i+1 \bmod w)$. This graph has hw vertices and $n = 2hw$ edges. We assign a boolean value z_e to each edge e, and define G_z to be the subgraph of H consisting of all the edges e such that $z_e = 1$. The *holographic function*[3] is defined as

$$\text{HOL}_{h,w}(z_1,\ldots,z_n) = \begin{cases} 1, & \text{if } G_z \text{ contains a directed cycle of length } w \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

For the correct choice of parameters ($w \approx \sqrt{h}$), we show that the holographic function (with certain modifications) is a $(n^\alpha, n^{-1/3-\alpha} \cdot \text{polylog}(n))$ amplifier for every constant α. These parameters are quantitatively inferior than those obtained from the DDNF function. However, the holographic function is known to be efficiently computable, so both HOL and HOL† are in NP. This makes the function suitable for non-uniform amplification.

1.5 Uniform amplification

O'Donnell's approach of hardness amplification in NP for polynomial-size circuits in the heuristic setting was extended by Trevisan [Tre03, Tre05] to work for randomized algorithms instead of

---

[3]The name "holographic function" comes from the property that with high probability over the input the function can be evaluated in a way that makes every input variable unlikely to be read [BSW05].

circuits (at some cost in the parameters).[4]

Our proof of uniform amplification mimics Trevisan's, though there are several differences that we point out here.

Amplification and decoding The fundamental difficulty in obtaining a uniform amplification result is information-theoretic: Most hardness amplification procedures are "black-box": Suppose we want to solve f on most inputs given an algorithm that solves some f' obtained from f on few inputs. The amplification proof can be viewed as an oracle procedure that, given access to an oracle that matches f' on some small fraction of inputs, computes another oracle that matches f on most inputs.

From this perspective [STV01, TV02, Tre03], black-box amplification proofs are local list-decoding algorithms in disguise: Think of f as the message, the proof as the encoding, f' as the codeword and the oracle for f' as a corrupted codeword. Uniform amplification corresponds to designing a unique decoding algorithm, but then we are limited by half the codeword distance. In the heuristic setting, this corresponds to amplifying from hardness $3/4 + \varepsilon$, and this was carried out by Trevisan [Tre03].

Beyond this range of parameters, unique decoding is impossible, so we settle for list-decoding. In the language of hardness amplification, given access to an oracle that matches f' on $1/2 + \varepsilon$-fraction of inputs, we want to produce a list of $\mathrm{poly}(1/\varepsilon)$ circuits one of which computes a function close to f.

In the errorless setting the coding-theoretic analogy is less natural: We are given a corrupted version of f' that agrees with it at an ε fraction of places, but all the corruptions are erasures; we want to recover a list of functions, one of which is close to f, but where all the places where this function disagrees with f are erasures.[5]

For $\varepsilon = O(\log n)^{-1/10+o(1)}$, our construction gives a list of $\log n$ such functions.

We wish to point out an interesting difference between decoding in the errorless and heuristic setting *within the unique decoding radius.* For this we need to look a bit more deeply into local decoding algorithms. The algorithm is given oracle access of a corrupted codeword f', obtains an input x and wants to compute $f(x)$ (for most x). It tosses some random coins, makes a few queries in f' and outputs an answer.

In the heuristic setting, the local decoding algorithm of Trevisan [Tre05] has the property that the algorithm can flip its coins before it sees the query x. In the errorless setting, we do not believe that this can be achieved; it is essential that the randomness is chosen after the input. In the language of reductions, Trevisan gives a randomized reduction that outputs a deterministic heuristic circuit. In contrast, we give a deterministic reduction that outputs a randomized errorless heuristic circuit.

Decision versus search, balancing, and advice Trevisan's construction [Tre05] consists of two main steps: a "black-box" step that gives a list of $\mathrm{poly}(n)$ candidate deterministic circuits, one of which with high probability solves a search version of L on a $1 - 1/n$ fraction of inputs, and a

---

[4]An alternative proof of Trevisan's result was recently provided by Buresh-Oppenheim et al. [BKS06]. Their proof is based on a different "advice-efficient" proof of the XOR lemma by Impagliazzo, Jaiswal, and Kabanets [IJK06].

[5]*Exact* decoding from erasures is very natural; it is this *approximate* decoding where the decoding itself has to differ from the message only by erasures that is less natural.

"non-black-box" step that uses the candidate circuits to solve L. The number of candidate circuits corresponds to the "advice" used by the reduction, which has two sources:

- Low-end amplification, which turns algorithms for L with very small advantage into algorithms for L with constant advantage. Advice is necessary in this stage because the corresponding local decoding problem is outside the unique decoding radius.

- A "balancing step", where a general language is turned into one that is almost balanced on infinitely many input lengths. Here, advice is needed to specify the corresponding length in the almost balanced language; the advice essentially encodes the approximate fraction of "yes" instances in the language.

The amount of advice of the first type depends only on the fraction of inputs that every language in $(\mathrm{NP}, \mathcal{U})$ can be solved on; this determines the range of parameters for which the amplification result applies. In contrast, the advice of the second type depends on how balanced we have to make our language; the more balanced we want it, the more advice we have to use. In fact, the ambiguity ($2^{\text{size of advice}}$) is always bigger than $1/\text{balance}$.[6]

Trevisan's high end amplification produces a circuit that, roughly, solves L on a $1 - 1/n$-fraction of inputs but only as long as it is $1/n$-balanced.[7] Thus to solve L on a $1 - 1/n$ fraction of inputs we must use $\log n$ advice.

In the heuristic setting, this is adequate: Given $\log n$ advice, we obtain a list of n circuits one of which solves the search version of L, run them on the input and accept if one of them produces a witness. The failure probability is only $1/n$.

However, in the errorless setting we cannot do so: Suppose we are given n circuits one of which is an errorless heuristic circuit for L, and the other ones may behave arbitrarily. On input x, we query all the circuits. If any of the circuits yields a witness, then we accept; but what if not? Should we reject or say "I don't know"? To be sure, we will have to output "I don't know" as long as *any one* of our n circuits says "I don't know". But most of the circuits are defective, and it may be the case that one of them outputs "I don't know" all the time, in which case we end up answering "I don't know" on all $x \notin L$.

One thing we can do is eliminate all the circuits that output "I don't know" more than, say, $2/n$ fraction of the time by sampling on random inputs. So now we have n circuits each one of which says "I don't know" on at most $2/n$ fraction of inputs. It may still be the case that for every input $x \notin L$, some circuit will always say "I don't know".

We now come to the main problem with using Trevisan's construction in its original form: The ambiguity is always bigger than the inverse of the error probability. To carry out the reduction in the errorless setting, we cannot afford so much ambiguity. In particular the advice that comes from the balancing step is too large.

To solve this problem, we show that in the errorless setting, the advice needed in the balancing step can be efficiently computed. This advice is simply the approximate probability p_n that a random

---

[6]It is inverse linear in terms of the input length of the original language, so a bit superlinear in terms of the input length of the "balanced" language.

[7]Here we are oversimplifying Trevisan's construction. We merely with to illustrate the point that the the fraction of inputs on which the algorithm fails always exceeds the advice.

instance of length n is a "yes" instance of L. We approximate p_n by sampling random instances of L and determining the fraction of yes instances. But how do we know if an instance of L is a "yes" instance? By Trevisan's main theorem (his *theorem*, not his proof), L has heuristic schemes, so we can use the heuristic scheme for L to determine if a random instance is a "yes" instance (with some small error).

In uniform amplification, the input size of the amplifier is roughly polynomial to the advice used by the reduction. Thus for the reduction to be efficient we have to restrict ourselves to logarithmic advice. In this setting we can use the DDNF function as an (optimal) amplifier since the input length of the amplifier is sublogarithmic in the input length of the instance, and DDNF is computable in exponential time. In contrast we cannot use Talagrand's random DNF since computing this function (e.g., the lexicographically smallest DNF with given amplification parameters) requires doubly exponential time.

1.6 Organization

In Section 2 we introduce the relevant concepts from probability, average-case complexity, and boolean functions. Section 3 begins with a proof of the errorless XOR lemma, which is not used anywhere but serves as a model for all the amplification lemmas in the paper and we urge the reader not to skip it. We then introduce the notion of monotone functions with large boundaries and prove a monotone amplification lemma for such functions.

In Section 4 we discuss DDNF and HOL as amplifiers. Section 5 contains a proof of the amplification result in the non-uniform setting. Section 6 contains a proof of our result on uniform amplification. Appendix A and Appendix B give proofs of the amplification properties of DDNF and the holographic function. Appendix C shows extremal properties of the DDNF function and limitations on the parameters of amplifiers.

2 Preliminaries

2.1 Distributions

For a discrete probability space Ω we use μ_Ω to denote the corresponding probability measure. Namely, for a set B, $\mu_\Omega(B)$ is the quantity $\Pr_{\omega \sim \Omega}[\omega \in B]$. When Ω is a finite set, we think of it as a probability space with the uniform distribution. When the probability space is clear from the context we may omit the subscript.

For integers n and $p \in [0,1]$, we use $\mu_{n,p}$ to denote the p-biased measure on $\{0,1\}^n$, that is,

$$\mu_{n,p}(x_1,\ldots,x_n) = (1-p)^{|\{i:x_i=0\}|} \cdot p^{|\{i:x_i=1\}|}.$$

We abbreviate $\mu_{n,1/2} = \mu_{\{0,1\}^n}$ by μ_n.

We use $x \sim \Omega$ to denote an x sampled uniformly from the set Ω, and $x \sim_p \{0,1\}^n$ for an x sampled from the p-biased distribution on $\{0,1\}^n$.

Distance between distributions Let Ω be a probability space with N elements, and μ, μ' be probability measures on Ω. We define

- The *statistical distance* $\mathrm{sd}(\mu, \mu') = \max_{T \subseteq \Omega}(\mu(T) - \mu'(T))$,

- The ℓ_1 *distance* $\ell_1(\mu, \mu') = \sum_{x \in \Omega} |\mu(x) - \mu'(x)|$, and

- The ℓ_2 *distance* $\ell_2(\mu, \mu') = \left(\sum_{x \in \Omega}(\mu(x) - \mu'(x))^2\right)^{1/2}$.

It holds that for all μ and μ'

$$2\,\mathrm{sd}(\mu, \mu') = \ell_1(\mu, \mu') \leq \sqrt{N} \cdot \ell_2(\mu, \mu').$$

It is also easy to check that for every $p \in [0, 1]$,

$$\ell_2(\mu_{n,p}, \mu_{n,1/2}) = \sqrt{\left(p^2 + (1-p)^2\right)^n - 2^{-n}},$$

so we obtain the following useful fact.

Fact 1. *For every n and $\gamma \in [-1/2, 1/2]$,*

$$\mathrm{sd}(\mu_{n,1/2+\gamma/2\sqrt{n}}, \mu_{n,1/2}) \leq \gamma.$$

2.2 Average-case complexity

We introduce the relevant notions and notation from average-case complexity. Some of the definitions are subtle and may appear counterintuitive. For motivation we refer the reader to the study [BT06], from where the definitions are taken. We specialize our definitions to the uniform distribution.

We give definitions for both circuits and randomized algorithms.

Definition 2 (Errorless Heuristic Circuits). *Let L be a language, $\delta : \mathbb{N} \to [0, 1]$. A family C of circuits is an* errorless heuristic *for $(L, \mathcal{U})$ with* failure probability δ *if*

- *For every x, $C(x)$ outputs either $L(x)$ or the special failure symbol $\perp$, and*

- *For every n, $\Pr_{x \sim \{0,1\}^n}[C(x) = \perp] \leq \delta(n)$.*

For randomized algorithms, the definition is a bit more delicate because randomized heuristics can fail in two ways: A bad choice of input and a bad choice of randomness.

Definition 3 (Errorless Heuristic Algorithms). *Let L be a language, $\delta : \mathbb{N} \to [0, 1]$. A randomized algorithm A is an* errorless heuristic *for $(L, \mathcal{U})$ with* failure probability δ *if A always outputs one of 1 ("yes"), 0 ("no"), or $\perp$ ("fail") and*

- *For every n and x, $\Pr_A[A(x) \notin \{L(x), \perp\}] \leq 1/4$, and*

- *For every n, $\Pr_{x \sim \{0,1\}^n}[\Pr_A[A(x) = \perp] \geq 1/4] \leq \delta(n)$.*

We define $\text{Avg}_{\delta(n)}\text{P}/\text{poly}$ and $\text{Avg}_{\delta(n)}\text{BPP}$ to be the classes of problems that admit polynomial-size errorless heuristic circuit families and polynomial-time errorless heuristic randomized algorithms, respectively.

We also have the following definition, which treats the error probability δ uniformly as part of the input.

Definition 4 (Errorless Heuristic Scheme). *A randomized algorithm A is a* (fully polynomial-time) *randomized errorless heuristic scheme for $(L,\mathcal{U})$ if each algorithm A takes two inputs x and δ, runs in time $\text{poly}(|x|/\delta)$, and*

- *For every $\delta > 0$ and every x, $A(x;\delta)$ outputs either $L(x)$ or $\perp$;*

- *For $\delta > 0$, $\text{Pr}_{x \sim \{0,1\}^n}[A(x,\delta) = \perp] \leq \delta$.*

We define AvgBPP as the class of all problems $(L,\mathcal{U})$ that admit fully polynomial-time randomized errorless heuristic schemes, respectively. It follows that $\text{AvgBPP} \subseteq \bigcap_{c>0} \text{Avg}_{n^{-c}}\text{BPP}$. For circuits, we simply define $\text{AvgP}/\text{poly} = \bigcap_{c>0} \text{Avg}_{n^{-c}}\text{P}/\text{poly}$.

Impagliazzo [Imp95b] (see also [BT06, Section 3.2]) observes the following, which holds for every constant $\gamma > 0$:

$$\text{If } (\text{NP},\mathcal{U}) \subseteq \text{Avg}_{n^{-\gamma}}\mathbf{C}, \text{ then } (\text{NP},\mathcal{U}) \subseteq \text{Avg}\mathbf{C}. \qquad (2)$$

where $\mathbf{C}$ can be either P/poly or BPP.

All the algorithms and circuits discussed in this paper are errorless heuristics, so when we say "The algorithm (or circuit) solves the problem on 99% inputs", what we mean is that on the remaining 1% it never outputs the incorrect answer in the case of circuits, and it does so with probability at most 1/4 over its internal randomness in the case of randomized algorithms.

For a circuit family C, we will say C *solves $L = \{f_n\}$ on a $(1 - \delta(n))$-fraction of zero inputs* if C is an errorless heuristic circuit family for L and for all n and x of length n,

$$\text{Pr}_{x \sim f_n^{-1}(0)}[C(x) = \perp] < \delta(n).$$

Similarly we define C *solves L on a $(1 - \delta(n))$-fraction of one inputs*, and we define an analogous notion for randomized algorithms.

Observe that a circuit (or algorihtm) that solves L on a $(1 - \delta(n))$-fraction of zero inputs is also required to be errorless on the one inputs; however the circuit is allowed to answer $\perp$ on as many of the one inputs as it wishes.

2.3 Boolean functions

For a boolean function $f : \{0,1\}^n \to \{0,1\}$, and an integer k, we use f^k for the function from $\{0,1\}^{n \times k}$ to $\{0,1\}^k$ given by

$$f^k(x_1, \ldots, x_k) = (f(x_1), \ldots, f(x_k)).$$

Given two functions $f : \{0,1\}^k \to \{0,1\}^m$ and $g : \{0,1\}^n \to \{0,1\}^k$, their *composition* is the function $(f \circ g)(x) = f(g(x))$. We often consider compositions of the type $f \circ g^k$, and we may omit k when it is notationally cumbersome and clear from context.

We say $f : \{0,1\}^n \to \{0,1\}$ is γ-*balanced* if $|\mu_n(f^{-1}(1)) - 1/2| \leq \gamma$. We say f is *monotone* if for every $x \preceq y$, it holds that $f(x) \leq f(y)$. Here, $\preceq$ is the partial order on $\{0,1\}^n$ defined by $(x_1, \ldots, x_n) \preceq (y_1, \ldots, y_n)$ if $x_i \leq y_i$ for all i. For a monotone f, its *monotone complement* is the function

$$f^\dagger(x_1, \ldots, x_n) = \overline{f(\overline{x}_1, \ldots, \overline{x}_n)}.$$

A special family of monotone functions are the *threshold functions*: f is a threshold function if for some integer k, $f(z) = 1$ when z has more than k ones and $f(z) = 0$ when z has fewer than k ones.

3 Amplification lemmas

3.1 The XOR lemma

The XOR lemma is the basic tool for achieving amplification of hardness for heuristic algorithms that make errors. The lemma has various proofs, though Impagliazzo's proof is the one that has to most fruitful generalizations in the context of hardness amplification.

However the known versions of the XOR lemma do not extend to the setting of errorless heuristics. Indeed, hardness amplification for errorless heuristics is much closer in spirit to analogous results about one-way functions than to results about amplification for heuristics that make errors. The proof is also considerably simpler in the errorless case.

Lemma 5 (XOR Lemma, Errorless Version). *Let $f : \{0,1\}^n \to \{0,1\}$ be a function, and k, ε, δ be parameters. Suppose the function*

$$f'(x_1, \ldots, x_k) = f(x_1) \oplus \ldots \oplus f(x_k)$$

can be solved on a $\delta + (1-\varepsilon)^k$ fraction of inputs by a circuit of size S. Then f can be solved on a $1 - \varepsilon$ fraction of inputs by a circuit of size $3nk^2 S/\delta$.

Setting $k = \varepsilon^{-1} \log 1/\delta$ yields amplification from 2δ to $1 - \varepsilon$ with the circuit blowup polynomial in $n/\varepsilon\delta$. Setting $\varepsilon, \delta = n^{-c}$ for large enough c we obtain the following corollary.

Corollary 6. *Let $\mathbf{C}$ be a complexity class closed under XOR. For every $c > 0$, if $(\mathbf{C}, \mathcal{U}) \subseteq \mathrm{Avg}_{1-n^{-c}}\mathrm{P}/\mathrm{poly}$ then $(\mathbf{C}, \mathcal{U}) \subseteq \mathrm{AvgP}/\mathrm{poly}$.*

Let us describe the main idea of the proof. Let A' be an errorless oracle that agrees with f' on an ε-fraction of inputs. For intuition, think of the oracle A' as an adversary that tries to reveal as little information about f as possible. We will roughly argue that the best possible strategy for A' is to choose some set $T \subseteq \{0,1\}^n$ and answer $f'(x_1, \ldots, x_k)$ when $x_i \in T$ for all i, and answer $\perp$ otherwise. In that case T^k must have measure at least $(1-\varepsilon)^k$ in $\{0,1\}^{n \times k}$, so T itself has measure $1 - \varepsilon$ in $\{0,1\}^n$.

To explain what this is the case we need the following definitions. For $x \in \{0,1\}^n$ and $i \in [n]$, define the ith slice with respect to x as the set

$$\mathrm{slice}_i(x) = \left\{ (x_1, \ldots, x_k) \in \{0,1\}^{n \times k} : x_i = x. \right\}$$

Let S denote the set of queries x where $A'(x) = f'(x)$, and consider an input x. If for all i, the set S has measure close to zero in $\text{slice}_i(x)$ then the circuit A for f has no good chance of finding any information about x by making queries that include x. However, we will show that if on the other hand the measure of $\text{slice}_i(x) \cap S$ is bounded away from zero then A can compute $f(x)$ by choosing random i and random queries in $\text{slice}_i(x)$. Thus, to prevent A from obtaining the correct answer on x, A' must answer $\bot$ on almost all queries that involve x. It follows that the set S must then be close to a set of the form T^k, where T includes all queries on which A' answers $\bot$ with probability not too close to one.

Proof. We begin by describing A as a randomized algorithm that uses bounded oracle access to f, and then turn it into a circuit by fixing the randomness as advice. On input x,

- For all $i \in [k]$, repeat the following $3nk/\delta$ times
    - Choose random $x_1, \ldots, x_k \in \{0,1\}^n$, except for x_i. Set $x_i = x$.
    - If $A'(x_1, \ldots, x_n) \neq \bot$, return the value

$$A'(x_1, \ldots, x_k) \oplus f(x_1) \oplus \ldots \oplus f(x_{i-1}) \oplus f(x_{i+1}) \oplus \ldots \oplus f(x_k)$$

- If no answer was found, return $\bot$.

Let $T \subseteq \{0,1\}^n$ be the set of all x such that $\mu_{\text{slice}_i(x)}(S) > \delta/k$ for at least one $i \in [k]$. We need the following technical claim, which is proved below.

Claim 7. *If $\mu_{n \times k}(S) > \delta + (1 - \varepsilon)^k$, then $\mu_n(T) > 1 - \varepsilon$.*

The algorithm A clearly never outputs an incorrect answer, so it suffices to show that it outputs the $f(x)$ with good probability when $x \in T$. When $x \in T$, it follows from sampling that $A(x)$ outputs $f(x)$ with probability more than 2^{-n}.

To dispense of the oracle access to f, by a union bound there exists a setting of the randomness for which the algorithm gives the correct answer for all $x \in T$. Let us fix such a setting of the randomness, and provide the circuit with the values $x_1^{(j)}, \ldots, x_k^{(j)}$ generated in the $3nk^2/\delta$ different runs of the algorithm, as well as the values

$$f(x_1^{(j)}) \oplus \ldots \oplus f(x_{i-1}^{(j)}) \oplus f(x_{i+1}^{(j)}) \oplus \ldots \oplus f(x_k^{(j)})$$

used by the algorithm. □

Proof of Claim 7. We prove the contrapositive, so assume that $\mu_n(T) \leq 1 - \varepsilon$.

$$
\begin{aligned}
\mu_{n \times k}(S) &= \Pr_{(x_1, \ldots, x_k) \sim \{0,1\}^{n \times k}}[(x_1, \ldots, x_k) \in S] \\
&\leq \Pr[(x_1, \ldots, x_k) \in S \text{ and } \exists i : x_i \notin T] + \Pr[\forall i : x_i \in T] \\
&\leq \sum_{i=1}^{k} \Pr[(x_1, \ldots, x_k) \in S \text{ and } x_i \notin T] + \prod_{i=1}^{k} \Pr[x_i \in T] \\
&\leq \sum_{i=1}^{k} \Pr[(x_1, \ldots, x_k) \in S \mid x_i \notin T] + \prod_{i=1}^{k} \Pr[x_i \in T] \\
&\leq k \cdot \delta/k + (1 - \varepsilon)^k.
\end{aligned}
$$

□

3.2 Monotone amplifiers

For a monotone function $C : \{0,1\}^k \to \{0,1\}$, call $z \in \{0,1\}^k$ *i-pivotal* if $C(z) \neq C(z \oplus e_i)$. Observe that the property of being *i*-pivotal is independent of the value of the *i*th coordinate z_i of z.

We use $h_C : \{0,1\}^k \to \mathbb{N}$ to denote the boundary size of C at $z \in \{0,1\}^k$, that is

$$h_C(z) = \big|\{i : z \text{ is } i\text{-pivotal}\}\big|.$$

We will make a distinction between the cases $C(z) = 0$ and $C(z) = 1$, so we define

$$h_C^{\uparrow}(z) = \big|\{i : C(z) = 0 \text{ and } z \text{ is } i\text{-pivotal}\}\big|,$$
$$h_C^{\downarrow}(z) = \big|\{i : C(z) = 1 \text{ and } z \text{ is } i\text{-pivotal}\}\big|.$$

Observe that $h_C(z) = h_C^{\uparrow}(z) + h_C^{\downarrow}(z)$. For instance for the majority function MAJ_k on an odd number of inputs, we have $h_{\text{MAJ}_k}^{\uparrow}(z) = (k+1)/2$ only when z has hamming weight $(k-1)/2$ and zero otherwise, and $h_{\text{MAJ}_k}^{\downarrow}(z) = (k+1)/2$ only when z has hamming weight $(k+1)/2$ and zero otherwise.

We can now define the type of function C which will be used to amplify hardness.

Definition 8. *A monotone function $C : \{0,1\}^k \to \{0,1\}$ is an $(t,\rho)_p^{\uparrow}$-amplifier if*

$$\Pr_{z \sim_p \{0,1\}^n}[h_C^{\uparrow}(z) \geq t] \geq 1 - \rho.$$

We similarly define $(t,\rho)_p^{\downarrow}$-amplifier.

When $p = 1/2$ we sometimes omit the index. Note that when $p = 1/2$ and ρ is small, amplifiers must be highly unbalanced. Amplifiers satisfy the following two simple properties.

Fact 9. *If C if a $(t,\rho)_p^{\uparrow}$-amplifier, then $C^{\dagger}$ is a $(t,\rho)_{1-p}^{\downarrow}$-amplifier.*

Fact 10. *Every $(t,\rho)_{1/2}^{\uparrow}$-amplifier on $\{0,1\}^k$ is a $(t,\rho+\gamma)_{1/2+\gamma/2\sqrt{k}}^{\uparrow}$-amplifier, where $\gamma \in [-1/2, 1/2]$.*

Fact 10 follows directly from Fact 1.

3.3 Amplification lemma for monotone funtions

We can now state the monotone errorless amplification lemma.

Lemma 11 (Errorless Amplification Lemma). *Suppose $C : \{0,1\}^k \to \{0,1\}$ is an $(t,\rho)_p^{\uparrow}$ amplifier (respectively, $(t,\rho)_p^{\downarrow}$ amplifier). Fix $\delta > 0$. Let $f : \{0,1\}^n \to \{0,1\}$ be a function such that $\mu(f^{-1}(0)) = p$. Suppose that the function*

$$f'(x_1, \ldots, x_k) = C(f(x_1), \ldots, f(x_k))$$

can be solved on an $(\rho + 2\delta)$-fraction by a circuit of size S. Then f can be solved on a $(1 - 4t^{-1} \cdot \ln(2/\delta))$-fraction of zero inputs (respectively, one inputs) by circuits of size $O(Sk^3 \ln(t/2 \ln(2/\delta))/\delta^2)$.

The two respective versions of the lemma have completely analogous proofs, so we consider the case when C is an $(t, \rho)^{\uparrow}_p$ amplifier and f is $4t^{-1} \cdot \ln(2/\delta)$-hard on zero.

The proof of the amplification lemma parallels the above proof of the XOR lemma. The principal difference is that the proof of the XOR lemma implicitly uses the symmetry of the XOR function. However, we do not know much about the symmetries of C, and in our application the function C will indeed not have much symmetry. The approach will be to first subdivide the cube $\{0,1\}^{n \times k}$ into 2^k subcubes depending on the values $z_1 = f(x_1), \ldots, z_k = f(x_k)$, then imitate the proof of the XOR lemma on each of the subcubes determined by $(z_1, \ldots, z_k)$. Namely, if f' is not hard, we will show that for not too few $(z_1, \ldots, z_k)$, f can be computed on a large fraction of zero inputs by querying points in the subcube determined by $(z_1, \ldots, z_k)$. Finally we will combine the results for the various subcubes using standard probabilistic arguments.

Proof. Suppose we are given an errorless circuit A' of size S that computes f' on a $1 - \rho - 2\delta$ fraction of inputs. We will use A' to derive a circuit A that computes f on a $1 - 2t^{-1} \cdot \ln(2/\delta)$ fraction of zero inputs. We describe the algorithm A for f first as a randomized algorithm that uses bounded oracle access to f, and then we turn it into a circuit by fixing the randomness and oracle answers. On input x,

- For all $i \in [k]$, repeat the following $12k^2 \ln\big(t/2\ln(2/\delta)\big)/\delta^2$ times
  - Choose random $x_1, \ldots, x_k \in \{0,1\}^n$, except for x_i. Set $x_i = x$.
  - If $(f(x_1), \ldots, f(x_k))$ is i-pivotal[8] for C and $A'(x_1, \ldots, x_k) \neq \bot$, return the value $A'(x_1, \ldots, x_k)$.

- If no answer was found, return $\bot$.

The fact that A is an errorless heuristic for f (assuming that A' is an errorless oracle for f') is immediate, so we focus on bounding the fraction of zero inputs for f on which A fails.

Let us first introduce some notation. For $z \in \{0,1\}^k$, let K_z be the subcube $\{\mathbf{x} : f^k(\mathbf{x}) = z\}$. We also define

$$I_i = \{(x_1, \ldots, x_k) : f(x_i) = 0 \text{ and } (f(x_1), \ldots, f(x_k)) \text{ is } i\text{-pivotal for } C\}.$$

In the proof of the XOR lemma, we defined S as the set of queries $\mathbf{x}$ for which $A'(\mathbf{x}) = f'(\mathbf{x})$. Here we will need to have a stricter definition, as the algorithm can also fail owing to the fact that it makes a query $\mathbf{x}$ such that $f^n(\mathbf{x})$ has few pivotal coordinates in C. To take care of this we define

$$S = \{\mathbf{x} \in \{0,1\}^{n \times k} : A'(\mathbf{x}) = f'(\mathbf{x}) = 0 \text{ and } h_C^{\uparrow}(f^n(\mathbf{x})) \geq t\}.$$

Alternatively,

$$\overline{S} = \{\mathbf{x} \in \{0,1\}^{n \times k} : f'(\mathbf{x}) = 1 \text{ or } A'(\mathbf{x}) \neq f'(\mathbf{x}) \text{ or } h_C^{\uparrow}(f^n(\mathbf{x})) < t\}.$$

The set S is not too small since

$$
\begin{aligned}
\mu_{n \times k}(\overline{S}) &\leq \mu_{n \times k}\big(\{\mathbf{x} : f(\mathbf{x}) = 1 \text{ or } h_C^{\uparrow}(f^n(\mathbf{x})) < t\}\big) + \mu_{n \times k}\big(\{\mathbf{x} : A'(\mathbf{x}) \neq f'(\mathbf{x})\}\big) \\
&= \mu_{k,p}\big(\{z : C(z) = 1 \text{ or } h_C^{\uparrow}(z) < t\}\big) + \mu_{n \times k}\big(\{\mathbf{x} : A'(\mathbf{x}) \neq f'(\mathbf{x})\}\big) \\
&< \rho + \big(1 - \rho - 2\delta\big) \\
&= 1 - 2\delta
\end{aligned}
$$

---

[8]We assume for now that the oracle also provides a list of the pivotal coordinates of C at every input.

17

so that $\mu_{n \times k}(S) > 2\delta$.

We now partition the set S among the various subcubes determined by $z \in \{0,1\}^k$: $S_z = S \cap K_z$. Let
$$T_z = \left\{ x : \mu_{K_z \cap \mathrm{slice}_i(x)}(S_z \cap I_i) > \delta/2k \text{ for some } i \in [k] \right\}.$$

We now need a technical claim which is the analog of Claim 7:

Claim 12. *If $\mu_{K_z}(S_z) > \delta$, then*
$$\mu_{f^{-1}(0)}(T_z) > 1 - \frac{1}{h_C^\uparrow(z)} \cdot \ln \frac{2}{\delta}.$$

Since $\mu_{K_z}(S_z) > 0$ also implies that $h_C^\uparrow(z) \geq t$, it follows that under the condition of the claim $\mu_{f^{-1}(0)}(T_z) > 1 - t^{-1} \ln(2/\delta)$, so for a suitable choice of t and δ the set T_z will be large.

We will show by a probabilistic argument that the fact that S is not too small and Claim 12 imply the existence of a single large set $T \subseteq \{0,1\}^n$ that contains queries x on which the algorithm A has a high chance of succeeding. We define the set T as follows:
$$T = \left\{ x : \mu_{\mathrm{slice}_i(x)}(S \cap I_i) > \delta^2/4k^2 \text{ for some } i \in [k] \right\}$$

and use the following claim, whose proof is shown below:

Claim 13. *Assuming $\mu_{n \times k}(S) > 2\delta$ and Claim 12, it follows that*
$$\mu_{f^{-1}(0)}(T) > 1 - \frac{2}{t} \cdot \ln \frac{2}{\delta}.$$

To finish the proof, we proceed as in the XOR lemma, however with a more careful analysis.[9] By our choice for the number of rounds of A, if $x \in T$, then with probability $1 - 2t^{-1} \ln(2/\delta)$ at least one round of the algorithm will choose values i and $x_1, \ldots, x_n$ such that $(x_1, \ldots, x_n) \in S \cap I_i$ (so that $A'(x_1, \ldots, x_k) = f'(x_1, \ldots, x_k) = 0$ and the algorithm A returns this value.) Therefore, there exists a single setting of the randomness of the algorithm for which for all but a $2t^{-1} \ln(2/\delta)$ fraction of strings in T the algorithm succeeds. We fix such a setting of the randomness and provide the circuit with the values $x_1^{(j)}, \ldots, x_k^{(j)}$ used in the different runs of the algorithm, as well as a list of all the pivotal coordinates of $\left(f(x_1^{(j)}), \ldots, f(x_k^{(j)}) \right)$. This algorithm will succeed on all the instances, except for those outside t and those inside t for which a bad fixing of the randomness was chosen. There are at most $2t^{-1} \ln(2/\delta)$ instances of each type. $\qquad\square$

Proof of Claim 12. We prove the contrapositive, so assume that $\mu_{f^{-1}(0)}(T_z) \leq 1 - h_C^\uparrow(z)^{-1} \cdot \ln(2/\delta)$.

---

[9]The same analysis as in the XOR lemma is sufficient for low-end amplification, however one must be slightly more careful with the size of the circuit for the argument to work in the high end.

Let $I^z = \{i : z \in I_i\}$.

$$
\begin{aligned}
\mu_{K_z}(S_z) &= \Pr_{(x_1,\dots,x_k) \sim K_z}[(x_1,\dots,x_k) \in S_z] \\
&\leq \Pr[(x_1,\dots,x_k) \in S_z \text{ and } \exists i \in I^z : x_i \notin T_z] + \Pr[\forall i \in I^z : x_i \in T_z] \\
&\leq \sum_{i \in I^z} \Pr[(x_1,\dots,x_k) \in S_z \text{ and } x_i \notin T_z] + \prod_{i \in I^z} \Pr[x_i \in T_z] \\
&\leq \sum_{i \in I^z} \Pr[(x_1,\dots,x_k) \in S_z \mid x_i \notin T_z] + \prod_{i \in I^z} \Pr[x_i \in T_z] \\
&\leq |I^z| \cdot \delta/2k + \left(1 - h_C(z)^{-1} \cdot \ln(2/\delta)\right)^{|I^z|} \\
&\leq \delta/2 + \delta/2 = \delta.
\end{aligned}
$$

For the last equation we use that $|I^z| = h_C^{\uparrow}(z) \leq k$ and the inequality $1 - \gamma \leq e^{-\gamma}$ for $\gamma \geq 0$. $\qquad\square$

Proof of Claim 13. Assume that $\mu_{n \times k}(S) > 2\delta$ and let Z denote the set of all $z \in \{0,1\}^k$ such that $\mu_{K_z}(S_z) > \delta$. By Claim 12, we have that for all $z \in Z$, $\mu_{f^{-1}(0)}(T_z) > 1 - t^{-1} \cdot \ln(2/\delta)$. It follows that

$$
\Pr_{x \sim f^{-1}(0), z \sim_p Z}[x \notin T_z] < t^{-1} \cdot \ln(2/\delta).
$$

Here, $z \sim_p Z$ denotes a z sampled from the p-biased distribution on $\{0,1\}^k$ conditioned on the set Z. By Markov's inequality, we have that

$$
\Pr_{x \sim f^{-1}(0)}\left[\Pr_{z \sim_p Z}[x \notin T_z] < 1/2\right] < 2t^{-1} \cdot \ln(2/\delta).
$$

On the other hand, it follows by conditional probabilities that

$$
\mu_{n \times k}(S) \leq 1 \cdot \mu_{k,p}(Z) + \delta \cdot \mu_{k,p}(\overline{Z})
$$

so that $\mu_{k,p}(Z) > \delta$, and

$$
\Pr_{x \sim f^{-1}(0)}\left[\Pr_{z \sim_p \{0,1\}^k}[x \in T_z] < \delta/2\right] < 2t^{-1} \cdot \ln(2/\delta).
$$

We will now show that any x that satisfies $\Pr_{z \sim_p \{0,1\}^k}[x \in T_z] \geq \delta/2$ must be inside T. This implies

$$
\Pr_{x \sim f^{-1}(0)}[x \notin T] \leq \Pr_{x \sim f^{-1}(0)}\left[\Pr_{z \sim_p \{0,1\}^k}[x \in T_z] < \delta/2\right]
$$

giving the desired conclusion. Fix an x such that $\Pr_{z \sim_p \{0,1\}^k}[x \in T_z] \geq \delta/2$. Unwinding the definition of T_z, we have

$$
\Pr_{z \sim_p \{0,1\}^k}\left[\mu_{K_z \cap \mathrm{slice}_i(x)}(S_z \cap I_i) > \delta/2k \text{ for some } i \in [k]\right] \geq \delta/2.
$$

It follows that there must exist an $i \in [k]$ for which

$$
\Pr_{z \sim_p \{0,1\}^k}\left[\mu_{K_z \cap \mathrm{slice}_i(x)}(S_z \cap I_i) > \delta/2k\right] \geq \delta/2k.
$$

For this i, we have that

$$
\mu_{\mathrm{slice}_i(x)}(S \cap I_i) = \mathrm{E}_{z \sim_p \{0,1\}^k}\left[\mu_{K_z \cap \mathrm{slice}_i(x)}(S_z \cap I_i)\right] > \frac{\delta}{2k} \cdot \frac{\delta}{2k} = \frac{\delta^2}{4k^2},
$$

so that $x \in T$. $\qquad\square$

4 Constructions of amplifiers

4.1 The DDNF function

Fix arbitrary integers t, d and $q \geq t$ which is a power of a prime. We identify the integers $1, \ldots, t$ with arbitrary distinct field elements in $\mathbb{F}_q$. Set s so that $q^{d+1} = 2^t/s$. In this section we investigate the monotone boolean function $\text{DDNF} : \{0,1\}^{t \times q} \to \{0,1\}$ given by

$$\text{DDNF}(z_{1,1}, \ldots, z_{t,q}) = \bigvee_p \bigwedge_{i=1}^t z_{i,p(i)}$$

where p ranges over all univariate polynomials of degree d over $\mathbb{F}_q$. (There are q^{d+1} such polynomials.)

Theorem 14. *The function* DDNF *is a* $(t/2s, 1/s + 4s/t + 4t/q + 4t/s^2q + 4t^2/q^2)^{\uparrow}$ *amplifier. More precisely,*

- $\Pr_z[\text{DDNF}(z) = 1] \leq 1/s$ *and*

- $\Pr_z[h^{\uparrow}_{\text{DDNF}}(z) < t/2s] \leq 1/s + 4s/t + 4t/q + 4t/s^2q + 4t^2/q^2.$

For a sufficiently large constant K, the setting $s = K$ and $q = Kt$ gives a function on qt variables where almost all the vertices have $\Omega(\sqrt{qt})$ neighbors. Observe that this is the best possible, as it is known that the total influence of any monotone boolean function on qt variables is $O(\sqrt{qt})$. This bound is also achieved by the majority function. However unlike majority, where the influence is concentrated over a very small set of inputs, the function DDNF distributes this total influence almost uniformly among inputs.

The proof of Theorem 14 is given in Appendix A.

Optimizing the amplification We can set the parameters $s = n^{1/5}/2$, $t = n^{2/5}$, $q = n^{3/5}$ in Theorem 14 to obtain DDNF as a $(n^{1/5}, O(n^{-1/5}))^{\uparrow}$-amplifier. However we can do better in terms of the second parameter by taking the OR of several independent copies of DDNF.

When we take the OR of k independent copies of a function, the probability that the function evaluates to 1 grows at most linearly with k, while the probability that a vertex hits the boundary increases exponentially with k. So if we start with a function with moderate boundary size but extremely high vanishing probability, the OR of several such functions has the effect of increasing the boundary while essentially preserving the vanishing probability.

The DDNF function is an ideal candidate for this transformation, as its vanishing probability can be made very large — as large as $1 - \Omega(1/\sqrt{n})$. (In contrast, for the tribes function to have pivotal coordinates, its vanishing probability must stay above $1 - O(1/\log n)$.)

Let's fix an arbitrary $0 \leq \alpha < 1/2$, set $\beta = 1/2 - \alpha$ and look at the following setting of parameters:

$$s = n^{\beta}\sqrt{\beta \log n/64} \quad t = \sqrt{n/16\beta \log n} \quad q = 4\sqrt{n/\beta \log n}.$$

Then DDNF is a function on $n/\beta \log n$ variables such that

$$\Pr_z[\text{DDNF}(z) = 1] \leq 8 \cdot n^{-\beta}/\sqrt{\beta \log n} \quad \text{and} \quad \Pr_z[h^{\uparrow}_{\text{DDNF}}(z) < n^{\alpha}] < 1/2.$$

Let OR-DDNF be the function obtained by taking the OR of $k = \beta \log n$ copies of DDNF over independent inputs $z_1, \ldots, z_k \in \{0,1\}^{n/\beta \log n}$. The number of inputs to OR-DDNF is n.

Theorem 15. *For every $0 \le \alpha < 1/2$, $\beta = 1/2 - \alpha$ and sufficiently large n, the OR-DDNF$_\alpha$ function on n variables is a $(n^\alpha, 9\sqrt{\beta \log n} \cdot n^{-\beta})$-amplifier.*

Proof. By a union bound,

$$\Pr_{z_1,\ldots,z_k}[\text{OR-DDNF}(z_1,\ldots,z_k) = 1] \le (\log n)/s \le 8\sqrt{\beta \log n} \cdot n^{-\beta}$$

while

$$\Pr_{z_1,\ldots,z_k}[h^\uparrow_{\text{OR-DDNF}}(z_1,\ldots,z_k) < n^\alpha] \le \Pr_{z_1,\ldots,z_k}[\forall i : h^\uparrow_{\text{DDNF}}(z_i) < n^\alpha]$$
$$= \prod_{i=1}^{k} \Pr_{z_i}[\text{DDNF}(z_i) < n^\alpha]$$
$$= (1/2)^k < n^{-\beta}. \qquad \square$$

On the complexity of DDNF   It is not difficult to see that the DDNF function is in NP, but it is much less clear if its monotone complement DDNF† is in NP or not. From the viewpoint of computational efficiency, we can view DDNF† as the following problem: Given integers t, q, d and a coloring of the rectangle $[t] \times [q]$, determine if there exists a polynomial $p : \mathbb{F}_q \to \mathbb{F}_q$ of degree at most d such that $(i, p(i))$ is colored black for all i.

If we put in the additional restriction that at most $O(t^2/qd)$ of the points are colored black, the problem becomes tractable via the Guruswami-Sudan list-decoding algotihm for Reed-Solomon codes [GS99]. Removing this restriction makes the problem more difficult, and in our setting there are colorings of the rectangle which correspond to lists of exponential size. However, unlike in the list-decoding setting, where the goal is to compute a list of all polynomials consistent (or partially consistent) with the given coloring, we merely ask for a certificate that the list is empty when this is the case. We are not aware of any hardness results for this problem.

Therefore we do not know if DDNF† can be used for hardness amplification beyond a certain range of parameters (logarithmic size inputs).

4.2   The holographic function

We consider the function HOL $=$ HOL$_{h,w}$ on $n = 2hw$ variables defined in equation (1). Let $\rho < 1/2$ and $s > 1$ be parameters satisfying $(2\rho)^w = 1/s$.

Theorem 16. *For every s such that $9w^2/h \le \log s \le w$, the function HOL satisfies $\Pr_z[\text{HOL}(z) = 1] \le 1/s$ and $\Pr_z[h^\uparrow_{\text{HOL}}(z) < w/4s] \le 0.8 + 2s/w$, where z is chosen from the distribution $\mu_{n,\rho}$.*

For sufficiently large m, we choose $w = m^{1/3}$, $h = m^{2/3}$, and $s = m^{1/3-\alpha}/4 \log m$. For sufficiently large m, we have $\Pr[\text{HOL}(z) = 1] \le m^{-1/3+\alpha} \log m$ and $\Pr_z[h^\uparrow_{\text{HOL}}(z) < m^\alpha \log m] < 0.9$. We also have $\rho = 1/2 - (1/3 - \alpha + o(1)) \log w/w$.

Now consider the function HOL-AMP $=$ OR $\circ$ HOL $\circ$ THR, where the OR function is on $\log n$ bits, HOL is the holographic function on $m = n/\log^2 n$ bits, and THR is a threshold function on $\log n$ bits with $\Pr[\text{THR}(z) = 1] = \rho \pm 1/n$ when $z \sim \mu_{\log n}$. The number of inputs to HOL-AMP is n.

As in the construction of OR-DDNF, the effect of the OR function here is to increase the fraction of inputs that have many pivotal coordinates. The effect of the threshold is to make the amplification work over balanced inputs.[10]

Theorem 17. *For every $\alpha > 0$ and sufficiently large n, HOL-AMP$_\alpha$ is a $(n^\alpha, n^{-1/3+\alpha} \cdot \text{polylog}(n))$ amplifier.*

Proof. The fact $\Pr[\text{HOL-AMP}(z) = 1] \leq n^{-1/3+\alpha} \cdot \text{polylog}(n)$ follows from a union bound. For the fraction of inputs with many pivotal coordinates, we first look at the function HOL $\circ$ THR. The HOL function has n^α pivotal coordinates over a $1/10$ fraction of its inputs chosen from distribution $\mu_{n,\rho}$. Let's fix an input z for HOL that has n^α pivotal coordinates and look at a corresponding random input w of HOL $\circ$ THR. Each pivotal bit of z gives rise to $O(\log n)$ pivotal bits of w with probability $O(1/\sqrt{\log n})$, so by Chernoff bounds w has $\Omega(n^\alpha \sqrt{\log n}) \leq n^\alpha$ pivotal bits with probability $1/2$ conditioned on z. Therefore HOL $\circ$ THR has n^α pivotal coordinates with probability at least $1/20$.

Taking the OR of $\log n$ such functions yields a function that has n^α pivotal coordinates with probability $1 - n^{-1/3}$ by the argument from the proof of Theorem 15. $\qquad\square$

5   Non-uniform amplification

In this section we prove the following theorem.

Theorem 18. *For every $\alpha > 0$, if $(\text{NP}, \mathcal{U}) \subseteq \text{Avg}_{1-n^{-2/9+\alpha}}\text{P/poly}$, then $(\text{NP}.\mathcal{U}) \subseteq \text{AvgP/poly}$.*

We use the function HOL-AMP from Theorem 17. First, using Lemma 11 and Fact 10, we obtain the following properties of HOL-AMP on inputs of size $n^{2-\alpha}$ as an amplifier.

Corollary 19. *For every $\alpha > 0$ there exists $\gamma > 0$ such that the following holds. Suppose that $f : \{0,1\}^n \to \{0,1\}$ is $n^{-1+\alpha}$-balanced.*

1. *If HOL-AMP $\circ f$ can be solved on $O(n^{-2/3+\alpha})$-fraction of inputs by circuits of size S, then f can be solved on a $1 - n^{-\gamma}$ fraction of zero inputs by circuits of size $S \cdot \text{poly}(n)$.*

2. *If HOL-AMP$^\dagger \circ f$ can be solved on $O(n^{-2/3+\alpha})$-fraction of inputs by circuits of size S, then f can be solved on a $1 - n^{-\gamma}$ fraction of one inputs by circuits of size $S \cdot \text{poly}(n)$.*

Apart from Corollary 19, the only issue that needs to be resolved is the assumption that the family of functions in question is balanced. We use a trick of O'Donnell and Trevisan: We simulate unbalanced functions with balanced ones, to which the amplification lemmas apply.

Specifically, we need the following technical lemma which was proved by Trevisan (see [Tre05, Section 6]).

Lemma 20 (Trevisan). *Let t be an arbitrary integer, and $\gamma > 0$ an arbitrary fraction. Suppose that every language L in NP can be solved on a $1 - n^{-\gamma}$-fraction of inputs of length n by circuits of size $S(n)$ for all n such that L is $O(n^{-1+1/t})$-balanced on inputs of length n. Then every language in NP can be solved on $1 - n^{-\gamma t}$-fraction of inputs of length n by circuits of size $(n+1)^{t-1} \cdot S((n+1)^t)$ for every n.*

---

[10]In our application we can tolerate somewhat unbiased inputs, but for simplicity we balance the holographic function instead.

Proof of Theorem 18. Suppose that $(\text{NP},\mathcal{U}) \subseteq \text{Avg}_{1-n^{-\gamma}}\text{P}/\text{poly}$, namely for every language in NP there is an errorless family of circuits that solves it on $m^{-1/3+\alpha}$ fraction of inputs for all lengths m. Now let $L = \{f_n\}$ be an NP-language and let n be an input length such that f_n is $n^{-1+\alpha}$-balanced. We apply HOL-AMP on $n^{2-\alpha}$ variables to f_n to obtain $g_{m(n)} = \text{HOL-AMP} \circ f_n$. Then $g_{m(n)}$ is a function on inputs of size $m(n) = n^{3-2\alpha}$. We define the language L_1 as the collection of all functions $\{g_{m(n)}\}$. Similarly we define $g^{\dagger}_{m(n)} = \text{HOL-AMP}^{\dagger} \circ f_n$ and $L_1^{\dagger}$ as the collection of all functions $\{g^{\dagger}_{m(n)}\}$.

Since both HOL-AMP and HOL-AMP† are efficiently computable (P is closed under monotone complement), the languages L and L' are both in NP. By assumption, $g_{m(n)}$ can be solved on a $O(m(n)^{-2/9+\alpha}) = O(n^{-2/3+\alpha})$ fraction of inputs by a circuit of size $\text{poly}(n)$, and the same holds for HOL-AMP $\circ\, g^{\dagger}_{m(n)}$ and one inputs. By Corollary 19, there exist polynomial-size circuits C and $C^{\dagger}$ that solve f_n on a $1 - n^{-\gamma}$ fraction of zero and one inputs, respectively, for every n such that f_n is $n^{-1+\alpha}$-balanced.

We obtain a circuit C' that solves f_n on a $1 - n^{-\gamma}$ fraction of all inputs by running both C and $C^{\dagger}$, outputting $\perp$ if both do so and outputing an answer otherwise. Since C and $C^{\dagger}$ are both errorless the answers they output will always be consistent.

To summarize, we have shown that every language in NP can be solved on $1 - n^{-\gamma}$ fraction of inputs on all input lengths n on which it is $n^{-1+\alpha}$-balanced.

By choosing $t = 1/\alpha$ in Lemma 20, we obtain that every language in NP can be solved on a $1 - n^{-\gamma/\alpha}$ fraction of inputs of every length n, namely

$$(\text{NP},\mathcal{U}) \subseteq \text{Avg}_{n^{-\gamma/\alpha}}\text{P}/\text{poly}.$$

Applying equation (2) concludes the proof. $\qquad\square$

6   Uniform amplification

In this section we show how to obtain uniform amplification at some cost in the error parameter. We follow the steps taken by Trevisan [Tre05] in the heuristic setting, though several modifications of Trevisan's approach will be necessary.

In short, Trevisan first designs a reduction that works when given an amount of randomness-dependent advice whose length is logarithmic in the size of the input, then shows how to remove the advice using a non-black-box reduction due to Ben-David et al. [BCGL92].

We outline how to adapt Trevisan's approach in the setting of errorless amplification. We assume that the reader is familiar with Trevisan's proof.

Theorem 21. *For every $\alpha > 0$, if $(\text{NP},\mathcal{U}) \subseteq \text{Avg}_{1-(\log n)^{-1/10+\alpha}}\text{BPP}$, then $(\text{NP},\mathcal{U}) \subseteq \text{AvgBPP}$.*

6.1   Low-end amplification

Using a slightly different analysis, one can derive the following advice-efficient version of Lemma 11, which will be useful for amplification in the low end.[11]

---

[11]Joe Kilian has pointed out to us that there is an alternative, "advice-free" approach to low-end amplification in the uniform setting when the hardness is an arbitrary constant independent of the instance size n.

Lemma 22. *There is a randomized reduction algorithm that on input n, k, δ, ρ runs in time $\mathrm{poly}(n, k, 1/\delta, 1/\rho)$ and outputs an oracle circuit A of size $O(1)$ that makes $\mathrm{poly}(k, 1/\delta)$ calls to the oracle with the following property.*

Suppose $C : \{0,1\}^k \to \{0,1\}$ is an $(t, \rho)_p^{\uparrow}$ amplifier (respectively, $(t, \rho)_p^{\downarrow}$ amplifier). Let $f : \{0,1\}^n \to \{0,1\}$ be a function such that $\mu(f^{-1}(0)) = p$. For every oracle A' that solves $f' = C \circ f^k$ on a $(\rho + 2\delta)$-fraction of inputs,

$$\Pr\left[A^{A'} \text{ solves } f \text{ on a } (1 - 4t^{-1}\ln(2/\delta))\text{-fraction of zero inputs}\right] > \left(t^2/(1 - \rho)\ln(2/\delta)\right)^{-O(k^3/\delta^2)},$$

where the probability is over the randomness of the reduction algorithm. The same holds for $C^{\dagger} \circ f^k$ and one inputs.

However Lemma 22 is inadequate for high-end amplification. In the high end the input size of the amplifying function will have to grow polynomially with the size of the instance, so the success probability of the reduction becomes exponentially small.

Proof sketch for Lemma 22. We use a slightly different version of the reduction in Lemma 11. The analysis is essentially the same so we just mention the points that are relevant for our application. Set $r = 12k^3 \ln\left(t/2\ln(2/\delta)\right)/\delta^2$. Given a circuit A' for f', the reduction algorithm does the following:

1. (Choice of randomness) For $1 \le j \le r$, choose random values $i^{(j)} \in [k]$ and $x_1^{(j)}, \ldots, x_k^{(j)} \in \{0,1\}^n$ except for $x_{i^{(j)}}^{(j)}$.

2. (Definition of circuit) Output the following circuit C: On input x,

   - For all $1 \le j \le r$, set $x_{i^{(j)}} = x$. Calculate $A'(x_1, \ldots, x_k)$ and return it if it does not equal $\bot$.
   - If no answer was found, return $\bot$.

We call a choice of the randomness by the algorithm good if for all j, $(f(x_1^{(j)}), \ldots, f(x_k^{(j)}))$ is $i^{(j)}$-pivotal for the function C. Observe that the randomness is good with probability at least $((1 - \rho)/t)^r$.

Conditioned on the randomness being good, the proof of Lemma 11 (with small modifications) shows that assuming A' solves f' on $\rho + 2\delta$ inputs, A solves f on a $(1 - 4t^{-1}\ln(2/\delta))$-fraction of zero inputs. $\qquad\square$

We now specialize C to the OR-DDNF function over inputs of size $(\log n)^{1/5}$. Both OR-DDNF and OR-DDNF† are now computable in time polynomial in n. Working out the parameters we obtain the following.

Corollary 23. *Fix arbitrary constants $\alpha, \varepsilon > 0$. There exists a randomized reduction algorithm R that, on input n, runs in time $\mathrm{poly}(n)$ and outputs an oracle circuit A with the following property. For every $f : \{0,1\}^n \to \{0,1\}$ that is $O((\log n)^{-1/10+\alpha})$-balanced and every oracle C' that solves OR-DDNF $\circ f$ on $(\log n)^{-1/10+\alpha}$ fraction of inputs,*

$$\Pr\left[C^{C'} \text{ solves } f \text{ on a } 1 - \varepsilon\text{-fraction of zero inputs}\right] = \Omega(1/n),$$

where the probability is over the randomness of the algorithm. The same holds for OR-DDNF$^{\dagger} \circ f$ and one inputs.

6.2 High-end amplification

We now sketch how to prove a uniform version of Lemma 11 that works for high-end amplification using recursive majorities. The proof is essentially identical to Trevisan's; however there is one important conceptual difference.

In both cases the reduction R produces an oracle circuit A that, when given access to an oracle that computes f on (say) 99% fraction of inputs, solves f on (say) $1 - 1/n^3$ fraction of inputs of length n.

In the heuristic setting, Trevisan gives a *randomized* reduction R that produces a *deterministic* oracle circuit A. In the errorless setting, however, we do not know how to produce a deterministic A. Instead, we give a deterministic reduction R that produces a randomized oracle circuit A.

Below we sketch Trevisan's proof as well as the necessary modifications for the errorless setting. First let us define the type of circuit we have in mind.

Definition 24. *We say that C is a* randomized errorless circuit *for $f : \{0,1\}^n \to \{0,1\}$ with error δ if C takes two inputs $x \in \{0,1\}^n$ and $r \in \{0,1\}^m$ and:*

- *For all $x \in \{0,1\}^n$, $\Pr_r[C(x;r) \in \{f(x), \bot\}] \geq 3/4$, and*

- $\Pr_x\left[\Pr_r[C(x;r) = \bot] \geq 1/4\right] \leq \delta$.

For simplicity we will omit at times the randomness of the circuit r from the description. We can naturally augment randomized circuits with oracle gates.

The following lemma is the errorless analog of [Tre05, Lemma 5].

Lemma 25. *There is a deterministic reduction algorithm that on input n, k, δ where k is odd and $100 < k \leq \delta^{-2/7}$ runs in time $\mathrm{poly}(n, k, 1/\delta)$ and outputs a randomized oracle circuit C of size $O(k \log k)$ with the following property.*

Suppose f is $1/12\sqrt{k}$-balanced and C' is a randomized errorless circuit for $\mathrm{MAJ}_k \circ f^k$ with error $\delta\sqrt{k}/12$. Then $C^{C'}$ is a randomized errorless circuit for f with error δ.

Proof sketch. We follow Trevisan's proof. We will sketch the main steps and outline the differences while skipping the calculations. Let $g = \mathrm{MAJ}_k \circ f^k$.

We relate the inputs to f and g via a bipartite graph as follows. The vertices on the left are elements of $\{0,1\}^n$, the vertices on the right are elements of $\{0,1\}^{n \times k}$, and we put an edge between x on the left and $(x_1, \ldots, x_k)$ on the right if $x = x_j$ for some j. The vertices on the left are labeled by $f(x)$. The vertices on the right are labeled by $\bot$ if $\Pr[C'(x_1, \ldots, x_k) = \bot] \geq 1/4$, and by $g(x_1, \ldots, x_k)$ otherwise.

We say that a vertex on the left is bad if more than a $1/12\sqrt{k}$ fraction of its neighbors on the right are labeled by $\bot$. Let A be the vertices on the right that are labeled by $\bot$, so that $\alpha = |A|/2^{nk} \leq \delta\sqrt{k}/12$. Let B be the bad vertices on the left and $\beta = |B|/2^n$.

Trevisan's calculation now shows that $\beta \leq \delta$.

To describe C, we will need to use a robust version C'_1 of C' that amplifies the error probability to $1/12\sqrt{k}$. This is achieved by making $O(\log k)$ independent queries to C' and taking the plurality answer. By Chernoff bounds, C'_1 has the property that

- For all $\mathbf{x} \in \{0,1\}^{n \times k}$, $\Pr[C_1'(\mathbf{x}) \in \{f(\mathbf{x}), \bot\}] \geq 3/4$, and

- If $\Pr[C'(\mathbf{x}) = \bot] < 1/4$ (i.e., $\mathbf{x} \notin A$), then $\Pr_{C_1'}[C_1'(\mathbf{x}) = \bot] < 1/12\sqrt{k}$.

We now describe the randomized circuit C. On input x, choose $t = O(k)$ inputs $\mathbf{x} = (x_1, \ldots, x_k)$ for C_1' where x is embedded in a random location x_i and all other x_j are random. Look at the outputs produced by C_1'. If more than a $1/4\sqrt{k}$ fraction of these are $\bot$, output $\bot$. Otherwise output the plurality of answers.

We show that C is errorless. Let's fix the input x. Since C outputs an incorrect answer only with probability $1/4$, by high deviation bounds with probability $3/4$ (over the randomness of C) both these events hold:

- The fraction of queries $\mathbf{x}$ made by C for which $g(\mathbf{x}) = f(x)$ is at least $1/2 + 1/3\sqrt{k}$.

- At most $1/12\sqrt{k}$ of the queries made by C are answered incorrectly by C_1' (i.e., C_1' outputs $\overline{g(\mathbf{x})}$).

Suppose both these conditions hold and consider two cases. If C sees more than $1/3\sqrt{k}$ $\bot$s, then it outputs $\bot$. Otherwise, the fraction of answers seen by C that disagree with $f(x)$ is at most $1/2 - 1/3\sqrt{k}$ ($C_1'(\mathbf{x})$ agrees with $g(\mathbf{x})$ but not $f(x)$) plus $1/12\sqrt{k}$ ($C_1'(\mathbf{x})$ outputs $\overline{g(\mathbf{x})}$) plus $1/4\sqrt{k}$ ($C_1'(\mathbf{x})$ outputs $\bot$), which is less than $1/2$.

Now we show that C is correct a $1 - \delta$ fraction of the time, in particular whenever $x \notin B$. Recall that if $x \in B$ then it has fewer than $1/12\sqrt{k}$ neighbors $\mathbf{x} \in A$, and if $\mathbf{x} \notin A$ then $C_1'(\mathbf{x})$ outputs $\bot$ with probability at most $1/12\sqrt{k}$. So if $x \notin B$, then for a random query $\mathbf{x}$,

$$\Pr[C'(\mathbf{x}) = \bot] \leq 1 \cdot \Pr[\mathbf{x} \in A] + 1/12\sqrt{k} \cdot \Pr[\mathbf{x} \notin A] \leq 1/6\sqrt{k}.$$

Then by high deviation bounds, with probability $3/4$ at most $1/4\sqrt{k}$ of the queries made by C will be answered by $\bot$, and C will not output $\bot$. $\square$

Why was it necessary to make the circuit C randomized? To determine whether an answer provided by the oracle C' is correct or not, C must have a good estimate of the fraction of queries to which C' does not know the answer. The only way to obtain such an estimate is by sampling, which is inherently probabilistic. In Trevisan's proof, the sampling is done by the reduction and not the circuit; but then there will be some small fraction of inputs x on which the estimate given by the sampler will be incorrect. In the heuristic setting, C can afford to give an incorrect answer for those inputs. In the errorless setting, however, C must always be correct, and it is crucial to embed the randomness into C itself.

Composing the lemma recursively with increasing values of k as in Trevisan, we obtain the following analog of [Tre05, Lemma 6].

Lemma 26. *Fix an absolute constant ε and let L be a language in* NP. *There is a language $L' \in$ NP, a polynomially bounded efficiently computable function $l(n)$, and a deterministic reduction that on input n, runs in time polynomial in n and outputs a randomized errorless oracle circuit C with the following properties:*

- *If L is $n^{-0.5}$-balanced on input length n, then L' is $n^{-\gamma}$-balanced on input length $l(n)$ for some $\gamma > 0$.*

- *If C' solves L' on input length $l(n)$ with error ε, then $C^{C'}$ solves L on input length n with error $2/n^{1/5}$, assuming L is $n^{-0.5}$-balanced on input length n.*

6.3 Balancing

To handle the requirement that high-end amplification only works for almost balanced functions, Trevisan gives a transformation that converts an arbitrary language L into a language L' such that every input length n of L corresponds to roughly n^t input lengths of L', one of which is almost balanced (almost is $N_n^{-t/(t+1)}$, where N_n is the input length in L' corresponding to n). Thus if we had an errorless heuristic for L' with sufficiently high success probability, we could obtain one for L at the cost of an additional $t \log n$ bits of advice.

In our application we will not be able to tolerate $t \log n$ extra bits of advice. Instead we give an errorless analog of Trevisan's reduction that uses *no advice at all*. Interestingly, in proving the reduction correct we crucially use Trevisan's theorem that if $(\mathrm{NP}, \mathcal{U})$ has heuristic algorithms that work with probability slightly above $1/2$ (taken both over choice of inputs and randomness of the algorithm), then $(\mathrm{NP}, \mathcal{U})$ has heuristic schemes.

Lemma 27. *Let $L \in \mathrm{NP}$ and $t > 0$ be an arbitrary constant. Suppose that L has a heuristic scheme S (not necessarily errorless). Then there is a language $L' \in \mathrm{NP}$ and a randomized reduction R that on input n, runs in time polynomial in n and with probability $7/8$ outputs an oracle circuit C and a number N with the following properties:*

- *N is between n^{t+1} and $n^{t+1} + n^t$*

- *The language L' is $4N^{-t/(t+1)}$-balanced on input length N, and*

- *If C' is a (randomized) errorless heuristic for L' on input length N with error $\delta(N)$, then $C^{C'}$ is a (randomized) errorless heuristic for L on input length n with error $2\delta(N)$.*

Proof sketch. The definition of L' is the same as in Lemma 7 of [Tre05]. We won't repeat all the details here. Each input length n of L is encoded into n^t consecutive input lengths of L', starting at n^{t+1} and going up to $n^{t+1} + n^t$. Each of these input lengths can be thought of as a guess for the balance of L (up to $1/n^t$), and the correct guess N rectifies the imbalance of L on length n by padding appropriately.

Let p_n be the fraction of "yes" instances of length n in L. In Trevisan's construction, L' will be balanced on input length $N = n^{t+1} + \lfloor n^t \cdot p_n \rfloor$, and this is the value that is used as advice. In fact, the value $N + 1$ can also be used as advice (this affects the balance by at most $1/2n^t$). Here we show that R can itself compute either N or $N + 1$ high probability.

By definition of N, we have that $N \leq n^{t+1} + n^t p_n < N + 1$. We choose $k = O(n^{2t})$ random samples $x_1, \ldots, x_k \sim \{0,1\}^n$ and run the heuristic scheme $S(x_1; \delta), \ldots, S(x_k; \delta)$ where $\delta = 1/8n^t$. We let $\hat{p}_n$ be the fraction of "yes" answers of S and output the closest integer to $n^{t+1} + n^t \hat{p}_n$.

We will now argue that with probability $7/8$, $|\hat{p}_n - p_n| < 1/2n^t$. It follows that the closest integer to $n^{t+1} + n^t \hat{p}_n$ is either N or $N + 1$.

This follows from standard deviation bounds, e.g. Chebyshev's inequality. With probability 7/8,

- The fraction of samples for which $S(x_k; \delta) \neq L(x_k)$ is at most $1/4n^t$, and

- The fraction of samples for which $L(x_k) = 1$ is within $1/4n^t$ of p_n.

It follows that $\hat{p}_n$ is then within $1/2n^t$ of p_n. $\qquad\qquad\qquad\qquad\qquad\qquad\qquad$ $\square$

6.4   Errorless search-to-decision reduction

We will also need to use the result of Ben-David et al. [BCGL92] which essentially says that search and decision are equivalent in the high end for average-case algorithms. Trevisan uses a version of this lemma for heuristic algorithms. The lemma also works for errorless heuristics.

To state the lemma we need to define the notion of "erroless heuristic search".

Definition 28 (Errorless heuristic search). *Let L be an NP-language and R a corresponding NP-relation. We say that C is a randomized errorless search circuit for L on input length n with error δ if*

- *For all $x \in L$ of length n, $\Pr_r[R(x, C(x, r)) = 1 \text{ or } C(x, r) = \perp] \geq 3/4$, and*

- *For all x of length n, $\Pr_x\big[\Pr_r[C(x, r) = \perp] \geq 1/4\big] \leq \delta$.*

Lemma 29 (Search to decision). *Let L be an NP-language and R a corresponding NP-relation. Suppose the length of a witness for $x \in L$ of length n is bounded by $w(n) = \mathrm{poly}(n)$. There is a language L', polynomial l and a deterministic reduction that on input n, runs in time polynomial in n and outputs a randomized oracle circuit C with the following property.*

Suppose that C' is a randomized errorless oracle circuit that solves L' on input length $l(n)$ with error δ. Then $C^{C'}$ solves the search version of L on input length n with error $\delta \cdot w(n)^2$.

6.5   Proof of the theorem

We now have all the ingredients in place. Suppose $(\mathrm{NP}, \mathcal{U}) \subseteq \mathrm{Avg}_{1-(\log n)^{-1/10+\alpha}}\mathrm{BPP}$. Since every $\perp$ answer can be converted into a guess by flipping a coin, it follows that for every $L \in \mathrm{NP}$ there is a randomized algorithm A such that

$$\Pr_{A, x \sim \{0,1\}^n}[A(x) = L(x)] \geq 1/2 + (\log n)^{-1/10+\alpha}.$$

Trevisan shows that under this assumption all of $(\mathrm{NP}, \mathcal{U})$ has heuristic schemes (not necessarily errorless).[12]

Now we are given a language L and want to design an errorless heuristic scheme for L. We define the following languages.

---

[12]To be precise, Trevisan's assumption is a bit stronger: The constant in the $\log n$ exponent may be smaller than $1/10$. However using Corollary 23 we can first amplify the errorless heuristic up to a range of parameters where Trevisan's result applies. This extra amplification step incurs $O(\log n)$ bits of advice, but this advice can be handled by Trevisan's reduction together with the rest of the advice incurred by the reduction.

1. Search version: Define $L_1 \in \mathrm{NP}$ to be the "search version" of L, as in Lemma 29.

2. Balancing: Choose t to be a sufficiently large constant. Define $L_2 \in \mathrm{NP}$ to be a version of L_1 that is infinitely often almost balanced as in Lemma 27.

3. High-end amplification: Define $L_3 \in \mathrm{NP}$ from L_2 as in Lemma 26.

4. Low-end amplification: Define $L_4, L_4^\dagger \in \mathrm{NP}$ from L_3 as in Corollary 23: L_4 and $L_4^\dagger$ are obtained by applying OR-DDNF and OR-DDNF†, respectively, to L_3.

We now describe a randomized errorless heuristic scheme for L. By equation (2) it is sufficient to give a randomized errorless heuristic algorithm with error $O(1/n)$.

1. By assumption, there are efficient randomized errorless heuristics A_4 and $A_4^\dagger$ that solve L_4 and $L_4^\dagger$ respectively on a $1 - (\log n)^{-1/10+\alpha}$ fraction of inputs of length n.

2. By Corollary 23, for every input length n we can randomly construct a deterministic circuit C_3 that with probability $\tilde{\Omega}(1/n^2)$ solves L_3 on a $1 - \varepsilon$ fraction of inputs of length n, provided L_3 is $(\log n)^{-1/10+\alpha}$-balanced on input length n.

   To construct C_3, we apply the reduction from Corollary 23 to the circuits representing both A_4 and $A_4^\dagger$ on appropriate input lengths, thereby obtaining two errorless circuits that solve L_3 on length n on 0-inputs and 1-inputs, respectively. The circuit C_3 will, on input x, run both these circuits, output $\perp$ if both of them do so, and output 0 or 1 if either of them does so. (Since the circuits are errorless there will be no inconsistency.)

3. By Lemma 26, for every input length n we can construct a randomized circuit C_2 such that if C_3 satisfies the above condition and L_2 is $n^{-0.5}$-balanced on input length n, then C_2 is an errorless circuit for L_2 with error $2/n^{1/5}$.

4. By Lemma 20, for every input length n with probability $3/4$ we can construct a randomized circuit C_1 and a number N such that L_2 is $N^{-0.5}$-balanced on input length N and if the circuit C_2 satisfies the above condition, then C_1 solves L_1 with error $4/n^{t/5}$. Here we use the fact that L_2 admits heuristic schemes.

5. Finally, by Lemma 29 we can construct a circuit C such that if C_1 satisfies the above condition then C is an errorless heuristic circuit that solves L on input length n with error $\varepsilon(n) = O(w(n)^2/n^{t/5})$, where $w(n)$ is the witness size for input length n.

We choose t to be sufficiently large so that $\varepsilon(n) \leq 1/n^3$. Multiplying the failure probabilities we obtain
$$\Pr\big[C \text{ is an errorless search circuit for } L \text{ with error } 1/n^3\big] = \Omega(1/n^2).$$
where the probability is over random coins of the reduction.

We now give a randomized algorithm A for L. On input x of length n,

1. Run the above reduction $N = O(n^2)$ times independently to obtain search circuits $C_1, \ldots, C_N$. Amplify the error probabilities of $C_1, \ldots, C_N$ individually by taking plurality over $O(n^3)$ independent settings of the randomness.

2. Sample each C_i on random inputs $O(n^3 \log n)$ times and eliminate circuit C_i if the fraction of times it outputs $\bot$ on the sample exceeds $4/n^3$.

3. Among the remaining circuits, run each C_i on input x. If any of the circuits outputs $\bot$, answer $\bot$. If any of the circuits produces a witness for x accept. Otherwise reject.

Claim 30. *A is a randomized errorless heuristic algorithm for L with error $O(1/n)$.*

To explain the claim, let's first pretend that the circuits $C_1, \ldots, C_N$ are deterministic because the reasoning is more clear. First, with high probability over the randomness of A, at least one of the circuits $C_1, \ldots, C_N$ will be an errorless search circuit for L on input length n with error $1/n^3$.

Without loss of generality suppose C_1 is an errorless search circuit for L. By Chernoff bounds, with high probability C_1 will not be eliminated in step 2 of the algorithm while all circuits that answer $\bot$ on more than $6/n^3$ fraction of inputs will be eliminated.

Everything that A did so far is completely independent of the input x. We now feed A its input x. If the choice of randomness in the two steps above was good, then A is an errorless heuristic for x: If $C_1(x)$ answers $\bot$, so does A. Otherwise, if $x \in L$, $C_1(x)$ produces a witness and $A(x)$ accepts. If $x \notin L$, then no circuit can produce a witness, but one of them can answer $\bot$, so $A(x)$ either answers $\bot$ or rejects.

Finally, since no circuit answers $\bot$ on more than $6/n^3$-fraction of inputs, A can answer $\bot$ on at most a $O(1/n)$-fraction of inputs.

Proof of Claim 30. With probability $7/8$, at least one of the circuits output by the reduction will be an errorless search circuit for L on input length n with error $1/n^3$. Let's call such a circuit good. Suppose that C_{i^*} is good. After amplification,

- For all $x \in L$ of length n, $\Pr_r[C_{i^*}(x,r)$ is a witness or $C_{i^*}(x,r) = \bot] \geq 7/8$, and

- For all x of length n, $\Pr_x\big[\Pr_r[C_{i^*}(x,r) = \bot] \geq 1/n^3\big] \leq 1/n^3$.

In particular, by the second condition, $\Pr_{x,r}[C_{i^*}(x,r) = \bot] \leq 2/n^3$. From this and Chernoff bounds, with probability $7/8$,

- C_{i^*} will not be eliminated in step 2, and

- For every C_i that survives step 2, $\Pr_{x,r}[C_i(x,r) = \bot] \leq 6/n^3$.

If this is the case we will say "step 2 succeeds" and denote this event by $S2$. Let I be the set of those i for which C_i survives step 2.

We now come to step 3. First we argue the fact that A is errorless. Fix an input x.

$$\Pr_A\big[A(x) \in \{L(x), \bot\}\big] \geq \Pr_A\big[A(x) \in \{L(x), \bot\} \mid \text{for some } i^*, C_{i^*} \text{ is good and } i^* \in I\big]$$
$$\cdot \Pr_A[\exists i^* : C_{i^*} \text{ is good and } i^* \in I].$$

For the first term, if $x \in L$, then $C_{i^*}(x)$ will output either a witness for x or $\bot$ with probability $7/8$, so $A(x) \in \{L(x), \bot\}$ with proability at least $7/8$. If $x \notin L$, then A will never accept because no C_i can produce a witness. So this term is bounded by $7/8$.

For the second term, we have that

$$\Pr_A[\exists i^* : C_{i^*} \text{ is good and } i^* \in I] = 1 - \Pr[\forall i : C_i \text{ is bad or } i \in I]$$

$$= 1 - \prod_{i=1}^{N} \Pr[C_i \text{ is bad or } i \in I]$$

$$= 1 - \prod_{i=1}^{N} \left(1 - \Pr[i \in I \mid C_i \text{ is good}] \cdot \Pr[C_i \text{ is good}]\right)$$

$$\geq 1 - \left(1 - \Omega(1/n^2)\right)^N$$

$$\geq 7/8,$$

thus $\Pr_A[A(x) \in \{L(x), \bot\}] \geq 7/8 \cdot 7/8 \geq 3/4$.

It remains to show that A does not err too often. First, let's fix x.

$$\Pr_A[A(x) = \bot] \leq \Pr_A[A(x) = \bot \mid S2] + (1 - \Pr_A[S2])$$

$$\leq \Pr_A[\exists i \in I : C_i(x) = \bot \mid S2] + 1/8$$

$$\leq \sum_{i \in I} \Pr_{C_i, r}[C_i(x, r) = \bot \mid S2] + 1/8.$$

We now bound the desired quantity.

$$\Pr_x\left[\Pr_A[A(x) = \bot] > 1/4\right] \leq \Pr_x\left[\sum_{i \in I} \Pr_{C_i, r}[C_i(x, r) = \bot \mid S2] > 1/8\right]$$

$$\leq 8 \, \mathrm{E}_x\left[\sum_{i \in I} \Pr_{C_i, r}[C_i(x, r) = \bot \mid S2]\right]$$

$$= \sum_{i \in I} 8 \Pr_{C_i, x, r}[C_i(x, r) = \bot \mid S2]$$

$$\leq \sum_{i \in I} 8 \cdot 6/n^3 = O(1/n).$$

The third step follows from the fact that the event $S2$ is independent of x: The algorithm A does not use the actual input x until step 3 of the algorithm. $\qquad\square$

This concludes the proof of Theorem 21.

Acknowledgments

References

[BCGL92] Shai Ben-David, Benny Chor, Oded Goldreich, and Michael Luby. On the theory of average case complexity. *Journal of Computer and System Sciences*, 44(2):193–219, 1992. 1, 23, 28

[BKS06]   Joshua Buresh-Oppenheim, Valentine Kabanets, and Rahul Santhanam. Uniform hardness amplification in NP via monotone codes. Technical Report TR06-154, Electronic Colloquium on Computational Complexity, 2006.   4, 9

[BSW05]   Itai Benjamini, Oded Schramm, and David B. Wilson. Balanced boolean functions that can be evaluated so that every input bit is unlikely to be read. In *Proceedings of the 37th ACM Symposium on Theory of Computing*, pages 244–250, 2005.   8, 37

[BT03]   Andrej Bogdanov and Luca Trevisan. On wost-case to average-case reductions for NP problems. In *Proceedings of the 44th IEEE Symposium on Foundations of Computer Science*, pages 308–317, 2003.   2

[BT06]   Andrej Bogdanov and Luca Trevisan. Average-case complexity. In Madhu Sudan, editor, *Foundations and Trends in Theoretical Computer Science*. Now Publishers, 2006. To appear.   12, 13

[Fei02]   Uriel Feige. Relations between average case complexity and approximation complexity. In *Proceedings of the 34th ACM Symposium on Theory of Computing*, pages 534–543, 2002.   2

[FK96]   Ehud Friedgut and Gil Kalai. Every monotone graph property has a sharp threshold. *Proc. Amer. Math. Soc.*, 124(10):2993–3002, 1996.   39

[Gol01]   Oded Goldreich. *The Foundations of Cryptography - Volume 1*. Cambridge University Press, 2001.   4

[GS99]   Venkatesan Guruswami and Madhu Sudan. Improved decoding of Reed-Solomon and algebraic-geometric codes. *IEEE Transactions on Information Theory*, 45(6):1757–1767, 1999.   21

[HR03]   Tzvika Hartman and Ran Raz. On the distribution of the number of roots of polynomials and explicit weak designs. *Random Structures and Algorithms*, 23:235–263, 2003.   7

[HVV04]   Alexander Healy, Salil Vadhan, and Emanuele Viola. Using nondeterminism to amplify hardness. In *Proceedings of the 36th ACM Symposium on Theory of Computing*, pages 192–201, 2004.   3, 5

[IJK06]   Russell Impagliazzo, Ragesh Jaiswal, and Valentine Kabanets. Approximately list-decoding direct product codes and uniform hardness amplification. In *Proceedings of the 47th IEEE Symposium on Foundations of Computer Science*, 2006.   9

[IL90]   Russell Impagliazzo and Leonid Levin. No better ways to generate hard NP instances than picking uniformly at random. In *Proceedings of the 31st IEEE Symposium on Foundations of Computer Science*, pages 812–821, 1990.   1, 2

[Imp95a]   Russell Impagliazzo. Hard-core distributions for somewhat hard problems. In *Proceedings of the 36th IEEE Symposium on Foundations of Computer Science*, pages 538–545, 1995.   4

[Imp95b]   Russell Impagliazzo. A personal view of average-case complexity. In *Proceedings of the 10th IEEE Conference on Structure in Complexity Theory*, pages 134–147, 1995.   1, 2, 3, 13

[Lev86]   Leonid Levin.   Average case complete problems.   *SIAM Journal on Computing*, 15(1):285–286, 1986.   1, 2

[MO03]   Elchanan Mossel and Ryan O'Donnell. On the noise sensitivity of monotone functions. *Random Structures and Algorithms*, 23(3):333–350, 2003.   39

[Nis91]   Noam Nisan. Pseudorandom bits for constant depth circuits. *Combinatorica*, 12(4):63–70, 1991.   7

[NW94]   Noam Nisan and Avi Wigderson. Hardness vs randomness. *Journal of Computer and System Sciences*, 49:149–167, 1994. Preliminary version in *Proc. of FOCS'88*.   7

[O'D02]   Ryan O'Donnell. Hardness amplification within NP. In *Proceedings of the 34th ACM Symposium on Theory of Computing*, pages 751–760, 2002.   3, 4, 5

[STV01]   Madhu Sudan, Luca Trevisan, and Salil Vadhan. Pseudorandom generators without the XOR lemma. *Journal of Computer and System Sciences*, 62(2):236–266, 2001.   4, 9

[Tal96]   Michel Talagrand. How much are increasing sets positively correlated? *Combinatorica*, 16(2):243–258, 1996.   39

[Tre03]   Luca Trevisan. List-decoding using the XOR Lemma. In *Proceedings of the 44th IEEE Symposium on Foundations of Computer Science*, pages 126–135, 2003.   3, 4, 8, 9

[Tre05]   Luca Trevisan. On uniform amplification of hardness in NP. In *Proceedings of the 37th ACM Symposium on Theory of Computing*, pages 31–38, 2005.   3, 8, 9, 22, 23, 25, 26, 27

[TV02]   Luca Trevisan and Salil Vadhan. Pseudorandomness and average-case complexity via uniform reductions. In *Proceedings of the 17th IEEE Conference on Computational Complexity*, pages 129–138, 2002.   4, 9

[Vio05]   Emanuele Viola. On constructing parallel pseudorandom generators from one-way functions. In *Proceedings of the 20th IEEE Conference on Computational Complexity*, 2005. 2

[Yao82]   Andrew C. Yao. Theory and applications of trapdoor functions. In *Proceedings of the 23th IEEE Symposium on Foundations of Computer Science*, pages 80–91, 1982.   4

A   DDNF as an amplifier

We now prove Theorem 14.

We identify the clauses of DDNF by the corresponding polynomial p. We say that assignment $z \in \{0,1\}^{t \times q}$ satisfies clause p if $z_{i,p(i)} = 1$ for all literals of p. We say that z almost satisfies p if

$z_{i,p(i)} = 1$ for all but exactly one of the literals of p. For $i \in [t]$ and two clauses $p \neq r$, we say that z satisfies p and q except at i if $p(i) = q(i)$, $z_{i,p(i)} = 0$, and z almost satisfies both p and q.

To prove the theorem, we will establish that for all but a $O(1/s + t/q)$ fraction of vertices z on the cube, the following three properties hold simultaneously:

1. No clause of DDNF is satisfied by z,

2. At least $t/2s$ clauses in DDNF are almost satisfied by z, and

3. There is no i, p, or r such that z satisfies p and r except at i.

Observe that these properties imply that $h_{\text{DDNF}}^{\uparrow}(z) \geq t/2s$, as property 1 says $\text{DDNF}(z) = 0$, property 2 says that at least $t/2s$ clauses contain pivotal coordinates, and property 3 says that no two of these variables can be the same.

We proceed with a probabilistic calculation. We will show that for a random z in the cube, property 2 holds with sufficiently high probability, while the complements of properties 1 and 3 hold with negligible probability.

Property 1 The expected number of clauses satisfied by DDNF is $q^{d+1} \cdot 2^{-t} = 1/s$, so by Markov's inequality $\Pr_z[\text{DDNF}(z) = 1] \leq 1/s$.

Property 2 To prove property 2 holds with high probability we introduce some notation. Consider a random assignment, and let Z_p be the indicator variable for the event that clause p is almost satisfied. Let $Z = \sum_p Z_p$. Observe that $\mathrm{E}[Z] = q^{d+1} \cdot t2^{-t} = t/s$. We will show a concentration bound on Z using the second moment method.

To do the second moment analysis, we will need some additional notation, which will be introduced as we go along. For a polynomial p, let $N(p)$ denote the number of zeros present among the values $p(1), \ldots, p(t)$.

We now proceed with the calculation. First, $E[Z^2] = E[Z] + \sum_{p \neq r} E[Z_p Z_r]$, where p and r range over polynomials of degree d. We bound the second quantity. For a pair of polynomials p and r, observe that $Z_p Z_r = 1$ if and only if clauses p and r are both almost satisfied, and the probability of this event is

$$E[Z_p Z_r] = \left[(t - N(p-r))^2 + N(p-r) \right] \cdot 2^{-2t+N(p-r)}.$$

Therefore

$$\sum_{p \neq r} \mathrm{E}[Z_p Z_r] \leq q^{2(d+1)} \, \mathrm{E}_{p \neq r}[\mathrm{E}[Z_p Z_r]]$$
$$= q^{2(d+1)} \, \mathrm{E}_{p \neq r}\left[[(t - N(p-r))^2 + N(p-r)] \cdot 2^{-2t+N(p-r)} \right]$$
$$= q^{2(d+1)} \, \mathrm{E}_{p \neq 0}\left[[(t - N(p))^2 + N(p)] \cdot 2^{-2t+N(p)} \right]$$
$$= q^{2(d+1)} \cdot 2^{-2t} \left(\mathrm{E}_{p \neq 0}[t^2 2^{N(p)}] - \mathrm{E}_p\left[(2t - N(p) - 1)N(p) \cdot 2^{N(p)} \right] \right)$$
$$\leq (t/s)^2 \, \mathrm{E}_{p \neq 0}[2^{N(p)}],$$

where the expectations are over a random choice of polynomials of degree d over $\mathbb{F}^q$. To bound the quantity $\mathrm{E}_{p \neq 0}[2^{N(p)}]$, we think of N as a random variable defined over the space of *nonzero*

34

polynomials of degree d under the uniform distribution. We will compare N with the random variable N^* which is a sum of t independent Bernoulli trials of probability $1/q$ each.

Claim 31. $\mathrm{E}[2^N] \leq \mathrm{E}[2^{N^*}]$.

Proof. We write $N = N_1 + \cdots + N_t$, where N_i is the indicator variable for the event that the random polynomial vanishes at i, and $N = N_1^* + \cdots + N_t^*$, where the N_i^* are independent Bernoulli.

$$\begin{aligned}
\mathrm{E}[2^{N^*}] - \mathrm{E}[2^N] &= \mathrm{E}[2^{N_1^* + \cdots + N_t^*}] - \mathrm{E}[2^{N_1 + \cdots + N_t}] \\
&= \mathrm{E}[(1 + N_1^*) \ldots (1 + N_t^*)] - \mathrm{E}[(1 + N_1) \ldots (1 + N_t)] \\
&= \sum\nolimits_{S \subseteq [t]} \big(\mathrm{E}[\textstyle\prod_{i \in S} N_i^*] - \mathrm{E}[\prod_{i \in S} N_i]\big) \\
&= \sum\nolimits_{j=1}^{t} \tbinom{k}{j} \big(\mathrm{E}[N_1^* \ldots N_t^*] - \mathrm{E}[N_1 \ldots N_t]\big).
\end{aligned}$$

We show that $\mathrm{E}[N_1^* \ldots N_t^*] \geq \mathrm{E}[N_1 \ldots N_t]$ for all t. Recall that $\mathrm{E}[N_1 \ldots N_t] = \Pr_{p \neq 0}[p(1) = \cdots = p(t) = 0]$. If $t > d$, this expression is zero, and otherwise,

$$\Pr_{p \neq 0}[p(1) = \cdots = p(t) = 0] \leq \Pr_p[p(1) = \cdots = p(t) = 0] = \Pr[N_1^* = \cdots = N_t^* = 0]$$

since the values of a random polynomial are d-wise independent. $\qquad\square$

Applying the claim, we obtain

$$\sum\nolimits_{p \neq r} \mathrm{E}[Z_p Z_r] \leq (t/s)^2 \, \mathrm{E}_{p \neq 0}[2^{N(p)}] \leq (t/s)^2 \, \mathrm{E}[2^{N^*}] = (t/s)^2 (1 + 1/q)^t \leq (t/s)^2 \cdot (1 + t/q + t^2/q^2).$$

We can now calculate the second moment deviation:

$$\Pr[Z < t/2s] \leq 4 \cdot \frac{\mathrm{E}[Z^2] - \mathrm{E}[Z]^2}{\mathrm{E}[Z]^2} \leq 4 \cdot \frac{t/s + (t/s)^2 \cdot (t/q + t^2/q^2)}{(t/s)^2} = 4 \cdot \frac{s}{t} + 4 \cdot \frac{t}{q} + 4 \cdot \frac{t^2}{q^2}.$$

Property 3 Fix an index i; without loss of generality we assume $i = 1$. We want to bound the quantity

$$\Pr_z[z \text{ satisfies } p \text{ and } r \text{ except at } 1 \text{ for some } p \neq r] \tag{3}$$

which is at most

$$\sum\nolimits_{p \neq r: p(1) = r(1)} \Pr_z[z_{1,p(1)} = 0 \text{ and } z_{j,p(j)} = z_{j,r(j)} = 1 \text{ for all } j \neq 1].$$

The above probability is $2^{-2t + N(p-r)}$, and we have

$$\begin{aligned}
\sum\nolimits_{p \neq r: p(1) = r(1)} 2^{-2t + N(p-r)} &\leq \frac{q^{2(d+1)}}{q} \, \mathrm{E}_{p \neq r: p(1) = r(1)}[2^{-2t + N(p-r)}] \\
&= \frac{1}{s^2 q} \, \mathrm{E}_{p \neq r: p(1) = r(1)}[2^{N(p-r)}] \\
&= \frac{1}{s^2 q} \, \mathrm{E}_{p \neq 0: p(1) = 0}[2^{N(p)}].
\end{aligned}$$

Now observe that if p is a random nonzero polynomial conditioned on $p(1) = 0$, then the polynomial $p'(u) = p(u)/(u-1)$ is a random nonzero polynomial of degree $d-1$, so we have that

$$\mathrm{E}_{p \neq 0 : p(1) = 0}[2^{N(p)}] = \mathrm{E}_{p' \neq 0}[2^{N'(p')+1}] = 2 \cdot \mathrm{E}_{p' \neq 0}[2^{N'(p')}],$$

where $N'(p')$ is the number of zeros of $p'(z)$ as z ranges from 2 to t. It follows from Claim 31 that $\mathrm{E}_{p' \neq 0}[2^{N'(p')}] \leq \mathrm{E}[2^{N^*}]$, where N^* is a sum of $t-1$ independent Bernoulli trials of probability $1/q$, so $\mathrm{E}[2^{N^*}] = (1 + 1/q)^{t-1} \leq 2$. It follows that the quantity in equation (3) is bounded from above by $4/s^2q$. By a union bound,

$$\Pr_z[z \text{ satisfies } p \text{ and } r \text{ except at } i \text{ for some } i, p, \text{ or } r] \leq \frac{4t}{s^2q}.$$

B   The holographic function as an amplifier

In this section we prove Theorem 16. We will think of $G = G_z$ as a random subgraph of H chosen according to the distribution $z \sim \mu_{n,p}$.

In what follows, by *cycle* we will always mean a directed cycle of length w in H, namely a sequence of vertices $(x_0, 0), \ldots, (x_{w-1}, w-1)$ where (x_i, i) and $(x_{i+1}, i+1)$ (modulo w) are edges in H (we think of the second index as representing a "time slice" and the first index is a position within that time slice.) For a cycle c and edge $e \in c$, we say c is a cycle in G open at e if $e \notin G$ and $e' \in G$ for all other edges e' in the cycle.

Let X be the number of cycles in G and Y be the number of edges e such that G has a *unique* cycle open at e. For fixed input z, each such edge yields a pivotal coordinate in HOL. To prove the theorem, it is sufficient to show that $\Pr[X > 0] \leq 1/s$ while $\Pr[Y < w/4s] \leq 0.8 + 2s/w$, since each edge counted by Y yields a pivotal coordinate.

For the analysis it helps to identify each cycle in H by a circular binary string c of length w. For each $0 \leq i < w$, the substring consisting of the bits $c_i c_{i+1} \ldots c_{i+\log_2 h - 1}$ when viewed as an integer in binary representation gives the position of the cycle at time slice i. In particular, there are 2^w cycles in H and by our choice of parameters

$$\Pr[X > 0] \leq E[X] = \sum_{\text{cycle } c} \Pr[c \text{ is a cycle in } G] = 2^w \cdot p^w = \frac{1}{s}.$$

To estimate Y, we let Y_1 indicate the total number of open cycles in G. Then $Y \leq Y_1$, since each unique open cycle is counted by Y_1. We let Y_2 indicate the total number of pairs of open cycles in G that are open at the same edge. Then $Y \geq Y_1 - Y_2$. We now compute a deviation bound on Y using the second moment method. First, we have

$$\mathrm{E}[Y_1] - \mathrm{E}[Y_2] \leq \mathrm{E}[Y] \leq \mathrm{E}[Y_1]$$

where

$$\mathrm{E}[Y_1] = \sum_{c, e \in c} \Pr[c \text{ in } G \text{ is open at } e] = w \cdot 2^w p^{w-1}(1-p) = \frac{1-p}{p} \cdot \frac{w}{s}.$$

For two cycles c, c' let $S(c, c')$ denote the number of common edges. Then

$$
\begin{aligned}
\mathrm{E}[Y_2] &= \sum_{c \neq c'} \sum_{e \in c \cap c'} \Pr[c, c' \text{ in } G \text{ are both open at } e] \\
&= \sum_{c \neq c'} S(c, c') p^{2w - S(c,c') - 1} (1 - p) \\
&\leq \frac{1 - p}{p} \cdot (2p)^{2w} \cdot \mathrm{E}_{C \neq C'}[S(C, C') \cdot p^{-S(C,C')}]
\end{aligned}
$$

The last expectation is over pairs of cycles $C \neq C'$ chosen uniformly at random from the collection of all cycles in H. To estimate this expectation, we say (as in [BSW05]) that i is a *join time* for two cycles if the cycles are at the same position in time slice i but not in time slice $i - 1$. Let $j(c, c')$ denote the set of join times for cycles c, c'. Then

$$
\begin{aligned}
\mathrm{E}_{C \neq C'} &\left[S(C, C') \cdot p^{-S(C,C')} \right] \\
&= \sum_{J \subseteq [w]} \mathrm{E}_{C \neq C'}[S(C, C') \cdot p^{-S(C,C')} \mid j(C, C') = J] \cdot \Pr[j(C, C') = J] \\
&\leq \sum_{l=1}^{w} \sum_{J : |J| = l} w \cdot \mathrm{E}_{C \neq C'}[p^{-S(C,C')} \mid j(C, C') = J] \cdot \Pr[j(C, C') = J].
\end{aligned}
\tag{4}
$$

If we view the cycles C and C' in string representation, the probability of $j(C, C') = J$ is exactly the probability that C and C' share the disjoint substrings[13] of length $\log_2 h + 1$ whose endpoints are indexed by elements of J at all but the first position, which is exactly $(2h)^{-l}$. For the other quantity, let $I_1, \ldots, I_l$ denote the intervals between the consecutive join times in J, where the end time is offset ahead by $\log_2 h + 1$. Then

$$
\mathrm{E}_{C \neq C'}\left[p^{-S(C,C')} \mid j(C, C') = J \right] = \mathrm{E}\left[p^{-(S_1 + \ldots, S_l)} \right] = \mathrm{E}\left[p^{-S_1} \right] \cdots \mathrm{E}\left[p^{-S_l} \right]
$$

where S_i is the length of the initial sequence in the interval I_i where the two cycles match. Then S_i is distributed as the number of leading zeros of a random string of length $w_i = |I_i|$ and

$$
\mathrm{E}\left[p^{-S_i} \right] = \sum_{j=0}^{w_i} p^{-j} \cdot \Pr[S_i = j] = \sum_{j=0}^{w_i} p^{-j} \cdot \frac{2^{-j-1}}{1 - 2^{-w_i - 1}} \leq \sum_{j=0}^{w_i} (2p)^{-j} \leq \frac{(2p)^{-w_i}}{1 - 2p}.
$$

Substituting into equation (4) we obtain

$$
\begin{aligned}
\mathrm{E}_{C \neq C'}\left[S(C, C') \cdot p^{-S(C,C')} \right] &\leq \sum_{l=1}^{w} \sum_{J : |J| = l} w \cdot \frac{(2p)^{-(w_i + \cdots + w_l)}}{(1 - 2p)^l} \cdot (2h)^{-l} \\
&\leq \sum_{l=1}^{w} w \cdot \binom{w}{l} \cdot (2p)^{-w} \cdot (2h \cdot (1 - 2p))^{-l} \\
&\leq w \cdot (2p)^{-w} \cdot \left[\exp\left(w / 2h(1 - 2p) \right) - 1 \right]
\end{aligned}
$$

The expression in the exponential can be upper bounded as follows:

$$
\begin{aligned}
\frac{w}{2h(1 - 2p)} &= \frac{w}{2h(1 - e^{-(\log s)/w})} && \text{by definition of } s \\
&\leq \frac{w}{(2h) \cdot (\log s / 2w)} = \frac{w^2}{h \log s} && \text{since } \log s \leq w \\
&\leq 1/9 && \text{since } \log s \geq 9w^2/h.
\end{aligned}
$$

---

[13] Join times must be spaced apart by at least $\log_2 h + 1$ by definition.

In the first inequality we used the identity $1 - e^{-t} \geq t/2$ which holds for $0 \leq t \leq 1$. Therefore $\exp\big(w/2h(1 - 2p)\big) - 1 \leq e^{1/9} - 1 \leq 0.12$. It follows that

$$\mathrm{E}[Y_2] \leq 0.12 \cdot \frac{1 - p}{p} \cdot \frac{w}{s}.$$

and therefore

$$0.88 \cdot \frac{1 - p}{p} \cdot \frac{w}{s} \leq \mathrm{E}[Y] \leq \frac{1 - p}{p} \cdot \frac{w}{s}. \tag{5}$$

We now upper bound $\mathrm{E}[Y^2]$. For a cycle c and edge $e \in C$, let Y_{ce} be an indicator variable for the event that c is the unique cycle in G open at e.

$$
\begin{aligned}
\mathrm{E}[Y^2] - \mathrm{E}[Y] &= \sum_{(c,e) \neq (c',e')} \mathrm{E}[Y_{ce} Y_{c'e'}] \\
&= \sum_{c \neq c'} \sum_{e \neq e'} \mathrm{E}[Y_{ce} Y_{c'e'}] &&\text{by uniqueness} \\
&= \sum_{c \neq c'} (w - S(c,c'))^2 \cdot p^{2w - S(c,c') - 2}(1 - p)^2 \\
&\leq \left(\frac{1 - p}{p}\right)^2 \cdot w^2 \cdot (2p)^{2w} \cdot \mathrm{E}_{C \neq C'}\big[p^{-S(C,C')}\big]
\end{aligned}
$$

Using a similar calculation as above we obtain

$$\mathrm{E}_{C \neq C'}\big[p^{-S(C,C')}\big] \leq \sum_{l=0}^{w} \binom{w}{l} \cdot (2p)^{-w} \cdot (2h \cdot (1 - 2p))^{-l} \leq (2p)^{-w} \cdot \exp\big(w/2h(1 - 2p)\big)$$

so that

$$\mathrm{E}[Y^2] - \mathrm{E}[Y] \leq \left(\frac{1 - p}{p}\right)^2 \cdot \left(\frac{w}{s}\right)^2 \cdot \exp\big(w/2h(1 - 2p)\big) \leq 1.12 \cdot \left(\frac{1 - p}{p}\right)^2 \cdot \left(\frac{w}{s}\right)^2.$$

Finally, from Chebyshev's inequality we obtain

$$\Pr\big[Y < \mathrm{E}[Y]/4\big] \leq \frac{16}{9} \cdot \frac{\mathrm{E}[Y^2] - \mathrm{E}[Y]^2}{\mathrm{E}[Y]^2} \leq 0.8 + \frac{2s}{w}.$$

C   Some properties of the DDNF function

C.1   Noise sensitivity

Consider the following setting of parameters for DDNF: Fix a constant $K \geq 25$, and set $s = 1/K$, $q = Kt$. Let $n = tq = Kt^2$. We prove the following.

Lemma 32. *Consider the following distribution on pairs (x, y): Choose x uniformly from $\{0,1\}^n$, and choose y by flipping each coordinate of x independently with probability $1/K\sqrt{n}$. Then*

$$\Pr[\mathrm{DDNF}(x) \neq \mathrm{DDNF}(y)] \geq 1/2K^2.$$

Thus DDNF is sensitive to noise as small as $1/\sqrt{n}$, which is optimal for a monotone function. The existence of such a function was studied before by Mossel and O'Donnell [MO03], who observed that a random DNF function considered by Talagrand [Tal96] has this property. However this function is obtained by a non-constructive argument, and its description size is $2^{\Omega(\sqrt{n})}$ bits. In contrast, the DDNF function is in NP.

Proof. Let $p = 1/K\sqrt{n}$, and let $F_p(x)$ denote the random variable that equals x with probability $1 - p$ and $\overline{x}$ with probability p.

We consider the following equivalent way of sampling from the distribution on pairs (x, y). First assume that $n = 1$ and choose x at random. Set

$$z = \begin{cases} 0, & \text{if } x = 0, \\ F_{p/(1-p)}, & \text{if } x = 1. \end{cases}$$

Then set

$$y = \begin{cases} F_p(z), & \text{if } z = 0, \\ 1, & \text{if } z = 1, \end{cases}$$

It is easy to check that (x, y) has the right distribution. Observe that in all cases, $z \le x$ and $y \ge z$. For larger n, generate each coordinate of (x_i, y_i) of (x, y) independently by this experiment and call the intermediate coordinate z_i, and $z = (z_1, \dots, z_n)$.

We now estimate the noise sensitivity of f. First observe that z is disributed according to the $\frac{1}{2}(1 - p/(1 - p))$-biased distribution on $\{0, 1\}^n$. Note that $p/(1 - p) \le 2/K\sqrt{n}$. Using Fact 1, we have that

$$\Pr_z[h_f^\uparrow(z) < \sqrt{n}/K] \le \Pr_x[h_f^\uparrow(x) < \sqrt{n}/K] + 2/K \le 10/K.$$

Now fix a z for which $h_f^\uparrow(z) \ge \sqrt{n}/K$ and consider the process of generating y from z. Let us focus on the pivotal coordinates of z: The probability that none of these is flipped is $(1-p)^{\sqrt{n}/K} \le e^{-1/K^2}$; if one of them is flipped however, we have $f(y) \ne f(x)$, and it follows that

$$\Pr[f(x) \ne f(y)] \ge (1 - 10/K)(1 - e^{-1/K^2}) \ge 1/2K^2. \qquad \square$$

C.2 Near optimality of boundary size

The OR-DDNF$_\alpha$ function was shown to be a $(n^\alpha, 9\sqrt{\beta \log n} \cdot n^{-\beta}))^\uparrow$-amplifier (Theorem 15) for every $0 \le \alpha < 1/2$, $\beta = 1/2 - \alpha$, and sufficiently large n. We show that this is optimal up to logarithmic factors.

Lemma 33. *For every $\alpha \ge 0, \beta > 0$ and sufficiently large n, if f is an $(n^\alpha, (K\beta \log n)^{-1/2} \cdot n^{-\beta})$ amplifier on n bits, then $\alpha + \beta < 1/2$. Here K is a universal constant.*

Proof. Set $p(\beta, n) = (K\beta \log n)^{-1/2} n^{-\beta}$. Let us look at the quantity $I(f) = \mathrm{E}_z[h_f^\uparrow(z)]$:

$$I(f) \ge n^\alpha(1 - p(\beta, n)) \ge n^\alpha/2 \tag{6}$$

for sufficiently large n. Now we fix n and look at all functions f on n bits such that $\mu(f) = \Pr_z[f(z) = 0] \le p(\beta, n)$. It is known that among such functions, the quantity $I(f)$ is maximized by a threshold function (a proof appears for instance in [FK96]):

Claim 34. *Fix n and $p \leq 1/2$. Among all functions f on n bits such that $\mu(f) = p$, $I(f) \leq I(\text{THR}^\star)$, where $\text{THR}^\star$ is the unique threshold function such that $\mu(\text{THR}^\star) = p$.*

Let $\text{THR}_{k,n}$ denote the n bit function which evaluates to 1 iff at least k of its inputs are ones. We can also show the following relations for threshold functions:

Claim 35. *Suppose $k \geq n/2$ and let $d = k - n/2$. The following hold for sufficiently large n:*

1. *(Chernoff bound) $\mu(\text{THR}_{k,n}) \leq \exp(-2d^2/n)$.*

2. *If $d \geq \sqrt{n}$, then $I(\text{THR}_{k-1,n}) \leq 54d \cdot \mu(\text{THR}_{k,n})$.*

3. *Let THR, THR' be threshold functions with $\mu(\text{THR}) \leq \mu(\text{THR}') \leq 1/2$. Then $I(\text{THR}) \leq I(\text{THR}')$.*

Let $\text{THR}^\star$ be the unique threshold function with $\mu(\text{THR}^\star) = p(\beta, n)$. We choose $k^\star$ as the largest integer for which $\mu(\text{THR}_{k^\star,n}) \geq p(\beta, n)$ and set $d^\star = k^\star - n/2$. It must be that $d^\star \leq d = \sqrt{\beta n \log n}$, because for $k = n/2 + d$ we have

$$\mu(\text{THR}_{k,n}) \leq \exp(-2d^2/n) = n^{-2\beta} < p(\beta, n)$$

using part (1) of Claim 35. Then

$$
\begin{aligned}
n^\alpha/2 &\leq I(f) && \text{by equation (6)} \\
&\leq I(\text{THR}^\star) && \text{by Claim 34} \\
&\leq I(\text{THR}_{k^\star,n}) && \text{by part (3) of Claim 35} \\
&\leq 54d^\star \cdot \mu(\text{THR}_{k^\star+1,n}) && \text{by part (2) of Claim 35} \\
&< 54d^\star \cdot p(\beta, n) && \text{by choice of } k^\star \\
&< n^{1/2-\beta}/2,
\end{aligned}
$$

from where $\alpha + \beta < 1/2$. $\qquad\square$

Proof of Claim 35. Part 1 is a standard version of the Chernoff bound. For part 2, we use the following estimate which holds for every $l \geq k$:

$$\mu(\text{THR}_{k,n}) = \sum_{i=k}^{n} \binom{n}{i} \geq (l - k + 1) \cdot \binom{n}{l}$$

from where

$$\frac{\mu(\text{THR}_{k,n})}{I(\text{THR}_{k-1,n})} \geq \frac{(l - k + 1) \cdot \binom{n}{l}}{2(k-1)\binom{n}{k-1}} \geq \frac{l - k + 1}{2n} \cdot \frac{\binom{n}{l}}{\binom{n}{k-1}}.$$

We choose $l = k + n/9d - 1$. Then

$$
\begin{aligned}
\binom{n}{l} \Big/ \binom{n}{k-1} &= \prod_{i=d}^{d+n/9d-1} \frac{n/2 - i}{n/2 + i} \geq \left(\frac{n/2 - d - n/9d}{n/2 + d + n/9d}\right)^{n/9d} \\
&= \left(1 - \frac{2(d + n/9d)}{l}\right)^{n/9d} \geq \left(1 - \frac{3d}{l}\right)^{n/9d} \geq 1 - \frac{n}{3l} \geq 1/3,
\end{aligned}
$$

since $l \geq k \geq n/2$. It follows that

$$\frac{\mu(\text{THR}_{k,n})}{I(\text{THR}_{k,n})} \geq \frac{l-k+1}{2n} \cdot \frac{1}{3} \geq \frac{1}{54d}.$$

For part 3, it is sufficient to consider two threshold functions THR and THR′ that differ in exactly one input z. It is easy to check that

$$I(\text{THR}') - I(\text{THR}) = |\{i : z_i = 0\}| - |\{i : z_i = 1\}|$$

and since $\mu(\text{THR}') \leq 1/2$, the last quantity is nonnegative. $\qquad\square$