Electronic Colloquium on Computational Complexity, Report No. 89 (2007)

Deterministic Hardness Amplification via Local GMD Decoding

Parikshit Gopalan
University of Washington
parik@cs.washington.edu

Venkatesan Guruswami*
U. Washington & IAS
venkat@cs.washington.edu

September, 2007

Abstract

We study the average-case hardness of the class NP against deterministic polynomial time algorithms. We prove that there exists some constant $\mu > 0$ such that if there is some language in NP for which no deterministic polynomial time algorithm can decide L correctly on a $1 - (\log n)^{-\mu}$ fraction of inputs of length n, then there is a language L' in NP for which no deterministic polynomial time algorithm can decide L' correctly on a $3/4 + (\log n)^{-\mu}$ fraction of inputs of length n. In coding theoretic terms, we give a construction of a monotone code that can be uniquely decoded up to error rate $\frac{1}{4}$ by a *deterministic* local decoder.

1 Introduction

The average case hardness of complexity classes such as EXP and NP has been studied intensively. There are several motivations for this study: in addition to being a natural alternative to worst-case hardness, average-case hardness is also necessary in cryptography. Functions with extreme average-case hardness also play a key role in the construction of pseudorandom generators and derandomization. This hardness is measured against a certain complexity class $\mathcal{C}$. We consider Boolean functions $f : \{0,1\}^* \to \{0,1\}$. The function f may not be defined for all input lengths n, but it should be defined for infinitely many n. We only consider such n in the definition below.

Definition 1. *We say that f is $(1-\delta)$-hard* almost everywhere (a.e) *for $\mathcal{C}$ if for every $\mathcal{A} \in \mathcal{C}$ there is some n_0 so that for all $n \geqslant n_0$, $\Pr_{x\in\{0,1\}^n}[\mathcal{A}(x) \neq f(x)] \geqslant \delta$. We say f is $(1-\delta)$-hard* infinitely often (i.o) *for $\mathcal{C}$ if for every $\mathcal{A} \in \mathcal{C}$ there are infinitely many n such that $\Pr_{x\in\{0,1\}^n}[\mathcal{A}(x) \neq f(x)] \geqslant \delta$.*

Our main focus in this work is on the average-case hardness of NP languages, and specifically on *hardness amplification* results that convert an NP language with mild average-case hardness into another NP language that is much harder. Previous hardness amplification results for NP focus on the case when the class $\mathcal{C}$ is either the class P/poly of poly-size circuits [O'D04, HVV06] or the class BPP of randomized polynomial time algorithms [Tre03, Tre05, BOKS06]. In this work we study the hardness of NP for the class P of *deterministic* polynomial time algorithms.[1] The following is our main result.

*Research supported in part by an Alfred P. Sloan Research Fellowship, NSF grant CCF-0343672, and a David and Lucile Packard Foundation Fellowship.

[1]Technically, P is the class of languages that admit deterministic polynomial time algorithms, and not the algorithms themselves. Likewise for BPP and P/poly. For ease of notation, we blur this distinction in the sequel.

ISSN 1433-8092

Theorem 1. *There is a constant $\mu > 0$ such that if there is some balanced function in* NP *which is $\left(1 - \frac{1}{(\log n)^{\mu}}\right)$-hard a.e/i.o for* P *then there is a function in* NP *which is $\left(\frac{3}{4} + \frac{1}{(\log n)^{\mu}}\right)$-hard a.e/i.o for* P*.*

This is the first hardness amplification result for NP against P. This matches the hardness parameters shown by Trevisan in [Tre03] for NP against BPP. Subsequently, this was improved to an amplification from $(1 - \frac{1}{\text{poly}(n)})$-hardness to $(\frac{1}{2} + o(1))$-hardness by [Tre05, BOKS06]. Stronger results are known for amplification against P/poly (see Section 1.1).

The conclusion of Theorem 1 still holds if the original mildly hard function is only close to balanced, and has bias bounded by $(\log n)^{-C}$ for some constant $C = C(\mu) > 0$ (with $C(\mu) \to 0$ for $\mu \to 0$). If we assume hardness a.e. against slightly non-uniform algorithms, we can eliminate the balance hypothesis entirely, and still conclude $(\frac{3}{4} + o(1))$-hardness i.o.

Theorem 2. *There is a constant $\mu > 0$ such that if there is some function in* NP *which is $(1 - \frac{1}{(\log n)^{\mu}})$-hard a.e for $\text{P}/\log n$ then there is a function in* NP *which is $(\frac{3}{4} + \frac{1}{(\log n)^{\mu}})$-hard i.o for* P*.* [2]

Our techniques also yield a simple proof of amplification from $(1 - 1/\text{poly}(n))$-hardness to $(3/4 + 1/\text{poly}(n))$-hardness, for languages in PSPACE (or any class closed under XOR) against P. This improves on the amplification from $1 - \frac{1}{\text{poly}(n)}$ to $\frac{7}{8} + \frac{1}{\text{poly}(n)}$ shown by Trevisan [Tre03]. Stronger amplification results are known for PSPACE and EXP against BPP and P/poly (see Section 1.1).

Theorem 3. *If there is some function in* PSPACE *which is $(1 - 1/\text{poly}(n))$-hard a.e/i.o for* P *then there is a function in* PSPACE *which is $(\frac{3}{4} + 1/\text{poly}(n))$-hard a.e/i.o for* P*.*

One can replace PSPACE by any complexity class which is closed under XOR. Though we note that for EXP and higher classes, very strong average case hardness is known unconditionally via diagonalization [GW00].

There is a well-studied connection between hardness amplification and error-correcting codes [STV01, Tre03]. Black-box hardness amplification amounts to taking a code with weak error-correction properties and converting it into a better error-correcting code which has a local decoding algorithm. A local decoding algorithm is one that can recover any bit of the message by querying the received word at a few locations. Buresh-Oppenheim *et al.* [BOKS06] define a family of codes called monotone codes and showed that error-correction for these codes can give amplification within NP. In what follows it will be helpful to think of the message $f \in \{0,1\}^N$ as the truth-table of a Boolean function on n bits, where $N = 2^n$, indexed by strings in $\{0,1\}^n$.

Definition 2. *An $[N, M]$ monotone code is a map $\text{C} : \{0,1\}^N \to \{0,1\}^M$ is a map such that each bit of the codeword $\text{C}(f)$ is a monotone function in the bits of the message f.*

Typically, we want $M = 2^{\text{poly}(n)}$. Also, we want our local decoder to run in time $\text{poly}(n)$ when asked for any bit of the message, which implies that it can only read a tiny fraction of the received word. It is easy to show that monotone codes cannot have good distance for all messages. However it is possible to have good distance if we restrict ourselves to balanced messages. Further one needs to settle for recovering the message approximately as opposed to exactly, we will show that this is an inherent limitation of such codes.

[2] In fact, it suffices to assume hardness against $\text{P}/(\log n)^{\varepsilon}$ for any fixed $\varepsilon > 0$, and the claimed statement will hold with some $\mu = \mu(\varepsilon) > 0$.

To show hardness amplification against P, one needs the local decoding algorithm $\mathcal{A}$ to be *deterministic*. This is a non-trivial requirement since in the conventional setting where we wish to recover the message exactly, it is impossible for a local decoder that tolerates a constant fraction of errors to be deterministic. This is because an adversary could corrupt all the bits of the input that are probed by the decoder to recover a particular bit of the message. The locality property of the decoder implies that this is a tiny fraction of the entire message. However, this does not rule out a deterministic local decoder that is able to recover all but a fraction of the message bits. Indeed, our result gives a monotone code with a deterministic local decoder that can recover from $\frac{1}{4} - o(1)$ errors.

Theorem 4. *For some constant $\mu > 0$, there is a monotone $[N, M]$ code C and a deterministic local decoder $\mathcal{A}$ which can $(\log n)^{-\mu}$-approximately decode C up to distance $\frac{1}{4} - (\log n)^{-\mu}$ on balanced messages which $M = 2^{n(1+o(1))}$. Further, $\mathcal{A}$ runs in time $o(n)$.*

Finally, we prove lower bounds on the error-correcting capabilities of monotone codes which can be used in hardness amplification. For EXP, one can obtain optimal average-case hardness starting from worst-case hardness assumptions [STV01, TV02]. One does not expect similar black-box reductions for NP, under some complexity theoretic assumptions [BT06]. The crux of the reduction for EXP is the existence of codes which can be decoded exactly even at very high error-rates. We observe that there exist monotone codes which have similar error-correcting properties. However, for use in black-box amplification, one requires an upper bound on the complexity of encoding each bit; namely that the functions used have small certificates. We show that this places certain restrictions on the error-correcting capacity of the code. On one hand we show that starting from worst case hardness, one can only get very weak-average case hardness $(1 - 2^{-n(1-o(1))})$. At the other end of the spectrum, to get a hardness of $1 - \eta$ for some constant $\eta > 0$, we show that one must start by assuming $1 - \frac{1}{\mathrm{poly}(n)}$ hardness (see Section 4 for precise statements).

1.1 Previous work

The has been a long body of work in computer science devoted to studying hardness amplification in various scenarios. In addition to being a natural question in its own right, hardness amplification has important applications in cryptography and derandomization. The first such result is the famous XOR Lemma due to Yao, which asserts that computing the XOR of k independent copies of a mildly hard function f is much harder than computing f [Yao82, GNW95]. There has been a long line of research that studies amplification for EXP against BPP and P/poly, motivated by derandomization [BFNW93, Imp95, IW97, IW98, STV01, TV02].

The study of hardness amplification within NP was initiated in a beautiful paper by O'Donnell [O'D04] who showed that one can amplify $(1 - 1/\mathrm{poly}(n))$-hardness to $(1/2 + 1/n^{1/3})$-hardness for NP against polynomial size circuits. The key technical ingredient in this work was the analysis of a variant of Yao's XOR Lemma when the XOR function is replaced by a monotone function. The efficacy of a particular monotone function in such a setting was rather precisely tied to the noise sensitivity of the function. O'Donnell's result was improved by Healy, Vadhan, and Viola [HVV06] who showed how to amplify $(1 - 1/\mathrm{poly}(n))$-hardness to $(1/2 + 1/\mathrm{poly}(n))$-hardness, also against polynomial size circuits. The non-uniformity in these proofs seemed inherent due to the use of Impagliazzo's powerful hard-core set lemma [Imp95].

The first *uniform* hardness amplification result for NP, albeit against randomized algorithms, was due to Trevisan [Tre03]. He was able to amplify $(1 - 1/(\log n)^{\alpha})$-hardness to $(3/4 + 1/(\log n)^{\alpha})$-

hardness (which is identical to what we achieve for deterministic polynomial time algorithms in this work). Trevisan's proof was based on a "more uniform" version of the hard-core set lemma, but the amount of non-uniformity was large enough to preclude amplifying from hardness fractions greater than $1 - 1/(\log n)^{\alpha}$.

In [Tre05], Trevisan built upon the result of [Tre03] to achieve amplification from $(1-1/\operatorname{poly}(n))$-hardness to $(1/2 + 1/(\log n)^{\alpha})$-hardness, for NP against randomized polynomial time algorithms. An alternate proof of this result was given by [BOKS06] based on monotone codes, and the powerful direct product theorem of Impagliazzo *et al.* [IJK06].

Our work seems to be the first to address the average-case hardness of NP against deterministic uniform algorithms. Hardness amplification against P was considered previously by Trevisan [Tre03] who proved an amplification from $(1 - 1/\operatorname{poly}(n))$-hardness to $(7/8 + 1/\operatorname{poly}(n))$-hardness for PSPACE (or any class that is closed under XOR). Goldreich and Wigderson use diagonalization to prove the existence of languages in EXP which are hard on average a.e for P [GW00].

There has been a body of work exploring limitations to the hardness amplification parameters that are achievable via various kinds of black-box reductions [TV02, O'D04, HVV06, BOKS06]. Our negative results in Section 4 complement these results.

1.2 Technical Contributions

Just as amplification against BPP requires uniform local decoding algorithms, amplification against P requires derandomized local decoding. Our main technical contribution is the construction of a monotone code that has an efficient, deterministic local decoder that can correct up to $\frac{1}{4} - o(1)$ errors. Note that $\frac{1}{4}$ is an upper bound on the unique-decoding radius of any non-trivial binary code, and due to the well-known connection between hardness amplification and coding theory, it is also the limit for results shown using uniform "black-box" reductions (see, for instance, the discussion in [Tre05] or [TV02, Sec. 6]). Thus the $(3/4 + o(1))$-hardness we achieve is a natural barrier for our techniques.

Our construction and proof is based on majority logic decoding of an expander-based code [ABN$^+$92, GI01] and *Generalized Minimum Distance* (GMD) decoding of concatenated codes [For66]. The first step in the encoding is to use the expander-based construction of Alon *et al.*[ABN$^+$92] which gives a good distance code over a larger alphabet as the outer code. In conjunction with the majority logic decoder for such codes due to [GI01], this gives a derandomized direct product theorem in the spirit of [IJK06].

A standard way to reduce the alphabet size is to use concatenation with an *inner* binary code. The advantage is that the binary code is now applied to a small message size, and thus can be decoded even by brute force. By adapting GMD decoding to this setting, Guruswami and Indyk [GI01, GI05] constructed binary codes that can decoded up to a $(1/4 - \varepsilon)$ fraction of errors in time linear in the block length. Trevisan [Tre03, Theorem 7] used the same construction with a simpler (but less powerful) local decoder to amplify $(1-1/\operatorname{poly}(n))$-hardness to $(7/8+1/\operatorname{poly}(n))$-hardness for PSPACE against the class P. However, translating this to the monotone setting and achieving a decoding radius of $\frac{1}{4}$ locally and deterministically require overcoming some obstacles which we detail below.

A significant barrier is that the naive decoder will only correct a fraction $1/8$ of errors and in order to decode up to a fraction $(1/4 - o(1))$ of errors, one needs to implement GMD decoding

in a *local* manner.[3] The standard approach behind GMD decoding is to pass weights from the inner decodings along with the decoded symbols. The outer decoder then picks a threshold, erases all symbols with weights below the threshold, and decodes from the remaining symbols using a majority vote for each bit of the message. This is repeated for each possible threshold. This seems hard to implement locally and uniformly since the same threshold needs to be used for all bits of the message and we do not know in advance which threshold will work.

We bypass this problem by using the weights as *soft information* representing the confidence of each inner decoder and take a weighted majority. This scheme is local and deterministic since to recover a message bit, we only need to decode the inner codes for its neighbors, and then take a majority vote. To prove that this scheme works, we use the spectral properties of the underlying expander graph.

In order for the final function to belong to NP the inner code has to be monotone. However monotone codes are only guaranteed to have good distance when the messages are far apart, and when each of them is balanced or close to balanced. Indeed, the code we use has the property that nearby messages have similar encodings, and this is crucial to our analysis. On the decoding side, we can only guarantee that messages are recovered approximately even when the decoding of the inner code succeeds (unlike in the conventional setting), which makes the analysis of the outer decoder much harder.

Our local GMD decoding technique can also be used to amplify hardness against P within EXP via concatenation with standard binary codes. In fact the proof is easier since none of the complications of monotone codes arise.

2 Preliminaries

2.1 Expander Graphs

A crucial ingredient in our construction is k-regular expander graphs with small second eigenvalue. We need graphs on N vertices where $N = 2^n$, whose degree is $k = (\log n)^\tau$ where $\tau \gg 1$ and whose second largest eigenvalue (in absolute value) λ is bounded by $k^{(1-\nu)}$ for some absolute constant $\nu > 0$. Finally, we need graphs which are highly explicit in the sense that the i'th neighbor of any vertex v can be computed in deterministic poly(n) time. The standard way to get graphs with good eigenvalue gap is to use the LPS construction of Ramanujan graphs. However, to construct such a graph with $N = 2^n$ vertices requires finding a large prime of size $N^{\Omega(1)}$ and it is not known how to perform this task in deterministic poly(n) time. Instead, we start with the construction of Margulis and its analysis due to Gabber-Galil [HLW06, Chap. 8].

Definition 3. *Let N be a square. The vertex set of G_N is $\mathbb{Z}_{\sqrt{N}} \times \mathbb{Z}_{\sqrt{N}}$. Let*

$$T_1 = \begin{pmatrix} 1 & 2 \\ 0 & 1 \end{pmatrix},\ T_2 = \begin{pmatrix} 1 & 0 \\ 2 & 1 \end{pmatrix},\ e_1 = \begin{pmatrix} 1 \\ 0 \end{pmatrix},\ e_2 = \begin{pmatrix} 0 \\ 1 \end{pmatrix}.$$

The vertex $v = (x, y)$ is adjacent to $T_1 v, T_2 v, T_1 v + e_1, T_2 v + e_2$ and four other vertices obtained by the inverse transformations.

[3] It is mentioned in [GK06, Remark 4.9] that the binary code construction from [GI01] can be used to boost hardness to $(3/4 + \varepsilon)$ in a uniform, deterministic way. However, it is unclear if the standard GMD algorithm can be implemented locally.

Note that since the transformations are invertible G_N has degree 8. The following result of Gabber and Galil bounds its second eigenvalue.

Theorem 5 ([HLW06])**.** *The graph G_N satisfies $\lambda \leqslant 5\sqrt{2} < 8$.*

Now take G_N^t, the t^{th} power of G_N, the (multi)graph where vertices are adjacent if they are connected by a length-t walk in G. It is easy to see that that this graph has N vertices, degree d^t and second eigenvalue $\lambda \leqslant (5\sqrt{2})^t = d^{t(1-\nu)}$ for some explicit constant $\nu > 0$. The parameters we want are obtained by taking $t = \Theta(\log \log n)$.

Theorem 6. *There is an absolute constant $\nu > 0$ such that for all $\tau > 0$ and for all large enough* even*integers n, there exists an integer $t = t(\tau, n)$ and a highly explicit graph G_N^t on $N = 2^n$ vertices with degree $k = \Theta\left((\log n)^\tau\right)$ and second eigenvalue $\lambda \leqslant k^{(1-\nu)}$.*

The above construction works only for even n. Note that if $f : \{0,1\}^n \to \{0,1\}$ is $(1-\delta)$-hard for $\mathcal{C}$, then the function $\tilde{f} : \{0,1\}^{n+1} \to \{0,1\}$ defined by $\tilde{f}(x,b) = f(x)$ for $b \in \{0,1\}$ is $(1-\delta/2)$-hard for class $\mathcal{C}$. For our application to hardness amplification, we can thus assume that the input length n of f is even, and thus use the above expanders.

Finally, let $G(A, B, E)$ be the $k = d^t$-regular bipartite multigraph obtained by taking the double cover of G_N^t. By the expander mixing lemma, for $S \subseteq A$ and $T \subseteq B$, we have

$$\left| |E(S,T)| - \frac{k|S||T|}{N} \right| \leqslant \lambda\sqrt{|S||T|}.$$

In the above, $|E(S,T)|$ is the number of edges between vertices in S and T, where multiple edges are counted as many times.

2.2 Monotone Codes

We begin by formally defining distance and decodability for monotone codes. We use Δ to denote the normalized Hamming distance between two strings of equal lengths.

Definition 4. *A monotone code* C *has distance (α, β) if for any two balanced messages f and g such that $\Delta(f,g) \geqslant \alpha$, $\Delta(\mathrm{C}(f), \mathrm{C}(g)) \geqslant \beta$.*
Algorithm $\mathcal{A}$ α-approximately decodes C *up to distance γ if when given a received word R, if there is a balanced f such that $\Delta(\mathrm{C}(f), R)) < \gamma$, $\mathcal{A}$ outputs g such that $\Delta(f,g) < \alpha$.*

Notice that we only require good distance between balanced messages that are far apart. This is not an inherent limitation of all monotone codes, [BOKS06] prove the existence of monotone codes that have good distance for any pair of balanced messages. However in Section 4, we show that monotone codes which can be used in black-box hardness amplification do have inherent limitations; they can only have good distance for messages which are far apart.

The following generic construction of monotone codes is from [BOKS06].

Definition 5. *Given a monotone function $g : \{0,1\}^r \to \{0,1\}$, we define the $[k, k^r]$ code $g^{k,r}$ as follows. The codeword is indexed by $i = (i_1, \ldots, i_r) \in [k]^r$. Given a message $x \in \{0,1\}^k$, the bit of $g^{k,r}(x)$ for index i is given by*

$$g^{k,r}(x)_i = g(x_{i_1}, \ldots, x_{i_r}).$$

To compute a random index of $g^{k,r}(x)$, we sample an r-tuple of coordinates in x with repetition, and apply the function g. To analyze the properties of the code, we use the notion of noise-sensitivity of a Boolean function.

Definition 6. *Given $x \in \{0,1\}^r$ let $\mathrm{N}_\delta(x)$ denote the random vector obtained by flipping each bit of x independently with probability δ. For a function $g : \{0,1\}^r \to \{0,1\}$, the noise-sensitivity of g at δ denoted $\mathrm{NS}_\delta(g)$ is defined as*

$$\mathrm{NS}_\delta(g) = \Pr_{x \in \{0,1\}^r, N_\delta(x)}[g(x) \neq g(N_\delta(x))].$$

The following Lemma is implicit in [BOKS06].

Lemma 7. *The code $g^{k,r}$ has distance $(\delta, \mathrm{NS}_\delta(g))$.*

Proof. Take two balanced messages $x, y \in \{0,1\}^k$ with $\Delta(x,y) = \delta$. If we pick a random $i \in [k]$,

$$\Pr[x_i = 1] = \Pr[y_i = 1] = \frac{1}{2}, \quad \Pr[x_i \neq y_i] = \delta.$$

Thus it follows that $\Delta(g^{k,r}(x), g^{k,r}(y)) = \mathrm{NS}_\delta(g)$. □

Note that Claim 7 of [BOKS06] about the list-decodability of $g^{k,r}$ can be derived from Lemma 7 by applying the Johnson bound. Another useful property of these codes is that nearby messages (even unbalanced) have similar encodings. This allows us to reason about the distance between the encodings of nearly-balanced messages by replacing them with nearby balanced messages.

Lemma 8. *For all $x, y \in \{0,1\}^k$ such that $\Delta(x,y) \leqslant \delta$, $\Delta(g^{k,r}(x), g^{k,r}(y)) \leqslant \delta r$.*

Proof. Pick a random index of $g^{k,r}$. Since x and y differ at no more than δk indices, the probability that sampling a random r-tuple gives different strings is bounded by δr, from which the claim follows. □

For each $r = 3^\ell$, the Recursive-Majority function $\mathrm{RecMaj} : \{0,1\}^r \to \{0,1\}$ is a function on r variables given by a depth-ℓ ternary tree of majority gates with the inputs at the leaves.

Fact 9 ([O'D04])**.** *There is a constant $\gamma \in (0,1)$, such that $\mathrm{NS}_{r^{-\gamma}}(\mathrm{RecMaj}) \geqslant \frac{1}{2} - r^{-\gamma}$.*

3 The Monotone Code construction

3.1 The expander-based construction

There is a standard way, originally due to [ABN+92], to use an expander graph to map a binary string into a string over a larger alphabet, such that a small Hamming distance between two binary strings is amplified into a much larger Hamming distance between their images.

Definition 7. *Given a string $f \in \{0,1\}^N$ and a k-regular bipartite (multi)-graph $G(A,B,E)$ with $|A| = |B| = N$ (we assume the elements of A and B are identified with $\{1,2,\ldots,N\}$ in some fixed order), the string $G(f) \in (\{0,1\}^k)^N$ is defined as follows. For $j \in B$, let $G(f)_j$, the j'th symbol of $G(f)$, be the vector obtained by concatenating the symbols of f corresponding to positions in the neighborhood of j in A in some fixed order. In other words, if $\Gamma_1(j), \ldots, \Gamma_k(j)$ are the k neighbors of $j \in B$, then*

$$G(f)_j = f_{\Gamma_1(j)} \circ f_{\Gamma_2(j)} \circ \cdots \circ f_{\Gamma_k(j)}.$$

In other words, we associate the vertices of the LHS with the message symbols. To each vertex on the RHS we associate the concatenation of the symbols of to its neighbors (considered in some fixed order). To reduce the alphabet size, we will further encode the k bits in each symbol of $G(f)$ by an "inner" code.

We can now state the construction of our code C. For some constant $\tau > 1$ that we fix later, we pick a graph $G(A, B, E)$ with $|A| = |B| = 2^n$, degree $k = \Theta((\log n)^\tau)$, and second largest eigenvalue at most $k^{1-\nu}$, as guaranteed by Theorem 6. We associate the message f of length N with the vertices on the LHS. We concatenate each symbol on the RHS (which lies in $\{0,1\}^k$) with the code $\mathrm{RecMaj}^{k,r}$ where $r = \Theta((\log n)^\rho)$ for any constant $\rho < 1$. It is clear from the choice of parameters that the encoding length is $M = Nk^r = 2^n(\log n)^{\tau(\log n)^\rho} = 2^{n(1+o(1))}$.

One can show that the distance of this code is close to $\frac{1}{2}$, as long as the messages are at distance at least $4r^{-\gamma}$ (here $\gamma > 0$ is the constant from Fact 9). We defer this proof to the appendix.

Lemma 10. *The code C has distance $(4r^{-\gamma}, \frac{1}{2} - 4r^{-\gamma})$.*

3.2 Deterministic Local Decoding

We analyze the following decoding procedure. Let R_j denote the portion of the received word corresponding to $C(f)_j$. We decode R_j to a balanced string y_j such that $\Delta_j = \Delta(\mathrm{RecMaj}^{k,r}(y_j), R_j)$ is minimized. We then define a confidence estimate $\beta_j \in [0,1]$ as

$$\beta_j = \max(1 - 4\Delta_j, 0).$$

Note that if Δ_j is small, then the confidence estimate is close to 1. As Δ_j approaches $\frac{1}{4}$, which is approximately half the relative distance of the inner code, β_j drops to 0.

After decoding the inner codes, for each vertex i on the LHS, we get a *vote* for the value of f_i from every vertex j in its neighborhood $\Gamma(i)$. We take a weighted majority of these votes, using the β_js are weights. Let $i \in A$, and let $\Gamma_0(i), \Gamma_1(i)$ denote the subset of $\Gamma(i)$ on the RHS which vote 0 and 1 respectively. We set $f_i = 1$ if

$$\sum_{j \in \Gamma_1(i)} \beta_j \geqslant \sum_{j \in \Gamma_0(i)} \beta_j \ ,$$

and $f_i = 0$ otherwise.

The decoding procedure outlined above is deterministic. To recover a bit f_i, we only need to decode the inner codes for vertices in $\Gamma(i)$. The size of $\Gamma(i)$ is bounded by $(\log n)^\tau$ and the time taken to decode each inner code is bounded by $(\log n)^{\tau(\log n)^\rho} = o(n)$ since $\rho < 1$. Thus the time needed to recover f_i is $o(n)$.

We now analyze the error-correction. Let f be a balanced message. Assume that an adversary corrupts $\Delta(R, C(f)) = \eta \leqslant \frac{1}{4} - 4r^{-\gamma/2}$ fraction of the indices. Also, let η_j denote the error-rate on the inner code for a vertex $j \in B$. We partition B into two sets, $\mathsf{Balanced} \subset B$ is the set of vertices j such that the bias of $G(f)_j$ is at most $\frac{1}{k^{\nu/2}}$, and its complement $\mathsf{UnBalanced}$.

Lemma 11. *We have $|\mathsf{UnBalanced}| \leqslant \frac{1}{k^\nu} N$.*

Proof. Take S to be the set of 1s on the RHS, so that $|S| = \frac{1}{2}N$. Let

$$U_1 = \left\{ j \in B \mid \mathrm{wt}(G(f)_j) > \frac{1}{2} + \frac{1}{k^{\nu/2}} \right\}, \ U_0 = \left\{ j \in B \mid \mathrm{wt}(G(f)_j) < \frac{1}{2} - \frac{1}{k^{\nu/2}} \right\}.$$

Note that $|E(S,U_1)| > (\frac{1}{2} + \frac{1}{k^{\nu/2}})k|U_1|$. Hence $|E(S,U_1)| - \frac{k|U_1|}{2} > k^{1-\nu/2}|U_1|$. Applying the expander mixing lemma,

$$k^{1-\nu/2}|U_1| \leqslant k^{1-\nu}\sqrt{|S||U_1|} \;\Rightarrow\; |U_1| \leqslant \frac{1}{2k^{\nu}}N.$$

A similar calculation for U_0 gives $|\mathsf{UnBalanced}| = |U_0| + |U_1| \leqslant \frac{N}{k^{\nu}}$. □

We note that for this lemma, $|S|$ need not be perfectly balanced, it follows that $|\mathsf{UnBalanced}| \leqslant O(\frac{1}{k^{\nu}}N)$ as long as the bias of f is $o(\frac{1}{k^{\nu/2}})$.

Thus $|\mathsf{Balanced}| \geqslant N(1 - \frac{1}{k^{\nu}})$. We further divide $\mathsf{Balanced}$ into two parts, Good and Bad. We say that $j \in \mathsf{Balanced}$ lies in Good if $\Delta(y_j, G(f)_j) \leqslant 2r^{-\gamma}$ else $j \in \mathsf{Bad}$.

Lemma 12. *We have*

$$\sum_{j\in\mathsf{Good}} \beta_j - \sum_{j\in\mathsf{Bad}} \beta_j \geqslant r^{-\gamma}N.$$

Proof. Fix a $j \in \mathsf{Balanced}$. Let x_j denote the balanced message so that $\Delta(x_j, G(f)_j) \leqslant \frac{1}{k^{\nu/2}}$. By Lemma 8,

$$\Delta(\mathrm{RecMaj}^{k,r}(x_j), C(f)_j) \leqslant \frac{r}{k^{\nu/2}}\ . \tag{1}$$

Case 1: $j \in \mathsf{Bad}$. In this case, we show that if vertex j has a high confidence β_j, then η_j needs to be large. Formally, we show that if $j \in \mathsf{Bad}$, then

$$\eta_j \geqslant \frac{1+\beta_j}{4} - 2r^{-\gamma}.$$

Since j is a bad vertex, $\Delta(y_j, G(f)_j) \geqslant 2r^{-\gamma}$, hence by the triangle inequality

$$\Delta(x_j, y_j) > 2r^{-\gamma} - \frac{1}{k^{\nu/2}} \geqslant r^{-\gamma} \quad (\text{since } r < k^{\nu/2})$$

Hence their encodings are far apart,

$$\Delta(\mathrm{RecMaj}^{k,r}(x_j), \mathrm{RecMaj}^{k,r}(y_j)) \geqslant \frac{1}{2} - r^{-\gamma}$$

which together with (1) gives

$$\Delta(C(f)_j, \mathrm{RecMaj}^{k,r}(y_j)) \geqslant \frac{1}{2} - r^{-\gamma} - \frac{r}{k^{\nu/2}} \geqslant \frac{1}{2} - 2r^{-\gamma}\ .$$

Now we consider two cases. If $\beta_j > 0$, then by definition

$$\Delta(R_f, \mathrm{RecMaj}^{k,r}(y_j)) = \frac{1-\beta_j}{4}$$

hence

$$\eta_j = \Delta(C(f)_j, R_j) \geqslant \frac{1}{2} - 2r^{-\gamma} - \frac{1-\beta_j}{4} = \frac{1+\beta_j}{4} - 2r^{-\gamma}.$$

When $\beta_j = 0$, we know that

$$\Delta(\mathrm{RecMaj}^{k,r}(x_j), R_j) \geqslant \Delta(\mathrm{RecMaj}^{k,r}(y_j), R_j) \geqslant \frac{1}{4}.$$

Together with (1), this gives

$$\eta_j = \Delta(C(f)_j, R_j) \geqslant \frac{1}{4} - \frac{r}{k^{\nu/2}} \geqslant \frac{1}{4} - 2r^{\gamma}$$

where the last inequality follows from $r^2 < k^{\nu/2}$.

Case 2: $j \in \mathsf{Good}$. We now show that if a good vertex has low confidence, then η_j is close to $\frac{1}{4}$. When $j \in \mathsf{Good}$, we have

$$\Delta(R_f, \mathrm{RecMaj}^{k,r}(x_j)) \;\geqslant\; \Delta(R_f, \mathrm{RecMaj}^{k,r}(y_j)) \;\geqslant\; \frac{1-\beta_j}{4}$$

and hence

$$\Delta(C(f)_j, R_j) \;\geqslant\; \frac{1-\beta_j}{4} - \frac{r}{k^{\nu/2}} \;\geqslant\; \frac{1-\beta_j}{4} - r^{-\gamma}.$$

Since the total fraction of errors is bounded by $\frac{1}{4} - 4r^{-\gamma/2}$,

$$\sum_{j \in \mathsf{Good}} \left(\frac{1-\beta_j}{4} - r^{-\gamma} \right) + \sum_{j \in \mathsf{Bad}} \left(\frac{1+\beta_j}{4} - 2r^{-\gamma} \right) \leqslant \left(\frac{1}{4} - 4r^{-\gamma/2} \right) N.$$

Rearranging terms, and using the fact that all but $\frac{1}{k^{\nu}}N$ vertices are balanced,

$$\sum_{j \in \mathsf{Good}} \beta_j - \sum_{j \in \mathsf{Bad}} \beta_j \geqslant N \left(2r^{-\gamma/2} - \frac{1}{k^{\nu}} \right) \geqslant r^{-\gamma/2} N$$

□

Thus overall, the total mass of the β_j's is higher on the good vertices than the bad vertices. Using the spectral properties of the graph, we can show that even locally, most vertices will receive more votes from good than bad vertices.

Lemma 13. *Define the set*

$$I_1 = \left\{ \; i \in A \;\middle|\; \sum_{j \in \Gamma(i) \cap \mathsf{Good}} \beta_j - \sum_{j \in \Gamma(i) \cap \mathsf{Bad}} \beta_j < \frac{r^{-\gamma/2}}{2} k \; \right\}$$

The $|I_1| \leqslant \frac{4r^{\gamma}}{k^{2\nu}} N$.

Proof. Define the vector $z = (z_1, \ldots, z_N)$ where $z_j = \beta_j$ for $j \in \mathsf{Good}$, $z_j = -\beta_j$ for $j \in \mathsf{Bad}$ and $z_j = 0$ otherwise. Define the vector χ to be the indicator of the set I_1. Let A_G be the adjacency matrix of G. Then

$$\chi^t A_G z = \sum_{i \in I_1} (A_G z)_i = \sum_{i \in I_1} \sum_{j \in \Gamma(i)} z_j = \sum_{i \in I_1} \left(\sum_{j \in \Gamma(i) \cap \mathsf{Good}} \beta_j - \sum_{j \in \Gamma(i) \cap \mathsf{Bad}} \beta_j \right) < \frac{r^{-\gamma/2}}{2} k |I_1|.$$

On the other hand, let us expand z and χ in the orthonormal basis of the eigenvectors $\{v_1, \ldots, v_N\}$ of A (with corresponding eigenvalues $k = \lambda_1 \geqslant \lambda_2 \geqslant \cdots \geqslant \lambda_N$ with $\lambda = \max\{\lambda_2, |\lambda_N|\}$) as: $z = \sum_N \zeta_\ell v_\ell$ and $\chi^T = \sum_\ell \xi_\ell v_\ell$. Note that $\xi_1 = |I_1|/\sqrt{n}$, and $\zeta_1 = (\sum_i z_i)/\sqrt{N} \geqslant r^{-\gamma}\sqrt{N}$. Now

$$\begin{aligned}
\chi^t A_G z &= \sum_\ell \xi_\ell \zeta_\ell \lambda_\ell \\
&= \xi_1 \zeta_1 d + \sum_{\ell \geqslant 2} \xi_\ell \zeta_\ell \lambda_\ell \\
&\geqslant \frac{|I_1|}{\sqrt{N}} \cdot r^{-\gamma/2}\sqrt{N} \cdot k - \lambda \langle \chi, z \rangle \\
&\geqslant r^{-\gamma/2} k |I_1| - \lambda \|\chi\| \|z\| \\
&\geqslant r^{-\gamma/2} k |I_1| - \lambda \sqrt{|I_1| N}.
\end{aligned}$$

We thus conclude

$$r^{-\gamma/2} k |I_1| - \lambda \sqrt{|I_1| N} < \frac{r^{-\gamma/2}}{2} k |I_1| \;\Rightarrow\; |I_1| < \frac{4r^\gamma}{k^{2\nu}} N.$$

□

Lemma 14. *The local decoding procedure $16r^{-\gamma/2}$-approximately decodes C up to distance $\frac{1}{4} - 4r^{-\gamma/2}$.*

Proof. We say that an edge is incorrect if the wrong symbol is sent to the LHS after decoding the inner code. We assume that all edges from vertices in Bad and vertices in UnBalanced are incorrect. In addition, each vertex in Good might have up to $2r^{-\gamma}$ incorrect edges. We will argue that even so, most of the vertices in A are decoded correctly.

Let $I_2 \subset A$ be the set of vertices which have at least $\frac{r^{-\gamma/2}}{4} k$ neighbors in UnBalanced. Since the total out-degree of the set UnBalanced is bounded by $k^{1-\nu} N$, we have

$$|I_2| \frac{r^{-\gamma/2}}{4} k \leqslant k^{1-\nu} N \;\Rightarrow\; |I_2| \leqslant \frac{4r^{\gamma/2}}{k^\nu} N.$$

Hence for any vertex in $A \setminus I_1 \cup I_2$,

$$\sum_{j \in \Gamma(i) \cap \mathsf{UnBalanced}} \beta_j \leqslant |\Gamma(i) \cap \mathsf{UnBalanced}| \leqslant \frac{r^{-\gamma/2}}{4} k.$$

$$\sum_{j \in \Gamma(i) \cap \mathsf{Good}} \beta_j - \sum_{j \in \Gamma(i) \cap \mathsf{Bad}} \beta_j \geqslant \frac{r^{-\gamma/2}}{2} k.$$

$$\text{Hence} \quad \sum_{j \in \Gamma(i) \cap \mathsf{Good}} \beta_j - \sum_{j \in \Gamma(i) \cap \mathsf{Bad}} \beta_j - \sum_{j \in \Gamma(i) \cap \mathsf{UnBalanced}} \beta_j \geqslant \frac{r^{-\gamma/2}}{4} k. \tag{2}$$

Finally, partition $\Gamma(i) \cap \mathsf{Good}$ into $\mathsf{Correct_i}$ and $\mathsf{Incorrect_i}$ based on whether or not the edge (i, j) is correct. The advantage of correct votes over incorrect votes at vertex i is given by

$$\begin{aligned}
\mathrm{adv}_i &\geqslant \sum_{j \in \mathsf{Correct_i}} \beta_j - \sum_{j \in \mathsf{Incorrect_i}} \beta_j - \sum_{j \in \Gamma(i) \cap \mathsf{Bad}} \beta_j - \sum_{j \in \Gamma(i) \cap \mathsf{UnBalanced}} \beta_j \\
&= \sum_{j \in \mathsf{Good}} \beta_j - 2 \sum_{j \in \mathsf{Incorrect_i}} \beta_j - \sum_{j \in \Gamma(i) \cap \mathsf{Bad}} \beta_j - \sum_{j \in \Gamma(i) \cap \mathsf{UnBalanced}} \beta_j \\
&\geqslant \sum_{j \in \mathsf{Good}} \beta_j - \sum_{j \in \Gamma(i) \cap \mathsf{Bad}} \beta_j - \sum_{j \in \Gamma(i) \cap \mathsf{UnBalanced}} \beta_j - 2|\mathsf{Incorrect_i}|
\end{aligned}$$

Hence for an incorrect decoding of i, we must have $|\mathsf{Incorrect_i}| \geqslant \frac{r^{-\gamma/2}}{4}k$. Denote the set of such vertices by I_3. The number of incorrect edges leaving Good is bounded by $(2r^{-\gamma})kN$. Thus,

$$|I_3|\frac{r^{-\gamma/2}}{4}k \;\leqslant\; (2r^{-\gamma})kN \;\Rightarrow\; |I_3| \leqslant 8r^{-\gamma/2}.$$

Thus overall, the fraction of vertices that are wrongly decoded is bounded by

$$\delta \leqslant \frac{|I_1| + |I_2| + |I_3|}{N} \leqslant 8r^{-\gamma/2} + \frac{4r^{\gamma/2}}{k^{\nu}} + \frac{4r^{\gamma}}{k^{2\nu}} \leqslant 16r^{-\gamma/2}$$

for sufficiently large k. □

3.3 Proofs of our main results

Theorem 4 now follows by a suitable choice of parameters. Recall that $r = (\log n)^{\rho}$ for some constant $\rho < 1$. We set $k = (\log n)^{\tau}$ for some large constant τ so that $r^2 < k^{2\nu}$, which ensures that error terms involving k are bounded by $r^{-\gamma/2}$. Thus our algorithm recovers balanced messages with error at most $(\log n)^{-\mu}$ up to error rate $\frac{1}{4} - (\log n)^{-\mu}$ for some constant μ and large n.

To prove Theorem 1, let f be a balanced function which is $1 - (\log n)^{-\mu}$ hard for P. Assume that there is a deterministic polynomial time algorithm $\mathcal{A}$ that computes $G(f)$ within accuracy $\frac{3}{4} + (\log n)^{-\mu}$. Then using the local decoding procedure in conjunction with $\mathcal{A}$ gives a deterministic polynomial time algorithm $\mathcal{A}'$ which computes f correctly on $1 - (\log n)^{-\mu}$ fraction of inputs, contradicting our assumption about its hardness. We can relax the condition that f is perfectly balanced. Indeed, our proof goes through even if f has bias $o(k^{-\nu/2})$. In this case, we can still use Lemma 11 to show that for most vertices $j \in B$, $G(f)_j$ has bias bounded by $k^{-\nu/2}$. The rest of the proof goes through unchanged.

To eliminate the balance hypothesis and prove Theorem 2, we use the padding argument used by Trevisan [Tre05, Lemma 7]. By suitable choice of parameters ρ, τ, assume that $k^{\nu/2} = o(\log n)$. We consider inputs lengths $n = 2^t$. We use f on $\{0,1\}^n$ to define a padded version f' on input lengths in the interval $[2^t, 2^t + t]$. On one of these lengths, f' has bias bounded by $O(1/t)$ and has about the same hardness as f. However, to get an algorithm for f from an algorithm for f', we need $O(\log t) = O(\log \log n)$ advice bits to tell us which *length* in the interval $[2^t, 2^t + t]$ to use, and $O(\log n)$ advice bits to tell us an appropriate suffix to pad the input for f with. The latter advice string is not needed in the BPP setting as one can pad the input with a random string. Since the argument is similar to ones used in [O'D04, Tre05], we omit the details.

To prove Theorem 3, we take an expander graph with degree $k = n^{O(1)}$. We concatenate it with a binary code that gives codewords of length $k^{O(1)}$ and can be unique decoded up to error rate $\frac{1}{4} - n^{-O(1)}$ in time poly(k). (There are several choices for such a code, for example Reed-Solomon concatenated with a Hadamard code would work.) For decoding, we define the confidence parameter β_j for each vertex as before. A vertex is now good if it recovers the message exactly and bad otherwise. We can prove statements similar to Lemma 12 showing that good vertices have more confidence overall, and Lemma 13 showing that this is also true for the neighborhoods of most vertices on the LHS. This concludes the analysis, since unlike in the monotone case, we do not have to worry about unbalanced vertices or wrong votes from good vertices. This gives a deterministic local decoder that runs in time poly(n) and can $\frac{1}{\text{poly}(n)}$-approximately decode codewords up to error rates of $\frac{1}{4} - \frac{1}{\text{poly}(n)}$, which implies Theorem 3.

4 Limitations to Hardness Amplification via Monotone Codes

For EXP, it is possible to show optimal average case hardness starting from worst-case hardness assumptions. Roughly, amplifying worst-case hardness to $(1-\delta)$-hardness requires codes that are exactly locally decodable up to error rate δ. The crux of this reduction for EXP is the existence of codes which can be locally list-decoded (exactly) up to $\frac{1}{2} - o(1)$ errors (the notion of local list-decoding is subtle and needs to be defined carefully, see [STV01]).

In contrast, for NP we do not expect to amplify worst case hardness to even $1 - 1/\text{poly}(n)$-hardness in a black-box manner, assuming certain complexity-theoretic hypotheses [BT06]. This suggests limitations to the error-correcting properties of monotone codes that can be used for amplification, namely that they cannot be decoded exactly. Our goal in this section is to prove these limitations without any complexity theoretic assumptions. We note that our lower bounds are purely combinatorial in nature, hence they apply also to randomized and non-uniform decoders.

[BOKS06] noted that there exist monotone codes that have good distance. We observe that additionally, they can have efficient and exact local decoding algorithms.

Lemma 15. *For any $\eta < \frac{1}{4}$, there exist monotone codes that can be locally decoded exactly up to error rate η.*

Proof. Let $\mathrm{C} : \{0,1\}^N \to \{0,1\}^M$ be a (not necessarily monotone) code which is locally decodable up to error rate η. We now define a monotone code $\mathrm{C}' : \{0,1\}^N \to \{0,1\}^M$ as follows: on balanced messages, $\mathrm{C}'(x) = \mathrm{C}(x)$. For each $i \in [M]$ and balanced string $x \in \{0,1\}^N$, this defines a function $\mathrm{C}'_{\mathrm{i}}(x)$ taking values in $\{0,1\}$. We can now extend C'_{i} to a monotone function that is defined on the entire hypercube $\{0,1\}^N$. The local decoder for C' is just the local decoder for C. On any balanced message, it is guaranteed to recover from up to η fraction of errors. □

The reason why this construction will not give hardness amplification is because we do not have an upper bound on the complexity of encoding. While it is certainly sufficient if the encoding is local, this is a strong condition and is in fact restrictive. Healy *et al.*[HVV06] show that one cannot hope for better that $\frac{1}{2} + \frac{1}{\text{poly}(n)}$ hardness via such reductions. Healy *et al.* [HVV06] observe that for black-box hardness amplification, the monotone code C must be such that each function C_i has small certificate complexity. Using this, they are able to break the $\frac{1}{2} + \frac{1}{\text{poly}(n)}$ barrier.

Definition 8. *For $S \subseteq N$ and $x \in \{0,1\}^N$, let x_S denote the projection of x onto the coordinates in S. For a Boolean function $f : \{0,1\}^N \to \{0,1\}$, and $x \in f^{-1}(1)$, we say that x_S is a certificate*

for x if $f(y) = 1$ for all $y \in \{0,1\}^N$ such that $y_S = x_S$. We define

$$N_1(f) = \max_{x \in f^{-1}(1)} \min |x_S|$$

over all certificates x_S for x.

Alternately $N_1(f)$ is the non-deterministic complexity of f, it is the maximum number of bits of x that a prover has to reveal to a verifier to convince her that $f(x) = 1$. It is easy to see that if f is monotone, then a minimal certificate for $x \in f^{-1}(1)$ can only consist of 1's. For a monotone code $\mathrm{C} : \{0,1\}^N \to \{0,1\}^M$, we define

$$N_1(\mathrm{C}) = \max_{i \in [M]} N_1(\mathrm{C}_i).$$

We will study black-box hardness amplification, where $N_1(\mathrm{C})$ is bounded by $\mathrm{poly}(n)$ and $N = 2^n$. This still allows each C_i to depend on all the 2^n message bits. Indeed this assumption that each bit can be encoded with small non-deterministic complexity seems to be the natural definition for amplification within NP. We will show that the condition $N_1(\mathrm{C}) = \mathrm{poly}(n)$ places bounds on the distance of the resulting monotone code.

Lemma 16. *A monotone code with $N_1(\mathrm{C}) \leqslant t$ can be exactly decoded from at most a fraction $\delta = \frac{t}{N}$ of errors.*

Proof. Let $x \in \{0,1\}^N$ be a random balanced message. Let A_0 and A_1 denote the subsets of indices where x_i is 0 and 1 respectively. Let $y \in \{0,1\}^N$ be generated from x by flipping a random bit from each of A_0 and A_1. It is clear that y is also a random balanced message.

For each $i \in [M]$, we bound $\Pr[\mathrm{C}_i(x) \neq \mathrm{C}_i(y)]$. Note that

$$\begin{aligned} \Pr[\mathrm{C}_i(x) \neq \mathrm{C}_i(y)] &= \Pr[\mathrm{C}_i(x) = 0,\ \mathrm{C}_i(y) = 1] + \Pr[\mathrm{C}_i(x) = 1,\ \mathrm{C}_i(y) = 0] \\ &= 2\Pr[\mathrm{C}_i(x) = 1,\ \mathrm{C}_i(y) = 0] \end{aligned}$$

by symmetry. By monotonicity, flipping a bit from A_0 cannot cause C_i to change from 1 to 0. Let S be a certificate that $\mathrm{C}_i(x) = 1$ of size at most t. The bit flipped from A_1 must belong to S, else $\mathrm{C}_i(y) = \mathrm{C}_i(x) = 1$ since $y_S = x_S$. The probability of this event is bounded by $\frac{t}{N}$. Thus,

$$\Pr[\mathrm{C}_i(x) \neq \mathrm{C}_i(y)] \leqslant \frac{2t}{N}.$$

By linearity of expectation,

$$\Delta(\mathrm{C}(x), \mathrm{C}(y)) \leqslant \frac{2t}{N}.$$

Thus there exists a pair of balanced messages x, y satisfying this condition. Thus any unique decoder for C can tolerate at most an error rate of $\frac{t}{N}$, which is exponentially small in n. □

The above lemma rules out getting $1 - \frac{1}{\mathrm{poly}(n)}$ hardness starting from worst-case hardness via black-box reductions. One can use a similar argument to also bound from below the error of any *approximate* decoder that decodes from a constant fraction of error.

Lemma 17. *Let C be a monotone code with $N_1(\mathrm{C}) \leqslant t$. Any local decoder that can tolerate η fraction of errors can only recover balanced messages $\frac{\eta}{2t}$-approximately.*

Proof. Define $x \in \{0,1\}^N, A_0 \subset [N], A_1 \subset [N]$ as before. Pick random subsets $B_0 \subset A_0$ and $B_1 \subset A_1$ of size $\frac{\eta}{2t}N$ each and flip these bits to obtain y. As before, we have

$$\Pr[\mathrm{C}_i(x) \neq \mathrm{C}_i(y)] = 2\Pr[\mathrm{C}_i(x) = 1,\ \mathrm{C}_i(y) = 0]$$

Let S be a certificate that $\mathrm{C}_i(x) = 1$ of size at most t. Some bit from S must lie in B_1 in order for $\mathrm{C_i}(y)$ to equal 0. It is easy to show that the expected size of $B_1 \cap S$ is at most η, hence the probability that it is non-empty is bounded by η. Hence using linearity of expectation,

$$\Pr[\mathrm{C}_i(x) \neq \mathrm{C}_i(y)] \leqslant 2\eta \ \Rightarrow\ \Delta(\mathrm{C}(x), \mathrm{C}(y)) \leqslant 2\eta$$

Thus there exist two balanced messages x, y such that

$$\Delta(x, y) = \frac{\eta}{t},\quad \Delta(\mathrm{C}(x), \mathrm{C}(y)) \leqslant 2\eta.$$

Hence there exists a received word $R \in \{0,1\}^M$ such that

$$\Delta(R, \mathrm{C}(x)) \leqslant \eta,\quad \Delta(R, \mathrm{C}(y)) \leqslant \eta.$$

Now consider the output of the decoder on input R. The decoded message is always at distance at least $\frac{\eta}{2t}$ from one of x and y. □

This lemma shows that to achieve $(1-\eta)$-hardness for any constant η by a black-box reduction, one has to start by assuming $1 - \frac{1}{\mathrm{poly}(n)}$ hardness.

5 Conclusions

The reason we are unable to amplify from $(1 - \frac{1}{\mathrm{poly}(n)})$-hardness for NP is due to the fact that we do not know anything but a brute-force exponential decoder for the inner monotone code that can correct a linear fraction of errors. In order to prove such a result via our concatenation scheme, we would need an inner monotone code with a deterministic decoder that on messages of size k, can correct up to $\frac{1}{4}$ fraction of errors with accuracy $k^{-\gamma}$ for some constant $\gamma > 0$ in $\mathrm{poly}(k)$ time. Of course, we would need some additional properties of the inner code (like Lemma 8) to deal with imbalance in the messages.

In the other direction, improving the hardness to $\frac{1}{2} + o(1)$ from $\frac{3}{4} + o(1)$ calls for a deterministic, local *list-decoder* that can correct a $\frac{1}{2} - \varepsilon$ fraction of errors approximately. We are unaware of any such decoders for any non-trivial code, say even the Hadamard code. Indeed, it would be interesting to improve Theorem 3 to get deterministic hardness amplification up to $\frac{1}{2} + o(1)$ even for PSPACE.

Acknowledgments

We are thankful to Dang-Trinh Huynh-Ngoc for several discussions during initial stages of this work. We thank Valentine Kabanets and Rahul Santhanam for useful comments, and Rahul again for pointing us to [GW00].

References

[ABN+92] N. Alon, J. Bruck, J. Naor, M. Naor, and R. M. Roth. Construction of asymptotically good low-rate error-correcting codes through pseudo-random graphs. *IEEE Transactions on Information Theory*, 38(2):509–516, 1992.

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

[BOKS06] J. Buresh-Oppenheim, V. Kabanets, and R. Santhanam. Uniform hardness amplification in NP via monotone codes. ECCC TR06-154, 2006.

[BT06] Andrej Bogdanov and Luca Trevisan. On worst-case to average-case reductions for NP problems. *SIAM Journal of Computing*, 36(4):1119–1159, 2006.

[For66] G. D. Forney. Generalized minimum distance decoding. *IEEE Transactions on Information Theory*, 12:125–131, 1966.

[GI01] V. Guruswami and P. Indyk. Expander-based constructions of efficiently decodable codes. In *Proc.* 42^{nd} *IEEE Symposium on Foundations of Computer Science (FOCS'01)*, pages 658–667, 2001.

[GI05] V. Guruswami and P. Indyk. Linear-time encodable/decodable codes with near-optimal rate. *IEEE Transactions on Information Theory*, 51(10):3393–3400, 2005.

[GK06] V. Guruswami and V. Kabanets. Hardness amplification via space-efficient direct products. In *Proc. of the 7th Latin American Symposium on Theoretical Informatics (LATIN'06)*, pages 556–568, 2006. Citations refer to journal version to appear in *Computational Complexity*.

[GNW95] O. Goldreich, N. Nisan, and A. Wigderson. On Yao's XOR-Lemma. ECCC TR95-050, 1995.

[GW00] O. Goldreich and A. Wigderson. On pseudorandomness with respect to deterministic observers. *ICALP Satellite Workshops*, pages 77–84, 2000.

[HLW06] S. Hoory, N. Linial, and A. Wigderson. Expander graphs and their applications. *Bull. Amer. Math Soc.*, 43:439–561, 2006.

[HVV06] A. Healy, S. Vadhan, and E. Viola. Using nondeterminism to amplify hardness. *SIAM Journal on Computing*, 35(4):903–931, 2006.

[IJK06] R. Impagliazzo, R. Jaiswal, and V. Kabanets. Approximately list-decoding direct product codes and uniform hardness amplification. In *Proc.* 47^{th} *Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*, pages 187–196, 2006.

[Imp95] R. Impagliazzo. Hard-core distributions for somewhat hard problems. In *Proc.* 36^{th} *Annual Symposium on Foundations of Computer Science (FOCS'95)*, pages 538–545. IEEE Computer Society, 1995.

[IW97] R. Impagliazzo and A. Wigderson. P = BPP if E requires exponential circuits: derandomizing the XOR lemma. In *Proc.* 27^{th} *annual ACM symposium on Theory of computing (STOC'97)*, pages 220–229, 1997.

[IW98] R. Impagliazzo and A. Wigderson. Randomness vs. time: De-randomization under a uniform assumption. In *Proc.* 39^{th} *Annual Symposium on Foundations of Computer Science (FOCS'98)*, pages 734–743, 1998.

[O'D04] R. O'Donnell. Hardness amplification within NP. *Journal of Computer and System Sciences*, 69(1):68–94, 2004.

[STV01] M. Sudan, L. Trevisan, and S. Vadhan. Pseudorandom generators without the XOR lemma. *Journal of Computer and System Sciences*, 62 (2):236–266, 2001.

[Tre03] L. Trevisan. List-decoding using the XOR lemma. In *Proc.* 44^{th} *Annual IEEE Symposium on Foundations of Computer Science (FOCS'03)*, page 126, 2003.

[Tre05] L. Trevisan. On uniform amplification of hardness in NP. In *Proc.* 37^{th} *annual ACM symposium on Theory of computing (STOC'05)*, pages 31–38, 2005.

[TV02] L. Trevisan and S. Vadhan. Pseudorandomness and average-case complexity via uniform reductions. In *IEEE Conference on Computational Complexity (CCC'02)*, pages 129–138, 2002. Citations refer to journal version to appear in *Computational Complexity.*

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

A Analyzing the Distance

We will need Lemma 11 showing that for a balanced message f and most vertices $j \in B$, $G(f)_j$ is unbiased. Next we argue that if f and g are both balanced and at sufficiently large distance, then $G(f)_j$ and $G(g)_j$ are also at a good distance for most $j \in B$.

Lemma 18. *Let f and g be balanced, such that $\Delta(f,g) = \delta > \frac{1}{k^\nu}$. Let T be the set of vertices $j \in B$ such that $\Delta(G(f)_j, G(g)_j) < \frac{\delta}{2}$. Then $|T| \leqslant \frac{4}{k^\nu} N$.*

Proof. Let $S \subset A$ denote the set of vertices on the LHS where f and g differ, so that $|S| \geqslant \delta N$. With T as above, we get $|E(S,T)| \leqslant \frac{\delta}{2} k|T|$, so that

$$\delta k|T| - |E(S,T)| > \frac{\delta}{2} k|T|.$$

By the expander mixing lemma,

$$\frac{\delta}{2} k|T| \leqslant \lambda\sqrt{\delta N|T|} \;\Rightarrow\; |T| < \frac{4}{k^{2\nu}\delta} N \leqslant \frac{4}{k^\nu} N.$$

□

We can now prove Lemma 10 stating that the code C has distance $(4r^{-\gamma}, \frac{1}{2} - 4r^{-\gamma})$.

Proof. We consider vertices $j \in B$ so that $G(f)_j$ and $G(g)_j$ have small bias and good distance, and call such vertices *nice*. Lemmas 11 and 18 imply that this is a $1 - \frac{5}{k^\nu}$ fraction of all vertices. Fix one such vertex j.

Since $G(f)_j$ and $G(g)_j$ have bias at most $\frac{1}{2} + \frac{1}{k^{\nu/2}}$, there are a balanced string x and y such that

$$\begin{aligned} \Delta(x, G(f)_j) < \frac{1}{k^{\nu/2}}, \quad \Delta(\mathrm{RecMaj}^{k,r}(x), C(f)_j) < \tfrac{r}{k^{\nu/2}} \\ \Delta(y, G(g)_j) < \frac{1}{k^{\nu/2}}, \quad \Delta(\mathrm{RecMaj}^{k,r}(y), \quad C(g)_j) < \tfrac{r}{k^{\nu/2}} \end{aligned} \tag{3}$$

But since $\Delta(G(f)_j, G(g)_j) \geqslant 2r^{-\gamma}$, we have

$$\Delta(x, y) \geqslant 2r^{-\gamma} - \frac{2}{k^{\nu/2}} > r^{-\gamma} \;\Rightarrow\; \Delta(\mathrm{RecMaj}^{k,r}(x), \mathrm{RecMaj}^{k,r}(y)) \geqslant \frac{1}{2} - r^{-\gamma}. \tag{4}$$

By Equations (3), (4) and the triangle inequality,

$$\Delta(C(f)_j, C(g)_j) \geqslant \frac{1}{2} - r^{-\gamma} - \frac{2r}{k^{\nu/2}} > \frac{1}{2} - 2r^{-\gamma}.$$

Since this holds for every nice vertex $j \in B$, and all but a fraction $O(\frac{1}{k^\nu})$ of vertices are nice, for large enough k

$$\Delta(C(f), C(g)) \geqslant \frac{1}{2} - 4r^{-\gamma}.$$

□

ECCC ISSN 1433-8092

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