Completeness amplification of PCP's and its applications

Prasant Gopal* Dana Moshkovitz †

June 22, 2012

Abstract

We construct the first PCP verifier that performs linear tests and has polynomially small completeness error simultaneously with logarithmic randomness, constant number of queries, constant alphabet and constant soundness error. Prior to our work, the smallest completeness error that was known in such circumstances was an arbitrarily small constant.

We use the verifier we construct to study Max-3Lin and Max-3Sat around their approximability thresholds. Specifically, we show that Max-3Lin and Max-3Sat become NP-hard to approximate at $1/2 + 1/(\log n)^\alpha$ and $7/8 + 1/(\log n)^\alpha$, respectively, for some $\alpha > 0$. The bottleneck for achieving an optimal (up to the constant α) hardness of $1/2 + 1/n^\alpha$ and $7/8 + 1/n^\alpha$ is in the technique for soundness amplification of projection games. If the technique improves in the future to give polynomially small soundness error, the $1/(\log n)^\alpha$ could be replaced with $1/n^\alpha$.

*prasant@csail.mit.edu, CSAIL, MIT
†dmoshkov@csail.mit.edu, CSAIL, MIT

1 Introduction

The Probabilistically Checkable Proofs (PCP) Theorem shows how to efficiently transform any mathematical proof to a format that can be probabilistically verified by reading only a constant number of locations. A PCP verifier has a number of parameters including the number of queries, the randomness, the alphabet, and the error parameters. There can be two types of errors: the verifier may reject given a correct statement and proof (completeness error), or it may accept given an incorrect statement (soundness error). If the verifier never rejects when given a correct statement and proof, we say that it has *perfect completeness*.

The basic PCP theorem was established through the works of Arora and Safra [AS98]; and Arora et al. [ALM+98]. They were inspired and(or) were greatly helped by a series of works including those of Feige et al. [FGL+96]; Lund et al. [LFKN90]; Babai et al. [BFL91, BFLS91]; Rubinfeld and Sudan [RS96]. The PCP Theorem states that every language in NP has a verifier that on input size n, uses $O(\log n)$ random bits, makes two queries to a proof over constant sized-alphabet; where the answer to the first query determines at most one satisfying answer to the second query (this is known as the "projection property"). The verifier always accepts a correct proof, and rejects any incorrect statement with a constant probability. The PCP theorem is the basis of essentially all hardness of approximation results known today. For more information about it, see [Din10].

Decreasing the error in PCP is the subject of considerable research. This is motivated both by the intrinsic interest in trustworthy verifiers, and by the use of PCP for hardness of approximation, in which PCPs with low error play a crucial role. Since there are PCP constructions with perfect completness, most of the work concentrated on decreasing the soundness error [Raz98, RS97, AS03, DFK+99, MR10, IKW09, DM11].

For certain verifiers, one cannot achieve perfect completeness. One example is unique verifiers, where the verifier makes two queries, and the answer to any one query determines uniquely a satisfying answer to the other query. Another example is linear verifiers, where the alphabet is a finite field, and the verifier performs linear tests over the field. Both types of verifiers were discovered to be very useful for hardness of approximation. The first gives rise to the host of UGC-based (UGC stands for Unique Games Conjecture [Kho02]; see the survey [Kho10]) hardness results we know today. The second allows using the Hadamard code in lieu of the long code [Kho01].

In this paper, we construct the first linear verifier with polynomially small completeness error:

Theorem 1.1. *There is a constant $\alpha > 0$, such that every NP language L has a linear PCP verifier with the projection property, such that on input size n, the verifier uses $O(\log n)$ random bits, queries a proof over a constant-size alphabet, has completeness error $1/n^{\alpha}$ and has soundness error 0.9.*

Given this theorem, it is possible to apply a soundness amplification technique and get a linear verifier with low soundness error in addition to low completeness error. Soundness amplification is done by testing many tests in parallel, and thus increases the rejection probability of the verifier. It cannot be performed without low completeness error. By the best soundness amplification technique known today for verifiers with the projection property [MR10] (which can be shown to preserve linearity), we deduce a statement similar to Theorem 1.1 but with lower, poly-logarithmically small, soundness error:

Corollary 1.1.1. *There is a constant $\alpha > 0$, such that every NP language L has a linear PCP verifier with the projection property, such that on input size n, the verifier uses $O(\log n)$ random bits, queries a proof over poly(n)-size alphabet, has completeness error $1/n^{\alpha}$ and has soundness error $1/(\log n)^{\alpha}$.*

One could hope for polynomially-small soundness error. The logarithmically small soundness error is due to the technique of [MR10]. We note that the alphabet of the proof must grow to allow lower soundness error, and this is the reason for the difference in the alphabet size between Theorem 1.1 and Corollary 1.1.1.

We use Corollary 1.1.1 to study the approximability of Max-3Lin and Max-3Sat around their thresholds, 1/2 and 7/8, respectively. For both problems, a simple random assignment algorithm gives an approximation up to the threshold. Moreover, for any constant larger than the threshold, Håstad proved that the problems become NP-hard to approximate [Hås01]. Moshkovitz and Raz refined the result to show that for an additive $1/(\log\log n)^\alpha$ beyond the threshold ($\alpha > 0$ is some constant), the problem has a nearly-exponential time lower bound, assuming Sat requires exponential time [MR10]. We explore the approximability of the problems in this window of width $1/(\log\log n)^\alpha$ beyond the threshold. This window is quite wide for practical values of n. Moreover, recent findings for other problems (e.g., [ABS10] for Unique-Games) reveal that sub-exponential time algorithms sometimes exist around the thresholds.

We prove that the window of efficient approximability is in fact much narrower than $1/(\log\log n)^\alpha$:

Theorem 1.2. *There is a constant $\alpha > 0$ for which it is NP-hard to approximate Max-3Lin to within a factor better than $1/2 + 1/(\log n)^\alpha$, and Max-3Sat to within a factor better than $7/8 + 1/(\log n)^\alpha$.*

Moreover, if there is a technique for projection soundness amplification that achieves polynomially small soundness error rather than logarithmically small soundness error, our methods yield hardness for Max-3Lin and Max-3Sat up to $1/2+1/n^\alpha$ and $1/2+1/n^\alpha$ for some constant $\alpha > 0$, respectively. Up to the constant α, this matches the polynomial-time approximation of $1/2+1/\sqrt{n}$ achieved by Håstad for Max-3Lin [Hås00].

1.1 Previous Work

The only previous work on completeness amplification that the authors are aware of is by Khot and Ponnuswami [KP06]. They construct PCP verifiers with arbitrarily small constant completeness error, and apply their construction to yield stronger hardness of approximation results for Max-Clique, Chromatic-Number and Min-3Lin-Deletion. Their constructions have a polynomial blow-up, where the exponent depends on the completeness error. Because of this exponent, they only achieve constant completeness error, and do not rule out sub-exponential algorithms.

2 Tensoring

We start by introducing the notion of Hamming weight and its corresponding metric the Hamming distance. Informally, the Hamming weight of block length n over an alphabet $\Sigma = \{0,1\}$ is the number of coordinates that are *not* zero. For any $\mathbf{x} \in \Sigma^n$, the hamming weight of $\mathbf{x}$ is the number of nonzero coordinates of $\mathbf{x}$, also known by $wt(\mathbf{x})$. The weight function is a norm and the correponding metric $d(\mathbf{x},\mathbf{y})$ is the weight of difference between $\mathbf{x}$ and $\mathbf{y}$, that is, $wt(\mathbf{x}+\mathbf{y})$. It is often useful to normalize the Hamming wight (resp. distance) w.r.t. n. From now on, we use normalized Hamming weight, which is $\mathrm{wt}(\mathbf{x})n$. Also, we use bold face small case letters $\mathbf{x}$, $\mathbf{y}$ to denote column vectors and upper case bold letters like $\mathbf{A}$, $\mathbf{X}$ to denote matrices. We use the notation $(\cdot\ )^T$ to denote the transpose of the corresponding vector or matrix.

Claim 2.1. *For every $\mathbf{x}$, $\mathbf{y}$, $\mathbf{z}$, $d(\mathbf{x},\mathbf{y}) = d(\mathbf{x}+\mathbf{y},\mathbf{y}+\mathbf{z})$.*

Proof. The key observation is that the Hamming distance of two vectors $\mathbf{x}$, $\mathbf{y}$ remain unchanged when a fixed vector $\mathbf{z}$ is added to $\mathbf{x}$, $\mathbf{y}$. Specifically, say $\mathbf{x}$, $\mathbf{y}$ agree on a coordinate i, then they continue to remain so even after adding $\mathbf{z}$ to both $\mathbf{x}$, $\mathbf{y}$. □

Let $\mathcal{L}$, $\mathcal{L} \equiv \mathbf{Ax} + \mathbf{y} = \mathbf{0}$, be a linear system with m equations over n variables. We use the notation UNSAT($\mathcal{L}$, $\mathbf{x}$) to denote the fraction of unsatisfied equations in $\mathcal{L}$ when its variables are assigned to $\mathbf{x}$. Note that, this is exactly same as the Hamming weight of $\mathbf{Ax} + \mathbf{y}$. We define the UNSAT value of $\mathcal{L}$ as follows.

Definition 1 (UNSAT($\mathcal{L}$))**.** *The UNSAT value of $\mathcal{L}$ is defined to be minimum fraction of unsatisfied equations over all possible assignments to the variables.*

Proposition 2.2. *The following are equivalent.*

$$UNSAT(\mathcal{L}) = \min_{\mathbf{x}} d(\mathbf{Ax}, \mathbf{y}) = \min_{\mathbf{x}} d(\mathbf{Ax} + \mathbf{y}, \mathbf{0}) \tag{1}$$

Proof. Recall that, UNSAT value of $\mathcal{L}$ is the minimum Hamming weight of $\mathbf{Ax}$ $+\mathbf{y}$ over all $\mathbf{x}$. This settles that the first and the last expression are the same. Now, subtract $\mathbf{y}$ from both terms of $d(\mathbf{Ax} + \mathbf{y}, \mathbf{0})$. We now invoke Claim 2.1 to establish that shifts preserve Hamming distances. And this establishes the truth of (1). □

We now wish to quantitatively establish the relation between the satisfiability of a linear system $\mathcal{L}$ over variables $X = \{x_1, x_2 \dots x_n\}$, and its tensor which is defined as follows. It involves taking the products of all possible pairs of linear equations from $\mathcal{L}$, and then linearizing them via replacing terms of the form $x_i \cdot x_j$ with x_{ij} and x_i with x_{ii} (since, we are working in $\mathbb{F}_2$). The claim that we make about the aforementioned transformation is that the UNSAT values of $\mathcal{L}$ and its tensor are very closely related. In what follows, we make this connection explicit. We start by defining the tensor of n-dimensional vector $\mathbf{x}$ over a binary alphabet.

Definition 2 (Tensoring)**.** *We define the tensor of a vector $\mathbf{x} \in \{0,1\}^n$ as $\mathbf{x}^{\otimes 2} : [n] \times [n] \to \mathbf{x}_i \cdot \mathbf{x}_j$.*

We also extend the above definition to matrices by treating them as vectors in an obvious way.

2.1 Tensoring Linear Systems

Formally, the tensor of a linear system $\mathcal{L}$, denoted by $\mathcal{L}^{\otimes 2}$, is defined as the following.

Definition 3. *Given a linear system $\mathcal{L}$, $\mathcal{L} \equiv \mathbf{Ax} + \mathbf{y} = \mathbf{0}$, over $\{x_1, \dots x_n\}$ variables and $[m]$ equations. The tensor of $\mathcal{L}$ is defined as $(\mathbf{Ax} + \mathbf{y}) \cdot (\mathbf{Ax} + \mathbf{y})^T$. We then linearize the quadratic system thus obtained by replacing a variable $x_i \cdot x_j$ by x_{ij} for $i, j \in [n]$. We denote this linearized system by $\mathcal{L}^{\otimes 2}$.*

Proposition 2.3 (Tensoring vs Solvability)**.** *For every linear system $\mathcal{L}$ and its tensor $\mathcal{L}^{\otimes 2}$ and every $\mathbf{x}$,*

$$UNSAT(\mathcal{L}^{\otimes 2}, \mathbf{x}^{\otimes 2}) \;=\; UNSAT(\mathcal{L}, \mathbf{x})^2.$$

Proof. Suppose, we are given an assignment $\pi : [n] \to \{0,1\}$ that satisfies $(1-\rho)$-fraction of $\mathcal{L}$, we claim that the assignment $\tilde{\pi} : [n] \times [n] \to \pi(i) \times \pi(j)$ satisfies at least $(1-\rho^2)$-fraction of $\mathcal{L}^{\otimes 2}$. It follows from the definition of $\mathcal{L}^{\otimes 2}$ that the only case an equation $\tilde{y}_{ij}$ from $\mathcal{L}^{\otimes 2}$ is unsatisfied iff π does not satisfy both y_i, y_j in $\mathcal{L}$. By a simple counting argument, we can check that there only ρ^2-fraction of such pairs. And thus the proposition holds. □

Proposition 2.3 was essentially restating a well known fact. In this work, we would like to use these operations to construct a PCP. In any PCP theorem, the challenging part to infer some structure if we accept certain proof. Informally, this corresponds to establishing that if we accept a purported proof, we must be able to decode an assignment that (approximately) achieves the claims made by the purported proof. If we would like to use tensoring to any end in PCP's, we would need a certain converse of Proposition 2.3. In other words, we would like to state something along the lines that the satisfiability of the tensored system is well behaved. We warm up via some nice, interesting observation that is exclusive to $\mathbb{F}_2$.

Claim 2.4. *For every* $\mathbf{x}$*, every row (column) of* $\mathbf{x}^{\otimes 2}$ *is either the vector* $\mathbf{x}$ *itself or* $\mathbf{0}$*. Moreover,* $\mathbf{x}^{\otimes 2}$ *is symmetric.*

Proof sketch. Follows from the definition of $\mathbf{x}^{\otimes 2}$ and the fact that we are working with $\mathbb{F}_2$. □

Claim 2.5. *For any vector* $\mathbf{x}$ *and* $\mathcal{L}$ *with UNSAT value* d*, the Hamming weight of* $(\mathbf{Ax}+\mathbf{y})$ *is at least* d*.*

Proof sketch. Follows from definition of Hamming weight and Proposition 2.2. □

Proposition 2.6. *For every linear system* $\mathcal{L}$*, the Hamming weight of* $\mathbf{y}$ *is at least as large as the UNSAT of* $\mathcal{L}$*.*

Proof. Set $\mathbf{x}=\mathbf{0}$ in Proposition 2.2 to see that the proposition holds. □

Lemma 2.7. *For every linear system* $\mathcal{L}$ *with UNSAT at least* d*, the UNSAT*$(\mathcal{L}^{\otimes 2})$ *is at least as large as* d^2*.*

Proof. $\mathcal{L}^{\otimes 2}$ can be rewritten as

$$\mathcal{L}^{\otimes 2} \equiv (\mathbf{Ax}+\mathbf{y})\cdot(\mathbf{Ax}+\mathbf{y})^T = \mathbf{Ax}\cdot\mathbf{x}^T\mathbf{A}^T + \mathbf{y}\,(\mathbf{Ax})^T + \mathbf{Ax}\cdot\mathbf{y}^T + \mathbf{y}\mathbf{y}^T \tag{2}$$

$$= \mathbf{AXA}^T + \mathbf{yAx}^T + (\mathbf{Ax}+\mathbf{y})\cdot\mathbf{y}^T \tag{3}$$

Let us denote the first two terms of Equation (2) by $\mathbf{P}$ and the latter two by $\mathbf{Q}$. We start by analyzing $\mathbf{Q}$. Both these terms are matrices with rows as $\mathbf{0}$ or $\mathbf{y}$. And their sum has $\mathbf{y}$ in row i whenever the vector $\mathbf{Ax}+\mathbf{y}$ has a 1 in the i-th index. We now invoke Claim 2.5 to infer that for any $\mathbf{x}$, $\mathrm{d}(\mathbf{Ax},\mathbf{y})$ is at least d. Hence, $\mathrm{wt}(\mathbf{Ax}+\mathbf{y}) \geq d$. Thus, we can conclude that at sum of these terms has at least d rows of vector $\mathbf{y}$. A quick glance at $\mathbf{P}$, reveals that the rows of $\mathbf{AXA}^T$ are essentially $\mathbf{Ax}$ for some $\mathbf{x}$ (possibly of a different $\mathbf{x}$ for each row/column). And so are the rows of $\mathbf{yAx}^T$. It follows from linearity that the sum of these terms, which is $\mathbf{P}$, has $\mathbf{Ax}$[1] in every row.

To wrap up the argument, we invoke Claim 2.5 over every non-zero row of matrix $\mathbf{Q}$. Notice that the corresponding row of $\mathbf{P}$ is of the form $\mathbf{Ax}$. Thus by Claim 2.5, their sum must have a Hamming weight at least d. Also, there are at least d non-zero rows in $\mathbf{Q}$. Hence, we can conclude that $\mathbf{P}+\mathbf{Q}$ has at least d non-zero rows each with d non-zero entries. □

Notice that, in the above lemma we never restricted the vectors $\mathbf{x}$ and $\mathbf{X}$ to be related in any possible way. In fact, the lemma also holds if a different $\mathbf{x}$ is used in the second, third terms of Equation (2). From now on, we denote the Hamming weight of $\mathbf{y}$ by y.

From now on, we no longer specify $\mathbf{x}$ explicitly. We use the diagonal entries of $\mathbf{X}$ as $\mathbf{x}$. We begin by showing that the UNSAT value of $\mathcal{L}^{\otimes 2}$ of any assignment specified by $\mathbf{X}$ is heavily influenced by $\mathbf{x}$.

[1] Again, not necessarily the same $\mathbf{x}$

Corollary 2.7.1. *For every* $\mathbf{A}$, $\mathbf{y}$, $\mathbf{x}$ *and* $\mathbf{X}$ *such that* $\mathbf{X}_{ii} = \mathbf{x}_i$ *for all* $i \in [n]$, *the Hamming weight of* $\mathbf{AXA}^T + \mathbf{yAx}^T + (\mathbf{Ax} + \mathbf{y}) \cdot \mathbf{y}^T$ *is at least* $wt(\mathbf{Ax} + \mathbf{y}) \cdot d$.

Proof. We begin by borrowing the nomenclature of $\mathbf{P}$, $\mathbf{Q}$ from Lemma 2.7. We then observe that it suffices to show that $\mathbf{Q}$ has at least $\text{wt}(\mathbf{Ax} + \mathbf{y})$ non-zero rows. In other words, $\mathbf{Q}$ has at least $\text{wt}(\mathbf{Ax} + \mathbf{y})$ rows equal to $\mathbf{y}$. Suppose this was indeed true. Also, recall that the rows of $\mathbf{P}$ are of the form $\mathbf{Ax}$. Thus, we can invoke Claim 2.5 to conclude the Hamming weight of $\mathbf{P} + \mathbf{Q}$ is at least $\text{wt}(\mathbf{Ax} + \mathbf{y}) \cdot d$.

Now we prove that $\mathbf{Q}$ has at least $\text{wt}(\mathbf{Ax} + \mathbf{y})$ rows equal to $\mathbf{y}$. $\mathbf{Q}$ has a $\mathbf{y}$ in row i whenever the vector $\mathbf{Ax} + \mathbf{y}$ has a 1 in the i-th index. In our setting, this is exactly $\text{wt}(\mathbf{Ax} + \mathbf{y})$. Thus, we can conclude that at sum of these terms has at least $\text{wt}(\mathbf{Ax} + \mathbf{y})$ rows of vector $\mathbf{y}$. And this completes the proof. □

In what follows from now, we only focus on symmetric matrices $\mathbf{X}$. We can enforce this using a technique known as *folding* in the proof verification literature.

Folding. Roughly, this translates to restricting ourselves to assignments which are symmetric, that is, $\mathbf{X}_{ij} = \mathbf{X}_{ji}$. This can be easily enforced by reading an $\mathbf{X}_{ij}$ only after ordering of $[n]$ in a canonical fashion. In the world of PCP's, this can viewed as *folding* the proof tables. Specifically, the proof consists of the assignment $\pi : \mathbf{X}_{ij} \to \{0, 1\}$, for $i \leq j \leq n$. In what follows, we will establish that it is possible to *uniquely* decode an assignment π for $\mathcal{L}$ from an assignment Π of $\mathcal{L}^{\otimes 2}$ under the guarantee that the $\text{UNSAT}(\mathcal{L}^{\otimes 2}, \Pi)$ exceeds a certain amount.

So, in essence we read $\mathbf{X}_{ij}$ and $\mathbf{X}_{ji}$ from the same location. Also, to read $\mathbf{x}_i$ we look at $\mathbf{X}_{ii}$. In other words, we set $\mathbf{x}_i = \mathbf{X}_{ii}$ for all $i \in [n]$.

2.2 Lower Bounds

Lemma 2.8. *For any linear system* $\mathcal{L}, \delta < d$ *and every* $\mathbf{X}$ *such that the Hamming weight of* $\mathbf{AXA}^T + \mathbf{yAx}^T + \mathbf{Axy}^T + \mathbf{yy}^T$ *is* $d \cdot (d + \delta)$, *it follows that the UNSAT*$(\mathcal{L}, \mathbf{x})$ *is at most* $(d + \delta)$. *Here,* $\mathbf{x}$ *denotes the diagonal vector of* $\mathbf{X}$. *Mathematically,*

$$UNSAT\left(\mathcal{L}^{\otimes 2}, \mathbf{X}\right) = d \cdot (d + \delta) \implies UNSAT(\mathcal{L}, \mathbf{x}) \leq (d + \delta)$$

Proof. We prove the contrapositive. Suppose, the UNSAT $\mathcal{L}(\mathbf{x})$ is greater than $d + \delta$. Thus, the Hamming weight of $\mathbf{Ax} + \mathbf{y}$ is greater than $d + \delta$. We can now invoke Corollary 2.7.1 on $\mathbf{x}$ with above parameters to conclude that the Hamming weight of $\mathbf{AXA}^T + \mathbf{yAx}^T + \mathbf{Axy}^T + \mathbf{yy}^T$ is at least $d \cdot (d + \delta)$. And this establishes the lemma. □

Corollary 2.8.1. *For every* $\mathbf{x}$ *and* $\mathbf{X}$ *that has* $\mathbf{x}$ *as its diagonal*

$$UNSAT(\mathcal{L}, \mathbf{x}) = (d + \delta) \implies UNSAT\left(\mathcal{L}^{\otimes 2}, \mathbf{X}\right) \geq d \cdot (d + \delta)$$

Sketch: Contrapositive of Lemma 2.8. □

A comment on the above lemma is due. Recall Proposition 2.3, it states that given an assignment $\mathbf{x}$ such that $\text{UNSAT}(\mathcal{L}, \mathbf{x}) = \varepsilon$, the $\text{UNSAT}(\mathcal{L}^{\otimes 2}, \mathbf{x}^{\otimes 2}) = \varepsilon^2$. Corollary 2.8.1 suggests that once the diagonal vector is fixed, the UNSAT value of any $\mathbf{X}$ having $\mathbf{x}$ as its diagonal can be at best a constant factor off from the UNSAT value of $\mathbf{x}^{\otimes 2}$ if $\delta \leq \mathcal{O}(d)$. Thus, no matter what $\mathbf{X}$ is the diagonal vector $\mathbf{x}$ is a good approximation of $\mathbf{X}$. In fact, we establish that $\mathbf{X}$ cannot be far-off from $\mathbf{x}^{\otimes 2}$.

Observation 2.9. *For every non-decreasing functions $\delta, d : \mathbb{Z}^+ \to [0,1]$ such that $\delta = O(d)$, $d \cdot (d+\delta) = \Theta(d+\delta)^2$.*

As already noted, Lemma 2.8 does not give any guarantees on the structure of $\mathbf{X}$. We now address this by showing that if $\mathbf{X}$ has an UNSAT in certain regime, then $\mathbf{X}$ and $\mathbf{x}^{\otimes 2}$ cannot be much different from each other.

Lower bounds from every row/column. But before we go any further, we remark that the lower bounds based on the diagonal vector can be extended to every non-zero row or column in $\mathbf{X}$. Thus, any non-zero row (column) is a lower bound on the UNSAT($\mathcal{L}^{\otimes 2}$) and also, the other way around. The trick is to set $\mathbf{x}$ to the corresponding non-zero row (column) in Lemma 2.8 and its Corollary 2.8.1. Thus, the non-zero rows (columns) $\mathbf{x}$ of $\mathbf{X}$ are sort of a primal-dual to each other in the sense that they bound one another.

2.3 Long Linear Systems

We now shift gears to focus on a class of linear systems that are central PCP's and their constructions. These are linear systems that are generated from the long code test. We shall deal with them at length in Section 3.1. For now, it suffices to know that these are linear systems that are overly determined. Also, the homogeneous part of these linear systems, essentially, looks like

$$x_i \oplus x_j \oplus x_{i+j} = 0, \qquad \forall\, i,j \in [n]$$

In other words, the linear systems, $\mathcal{L} \equiv \mathbf{A}\mathbf{x} + \mathbf{y} = \mathbf{0}$, are such that every row of $\mathbf{A}$ will have only 3 non-entries and every column has a Hamming weight of n. Also, $\mathbf{A}$ is a $n^2 \times n$ matrix.

Proposition 2.10. *For every $\mathbf{A} \in \mathbb{F}_2^{n^2 \times n}$ such that the Hamming weight of every row and column of $\mathbf{A}$ is 3 and n respectively. Then, for every $\mathbf{x} \in \mathbb{F}_2^n$ and any $\tilde{\mathbf{x}}$ that is δ-far from $\mathbf{x}$ the Hamming weight of $\mathbf{A}\cdot(\mathbf{x}+\tilde{\mathbf{x}})$ is at least $2\cdot\delta(1-\delta)$.*

Proof. By assumption, the Hamming weight of $(\mathbf{x}+\tilde{\mathbf{x}})$ is at least δ. A simple counting reveals that there are at least $2\cdot\delta(1-\delta)$ that $\mathbf{x}+\tilde{\mathbf{x}}$ disagree. Thus, the Hamming weight of $\mathbf{A}\cdot(\mathbf{x}+\tilde{\mathbf{x}})$ is at least $2\cdot\delta(1-\delta)$. □

Proposition 2.10 can be easily extended to claim that the distance between $\mathbf{A}\mathbf{X}\mathbf{A}^T$ and $\mathbf{A}\tilde{\mathbf{X}}\mathbf{A}^T$ is roughly same as the distance between $\mathbf{X}$ and $\tilde{\mathbf{X}}$. In fact, we get the same tradeoff.

In what follows, we shall show that if the prover provides us with any assignment $\mathbf{X}$ other than one equal to $\mathbf{x}^{\otimes 2}$, such that UNSAT($\mathcal{L}$, $\mathbf{x}$) is not optimum (which is d), then UNSAT($\mathcal{L}^{\otimes 2}$, $\mathbf{X}$) strictly larger than d^2.

Lemma 2.11 (Unique Tensor Lemma)**.** *For any linear system $\mathcal{L}$ with UNSAT value d and every $\mathbf{X} \notin \{\mathbf{x}^{\otimes 2}\}$, UNSAT($\mathcal{L}^{\otimes 2}$, $\mathbf{X}$) is strictly larger than d^2.*

Proof. We kick off by invoking Proposition 2.3 to infer that if $\mathbf{X} = \mathbf{x}^{\otimes 2}$, for some $\mathbf{x}$ with UNSAT($\mathcal{L}$, $\mathbf{x}$) larger than d, then the UNSAT($\mathcal{L}^{\otimes 2}$, $\mathbf{x}^{\otimes 2}$) is also larger than d^2. We now move on to show that if $\mathbf{X} \notin \{\mathbf{x}^{\otimes 2} : \text{UNSAT}(\mathcal{L}, \mathbf{x}) = d\}$, then its UNSAT is larger than d^2. A crucial observation based on Claim 2.4 and the hypothesis here is that $\mathbf{X}$ has at least two non-zero rows that are distinct. In particular, there is another non-zero vector $\tilde{\mathbf{x}}$ different from the diagonal vector ($\mathbf{x}$) of $\mathbf{X}$. Recall that for every $\mathbf{X}$

$$\text{UNSAT}(\mathcal{L}^{\otimes 2}, \mathbf{X}) = \text{wt}(\mathbf{A}\mathbf{X}\mathbf{A}^T + \mathbf{y}\mathbf{A}\mathbf{x}^T + (\mathbf{A}\mathbf{x}+\mathbf{y})\cdot\mathbf{y}^T)$$

Let us denote the first two terms of above equation by $\mathbf{P}$ and the latter by $\mathbf{Q}$. We infer conclusions of Lemma 2.7 that every row in $\mathbf{P}$ is in the form of $\mathbf{A}\mathbf{x}$[2] in every row. Now, we have

[2] Again, not necessarily the same $\mathbf{x}$

additional information in that $\mathbf{X}$ has at least non-zero twos (say, $\mathbf{x}$ and $\tilde{\mathbf{x}}$) that are distinct. Say, they are δ-far from one another, for some $\delta > 0$. We now invoke Proposition 2.10 to infer the $\mathbf{Ax}$ and $\mathbf{A}\tilde{\mathbf{x}}$ are δ'-far. By Claim 2.1, $(\mathbf{Ax} + \mathbf{y})$ and $(\mathbf{A}\tilde{\mathbf{x}} + \mathbf{y})$ are also δ'-far. Now observe that there are at least d rows in $\mathbf{Q}$, each of which is equal to $\mathbf{y}$. Since, $\mathrm{wt}(\mathbf{Ax} + \mathbf{y})$ is least d, $\mathbf{P} + \mathbf{Q}$ has at least d non-zero rows. However, there are two rows that differ in at least δ'-fraction of entries. Thus, there are at least $d + \delta'$ non-zero columns in $\mathbf{P} + \mathbf{Q}$.

At this juncture, we recall that every $\mathbf{X}$ is symmetric and is the matrix $\mathbf{P} + \mathbf{Q}$[3] and every row/column of $\mathbf{P} + \mathbf{Q}$ is either $\mathbf{0}$ or $\mathbf{Ax} + \mathbf{y}$, for some $\mathbf{x}$. Hence, the Hamming of every row/column is either 0 or at least d. Thus, the Hamming weight of $\mathbf{P} + \mathbf{Q}$ is at least $d \cdot (d + \delta')$. Because $\delta' > 0$, the UNSAT$(\mathcal{L}^{\otimes 2}, \mathbf{X})$ is strictly larger than d^2. □

A comment on the above lemma is due. Lemma 2.11 implies that if a tensored system satisfies a certain fraction of equations, then the tensored assignment is constructed from a unique assignment of $\mathcal{L}$. The key idea was that we set up ourselves such that we operate in the *unique decodability regime* of the tensored system.

3 The PCP verifier

We denote by $PCP_{c,s}[r, q]_{\Sigma}$ the class of languages that have a PCP verifier with completeness c, soundness s, randomness r and makes q queries to the purported proof over an alphabet Σ. It is often useful to imagine all the parameters as a function of n.

3.1 Assignment Testers

We now apply the tools we have constructed to PCPs. In what follows, we work with assignment testers. The notion of assignment testers was pioneered by [DR06, BSGH+06]. Informally, an assignment tester not only needs to accept a correct proof almost always but reject a purported proof which is far from any valid proof. Note that this is a stronger requirement to that of a PCP, where one aspires to verify only if there exists an satisfying assignment to that instance of the problem.

We restrict ourselves to *linear* assignment testers. These are assignment testers where every output circuit in Ψ computes a linear function over its variables.

Definition 4 (Assignment Tester)**.** *An (ϵ, δ)-Assignment Tester is a reduction whose input is a Boolean constraint ψ over a set of Boolean variables X. The output of the reduction is a system of constraints Ψ over variables X and auxiliary variables Y such that for every assignment $\pi : X \to \{0,1\}$,*

- Completeness: *If π satisfies ψ then there exists an assignment $\tilde{\pi} : Y \to \{0,1\}$ such that $\pi \cup \tilde{\pi}$ satisfies at least $(1 - \epsilon)$ fraction of the constraints in Ψ.*
- Soundness: *For every $\delta' < \delta$, if π is δ'-far from a satisfying assignment for ψ, then for every assignment $\tilde{\pi} : Y \to \{0,1\}$, at least $\Omega(\delta')$ of the constraints in Ψ reject $\pi \cup \tilde{\pi}$.*

[3]The set of symmetric matrices is closed under addition.

Standard Definitions. We identify the *long code* of $\mathbf{x} \in \{\pm 1\}^n$ by $\mathsf{LC}(\mathbf{x} = \{f(x) | f : [n] \to \{\pm 1\}\}$. Informally, we evaluate $\mathbf{x}$ on every Boolean function on n bits. We usually identify the domain of $\mathsf{LC}$with the Boolean hypercube $\{\pm 1\}^n$. We use the letters f, g for points on the hypercube. We use A, B and χ to denote functions whose domain is in the hypercube. In particular, we consider functions whose domain is an arbitrary set of size n. In this application, this set is usually some $\{\pm 1\}^s$ such that $n = 2^s$. For $\alpha \subset [n]$, define

$$\chi_\alpha : \{\pm 1\}^n \to \{\pm 1\}, \chi_\alpha(f) \triangleq \prod_{i \in \alpha} f(i)$$

It is easy to check that the characters $\{\chi_\alpha\}_{\alpha \subseteq [n]}$ form an orthonormal basis for the space of functions $\{A : \{\pm 1\}^n \to \mathbb{R}\}$, where inner product is defined by $\langle A, B \rangle = \mathbb{E}_f[A(f)B(f)] = 2^{-n} \sum_f A(f)B(f)$. It follows that any function $A : \{\pm 1\}^n \to \{\pm 1\}$ can be written as $A = \sum_\alpha \hat{A}_\alpha \cdot \chi_\alpha$, where $\hat{A}_\alpha = \langle A, \chi_\alpha \rangle$. We also have the parseval's identity, $\sum_\alpha |\hat{A}_\alpha|^2 = \langle A, A \rangle = 1$.

Folding. As done in [BGS98], we fold the long code tables over true and the respective constraint ψ. This means that whenever the test needs to read $A[f]$, it reads $A[\psi \wedge f]$ instead. In addition, we fold over true which means for every pair f and $-f$, we let A specify only one and access the other via the identity $A[f] = -A[f]$. In short, we assume that $A[f] = A[f \wedge \psi]$ and $A[f] = -A[-f]$ for all f. It is well known that after folding $\hat{A}_\alpha = 0$ whenever $|\alpha|$ is even or there exists an i in α for which $\psi(i) = 1$ (recall that 1 corresponds to false).

The Long Code Test. Let $\Psi : [n] \to \{\pm 1\}$, be some predicate, and fix some $\epsilon > 0$. let $A : \{\pm 1\}^n \to \{\pm 1\}$. The test picks two uniformly random vectors $\mathbf{x}, \mathbf{y} \in \{\pm 1\}^W$ and then a vector $\mathbf{z} \in \{\pm 1\}^W$ according to the following distribution: for every coordinate $i \in [W]$, with probability $1 - \rho$ we choose $z_i = 1$ and $z_i = -1$ otherwise. It is useful to imagine $\mathbf{z}$ as a noise vector. The test accepts iff $f(\mathbf{x})f(\mathbf{y}) = f(\mathbf{xyz})$. In other words, iff $z_w = 1$, which happens with probability $1 - \rho$. It follows from the construction that the test accepts any valid long code encoding with probability $1 - \rho$.

Lemma 3.1 (Håstad's lemma [Hås01])**.** *If the test accepts with probability $1/2 + \delta$, then $\sum_\alpha \hat{f}_\alpha^3 \cdot (1 - 2\rho)^{|\alpha|} \geq 2\alpha$.*

Corollary 3.1.1. *If f passes the long code test with probability with $1/2 + \delta$, then for $k = \frac{1}{2\rho} \log \frac{1}{\epsilon}$, there exists α with $|\alpha| \leq k$ such that $\hat{f}_\alpha \geq 2\delta - \epsilon$.*

Lemma 3.2 (Tester Lemma)**.** *For every $\delta < 1/4$ there is a constant $\eta(\delta) > 0$, such that a table A which is δ-far from any valid long code encoding of a with $\psi(a) = \mathsf{T}$, is rejected with probability $\geq \Omega(\delta)$.*

Proof. Assume that A test passes with probability $> 1 - \delta$. That is,

$$\begin{aligned} &\Pr[A \text{ passes }] \geq 1 - \delta \\ &\qquad \implies \exists k \leq \eta,\ \hat{f}_k \geq 2 \cdot (1/2 - \delta) - \epsilon \\ &\qquad \implies \exists \alpha \subset [n], |\alpha| \leq k \text{ s.t. } \chi_\alpha \text{ is } \Omega(\delta')\text{-far from } A \end{aligned}$$

Note that χ_α can be expressed as $\prod_{i \in \alpha} \chi_i$ and $|\alpha|$ is a constant. Thus, any of the χ_i's agrees with χ_α on at least $2^{-|\alpha|}$ fraction of the inputs. Putting this in the context of agreement between A and χ_i: χ_i disagrees with A on at least $1/2^{|\alpha|} \cdot \delta'$ fraction, which is essentially $\Omega(\delta)$.

□

We are now ready to present the assignment tester needed for our construction. Let ψ be a Boolean constraint over Boolean variables $x_1, x_2 \dots x_s$. We describe an algorithm $\diamond$ whose input is ψ and whose output will be a system of linear equations satisfying the requirements of Definition 4.

There is nothing new in the construction. It essentially takes two steps (which have become quite standard):

1. Long Code Constraints: Lets start with the variables $X = \{x_1, x_2 \dots x_s\}$. An assignment now is made of two tables $\sigma : X \to \{\pm 1\}$ and $\pi : L \to \{\pm 1\}$. We place two constraints. The first set includes 3 variable constraints based on π derived from the *long code* test. Specifically, one constraint per each coin toss.

2. Consistency Constraints: The second set of constraints include the following: For each choice of $i \in [s]$ and $f \in L$ place a constraint that is satisfied iff $\sigma(x_i) = A(f) \oplus A(f \oplus e_i)$[4]. For the ease of argument, assume that are an equal number of constraints of each type[5].

Lemma 3.3 (Assignment Tester)**.** *For any $\rho > 0$, the constraint system constructed above is a linear assignment tester for any Boolean function $\psi : [s] \to \{0, 1\}$, with completeness $1 - \rho$, $\delta < 1/4$ and $|\Sigma| = 2$.*

Proof. Linearity of the constraints follows from construction. We now analyze the parameters of the tester below parameter by parameter.

- Completeness: For any good assignment that satisfies the constraint ψ, the second set of constraints are always satisfied. The only case where we reject a good proof is during the long code test and this happens with probability $1 - \rho$.

- Soundness: Assume that σ is an assignment which is δ-far from every satisfying assignment for ψ. We first show that at least $\Omega(\delta)$ constraints of the long code test are rejected. It follows from Lemma 3.2 that any A that is δ-far from a valid long code table is rejected with probability greater than $\Omega(\delta)$ and by our construction, these constraints constitute at least $\Omega(\delta)$-fraction of the total constraints.

  If A is not δ-far from a legal long code, then it is δ-close to the long code encoding of some σ' which satisfies σ'. From the hypothesis, $Pr_i[\sigma(x_i) \neq \sigma'(x_i)] \geq \delta$. This implies that at least $\Omega(\delta)$ of constraints of the second type. This is clear as for each i, we have

$$\Pr_{f \in L} \left[A(f) \oplus A(f + e_i) = f(\sigma') \oplus (f \oplus e_i)(\sigma') \right] \geq 1 - 2\delta \tag{4}$$

  Also, if $\sigma(x_i) \neq \sigma'(x_i)$, then $f(\sigma') \oplus (f \oplus e_i)(\sigma') = \sigma'(x_i) \neq \sigma(x_i)$. Hence, every f that satisifies Equation (4) causes the corresponding comparision constraint to reject. Hence, even in this case at least $\Omega(\delta)$-fraction of the constraints are rejected.

- Query Size: The tester makes 3 queries for the long code test and 3 for the comparisions totalling to 6 queries.

- Randomness: The tester uses $2 \cdot s$ bits of randomness.

[4] e_i is the vector of dimension s with a -1 in i-th index and 1 elsewhere.

[5] This can be easily achieved by placing multiple copies of the constraint of each type with appropiate multiplicity.

□

We denote the size of a linear tester by the n and it refers to the quantity $q \cdot m$, where q is the number of queries made by the tester and m is the total number of equations in the tester. Linear assignment testers cannot have perfect completeness. For otherwise, one may use guassian elimination to figure out the if there is an assignment that satisfies all the linear constraints and reject if there isn't one. Hence, any linear tester is forced to have a non-zero completeness error. Also, the soundness is always less than $1/2$ as a random assignment satisfies at least $1/2$ of the equations in $\mathcal{L}$ in an expected sense (follows from the fact that the equations are over $\mathbb{F}_2$). As highlighted earlier, having low completeness error becomes handy in several scenarios. We now highlight on how we intend to use the tools we developed in the previous section to achieve completeness amplification.

4 Completeness Amplification

Lemma 4.1 (Tensor Tester)**.** *There is an algorithm that transforms a (c, s)-tester for any Boolean function with q queries into a (c^2, s^2)-tester. Morever, this transformation runs in time polynomial in the size of the input instance.*

Proof. Let V denote the (c, s)-tester and $\mathcal{L}$ denote the tests performed by V. The idea is tensor the tester. So, the new tester will be $V^{\otimes 2} \equiv \mathcal{L}^{\otimes 2}$. Alternatively, the tests of $V^{\otimes 2}$ are the linear equations in $\mathcal{L}^{\otimes 2}$. So, the proof to the new tester will be $(\sigma \circ \pi)^{\otimes 2}$. We shall now analyze the parameters of $\mathcal{L}^{\otimes 2}$.

- Completeness: In the good case, if there is a proof $\sigma \circ \pi$ which $\mathcal{L}$ accepts with probability at least $1 - c$. We invoke Proposition 2.3 to conclude that the probability that the verifier whose tests are $\mathcal{L}^{\otimes 2}$ accepts $(\sigma \circ \pi)^{\otimes 2}$ is at least $1-$ UNSAT$(\mathcal{L}, \sigma \circ \pi)^2$, which is $1 - c^2$.

- Soundness: We use Lemma 2.7 to claim that the tensored verifier rejects an invalid proof with probability at least s^2. In order to verify that verifier also rejects any proof that is s^2-far from any satisfying assignment with probability at least $\Omega(s^2)$, we show the following. For any proof that the tensored verifier $V^{\otimes 2}$ accepts with probability at least s^2, we can *uniquely* decode a proof that V accepts with probability at least s. Once, V accepts anything with probability s, Lemma 3.3 implies that it is close to a good proof. We now elaborate on what we said in the prequel.

  Say, we are given a proof Π that the tensored verifier accepts with probability at least $1 - s^2$. We can by the virtue of Lemma 2.11, decode a proof which the input tester accepts with probability at least $1 - \Omega(s)$. And from Lemma 3.3, we know that if the input tester accepts a proof, it is indeed close to a satisfying proof. Specifically, if it accepts with probability at least $\Omega(\delta)$, it is at most $(1 - \delta')$-far from a valid proof. It follows from the definition of tensoring and uniqueness of Lemma 2.11 that if a tester accepts a proof is at most $(1 - s')$-far from a satisfying assignment π, the tensored proof is at most $(1 - s'^2)$-far from the satisfying assignment $\pi^{\otimes 2}$ to the tensored system. This establishes that if the tensored tester accepts any proof with probability more than $1 - s^2$, it is at most $(1 - s'^2)$-far from a valid proof.

- Query size: $V^{\otimes 2}$ needs to make as many queries as the number of variables in an equation of $\mathcal{L}^{\otimes 2}$. And this happens to be q^2.

- Randomness: The randomness used by $V^{\otimes 2}$ is $\log \mathcal{L}^{\otimes 2}$, which is $2 \cdot \log \mathcal{L}$. Thus, we double the randomness used whenever we tensor V.

□

Notice that the above transformation preserves the linearity of the tester. The downside of tensoring is that while it enhances the completeness, it also destroys the soundness. Ideally, we would like to improve the acceptance probability of the verifier in the good case and leave the acceptance probability in the bad case unperturbed. Thus to improve the soundness of the tester after the tensoring operation, we resort to sequential repetition.

Soundness Amplification of the Tester. Sequential repetition of a Tester involves repeating the tests sequentially with independent trails. For instance, performing sequential repetition once would involve creating a new linear system by picking every pair of equations in $\mathcal{L}$ and then summing them up to obtain a new linear system $\mathcal{L}'$ whose size is twice the size of $\mathcal{L}$. This transformation converts a (c, s)-tester into a $(2c, 2s)$-tester. In general, performing it ϑ times on (c, s)-tester leaves us with a $(\vartheta \cdot c, \vartheta \cdot s)$-tester.

Lemma 4.2 (Soundness Amplification)**.** *For every $\vartheta < \min(1/c, 1/s)$, there is an efficient procedure which on input a (c, s)-tester outputs a $(\vartheta \cdot c, \vartheta \cdot s)$-tester.*

Proof. Let us denote the (c, s)-tester by V and the tests performed by V with $\mathcal{L}$. Roughly, our approach will be to design a new verifier V' whose tests are the sum of ϑ tests of the old verifier V, each of which is chosen independently at random. Thus, $\mathcal{L}' = \{\phi_1 \oplus \phi_2 \ldots \oplus \phi_\vartheta : \phi_i \in \mathcal{L} \ \ \forall\ i \ \in [\vartheta]\}$. Note that even though the tests of V' are different from V, both of them expect the same proof as in Lemma 3.3. We now establish that V' is in fact a $(\vartheta \cdot c, \vartheta \cdot s)$-tester.

- Completeness: In the good case, V rejects a valid proof $\sigma \circ \pi$ with probability at most c. Now by the union bound, the probability that V' rejects it is at most $\vartheta \cdot c$.
- Soundness: Consider the scenario that V' accepts a proof π' with probability higher than $\vartheta \cdot s$. It follows from basic probability that if a bernoulli experiment(tests of V) with success probability p is repeated k times, then the probability that there are no successes is exactly $1-(1-p)^k$. A simple calculation will reveal that if we are given that this quantity is $s \cdot k$, then p is at least s. In other words, V could accept π' with at least s. To conclude the argument, we invoke Lemma 3.3 to infer that if V accepts π with probability s, then the purported proof is close to a satisfying proof.
- Query size: The arity of the equations in $\mathcal{L}'$ is $\vartheta \cdot q$, where q is the arity of V.
- Randomness: Since, we are drawing ϑ independent samples from $\mathcal{L}$, we would be needing $\vartheta \cdot r_V$, where r_V is the randomness used by V.

□

The randomness used in the above construction is $\vartheta \cdot r_V$. This is multiplicative in the number of sequential repetitions. However, we would like to be frugal with regard to the randomness. Since, we are using sequential repetition; it can be derandomized via randoms walks on expanders or essentially any one of its variants like samplers. In this work, we make use of a samplers based approach. Samplers are pseudo-random objects that are extremely useful in simulating a random walk or drawing random samples. In what follows, we denote the neighbours of a vertex u in a graph by $\Gamma(u)$.

Definition 5 (Samplers). *A bipartite graph $H = (A, B, E)$ is said to be an (ϵ, δ)-sampler if for every subset $S \subseteq A$,*

$$\Pr_{b \in B}\left[\frac{|\Gamma(b) \cap S|}{|\Gamma(b)|} > \frac{|S|}{|A|} + \epsilon\right] \leq \delta$$

In other words, neighbourhoods of most vertices b behave like a random sample of A, in the sense that that their density within any fixed S is close to what is expected. The following article [Gol97] by Oded Goldreich is an excellent exposition on samplers, their constructions and applications.

Theorem 4.3. *[Gol97] There exists a polynomial time algorithm that, given an integer n and a parameter $\epsilon > 0$, outputs an (ϵ, δ)-sampler with $|A| = |B| = n$ and the right degree $4/\epsilon^4$. Moreover, the randomness used by the algorithm is bounded by $\log n + \mathcal{O}(\log 1/\epsilon) + \mathcal{O}(\log 1/\delta)$.*

Lemma 4.4 (Derandomization Lemma). *For every $c > 0, s < 1/4$, $\vartheta < \min(1/c, 1/s)$ and ϑ sequential repetitons of every (c, s)-tester can be achieved using only $r_v + \mathcal{O}(\log \vartheta)$ random bits.*

Proof. Follows by invoking Theorem 4.3 to construct a (ϵ, δ)-sampler and use it instead of sampling ϑ tests of verifier at random. Specifically, set $n = 2^{r_v}$, $\delta = 1 - \vartheta \cdot s$, ϵ will be fixed latter. We now establish that this process actually simulates sequential repetition in a randomness efficient way. During sequential repetition, we wish to test independent samples of the verifier's test. Consider the soundness case, here we would like to sample one of the constraints that fail the verifier's test. Say, a set S of the tests fail. So, we would like to hit one of these tests. In a sequential repetition, the probability that we hit this set is $\vartheta \cdot \frac{|S|}{|A|}$.

We aim to achieve the aforementioned claims via a sampler. For a sampler to have failed us, it must be that the vertex b on the right must have had all its neighbours in the complement (denote this set by $\overline{S}$) of S that we wanted to hit. And, the probability that this would happen is

$$\Pr_{b \in B}\left[\frac{|\Gamma(b) \cap \overline{S}|}{|\Gamma(b)|} = 1\right] = \Pr_{b \in B}\left[\frac{|\Gamma(b) \cap \overline{S}|}{|\Gamma(b)|} = 1 - \frac{|S|}{|A|} + \frac{|S|}{|A|}\right] = \Pr_{b \in B}\left[\frac{|\Gamma(b) \cap \overline{S}|}{|\Gamma(b)|} = \frac{|\overline{S}|}{|A|} + \frac{|S|}{|A|}\right]$$

We invoke Theorem 4.3 for $\epsilon \in (0, \frac{|S|}{|A|}]$.

$$\Pr_{b \in B}\left[\frac{|\Gamma(b) \cap \overline{S}|}{|\Gamma(b)|} > \frac{|\overline{S}|}{|A|} + \epsilon\right] \leq 1 - \vartheta \cdot s \implies \Pr_{b \in B}\left[\frac{|\Gamma(b) \cap S|}{|\Gamma(b)|} > \epsilon\right] > \vartheta \cdot s$$

Hence, the completeness and soundness of this new tester is $\vartheta \cdot c$ and $\vartheta \cdot s$ respectively. This is essentially what we achieve with ϑ sequential repetitions as in Lemma 4.2 but with a larger query complexity $1/\epsilon^4$. If $\epsilon = \Theta(1/\vartheta)$, then the query complexity happens to be $\mathcal{O}(\vartheta^4)$. The bounds on randomness follows from Theorem 4.3. □

Corollary 4.4.1 (Tensoring + Repetition). *After performing a round of tensoring and ϑ' rounds of sequential repetition on a (c, s)-tester that uses r_V random bits to make q queries, we obtain a $\mathcal{O}\left(c^2 \cdot \vartheta\right), s'$-tester. The new tester uses $r_v + \mathcal{O}(\log \vartheta)$ random bits and makes $\vartheta^4 \cdot q^2$ queries.*

Proof. Invoke Lemma 4.1 followed by Lemma 4.4 with $\vartheta = 1/s$ and $c = 1/\left(2 \cdot \vartheta\right)$. □

Parameters	Tensoring	ϑ Sequential Repetitions	After k Iterations
`Completeness`	c^2	$\mathcal{O}\left(\vartheta' \cdot c^2\right)$	$\mathcal{O}\left(\vartheta' \cdot c^{k+1}\right)$
`Soundness`	s^2	s	s
`Queries`	q^2	$\vartheta^4 \cdot q^2$	$\vartheta^{4\cdot(k+1)} \cdot q^k$
`Alphabet`	$\{0,1\}$	$\{0,1\}$	$\{0,1\}$

Table 1: Parameters of a (c, s)-tester after various operations.

Lemma 4.5 (Completness Amplification Lemma)**.** *There exist an infinite sequence $\{k_i\}_{i\in\mathbb{N}}$, such that for every k in the sequence, a (c, s)-tester of size 2^{r_v} can be transformed into a $(\mathcal{O}(c^k), s')$-tester in time polynomial in $2^{k\cdot\left(r_v+\mathcal{O}(\log\vartheta)\right)}$.*

Proof. The first observation we make is that the after a (c, s)-tester, of size n, under goes r iterations of the Tensoring followed by σ rounds of sequential repetition, the size of the resulting tester is $n^{2^{2^r-1}}$. This can be verified once we observe that we are squaring the size of the input instance in every iteration. By invoking Corollary 4.4.1, we infer that the completeness error is also squared in each iteration. Hence after r iterations, the new error parameter is $c^{2^{2^r-1}}$. However, the soundness of the tester remains uneffected. With this background, the Lemma follows by setting $r = \log\log k$. The parameters are summarized in Table 4.

Recall Corollary 4.4.1 to infer that after the first iteration of (c, s)-tester we obtain a $(\mathcal{O}(c^2 \cdot \vartheta), s')$-tester. Since, Lemma 4.1 and 4.2 guarantee that we preserve the properties of tester and are not related in any way, this implies that every iteration produces with the corresponding parameters. In every tensoring, we square the randomness (ref. Lemma 4.1. Thus, the *randomness* of the tester after r iterations is $k\cdot\left(r_v + \mathcal{O}\left(\log\vartheta\right)\right)$, where r_v is the randomness used by the (c, s)-tester. □

5 Towards Reducing Queries, Soundness Amplification

One parameter we have not paid attention during the completeness amplification of the tester is the query size. As was the case with every other parameter, we also square the number of the queries the input tester makes. In general, the number of queries the tester makes is $\mathcal{O}(1/c)$. To establish PCPs for NP, one must be frugal when it comes to queries. So, we must apply some technique to keep reduce the queries. The obvious approach to break into $3 - Lin$ does not work as we would lose huge factors $(1/c)$ in soundness. At which point, it will be hard to keep the randomness used by the verifier to logarithmic amount. Thus, we use *Locally Decode/Reject Codes* (LDRC) to reduce the number of queries in the PCP (from the perspective of GAP-LIN, we reduce the size of equation). The advantage of using LDRC for query reduction is that no additional completeness error is incurred.

Definition 6 (Construction Algorithm, Theorem 15 in [MR10])**.** *A (k_{max}, δ_{min})-construction algorithm for bipartite LDRCs with parameters $\langle size, block_A, block_B\rangle$ is an efficient algorithm that given a collection of k-tuples, where $k \leq k_{max}$, outputs a (δ_{min}, l_{max})-bipartite LDRC for the tuples, where $l_{max}(\delta) \leq \delta^{O(1)}$. The size of the output is size, the alphabet size of the A vertices is 2^{block_A} and the alphabet size of the B vertices is 2^{block_B}.*

Theorem 5.1 (Query Reduction, Theorem 16 in [MR10])**.** *If there is a (q, ϵ)-construction algorithm for bipartite LDRCs with parameters $\langle size \leq (N+n)\cdot n^{o(1)}, block_A, block_B\rangle$, then for some $\epsilon_0 \geq \epsilon^{O(1)}$,*

$$PCP_{1,\epsilon_0}\left[\left(1+o\left(1\right)\right)\cdot\log n, q\right] \subseteq PCP_{1,O(\epsilon)}\left[\left(1+o\left(1\right)\right)\cdot\log n, 2\right]_{\{0,1\}^{block_A}}$$

Moreover, the transformation yields linear projection tests if the original instance had linear constraints.

Though, the claim on linearity is not explicitly stated in [MR10], it does hold. The key observation in establishing this invariant is that locally decode/reject codes are based on linear codes and are themselves linear by definition. One advantage of using LDRC for query reduction is that the composed verifier has all the qualities of a tester and thus, will be for ready for composition with another verifier in a seamless manner.

Theorem 5.2 (Tester with Low Error)**.** *For every $\alpha, \beta > 0$, any (c, s)-tester that can verify a Boolean function $\psi : [s] \to \{0,1\}$ by making q queries using r_v random bits can be transformed into a $(\mathcal{O}(c^{\alpha'}), \mathcal{O}(s^{\beta}))$-tester that uses $(1+o(1))\cdot\alpha\cdot(r_v + \mathcal{O}(\log\vartheta))$ random bits and has an alphabet size $\{0,1\}^{1/c^{\alpha'}\cdot\mathsf{poly}(1/s^{\beta})}$. Moreover, the new linear tester only makes two queries with projection property to the purported proof.*

Proof. Follows by invoking Theorem 5.1 on the tester obtained in Lemma 4.5. Particularly, we invoke it with $k = \alpha$ to obtain a $((\mathcal{O}(c^{\alpha}), s')$-tester. We then use the $(q^{\alpha}, 1/s^{\beta})$-construction algorithm for query reduction and soundness amplification. We now analyze the parameters of the tester.

- *Completeness:* The completeness error of the tester obtained in Lemma 4.5 is c^{α}. Theorem 5.1 introduces an error which is proportional to the soundness error of the PCP obtained by applying it. In fact, Theorem 5.1 is essentially a parallel-repetition theorem in the low-error regime, albeit with a much worse alphabet tradeoff. Thus, the completeness error is that product of $c^{\alpha}\cdot 1/s^{\beta}$. In other words, the error is not more than $c^{\alpha-\beta'}$, choose $\alpha = \omega(\beta')$ to get the parameters we want.
- *Soundness:* It follows from Theorem 5.1 that the soundness of the new construction is $\mathcal{O}(s^{\beta})$.
- *Alphabet:* We inherit the alphabet of Theorem 5.1, which happens to be $\{0,1\}^{1/c^{\alpha}\cdot\mathsf{poly}\left(1/s^{\beta}\right)}$, where $\mathsf{poly}(\cdot)$ is implicit in Definition 6.
- *Randomness:* Theorem 5.1 blows up the randomness of the input tester to $1+o(1)\cdot\mathsf{R}$, where R is randomness of the tester obtained from Lemma 4.5. For the parameters that we have chosen, it is $\left(1+o\left(1\right)\right)\cdot\alpha\cdot(r_v+\mathcal{O}(\log\vartheta))$.

□

6 Final Brick: Composition

The overall strategy will be to start with an outer PCP which has perfect completeness and is also robust. The notion of robust composition PCP's was pioneered by [BSGH+06, DR06]. It involves two steps:

- Robustization: The key idea is to have a stronger soundness condition. In a robust PCP not only that the verifier must reject any invalid proof with a good probability, also the answers to the queries of PCP verifier are sufficiently far from any satisfying answer. Robust PCP's can be constructed from the extant PCP literature. We highlight two constructions.

	Robust Outer	Linear Tester	Final PCP
Soundness	$1/(\log n)^{\beta}$	$1/(\log n)^{\beta}$	$1/\mathcal{O}\left((\log n)^{\beta}\right)$
Completeness	1	$1-1/(\log n)^{\alpha}$	$1-1/(\log n)^{\alpha}$
Size	n	$(\log n)^{\beta}$	$n \cdot (\log n)^{\beta}$

Table 2: Various Parameters during Composition

1. *Robust PCP* $\equiv$ Label-Cover. The formulation of Robust PCP's might have arrived late, but they have been ever-existent and ubiquitous in the PCP world disguised as Label-Cover (ref. Section 6 to know more about Label-Cover). An interested reader might refer Lemma 2.5, page 11 of [DH09] to understand this equivalence.
2. *Via Error Correcting Codes.* Dinur and Reingold [DR06] give a generic transformation of any PCP into a robust one. It roughly involves encoding the alphabet with an error correcting code $E : \Sigma \to \{0,1\}^l$. The distance required from the code is determined by the robustness parameter ρ. Also, if one would additional require that the size of the reduction is quasi-linear, then they must use an error correcting code E to with a linear rate.

- Composition: We now run the assignment tester given by Theorem 5.2 (with required parameters) on every test $c : \Sigma^k \to \{0,1\}$ of the aforementioned robust PCP verifier. Notice that C can be transformed into a Boolean constraint over Boolean variables. To this end, replace each variable v by a set of l Boolean variables denoted by $[v]$ and the circuit for c by Boolean gates (essentially its binary encoding). So, we would end up with a new constraint $\tilde{c}$ over new variables $[x_1] \cup [x_2] \dots [x_s]$. c would be satisfied iff the assignment for $[x_1] \cup [x_2] \dots [x_s]$ is a satisfying assignment for c. After carrying out the above transformation, we shall have a system of constraints $\{\tilde{c}_i\}_{i \in [n]}$. Thus, it is well defined to run the tester on c.

  After composing them with our tester, we will end up with $\{\widehat{c}_i\}$ over the old variables $[x_1] \cup [x_2] \dots [x_s]$ and new auxiliary variables Y. Firstly, it is easy to check that the size remains polynomial in n [6]. The outer will be a Robust PCP verifier instance. We start with a verifier possessing perfect completeness [7] and low soundness error. Hence, the error parameters of the "new" composed verifier are solely determined by the errors of the assignment tester Π. This completes the outline of the composition.

We state the robust outer that we intend to use for our composition. We use the low error Label-Cover (Definition 7) instance generated by Moshkovitz and Raz [MR10]. For the sake of completeness, we state it here.

Theorem 6.1 (Low-Error Label-Cover)**.** *For every n, and every $\epsilon > 0$ (that can be any function of n) the following holds. Solving* 3SAT *on inputs on size n can be reduced to distinguishing between the case that a* Label-Cover *instance of size $n^{1+o(1)} \cdot \mathsf{poly}(\frac{1}{\epsilon})$ and parameters $\log |\Sigma_A| \leq \mathsf{poly}(\frac{1}{\epsilon})$ and $|\Sigma_B| \leq |\Sigma_A|$ is completely satisfiable and the case that at most ϵ fraction of the edges are satisfiable.*

[6] Follows from the fact that the size of every inner is $\mathcal{O}(\log n)^{\beta}$, for some $\beta < 1$ and there are at most $\mathcal{O}(n^{1+o(1)})$ constraints to compose.

[7] In other words, no error in completeness.

Theorem 6.2. *For some $0 < \alpha, \beta \leq 1$,*

$$3SAT \in PCP_{1-\frac{1}{(\log n)^{\alpha}}, \frac{1}{(\log n)^{\beta}}} \left[\left(1 + o\left(1\right)\right) \cdot \log n, 2 \right]_{\{0,1\}^{\log n}}$$

Moreover, the verifier performs linear tests which have the projection property.

Proof. We now make the aforementioned construction explicit. We start with a randomness efficient robust PCP with perfect completeness and sub-constant soundness error. To obtain a robust PCP, we use its equivalence to Label-Cover. In particular, we start with a Label-Cover generated by Theorem 6.1 with $\epsilon \simeq 1/(\log n)^{\beta}$. Now we invoke Theorem 5.2 to construct a $(1/(\log n)^{\alpha}, 1/(\log n)^{\beta})$-tester ($\alpha, \beta$ to be fixed at a later point). We then compose the inner with the Label-Cover instance in an obvious way. We summarize the parameters in Table 2. We analyze the parameters of the newly composed verifier.

- Completeness: Since the Label-Cover has perfect completeness, the error is solely contributed by the inner. Invoking Theorem 5.2, we conclude that the completeness is $1 - \frac{1}{(\log n)^{\alpha}}$.
- Soundness: The inner and the outer each contribute $1/(\log n)^{\beta}$ to the error. Hence, the overall soundness is $\frac{1}{2 \cdot (\log n)^{\beta}}$.
- Query Size: This is inherited from the query size of the inner. Since, we use [MR10] to obtain Theorem 5.2, the inner makes 2 queries.
- Alphabet: Again, follows from Theorem 5.2 that alphabet is $\{0,1\}^{(\log n)^{\alpha}} \cdot \mathsf{poly}((\log n)^{\beta})$. We choose α and β such that the product of $(\log n)^{\alpha}$ and $\mathsf{poly}(\log^{\beta} n)$ is $\log n$. Hence, the alphabet used by the proof is $\{0,1\}^{\log n}$.
- Randomness: The randomness of the composed verifier is the sum total of the randomness used by inner and that used by the outer. In our case, it follows from Theorem 5.2 and Lemma 4.5 that the inner uses $\log \log n \cdot \alpha' \cdot \mathcal{O}(r_v + \log \vartheta)$ random bits and the outer uses $\log n \cdot (1 + o(1))$. So, the total number of random bits is $(1 + o(1)) + \log n + \mathcal{O}(1) \cdot \log \log n \simeq (1 + o(1)) \cdot \log n$.

□

Hardness of Label-Cover. PCP's with the projection property can also formulated a certain CSP called the Label-cover problem. An instance of a Label-cover is a bi-partite graph whose edges are between questions of first prover and those of the second prover. For every edge, there is an associated projection that uniquely identifies the label of second vertex given the label of a vertex from the first set. The goal is find a labelling that maximizes the number of satisfied edges. The following formalizes the notion we have been talking in the prequel.

Definition 7 (Label-cover)**.** *An instance of* Label-cover *contains a regular bi-partite multi-graph $G = (A, B, E)$ and two finite sets Σ_A and Σ_B, where $|\Sigma_A| \geq |\Sigma_B|$. Every vertex in A is supposed to get a label from Σ_A, and every vertex in B is supposed to get a label from Σ_B. For each edge $e \in E$ there is a projection $\pi_e : \Sigma_A \to \Sigma_B$ which is a partial function.*

Given a labeling to the vertices of the graph, that is, functions $\psi_A : A \to \Sigma_A$ and $\psi_B : B \to \Sigma_B$, an edge $e = (a, b)$ is said to be "satisfied" if $\pi_e(\psi_A(a)) = \psi_B(b)$ (if $\pi(\psi_A(a))$ is undefined; in which case the edge is deemed to be unsatisfied.) The goal is to find a labeling which maximizes the number of satisfied edges.

We say that γ fraction of the edges are satisfiable if there exists a labeling that satisfies γ fraction of the edges. In the Label-Cover notation, the size corresponds to the number of vertices $|A|+|B|$. The alphabet corresponds to the larger set of labels Σ_A.

The Label-Cover seems to be extremely useful in establishing inapproximability results. Khot's survey [Kho05] is an excellent exposition on the gamut of hardness results that can be obtained from Label-Cover. In the world of provers, the projection property is equivalent to a 2 prover 1 round game and has the following correspondence with the Label-cover: The vertices in A and B are the set of questions that can posed by the verifier to the first prover and the second prover respectively. The set of labels corresponds to the answers of the provers. So, upon receiving an answer from the first prover, the verifier rejects the claim made by the provers or has uniquely determined the answer to the second prover's question.

Theorem 6.2 can be reformulated in the language of Label-Cover as follows.

Corollary 6.2.1. *For every n and for some $\alpha, \beta > 0$ the following holds. Solving an instance of SAT of size n can be reduced to distinguishing between the following two cases of a* Label-Cover *instance.*

- Yes: *There is a labeling that satisfies at least $(1 - \frac{1}{\log n)^\alpha})$-fraction of the edges.*
- No: *Any labeling satisfies at most $(\frac{1}{(\log n)^\beta})$-fraction of the edges.*

Moreover, the size of the Label-Cover *instance is $n^{1+o(1)}$ and every projection is linear in nature.*

7 Hardness Results

We begin by borrowing some machinery from the PCP literature. We will deal with Boolean functions $A : \mathbb{F}_2^u \to \{-1, 1\}$. A function is called linear if $A(x \oplus y) = A(x) \cdot A(y)$, where $\oplus$ denotes vector addition over $\mathbb{F}_2$. Note that there are precisely 2^u linear functions: for every $\alpha \in \mathbb{F}_2^u$, χ_α defined as

$$\chi_\alpha(a) = (-1)^{a \cdot \alpha} \quad \forall a \in \mathbb{F}_2^u$$

Hadamard Codes. Our PCP verifier will expect an encoding of the proof described in the prequel. Specifically, we will use Hadamard codes which we define now.

Definition 8 (Hadamard Encoding)**.** *Hadamard code of $p \in \mathbb{F}_2^u$ is the 2^u bit string $\{\chi_p(a)\}_{a \in \mathbb{F}_2^u}$. We denote it by Hadamard(p).*

Recall that the string $x = Q(W)$ read by the aforementioned verifier is supposed to satisfy certain linear conditions modulo 2. Let these constraints be $h_1 \cdot x = \eta_1, \ldots h_u \cot x = \eta_u$, where $h_1, h_2, h_3 \ldots h_u \in \mathbb{F}_2^{3u}$ and $\eta_1, \eta_2 \ldots \eta_u \in \mathbb{F}_2$.

Folding. Informally, *Folding* is a technique used to enable the verifier to ignore the linear constraint test. Suppose that C is a Hadamard code of x and x satisfies the constraints mentioned above. Let $\mathcal{H}$ be the linear subspace spanned by the vectors $h_1, h_2 \ldots h_u$. Then for any vector b and any vector $h \in \mathcal{H}$, $h = \oplus \rho_i \cdot h_i$, we have

$$C(b \oplus h) = C(b) \cdot (-1)^{\sum_i \rho_i \eta_i}$$

One may generalize this observation, for any function $F : \mathbb{F}_2^{3u} \to \{-1, 1\}$, Let v_b denote the lexicographically smallest vector in the set of vectors $b \oplus \mathcal{H}$ (coset of $\mathcal{H}$). we one may define a function F' as:

$$b = v_b \oplus \rho_i \cdot h_i, \quad \rho_1, \ldots \rho_u \in \mathbb{F}_2 F'(b) = F(b) \cdot (-1)^{\sum_i \rho_i \eta_i}$$

F' is said to be a folding of F over the linear constraints.

The verifier we design requires the supposed Hadamard codes be folded over the respective constraints. This does not alter much. One difficulty that it may introduce is ensuring that this requirement is enforced. This is done as follows: If the verifier is required to read $F(b)$ from the code, it can read $F(v_b)$ and can calculate $F(b)$ from the expression given in the prequel.

A comment on folding of Hadamard codes is due. We will eventually need to show that verifier accepts the purported proofs with a good enough probability, then these "supposed" proofs can be decoded to construct an assignment of the variables which satisfies a significant fraction of the constraints. Decoding of F gives h with probability $\widehat{H}_h^2$. Now, *folding* ensures that any h given by this decoding procedure satisfies the linear constraints on variables. Thus, folding lets us to ignore the linear constraints altogether.

7.1 The PCP verifier

We now define and analyze our PCP verifier whose test s will be linear equations over 3 variables. The verifier expects to have proofs (Π_A, Π_B) which are the Hadamard encoding of the assignments given to the vertices in A and B. The verifier does the following:

1. Pick at random edge $e = (x, y)$. Let f, g be the corresponding Hadamard encoding of the labels of vertices x and y respectively; and h be the corresponding projection function between Σ_A and Σ_B. $h^{-1}(q)$ denotes the unique vector $p \in \Sigma_A$ such that $h(p) = q$.

2. Choose $\mathbf{v}, \mathbf{y}$ at random conditioned on $\mathbf{v} \in \{\pm 1\}^{\log |\Sigma_A|}$ and $\mathbf{y} \in \{\pm 1\}^{\log |\Sigma_B|}$. Accept iff

$$f(\mathbf{v}) \cdot g(\mathbf{y}) = f(h^{-1}(\mathbf{y}) \oplus \mathbf{v})$$

We begin by stating a Theorem of Khot [Kho01]. Khot uses a Hadamard based PCP on an instance of Label-cover produced by applying parallel repetition on a linear PCP with completeness $1 - \epsilon$ and soundness μ. The linear PCP is queried k times in parallel and has a soundness of S. We start with a Label-Cover with better parameters, so we set $k = 1$. We now take the liberity and rephrase Khot's Theorem about the guarantees provided by the verifier V_{lin} (Page 4, Theorem 3.1 in [Kho01]) in the language of Label-Cover.

Theorem 7.1. *Every* Label-Cover *instance with linear projection tests and size n has a verifier V_{lin} which*

- *Uses $r = \log n + \mathcal{O}(1)$ random bits.*
- *Performs linear tests each with arity 3.*
- *Has completeness at least $1 - \epsilon$.*
- *Has soundness $\frac{1}{2} + \delta$ provided $S < \delta^2$, where S is the soundness of* Label-Cover *instance.*

Theorem 7.2. *For some $\alpha, \beta > 0$,* 3SAT *has a verifier $\mathcal{V}$ that has the following properties.*

- *Uses $\log n \cdot \big(1 + o\,(1)\big)$ random bits.*

- *The tests performed by $\mathcal{V}$ are linear over 3 variables.*
- *If the* SAT *instance is satisfiable, $\mathcal{V}$ accepts it with probability at least $1 - \frac{1}{(\log n)^{\alpha}}$.*
- *If it is unsatisfiable, $\mathcal{V}$ accepts any proof with probability at most $\frac{1}{2} + \frac{1}{(\log n)^{\beta'}}$.*

Proof. We now use Khot's verifier V_{lin} on the Label-Cover instance in Corollary 6.2.1. Specifically, set $\epsilon = \frac{1}{(\log n)^{\alpha}}$, and $\delta = \frac{1}{(\log n)^{\sqrt[3]{\beta}}}$. Completeness follows trivially from Theorem 7.1. Since $S < \delta^2$, the soundness of V_{lin} is $\frac{1}{2} + \frac{1}{(\log n)^{\beta'}}$. This completes the proof. □

7.2 Inapproximability Results

The PCP theorem 7.2 immediately yields a hardness of $\frac{1}{2} + \frac{1}{\log n)^{\beta}}$ for 3Lin assuming $\mathsf{P} \neq \mathsf{NP}$. This translates into a hardness factor of $\frac{7}{8} + \frac{1}{(\log n)^{\beta}}$ for 3Sat.

Corollary 7.2.1 (3Lin Hardness)**.** *For some $\alpha, \beta > 0$, given a* 3Lin *instance Φ of size n it is* NP*-hard to distinguish between the following two cases.*

- *There is an assignment which satisfies $(1 - \frac{1}{(\log n)^{\alpha}})$-fraction of Φ.*
- *Every assignment can satisfy at most $(\frac{1}{2} + \frac{1}{(\log n)^{\beta}})$-fraction of Φ.*

Proof. The constraint satisfaction version of Theorem 7.2. □

Corollary 7.2.2 (3Sat Hardness)**.** *For some $\alpha, \beta > 0$, it is* NP*-hard to distinguish if a given* 3Sat *instance Υ of size n which of the following holds.*

- *There is an assignment which satisfies $(1 - \frac{1}{(\log n)^{\alpha}})$-fraction of Υ.*
- *Every assignment can satisfy at most $(\frac{7}{8} + \frac{1}{(\log n)^{\beta}})$-fraction of Υ.*

8 Polynomially Small Completeness Error

In this section, we construct PCPs with polynomially low completeness error and yet, have a low (even sub-constant) soundness error. The bottleneck to achieving polynomially low completeness error was that the number of queries grew hand in hand with the completeness error. Thus, if we set $r = \log \log n$ in Lemma 4.5 to get polynomially small completeness error, the queries made will be $\mathsf{Poly}(n)$. At which point, any technique to rescue the query size will result in either perturbing the completeness error to a constant or soundness error into the $1/\mathsf{polylog}$ regime. And both these means defeat the ends that we set out to achieve as the former the destroys the completeness error. While the latter shall take us into a region which we currently cannot come out of. Thus, both these approaches are ruled out.

However, we cannot allow the query size to grow along the completeness if we are to achieve out ends. The approach we adopt is simple, we use composition to reduce the query size and at this point of time, there are a lot of already established composition theorems. However, none of them will be suitable to our needs as we need a low completeness error and almost all the exant theorems either are non-linear or have a constant error in completeness. So, in a sense we are forced to self-compose with our constructions. And as it turns out it is not the worst thing that can happen. To that end, we start by recollecting Corollary 4.4.1 that we construct after a tensoring and performing sequential repetition on a basic tester obtained from Lemma 3.3.

Corollary. *After performing a round of tensoring and ϑ rounds of sequential repetition on a (c, s)-tester that uses r_v random bits to make q queries, we obtain a $(\mathcal{O}(c^2 \cdot \vartheta), s')$-tester. The new tester uses $r_v + \mathcal{O}(\log \vartheta)$ random bits and makes $\vartheta^4 \cdot q^2$ queries.*

Let us take a moment and reflect the hypothetical parameters that one might obtain if we composed the above Corollary with itself (presented in Table 3). In short, we would double the error while reducing the queries as a function of n (ref. Claim 8.1)[8]. So, our blue print of the construction will be as follows.

Parameters	Inner	Self Composed Inner
`Completeness error`	$c(n)$	$2 \cdot c(n)$
`Soundness error`	$s(n)$	$2 \cdot s(n)$
`Queries`	$q(n)$	$\mathsf{q}(\mathsf{q}(n))$
`Randomness`	$r(n)$	$r(r(n))$
`Alphabet`	$\{0,1\}$	$\{0,1\}$

Table 3: Parameters after Self-Composition of Inner

Claim 8.1. *For every non-decreasing $f : \mathbb{N} \to \mathbb{N}$ such that $f(n) = o(n)$, the following holds for every x.*

- $f\left(f\left(x\right)\right) \leq f(x)$.
- $f\left(f\left(x\right)\right) < x/k$, *for all* $k > 0$.

Proof. The first part of the claim follows from the hypothesis that f is a non-decreasing, sub-linear function. For the latter half, suppose that $f\left(f\left(x\right)\right) = x/k$, for some constant k. This implies $f(x) = O(n)$ and this is a contradiction. Thus, the claim holds. □

Modified Composition

1. We start with the basic (c, s)-tester constructed in Lemma 3.3. We then tensor the (c, s)-tester to obtain a (c^2, s^2)-tester. As in Lemma 4.4.1 we perform ϑ sequential repetitions using a (ϵ, δ)-sampler, for some constant $\epsilon, \delta > 0$, in a randomness efficient manner to obtain $(\vartheta \cdot c^2, \vartheta \cdot s^2)$-tester, denoted by $\mathcal{V}$. The query complexity $\mathsf{q}(n)$ is $\vartheta^4 \cdot \mathsf{q}^2$, where q is the number of queries made by the tester in Lemma 3.3.

2. We now use self-composition to compose $\mathcal{V}$ with itself. As we have witnessed in Table 3, this doubles the error parameters[9]. Thus, we end up with a $(2 \cdot \vartheta \cdot c^2, 2 \cdot \vartheta \cdot s^2)$-tester with a query complexity of $\mathsf{q}(\mathsf{q}(n))$, where n is the size of the outer. Set $\vartheta = 1/s, c = 1/(8 \cdot \vartheta)$ to check that we essentially have a $(\mathcal{O}(c^2), s)$-tester

3. We then iterate over the first two steps $\log n$ times. It follows that the tester we achieve after this is $(1/\mathcal{O}(c^{\log n}), s')$-tester. This is what we would like to achieve as $\mathcal{O}(c^{\log n}) \simeq \mathcal{O}(1/n^{\Omega(1)})$. The query complexity of our construction decreases over each iteration in n. It take a bit of work but it is not hard to show that at the end of it all, we end up with a tester that only makes a constant number of queries.

[8] We prove a strong version of this Claim at a later point
[9] Will be established via Lemma 8.3

However, all is not well as the above sketch suggests. For one, we have designed the basic tester to be utilized as an inner. To be eligible for composition as an outer, we need an robust variant of the same. We shall use this opportunity to refresh the notion of verifier composition. The prover would need to provide an inner-proof ψ_R for every possible random coin R of the outer PCP. Thus, the proof for the composed verifier is $\Psi = \{\psi_R \ : \ R \text{ outer random coins}\}$. Thus, if our tester to be eligible to be used as an outer, it needs to be robust. Informally, any satisfying assignment must be far from an unsatisfying assignment. We now formalize this notion via *robust* assignment testers first introduced by [DR06, BSGH+06].

Definition 9 (Robust Assignment Testers)**.** *An assignment-tester is called ρ-robust if in the soundness case in Definition 4 of an assignment tester, for every assignment $\tilde{\pi} : Y \to \Sigma$, the assignment $(\pi \cup \tilde{\pi})|_{\psi_i}$ is ρ-far from any satisfying assignment of ψ_i on least $1-\epsilon$ fraction of $\Psi = \{\psi_1, ..., \psi_R\}$.*

Dinur and Reingold [DR06] provide a generic transformation that transforms any assignment-tester into a robust one. In particular, they prove the Robustization lemma which is very useful in our quest. Specifically, we use a slight variant of their result – one that includes completeness error. Let us denote a tester that uses R random bits, has a completeness error c, size S, makes q queries, distance parameter δ, soundness ϵ and robustness parameter ρ by $(R, c, S, \mathsf{q}, \delta, \epsilon, \rho)$.

Lemma 8.2 (Robustization Lemma, Lemma 3.6 in [DR06])**.** *There exists some $c_1 > 0$ such that given an assignment tester $\mathcal{A}$ with parameters $(R, c, S, \mathsf{q}, \delta, \epsilon)$, we can construct a ρ-robust assignment tester $\mathcal{A}' = \Psi'$ with parameters $\big(R, c, c_1 \cdot S, \mathsf{q}, \delta, \epsilon, \rho = \Omega(1/q)\big)$.*

Based on Lemma 8.2, we can now transform our tester obtained from Lemma 3.3 into a robust tester. The key parameter for us is the completeness error and this is left untouched by this generic robustization. We also recollect the robust composition lemma from [DR06] which basically establishes that robust assignment-testers compose in a modular fashion.

Lemma 8.3 (Robust Composition Lemma 3.5 in [DR06])**.** *Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be two assignment testers with parameters $(R_1, c_1, \mathsf{S}_1, \mathsf{q}_1, \delta_1, \epsilon_1)$ and $(R_2, c_2, \mathsf{S}_2, \mathsf{q}_2, \delta_2, \epsilon_2)$ respectively. If $\mathcal{A}_1$ is ρ-robust with $\rho = \delta_2$ then one can construct an assignment tester $\mathcal{A}_3$ with parameters $(R_3, c_3, \mathsf{S}_3, \mathsf{q}_3, \delta_3, \epsilon_3)$ such that:*

$$R_3 = R_1(n) \cdot R_2(\mathsf{S}_1(n)), \quad \mathsf{S}_3(n) = \mathsf{S}_1(\mathsf{S}_2(n)), \quad \mathsf{q}_3(n) = \mathsf{q}_2(\mathsf{S}_1(n)), \quad c_3 = c_1(n) + c_2(\mathsf{S}_1(n)) - c_1(n) \cdot c_2(\mathsf{S}_1(n))$$

and

$$\epsilon_3 = \epsilon_1(n) + \epsilon_2(\mathsf{S}_1(n)) - \epsilon_1(n) \cdot \epsilon_2(\mathsf{S}_1(n)), \quad \delta_3(n) = \delta_1(n)$$

8.1 Construction of Tester

We now highlight the iterative construction aimed at constructing PCP's with polynomially small completeness error. The construction involves *three* phases as outlined in Section 8.

Tensoring and Soundness Amplification. We begin by recalling the tester we obtained after tensoring a (c, s)-tester followed by ϑ sequential repetitions in a randomness efficient manner via Corollary 4.4.1.

Corollary (Tensoring + ϑ Repetitions)**.** *After performing a round of tensoring and ϑ rounds of sequential repetition on a (c, s)-tester that uses r_v random bits to make q queries, we obtain a $\left(\mathcal{O}\left(c^2 \cdot \vartheta\right), s'\right)$-tester. The new tester uses $r_v + \mathcal{O}(\log \vartheta)$ random bits and makes $\vartheta^4 \cdot q^2$ queries.*

Proof. Set $\vartheta = 1/s$ and $c = 1/(2 \cdot \vartheta)$ in Lemma 4.4. □

Robustization. In order to compose the tester we have built with itself. It must be robust in the sense of Definition 9. So, our approach will be to robustize the tester via Lemma 8.2 and then compose it with the tester obtained by Corollary 4.4.1 using Lemma 8.3.

Lemma 8.4 (Robust Tester)**.** *There exists some $\gamma > 0, s, delta < 1/4$, such that a (c, s)-tester $\mathcal{T}$ that uses r_v random bits to make q queries to verfiy a Boolean property ψ can be made a ρ-robust assignment tester $\mathcal{A}$ with parameters $\left(r_v, c, \gamma \cdot \mathsf{S}(\mathcal{T}), \mathsf{q}, \delta, \epsilon, \rho = \Omega(\frac{1}{\mathsf{q}})\right)$.*

Proof. We invoke Lemma 8.2 on the (c, s)-tester from Lemma 3.3. □

Self Composition. We now compose the robust tester obtained by Lemma 8.4 with the tester obtained via Corollary 4.4.1 as its inner.

Lemma 8.5. *For some $\gamma, \epsilon > 0$ and every $c > 0$, one can transform the tester obtained in Lemma 4.4.1 into a $\mathcal{V}$ with the following parameters ($\mathcal{O}(r_v)$, $\mathcal{O}(c^2)$, $\mathsf{S}(\mathsf{S}(n))$, $\mathsf{q}(\mathsf{q}(n))$, δ, ϵ)-tester.*

Proof. The idea is very simple. We start with the basic tester (r_v, c^2, $\mathsf{S}(n)$, $\mathsf{q}(n)$, δ, s')-tester obtained from Corollary 4.4.1. We then invoke Lemma 8.2 on it to obtain its robust cousin $\left(r_v, c^2, \gamma \cdot \mathsf{S}(n), \mathsf{q}(n), \delta, \epsilon, \rho = \Omega(1/\mathsf{q}(n))\right)$-tester. We now compose them in an obvious way using Lemma 8.3 to claim the result.

In particular, we begin by setting $\vartheta = 1/s, c = 1/(8 \cdot \vartheta)$ in Lemma 4.4.1 to obtain an $(1/64 \cdot \vartheta, s')$-tester with q queries. Upon composition with a robust version of itself we end up with a $c = 1/32 \cdot \vartheta = 1/2 \cdot 1/64 \cdot \vartheta \simeq \mathcal{O}(c^2)$ and $s = \Omega(\frac{1}{\mathsf{q}}) = \mathcal{O}(1) \simeq \epsilon$[10]. The rest of the parameters follow vacuously. □

Iterated Tester

We are now almost set to iterate over Corollary 4.4.1, Lemmas 8.4 and 8.5 to obtain our ends. But before we go any further, we would like to introduce a couple of definitions and make a few observations about the function $\mathsf{q}(n)$. At the very outset, we make it clear that in what follows we deal with only sub-linear functions. That is, functions whose output is small than its input by a $\mathsf{poly}(\cdot)$ quantity. Examples of such functions include $\log n$, $\mathsf{poly}\log n$, n^ϵ, for $\epsilon < 1$.

Definition 10 (`Repetition`)**.** *We define* `Repetition`$(\mathcal{V})$ *as an invocation of Corollary 4.4.1 on $\mathcal{V}$, followed by robustization using Lemma 8.4, and finally self composition using Lemma 8.3. Further, for any $k > 0$ we define* `Repetition`$^k(\mathcal{V})$ *recursively as* `Repetition`$($`Repetition`$^{k-1}(\mathcal{V}))$.

Corollary 8.5.1. *For every $c > 0$ and $\delta < 1/4$, there are constants $k, \delta, > 0$ such that* `Repetition`$(\mathcal{T})$ *yields a $(k \cdot r_v, \mathcal{O}(c^2), \mathcal{O}(\mathsf{S}^k), \delta)$-tester.*

Proof. Essentially restating Lemma 8.5 expect for we use an explicit constant to account for the randomness used by in a single invocation of `Repetition`. □

[10] Since, this is a one-time composition, it is a constant.

Definition 11. *Let $\mathcal{T}$ be an assignment tester with a query function $\mathsf{q} : \mathbb{N} \to \mathbb{N}$. For any i, we define $\mathsf{q}^i(n)$ as the query size of the tester produced after $\texttt{Repetition}^i(\mathcal{T})$.*

Proposition 8.6. *For every tester $\mathcal{T}$ with query complexity $\mathsf{q}(n)$, $k > 0$, $\mathsf{q}^k(n) = \left[\mathsf{q}^{k-1}\left((\mathsf{q}^{k-1}(n))^2\right)\right]^2$.*

Proof. Follows from the definition of `Repetition`. The query size of the input tester to the $\texttt{Repetition}^i$ is q^{i-1}. Recall Corollary 4.4.1 to conclude that the tensor and sequential repetition operations increase the query function to $Q = \mathcal{O}(\mathsf{q}^{i-1}(n))^2$. The robustization step does not increase the number of queries. And from Lemma 8.3, the self composition step results in a query size of $Q(Q(n))$, which is $\left[\mathsf{q}^{k-1}\left((\mathsf{q}^{k-1}(n))^2\right)\right]^2$. □

Lemma 8.7. *The sequence $\mathcal{Q}_{i\in\mathbb{N}} = \{\mathsf{q}^i(n)\}$ is well behaved in the sense that given q^k, one can compute q^{k-1} and q^{k+1} upto constant factors.*

Proof. In order to be able to compute q^{k-1} we would need uncompute the effect of the k-th `Repetition` and we are perfectly capable to do *à la* Proposition 8.6. The argument for computing q^{k+1} follows arguments from Proposition 8.6. □

Corollary 8.7.1 (Smoothness Property)**.** *The sequence $\mathcal{Q}_{i\in\mathbb{N}} = \{\mathsf{q}^i(n)\}$ is smooth in the following sense.*

$$\mathsf{q}^k = \Theta(f) \implies \mathsf{q}^{k-1} = \Theta(\tilde{f}) \quad \text{and} \quad \mathsf{q}^{k+1} = \Theta(\hat{f})$$

Moreover, both $\tilde{f}$ and $\hat{f}$ are sub-linear in their input.

Definition 12 ($\mathcal{Q} = \Upsilon(f)$)**.** *For any $f : \mathbb{N} \to \mathbb{N}$ and any $\mathcal{Q}_{i\in\mathbb{N}} = \{\mathsf{q}^i(n)\}$, we say $\mathcal{Q} = \Upsilon(f)$ if there is a constant C, such that $\mathsf{q}^i(n) = \Theta(f^i(n))$ for every $i > C$, where $f^i(n)$ is the function obtained by composing f with itself for i times.*

Lemma 8.8. *The sequence $\mathcal{Q}_{i\in\mathbb{N}} = \{\mathsf{q}^i(n)\}$ decreases faster than any function in n. In other words, there is no f such $\mathcal{Q} = \Upsilon(f)$.*

Proof. We give a proof by contradiction. Suppose there is a function f such that $\mathcal{Q} = \Upsilon(f)$. Thus for every $i > C$, $\mathsf{q}^i(n) = \Theta(f^i(n))$. Since, every element in $\mathcal{Q}$ can be used to good effect in computing its predecessor, we can go backwards until the point where we have reached the start of the sequence or q^j is no longer sub-linear. In the former, we arrive at a contradiction as the initial tester in the sequence had a query complexity independent of n or a constant number of queries. In the latter case, we contradict the smoothness property of Corollary 8.7.1 as $\mathsf{q}^j = \Theta(f)$ implies the existence of $\mathsf{q}^{j-1} = \Theta(\hat{f})$. □

Proposition 8.9. *$\mathcal{Q}_{i\in\mathbb{N}} = \{\mathsf{q}^i(n)\}$ is independent of n. In other words, any $\mathsf{q}^i(n) = \mathcal{O}(1)$.*

Proof. It follows from Lemma 8.8 that the sequence $\mathcal{Q}$ is not bounded from below by any $f(n)$. Thus, $\mathcal{Q}$ is independent of n. □

Theorem 8.10. *For every $k > 0, c, \delta < 1/4$, there exists $m = k(\cdot)$ and constants $\mathsf{q}', \delta, \epsilon > 0$ such that $\texttt{Repetition}^m(r_v, c, \mathsf{S}, \mathsf{q}, \delta)$ tester yields a $(r_v \cdot \log k, c^k, \mathsf{S}^k, \mathsf{q}', \delta, \epsilon)$-tester. Moreover, the transformation takes time polynomial in $2^{\log k \cdot r_v}$. Also, the tester performs only linear tests.*

Proof. Recall that even though sequential repetition, self composition increase the size, the tensoring step dominates the rest of them asymptotically. Since in this study, we primarily deal with asymptotics, we can focus on the blow-up due to tensoring. Repeated tensoring essentially squares the size of the input instance. It is easy to verify that the sequence generated is $\{2^{2^{i-1}}\}_i$.

Corollary 8.5.1 gives the parameters after a single invocation of `Repetition`. To establish the Theorem would essentially require us to export each parameter across each invocation of `Repetition`. We invoke it with $m = \log\log k$. We now analyze and argue the parameters of the tester.

- *Randomness:* As highlighted in the prequel, the randomness used is asympotically dominated by the tensoring operation. Thus, we would square the randomness used at every iteration. Hence if the input tester uses r_v bits, after m iterations the randomness blows up to $r_v \cdot 2^{2^m} \simeq r_v \cdot k$.
- *Completeness:* Again, we square the completeness error in each iteration, the error falls doubly exponential in m, which is $\mathrm{O}(k)$. Hence, the completeness error at the end of m iterations is $\mathcal{O}(c^k)$.
- *Query Size:* We recall Proposition 8.9 to conclude that the query $\mathsf{q}(n)$ is independent of n and thus, we maintain a constant query size after each invocation.
- *Soundness:* We invoke Corollary 8.5.1 and Proposition 8.9 to infer that after each invocation of `Repetition`, the soundness remains independent of other parameters and thus, we end up with constant soundness.
- *Distance Parameter:* In our construction, this is same as the soundness.

The linearity of the tests follows from the observation that that the basic steps of this operation including tensoring, sequential repetition, self composition preserve linearity. □

8.1.1 Soundness Amplification

The tester obtained by Theorem 8.10 has a constant soundness, for some arbitrary constant. We now amplify the soundness of the tester whilst losing modest factors in the completeness. Again, we would like to amplify the soundness whilst retaining the good properties which are helpful in making the composition modular even possibly at the expense of a non-binary alphabet. In fact, to achieve non-constant soundness we would need larger alphabet. To this end, we shall use Locally Decode and Reject Codes of Moshkovitz and Raz [MR10]. To amplify the soundness, we use Theorem 5.1 on the tester obtained from Theorem 8.10. The construction follows.

Lemma 8.11 (Soundness Amplification)**.** *For every n and $\epsilon > 0$ (that can be function of n), every $(r_v \cdot k, c^k, \mathsf{S}^k, \mathsf{q}, \epsilon_o)$ can be transformed into $(r_v \cdot k \cdot (1 + o(1)), \mathcal{O}(c^k), \mathsf{S}^k \cdot (1 + o(1)), 2, \epsilon)$.*

Proof. Follows by invoking Theorem 5.1 on the tester given by Lemma 8.10. We now analyse the parameters of the tester.

- *Completeness:* The completeness error of the tester obtained in Theorem 8.10 is c^k. Theorem 5.1 introduces an error which is proportional to the soundness error of the PCP obtained by applying it. In fact, Theorem 5.1 is essentially a parallel-repetition theorem in the low-error regime, albeit with a much worse alphabet tradeoff. Thus, the completeness error is that product of $c^k \cdot 1/\epsilon$. In other words, the error is not more than $c^{k-\epsilon'}$, we choose $k = \omega(\epsilon')$ to get the parameters we want.
- *Soundness:* It follows from Theorem 5.1 that the soundness of the new construction is some $\epsilon^{O(1)} > \epsilon$.

- *Alphabet:* We inherit the alphabet of Theorem 5.1, which happens to be $\{0,1\}^{\mathsf{q}\cdot\mathsf{poly}\frac{1}{\epsilon}}$, where $\mathsf{poly}(\cdot)$ is implicit in Definition 6.
- *Randomness:* Theorem 5.1 blows up the randomness of the input tester to $1+o(1)\cdot\mathsf{R}$, where R is randomness of the tester obtained from Lemma 4.5. For the parameters that we have chosen, it is $\big(1+o\left(1\right)\big)\cdot k\cdot r_v$.

□

8.2 Final Composition

Theorem 8.12 (Main Theorem)**.** *For every n and $\alpha,\epsilon>0$,*

$$3SAT\in PCP_{1-n^{\frac{\alpha'}{\alpha'+1}},\epsilon}\big[(1+\alpha)\cdot\log n,2\big]_{\{0,1\}^{\mathsf{poly}\frac{1}{\epsilon}}}$$

Moreover, verifier's test are linear, and also have the projection property.

Proof. We start with a randomness efficient robust PCP with perfect completeness and sub-constant soundness error. To obtain a robust PCP, we use its equivalence to Label-Cover. In particular, we start with a Label-Cover generated by Theorem 6.1 with a soundness parameter ϵ. Now we invoke Theorem 8.11 to construct a $\big(1/n^{\alpha},\epsilon\big)$-tester. We then compose the inner with the Label-Cover instance in an obvious way. We analyze the parameters of the newly composed verifier.

- Completeness: Since the Label-Cover has perfect completeness, the error is solely contributed by the inner. Invoking Theorem 5.2, we conclude that the completeness is $1-n^{\alpha'+1/\alpha'}$, which is $1-n^{\frac{\alpha'+1}{\alpha'}}$.
- Soundness: The inner and the outer each contribute $1/\epsilon$ to the error. Hence, the overall soundness is $1/\mathcal{O}\left(\epsilon\right)$.
- Query Size: This is inherited from the query size of the inner. Since, we use [MR10] to obtain Theorem 5.2, the inner makes 2 queries.
- Alphabet: Again, follows from Theorem 5.2 that alphabet is $\{0,1\}^{\mathsf{poly}\frac{1}{\epsilon}}$.
- Randomness: The randomness of the composed verifier is the sum total of the randomness used by inner and that used by the outer. In our case, it follows from Theorem 5.2 and Lemma 4.5 that the inner uses $\log n\cdot\alpha'\cdot(1+o(1))$ random bits and the outer uses $\log n\cdot\big(1+o(1)\big)$. So, the total number of random bits is $\big(1+o\left(1\right)\big)\cdot\alpha'\cdot\log n+\big(1+o\left(1\right)\big)\cdot\log n\ \simeq\ \big(1+\alpha+o(1)\big)\cdot\log n$.

□

Corollary 8.12.1 (Label-Cover)**.** *For every n, α and some $\beta>0$ the following holds. Solving an instance of SAT of size n can be reduced to distinguishing* Label-Cover *instances between the following two cases.*

- Yes: *There is a labeling that satisfies at least $(1-\frac{1}{n^{\alpha'}})$-fraction of the edges.*
- No: *Any labeling satisfies at most $(\frac{1}{(\log n)^{\beta}})$-fraction of the edges.*

The size of the Label-Cover *instance is $n^{1+\alpha}$ and every projection is linear in nature.*

Corollary 8.12.2 (3Lin Hardness)**.** *For every α, there is some $\beta > 0$, the following holds. Given a* 3Lin *instance Φ of size n, it is* NP*-hard to distinguish between the following two cases.*

- Yes: *There is an assignment which satisfies $(1 - \frac{1}{n^{\alpha'}})$-fraction of Φ.*
- No: *Every assignment can satisfy at most $(\frac{1}{2} + \frac{1}{(\log n)^{\beta}})$-fraction of Φ.*

Proof. Plugging in the Label-Cover obtained from Corollary 8.12.1 instead of the one obtained by Corollary 6.2.1 in Theorem 7.2 by setting $\epsilon = \frac{1}{n^{\alpha}}$, and $\delta = \frac{1}{(\log n)^{\sqrt[3]{\beta}}}$. Completeness follows trivially from Theorem 7.1. Since $S < \delta^2$, the soundness of V_{lin} is $\frac{1}{2} + \frac{1}{(\log n)^{\beta'}}$. This completes the proof. □

Corollary 8.12.3 (Min-3Lin-Deletion Hardness)**.** *For every α and some $\beta > 0$ the following holds. Given a* Max-3Lin-Deletion *$\mathcal{D}$ instance of size n it is* NP*-hard to distinguish between the following two cases.*

- Yes: *There is a $(\frac{1}{n^{\alpha'}})$-fraction of equations whose removal makes $\mathcal{D}$ satisfiable.*
- No: *At least β-fraction of the equations in $\mathcal{D}$ need to be removed from it to make it satisfiable.*

Proof. An alternate view of Corollary 8.12.2. □

Corollary 8.12.4 (3Sat Hardness)**.** *For every $\alpha > 0$ and some $\beta > 0$, given a* 3Sat *instance Υ of size n it is* NP*-hard to distinguish between the following two cases.*

- Yes: *There is an assignment which satisfies $(1 - \frac{1}{n^{\alpha'}})$-fraction of Υ.*
- No: *Every assignment can satisfy at most $(\frac{7}{8} + \frac{1}{(\log n)^{\beta}})$-fraction of Υ.*

9 Nearest Codeword Problem

The Nearest Codeword Problem (NCP) (also known as the maximum likelihood decoding problem) is the following. The input instance consists of a generator matrix $\mathbf{A} \in \mathbb{F}_2^{k \times n}$ and a received word $\mathbf{x} \in \mathbb{F}_2^n$ and the goal is to find the nearest codeword $\mathbf{y} \in \mathcal{A}$ to $\mathbf{x}$. One relaxation that received attention was estimating the minimum error weight to the nearest codeword. In other words, the distance $d(\mathbf{x}, \mathbf{y})$ to the nearest codeword, without necessarily finding the codeword $\mathbf{y}$.

Prior Results. Berlekamp, McEliece and van Tilborg [BMvT78] showed that even the relaxed version is NP-hard. Arora *et al.* [ABSS93] established that the error weight is hard to approximate to any constant is NP-hard and $2^{\log^{(1-\epsilon)} n}$ for any $\epsilon > 0$ under $\mathsf{NP} \not\subseteq \mathsf{DTIME}(2^{\mathsf{polylog}})$. This was improved to inapproximability to within $n^{1/\mathcal{O}(\log \log n)}$ under $\mathsf{P} \neq \mathsf{NP}$ by Dinur *et al.* [DKRS03]. On the algorithmic front, Berman and Karpinski [BK02] gave a randomized algorithm for $\epsilon \cdot n / \log n$ approximation, for any fixed $\epsilon > 0$ and $\epsilon \cdot n$ approximation in deterministic time.

In the gap version of nearest codeword problem (NCP), we are given a linear code $\mathcal{A}$, target vector $\mathbf{v}$ and an integer t with a promise that either the Hamming distance of $\mathbf{v}$ to $\mathcal{A}$ is *less than* $\gamma \cdot t$ or greater than t, one must decide which of them is true. In this section, we show that for the nearest codeword problem of any linear code over a *binary* alphabet cannot be approximated within n^{α}, for some $\alpha > 0$ unless $\mathsf{P} = \mathsf{NP}$. Before we establish the theorems, we shall define the corresponding promise problems which capture the hardness of approximating the aforementioned problems within a factor of γ.

Definition 13 (Nearest Codeword Problem)**.** *For $\gamma \geq 1$ and $t < 1$, an instance of* GapNcp$_\gamma$ *is a triplet $(\mathbf{A}, \mathbf{v}, t)$, where $\mathbf{A} \in \mathbb{F}_2^{k \times n}$, $\mathbf{v} \in \mathbb{F}_2^n$ and $t \in \mathbb{Z}^+$, such that*

- *$(\mathbf{A}, d)$ is a* Yes *instance if $d(\mathcal{A}, \mathbf{v}) \leq \gamma t$.*
- *$(\mathbf{A}, d)$ is a* No *instance if $d(\mathcal{A}, \mathbf{v}) > t$.*

We begin by recollecting Corollary 8.12.3.

Corollary (Min-3Lin-Deletion Hardness)**.** *For some $\alpha, \beta > 0$ and* Max-3Lin-Deletion *$\mathcal{D}$ instance of size n it is* NP*-hard to distinguish between the following two cases.*

- *There exists a $\left(\frac{1}{n^\alpha}\right)$-fraction of equations whose removal makes $\mathcal{D}$ satisfiable.*
- *One needs to delete more than β-fraction of the equations in $\mathcal{D}$ to make it satisfiable.*

We are now ready to prove the hardness of approximation of GapNcp$_{n^{-\beta}}$.

Theorem 9.1. *There exists an absolute constant $\beta > 0$ such that it is* NP*-hard to approximate* GapNcp$_{n^{-\beta}}$.

Proof. We reduce Min-3Lin-Deletion$_{\frac{1}{n^\beta}, \frac{1}{2}}$ to GapNcp$_{n^{-\beta}}$. Given an instance $\mathcal{L} \equiv \mathbf{AX} + \mathbf{y} = \mathbf{0}$ of Min-3LinDel$_{\frac{1}{n^\beta}, \frac{1}{2}}$, we use the $n^{-\beta}$ approximation scheme for GapNcp available to us on $(\mathbf{A}, \mathbf{y}, 1/2)$.

- Yes Case: Here the deletion value of the instance is less than $1/n^\beta$. Thus, number of unsatisfied equations is at most $n^{-\beta}$. So, Hamming weight of $\mathbf{AX} + \mathbf{y}$ is at most $n^{-\beta}$. Invoking Proposition 2.2, we infer that the Hamming distance between $(\mathbf{AX}, \mathbf{y})$ is at most $n^{-\beta}$. Since, we have an access to $\mathcal{O}\left(n^{-\beta}\right)$ approximation algorithm to estimate the nearest codeword of any linear code. For the parameters, the algorithm must accept $(\mathbf{AX}, \mathbf{y}, 1/2)$.
- No Case: In this case, more than $1/2$ constraints are left unsatisfied by every assignment. Thus, the Hamming distance between $(\mathbf{AX}, \mathbf{y})$ is strictly greater than $1/2$ and the approximation algorithm is forced to reject the instance $(\mathbf{AX}, \mathbf{y}, 1/2)$.

This completes the reduction. □

10 Concluding Remarks

In a nutshell, we are have laid out a plan to initiate the study of completeness amplification of PCPs. We, have, also identified some immediate applications to our new constructions. We are sure that the theory of completeness amplification will find more applicability elsewhere in near future.

References

[ABS10] S. Arora, B. Barak, and D. Steurer. Subexponential algorithms for unique games and related problems. In *Proc. 51st IEEE Symp. on Foundations of Computer Science*, 2010. 3

[ABSS93] S. Arora, L. Babai, J. Stern, and Z. Sweedyk. The hardness of approximate optima in lattices, codes, and systems of linear equations. *Foundations of Computer Science, IEEE Annual Symposium on*, 0:724–733, 1993. 27

[ALM+98] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and the hardness of approximation problems. *Journal of the ACM*, 45(3):501–555, 1998. 2

[AS98] S. Arora and S. Safra. Probabilistic checking of proofs: a new characterization of NP. *Journal of the ACM*, 45(1):70–122, 1998. 2

[AS03] S. Arora and M. Sudan. Improved low-degree testing and its applications. *Combinatorica*, 23(3):365–426, 2003. 2

[BFL91] L. Babai, L. Fortnow, and C. Lund. Nondeterministic exponential time has two-prover interactive protocols. *Computational Complexity*, 1:3–40, 1991. 2

[BFLS91] L. Babai, L. Fortnow, L. A. Levin, and M. Szegedy. Checking computations in polylogarithmic time. In *Proc. 23rd ACM Symp. on Theory of Computing*, pages 21–32, 1991. 2

[BGS98] M. Bellare, O. Goldreich, and M. Sudan. Free bits, PCPs, and nonapproximability—towards tight results. *SIAM Journal on Computing*, 27(3):804–915, 1998. 9

[BK02] Piotr Berman and Marek Karpinski. Approximating minimum unsatisfiability of linear equations. In *SODA*, pages 514–516, 2002. 27

[BMvT78] E. Berlekamp, R. McEliece, and H. van Tilborg. On the inherent intractability of certain coding problems (corresp.). *Information Theory, IEEE Transactions on*, 24(3):384 – 386, may 1978. 27

[BSGH+06] E. Ben-Sasson, O. Goldreich, P. Harsha, M. Sudan, and S. Vadhan. Robust PCPs of proximity, shorter PCPs, and applications to coding. *SIAM Journal on Computing*, 36(4):889–974, 2006. 8, 15, 22

[DFK+99] I. Dinur, E. Fischer, G. Kindler, R. Raz, and S. Safra. PCP characterizations of NP: Towards a polynomially-small error-probability. In *Proc. 31st ACM Symp. on Theory of Computing*, pages 29–40, 1999. 2

[DH09] I. Dinur and P. Harsha. Composition of low-error 2-query pcps using decodable pcps. In *FOCS*, pages 472–481, 2009. 16

[Din10] I. Dinur. Probabilistcally checkable proofs and codes. Technical report, Plenary Talk at International Congress of Mathematicans (ICM), 2010. 2

[DKRS03] Irit Dinur, Guy Kindler, Ran Raz, and Shmuel Safra. Approximating cvp to within almost-polynomial factors is np-hard. *Combinatorica*, 23(2):205–243, 2003. 27

[DM11] I. Dinur and O. Meir. Derandomized parallel repetition via structured pcps. *Computational Complexity*, 20(2):207–327, 2011. 2

[DR06] I. Dinur and O. Reingold. Assignment testers: Towards a combinatorial proof of the PCP theorem. *SIAM Journal on Computing*, 36(4):975–1024, 2006. 8, 15, 16, 22

[FGL+96] U. Feige, S. Goldwasser, L. Lovasz, S. Safra, and M. Szegedy. Interactive proofs and the hardness of approximating cliques. *Journal of the ACM*, 43(2):268–292, 1996. 2

[Gol97] O. Goldreich. A sample of samplers - a computational perspective on sampling. Technical Report TR97-020, Electronic Colloquium on Computational Complexity, 1997. 13

[Hås00] J. Håstad. On bounded occurrence constraint satisfaction. *Inf. Process. Lett.*, 74(1-2):1–6, 2000. 3

[Hås01] J. Håstad. Some optimal inapproximability results. *Journal of the ACM*, 48(4):798–859, 2001. 3, 9

[IKW09] R. Impagliazzo, V. Kabanets, and A. Wigderson. New direct-product testers and 2-queryPCPs. In *Proceedings of the 41st annual ACM symposium on Theory of computing*, STOC '09, pages 131–140, 2009. 2

[Kho01] S. Khot. Improved inapproximability results for maxclique, chromatic number and approximate graph coloring. In *Proc. 42nd IEEE Symp. on Foundations of Computer Science*, pages 600–609, 2001. 2, 19

[Kho02] S. Khot. On the power of unique 2-prover 1-round games. In *Proc. 34th ACM Symp. on Theory of Computing*, pages 767–775, 2002. 2

[Kho05] S. Khot. Guest column: inapproximability results via long code based PCPs. *SIGACT News*, 36(2):25–42, 2005. 18

[Kho10] S. Khot. On the unique games conjecture. In *2010 IEEE 25th Annual Conference on Computational Complexity (CCC)*, pages 99 – 121, 2010. 2

[KP06] S. Khot and A. K. Ponnuswami. Better inapproximability results for maxclique, chromatic number and min-3lin-deletion. In *Proc. Automata, Languages and Programming, 33rd International Colloquium*, pages 226–237, 2006. 3

[LFKN90] C. Lund, L. Fortnow, H. J. Karloff, and N. Nisan. Algebraic methods for interactive proof systems. In *IEEE Symposium on Foundations of Computer Science*, pages 2–10, 1990. 2

[MR10] D. Moshkovitz and R. Raz. Two query PCP with sub-constant error. *Journal of the ACM*, 57(5):1–29, 2010. 2, 3, 14, 15, 16, 17, 25, 26

[Raz98] R. Raz. A parallel repetition theorem. In *SIAM Journal on Computing*, volume 27, pages 763–803, 1998. 2

[RS96] R. Rubinfeld and M. Sudan. Robust characterizations of polynomials with applications to program testing. *SIAM Journal on Computing*, 25(2):252–271, 1996. 2

[RS97] R. Raz and S. Safra. A sub-constant error-probability low-degree test and a sub-constant error-probability PCP characterization of NP. In *Proc. 29th ACM Symp. on Theory of Computing*, pages 475–484, 1997. 2