Algebraic Construction of Projection PCPs

Dana Moshkovitz *

February 1, 2012

Abstract

In a projection PCP, also known as Label-Cover, the verifier makes two queries to the proof, and the answer to the first query determines at most one satisfying answer to the second query. Projection PCPs with low error probability are the basis of most NP-hardness of approximation results known today. In this essay we outline a construction of a projection PCP with low error and low blow-up. This yields sharp approximation thresholds and tight time lower bounds for approximation of a variety of problems, under an assumption on the time required for solving certain NP-hard problems exactly. The approach of the construction is algebraic, and it includes components such as low error, randomness-efficient low degree testing and composition of projection PCPs.

1 Introduction

In 1991 it was discovered [11] that an NP-hardness result for approximating Clique could be obtained from what would later be known as the PCP Theorem [2, 1]:

> *Every NP language L has a verifier that uses $O(\log n)$ random bits to make $O(1)$ queries to a proof over alphabet of size $O(1)$. Given an input $x \in L$, there is a proof that the verifier always accepts; given an input $x \notin L$, for any proof, the probability the prover accepts is at most $\varepsilon = \frac{1}{2}$.*

Ever since, researchers have attempted to prove strong PCP theorems in order to achieve strong hardness of approximation results. The PCP theorem that turned out to be most useful for hardness of approximation was a "projection" PCP: the verifier makes two queries to the proof, upon seeing the answer to the first query, there is at most one acceptable answer to the second query (projection). The verifier accepts a correct proof with probability 1, and accepts an alleged proof of an incorrect statement with at most a small error probability ε.

The strongest projection PCP theorem known, proved by the author and Raz in 2008, has an error ε that tends to 0 with the proof length, while incurring almost-linear blow-up when transforming a proof in a standard format to a probabilistically checkable format [22]. Using techniques pioneered by Bellare-Goldreich-Sudan [4] and Håstad [13], this PCP implies a large number of sharp hardness results, both in terms of the approximation factor and in terms of the time lower bound. For instance, consider Max-3Sat, where given clauses, each over three Boolean variables, the task is to find an assignment to the variables that satisfies as many clauses as possible. By the probabilistic method, there is always an assigment that satisfies 7/8 fraction of the clauses. However, approximating Max-3Sat to within a factor which is

*dmoshkov@csail.mit.edu. Department of Electrical Engineering and Computer Science, MIT.

$7/8 + o(1)$ requires nearly the same time as solving SAT exactly, and the latter is conjectured to require exponential time. For an exposition of the technique of basing optimal hardness results on projection PCP, see Khot's complexity column 36 [15].

The purpose of this article is to outline the construction of a projection PCP as above from the beginning. The emphasis is on what is done and why it is done. Some details are omitted, and proofs are only sketched. This allows us to present in the next pages a proof which in its fullest spans a number of long research papers [6, 21, 20, 22, 10].

1.1 Organization

In this essay we describe the algebraic construction of a low-error projection PCP. The construction consists of the following steps:

1. Translate 3SAT to the problem of verifying whether low degree multivariate polynomials over a finite field have a common root.
2. Perform the "sum-check" protocol that yields a simplified PCP for the common root problem.
3. Convert the simplified PCP into a projection PCP, using "low degree testing", i.e., a probabilistic test for whether a function is approximately of low degree. This projection PCP has a large alphabet inherited from the low degree testing.
4. Decrease the alphabet of the projection PCP by an operation called "composition".

This essay is organized as follows. In Section 2, we give some preliminaries, and in Section 3 we give formal definitions of PCPs and projection PCPs, as well as survey existing constructions. In Section 4 we discuss encoding by low degree polynomials. In Section 5 we describe low degree testing, including a sketch of the analysis, and construction of randomness-efficient tests. In Section 6 we discuss the sum-check protocol that yields PCPs with a small number of queries. In Section 7 we describe how to combine sum-check with low degree testing to achieve a projection PCP with large alphabet. In Section 8 we describe alphabet reduction that preserves projection and low error, and thus finishes the construction of the projection PCP. The presentation is modular. Section 5 that describes low degree testing can be read by itself. Sections 4 and 6 that present the sum-check protocol can be read by themselves. Section 8 that describes alphabet reduction via composition can also be read separately.

2 Preliminaries

2.1 Error Correcting Codes

The (relative) *Hamming distance* between two strings $x, y \in \Sigma^n$ is the fraction of positions on which they differ, i.e., $\Delta(x,y) = |\{1 \le i \le n | x_i \neq y_i\}|/n$. The (relative) *agreement* between x and y is $1 - \Delta(x,y)$.

Definition 1 (Code)**.** *An $(n,k,d)_\Sigma$-code is a subset $C \subseteq \Sigma^n$, where $|C| = |\Sigma|^k$, and for every two different codewords $x \neq y \in C$, it holds that $\Delta(x,y) \ge d/n$.*

Above n is the *length* of the code, k is the *dimension* of the code, d is the *distance* of the code, and Σ is the *alphabet* of the code. If $E : \Sigma^k \to \Sigma^n$ is such that $E(\Sigma^k) = C$, then we call

E an *encoding* function of C. An encoding function maps every possible message $x \in \Sigma^k$ to a codeword of C.

The ratio $R \doteq k/n$ is the *rate* of the code, and we want it to be as large as possible. The ratio $\delta \doteq d/n$ is the *relative distance* of the code, and we also want it to be as large as possible. The Plotkin bound states that $R \leq 1 - \frac{|\Sigma|}{|\Sigma|-1}\delta + o(1)$ [24]. It implies that the rate and the distance cannot be close to 1 simultaneously, as well as that the alphabet has to be sufficiently large to allow relative distance close to 1. Constructions that come close to matching the Plotkin bound are known, e.g., algebraic geometric codes [29]:

Lemma 2.1. *For every $n \geq 1$, $\varepsilon > 0$, there is an explicit $(n, k, d)_\Sigma$-code with relative distance $1 - \varepsilon$, alphabet $|\Sigma| = poly(1/\varepsilon)$ and rate $poly(\varepsilon)$.*

We also need the following proposition that states that no string can be close in Hamming distance to too many codewords:

Proposition 2.2. *Let C be an $(n, k, d)_\Sigma$ code. For any $w \in \Sigma^n$, for any $\alpha \geq 2\sqrt{\delta}$, there are at most $2/\alpha$ codewords in C with agreement at least α with w.*

For more details about coding theory, see the lecture notes of a course by Madhu Sudan [28].

2.2 Expanders

An *expander* is an undirected graph that resembles a random graph. In a D-regular expander, the number of edges between any two sets of vertices is approximately the expected number of edges between those sets in a *random* D-regular undirected graph.

For a graph $G = (V, E)$ and two sets of vertices $X, Y \subseteq V$, let the set of edges between X and Y be $E(X, Y) \doteq \{(x, y) \in E \mid x \in X, y \in Y\}$. It can be shown that any D-regular undirected graph whose adjacency matrix has low second largest eigenvalue (in absolute value) λ satisfies the following.

Lemma 2.3 (Expander mixing lemma)**.** *Let $G = (V, E)$ be a D-regular undirected graph, whose adjacency matrix has second largest eigenvalue (in absolute value) λ. Then, for any two sets $X, Y \subseteq V$,*

$$\left| \frac{|E(X,Y)|}{D\,|V|} - \frac{|X|}{|V|} \cdot \frac{|Y|}{|V|} \right| \leq \frac{\lambda}{D} \cdot \sqrt{\frac{|X|}{|V|} \cdot \frac{|Y|}{|V|}}$$

It can be shown that $\lambda \leq O(\sqrt{D})$ (see, e.g., [14]). We use the explicit construction guaranteed by the following lemma (for a proof see, e.g., [22]):

Lemma 2.4 (Explicit construction of expanders)**.** *Given $n \geq 1$ and $D \geq 3$, there is an explicit construction of a graph $G = (V, E)$ with $|V| = n$ that is regular with degree $O(D)$ and whose adjacency matrix has second largest eigenvalue (in absolute value) $\lambda \leq O(D^\alpha)$ for some constant $\alpha < 1$.*

2.3 Subspaces

Let $\mathbb{F}$ be a field, and let m be a natural number. An *affine subspace* of dimension k is defined by a shift $x \in \mathbb{F}^m$ and linearly independent directions $y_1, \ldots, y_k \in \mathbb{F}^m$ as

$$\{x + t_1 y_1 + \cdots + t_k y_k \mid t_1, \ldots, t_k \in \mathbb{F}\}.$$

A *line* is an affine subspace of dimension 1, a *plane* is an affine subspace of dimension 2, and a *cube* is an affine subspace of dimension 3. A k-dimensional *manifold* of degree d is defined by k-variate polynomials $p_1, \ldots, p_m$ of degree at most d, as $\{ (p_1(t_1, \ldots, t_k), \ldots, p_m(t_1, \ldots, t_k)) \mid t_1, \ldots, t_k \in \mathbb{F}\}$. An affine subspace can be thought of as a manifold of degree $d = 1$.

3 PCPs and Projection PCPs

Projection PCPs can be described in terms of the Label-Cover problem:

Definition 2 (Label Cover)**.** *The input to* Label-Cover *consists of a bipartite graph $G = (A, B, E)$, finite alphabets Σ_A, Σ_B, and projections $\{\pi_e\}_{e \in E}$, where the π_e's are partial functions $\Sigma_A \to \Sigma_B$. An assignment $f_A : A \to \Sigma_A$, $f_B : B \to \Sigma_B$ satisfies an edge $e = (a, b) \in E$ if $\pi_e(f_A(a)) = f_B(b)$ (if $\pi_e(f_A(a))$ is undefined, the equality does not hold). The objective is to find an assignment that satisfies as many edges as possible.*

The vertices of the graph correspond to the locations in the proof. The assignments to the vertices correspond to the symbols that appear in these locations. The edges and the projections on them correspond to the tests of the verifier. The verifier picks an edge $e = (a, b)$ uniformly at random from E, it queries the proof locations a and b, gets answers $f_A(a)$ and $f_B(b)$, and tests whether $\pi_e(f_A(a)) = f_B(b)$. If so, the verifier accepts. If not, it rejects.

The projection PCP theorem states that verification of the NP-complete problem Sat can be done by checking assignments to an appropriate Label-Cover instance:

Theorem 3 (Projection PCP, [22])**.** *Given a* Sat *instance φ of size n and $\varepsilon = \varepsilon(n) > 0$, one can efficiently translate φ into a* Label-Cover *instance on a graph $G = (V, E)$ of size $|V| + |E| = n \cdot 2^{(\log n)^a} poly(1/\varepsilon)$ for some $a < 1$, and alphabets Σ_A, Σ_B of size $|\Sigma_A| = exp(1/\varepsilon^{O(1)})$, $|\Sigma_B| = poly(1/\varepsilon)$. The construction satisfies the following two conditions: If φ is satisfiable, then all the edges of G can be satisfied. If φ is not satisfiable, then at most ε fraction of the edges of G can be satisfied.*

In this theorem, the blow-up of the reduction is "almost linear", $n \cdot 2^{(\log n)^a} poly(1/\varepsilon) = n^{1+o(1)} poly(1/\varepsilon)$. Smaller blow-up of $n \cdot poly \log n$ is known for constant error ε [7, 8] (in that construction the number of queries depends on $1/\varepsilon$). At present, it is not known whether a blow-up of $n \cdot poly \log n \cdot poly(1/\varepsilon)$ is possible for all $\varepsilon = \varepsilon(n) > 0$, and it is open whether a linear blow-up $O(n)$ is possible even for constant $\varepsilon > 0$.

Another open question is whether one can obtain the theorem with alphabet size $poly(1/\varepsilon)$ rather than $exp(1/\varepsilon^{O(1)})$. The known lower bound on the alphabet is $\varepsilon \geq 1/|\Sigma|$, and other PCP theorems achieve $|\Sigma| = O(1/\varepsilon)$ or $|\Sigma| = poly(1/\varepsilon)$ (see Section 3.1).

We conclude this section with the general definition of a PCP verifier. A general PCP verifier may make more than two queries, its tests can be general predicates and not just projection tests, and there can be an error even for inputs in the language:

Definition 4 (PCP Verifier)**.** *A $PCP_{c,s}[r, q]_\Sigma$ verifier V for a language L is an algorithm that is given an input x as well as oracle access to a proof over alphabet Σ. The algorithm uses r random bits to make q queries to the proof. Completeness: If $x \in L$, then there exists a proof that V accepts with probability at least c. Soundness: If $x \notin L$, then for every proof, V accepts with probability at most s.*

Note that the size of the proof does not appear as a parameter in this definition. The reason is that we can use the randomness of the verifier as a proxy for the size: the size is at most $2^r \cdot q$.

3.1 Other Low Error PCP Theorems

In 1994, Raz proved the parallel repetition theorem [25], which results in a projection PCP with arbitrarily small error ε when applied to the basic PCP theorem [1]. This PCP has become the starting point of most NP hardness of approximation results proved since. It has low alphabet $poly(1/\varepsilon)$, but inherently large size $n^{\Omega(\log 1/\varepsilon)}$ [12]. This means that taking ε that tends to 0 as $n \to \infty$ results in a super-polynomial construction. For constant ε, however, the construction is polynomial. Because of its large blow-up, the parallel-repetition based PCP yields hardness results in which the time lower bound on the approximation is much lower than the time lower bound on SAT.

Low error simultaneously with polynomial, or almost-linear, size is known for PCP with more queries, but is less useful for hardness of approximation. We have PCP theorems with $\varepsilon = 2^{-(\log n)^\beta}$ for some constant $\beta > 0$ and $q = 3$ queries [26, 3]. This construction was adapted to almost linear size in [20] though no attempt was made to optimize the exact number of queries, and the number of queries was 7. For a large constant number of queries, an error of $2^{-(\log n)^{1-\alpha}}$ is known for *any* $\alpha > 0$. The number of queries grows with $1/\alpha$ and is $poly(1/\alpha)$ [9].

4 Encoding By Multivariate Low Degree Polynomials

In this section we discuss how to encode a string $a \in \{0,1\}^n$ (e.g., an NP witness) by a multivariate low degree polynomial, while leaving a as part of the polynomial's table. This is a basic step in the algebraic approach to the proof of the PCP theorem. Here "low degree" refers to low relative degree, i.e., low ratio between the degree and the field size, and not necessarily to a particularly low degree.

The encoding of a is done as follows: Consider a finite field $\mathbb{F}$ and let $H \subseteq \mathbb{F}$ be a set of finite field elements so that $n = |H^m|$ for some parameter m. We identify $\{1, \ldots, n\}$ with H^m and interpret the n bits of a as the evaluations of a polynomial on the product subset H^m. Then, we interpolate an m-variate polynomial p of degree at most $|H| - 1$ in each one of its variables. This results in a polynomial of degree[1] $d = (|H| - 1)m$. The field size $|\mathbb{F}|$ is determined so $d/|\mathbb{F}|$ is ε^c for a sufficiently large constant $c \geq 1$.

There are various ways to choose the parameter m as a function of n. Each choice determines different $|H|$, d and $|\mathbb{F}|$, and gives rise to different relative degree $d/|\mathbb{F}|$ and to different size $|\mathbb{F}^m|$. Some useful choices of parameters are listed in Table 1.

m	$\lvert\mathbb{F}\rvert$	$d/\lvert\mathbb{F}\rvert$	
$(\log n)^{1-\alpha}$	$2^{(\log n)^\alpha(1+o(1))}$	$2^{-(\log n)^{\Theta(\alpha)}}$	almost-linear size
$\log n / \log\log n$	$poly \log n$	$1/poly \log n$	polynomial size
$O(n)$	$poly(1/\varepsilon)$	ε	exponential size

Table 1: Polynomial parameters.

Low degree multivariate polynomials serve two main purposes in the proof of the PCP theorem: they form a good code, and they have a recursive structure. We discuss those two next.

[1] Throughout this essay, the degree of a polynomial is its total degree.

Error correcting code. For every m, d, $\mathbb{F}$, the family of m-variate polynomials of degree at most d over the field $\mathbb{F}$ form an error correcting code, known as the "Reed-Muller code". The relative distance of this code is $1 - d/|\mathbb{F}|$, which is close to 1 when the relative degree is small:

Proposition 4.1. *For every two different m-variate polynomials p, q of degree at most d over a finite field $\mathbb{F}$, when picking uniformly at random $x \in \mathbb{F}^m$, the probability that $p(x) \neq q(x)$ is at least $1 - d/|\mathbb{F}|$*

Proof. See, e.g., [18]. □

In PCP context, this means that checking equality between two low degree polynomials probabilistically can be done by making just one query to each!

Recursive structure. Multivariate polynomials have a recursive structure: the restriction of a polynomial to fewer variables, by substituting values to the other variables, is again a low degree polynomial. Similarly, restricting a polynomial to a line, or some other low degree subspace, leaves us again with a low degree polynomial, only of a smaller dimension.

5 Low Degree Testing

Our first test case for probabilistic checking, as well as one of the key steps in the algebraic construction of PCP, is the task of *low degree testing*, i.e., checking whether a multivariate function over a finite field, given by its table, is (approximately) a multivariate polynomial of certain given degree. The low degree testing problem is not NP-hard, so what we show in this section does not imply a PCP theorem by itself, only assists in the proof of the PCP theorem.

The low degree testing problem is formally as follows:

> Fix a finite field $\mathbb{F}$, a dimension m and a degree d. Given oracle access to a function $f : \mathbb{F}^m \to \mathbb{F}$, test by making as few queries to f as possible, whether f is close (in Hamming distance) to an m-variate polynomial of degree at most d over $\mathbb{F}$.

Note that we cannot hope to check whether f is a polynomial of degree at most d, rather than just close to a polynomial of degree at most d, because a tester that makes very few queries to f is not likely to detect changes in a few points in $\mathbb{F}^m$.

One can show that when f alone is given, low degree testing requires more than d queries to f. Thus, to decrease the number of queries, the tester is given an auxiliary table $\mathcal{A}$ in addition to f. In PCP context, this auxiliary table is part of the probabilistically checkable proof. In the next section we go over an illustraive example.

5.1 The Line vs. Point Test

An example of a low degree test is the Line-vs.-Point Test [27]. For this test, the auxiliary table associated with a low degree polynomial $p : \mathbb{F}^m \to \mathbb{F}$ consists of p's restrictions to the lines in $\mathbb{F}^m$. If a line ℓ is of the form[2] $\ell = \{ x + ty \mid t \in \mathbb{F}\}$ for $x, y \in \mathbb{F}^m$, then the restriction of p to ℓ is $p_{|\ell}(t) = p(x+ty)$. For every line ℓ in $\mathbb{F}^m$, $\mathcal{A}(\ell)$ is a univariate polynomial of degree at most d, which is supposed to be $p_{|\ell}$. One can show that if the restriction of a function to each and every

[2]Every line ℓ has many different representations as $\ell = \{ x + ty \mid t \in \mathbb{F}\}$ for different $x, x + y \in \ell$. We pick arbitrarily one such representation.

line in $\mathbb{F}^m$ is of degree at most d, then the function has to be an m-variate polynomial of degree at most d. However, $\mathcal{A}$ is not necessarily the restriction of any single function to the different lines, and so what the LINE-VS.-POINT TEST does is check that the lines are consistent with a single function:

LINE-VS.-POINT TEST:

1. Pick uniformly at random a line ℓ and a point $x \in \ell$.
2. Test whether[3] $\mathcal{A}(\ell)(x) = f(x)$.

This test makes only one query to $\mathcal{A}$ and one query to f, and performs a projection test: given $\mathcal{A}(\ell)$, the acceptable value for x is determined uniquely.

When f is a polynomial of degree at most d, and $\mathcal{A}$ gives the restrictions of f to the different lines, the test passes with probability 1. Moreover, "low degree testing theorems" show that whenever f and $\mathcal{A}$ pass the test with probability at least γ for some $0 < \gamma < 1$, it holds that $\mathcal{A}$ and f are $\approx \gamma$-close to representing a polynomial of degree at most d:

Theorem 5. *[3] For every $0 < \gamma < 1$, if the* LINE-VS.-POINT TEST *passes with probability γ, then there is an m-variate polynomial p of degree at most d that agrees with $\mathcal{A}$ on at least $\gamma - \varepsilon$ fraction of the lines and with f on at least $\gamma - \varepsilon$ fraction of the points, where $\varepsilon = Cm^a d^b / |\mathbb{F}|^c$ for some constants $a, b, c, C > 0$.*

The alphabet size of $\mathcal{A}$ is $|\mathbb{F}|^{d+1}$, which is the number of univariate polynomials of degree at most d over $\mathbb{F}$. The size of $\mathcal{A}$, i.e., the number of different entries in the table, is the number of lines in $\mathbb{F}^m$, which is quadratic in $|\mathbb{F}^m|$, the size of f. The alphabet and the size of the low degree test eventually influence the alphabet and the size, respectively, of the PCP. In Section 5.3 we discuss how to decrease the size of $\mathcal{A}$ to almost-linear in $|\mathbb{F}^m|$. We leave the large alphabet as is. This will yield a construction of a projection PCP of almost-linear size and large alphabet in Section 7. In Section 8 we decrease the alphabet of the PCP.

5.2 Proof of Low Degree Testing Theorem

Similarly to the LINE-VS.-POINT TEST, one can consider the PLANE-VS.-POINT TEST that compares a plane and a point on it. The advantage of this variant is that it has a different and elegant analysis by Raz and Safra [26, 21]. The analysis gives $\varepsilon = Cm^a(d/|\mathbb{F}|)^b$ for some constants $a, b, C > 0$. This improvement turns out to be crucial for achieving almost-linear size, as it allows a field size that is of the same order of magnitude as the degree. In this section we sketch the Raz-Safra analysis.

Consider the graph G whose vertices are the planes in $\mathbb{F}^m$ and whose edges connect planes s_1, s_2 if for every intersection point of the planes, $x \in s_1 \cap s_2$, the two planes agree, i.e., $\mathcal{A}(s_1)(x) = \mathcal{A}(s_2)(x)$. We want to show that there is a set S of at least $\gamma - \varepsilon$ fraction of the planes and a polynomial p of degree at most d, such that for every plane, $s \in S$, the polynomial $\mathcal{A}(s)$ is $p_{|s}$. Similarly, we want to show that there exists a set S' of at least $\gamma - \varepsilon$ fraction of the points in $\mathbb{F}^m$, such that for every $x \in S'$, the evaluation $f(x)$ is $p(x)$.

We start by proving such a result for the planes and points inside a cube, i.e., a three-dimensional affine subspace in $\mathbb{F}^m$. For every cube $u \subseteq \mathbb{F}^m$, we consider the subgraph G_u of G induced by the planes contained in u. Denote the acceptance probability of the test when

[3]For convenience, to refer to the value induced by $\mathcal{A}(\ell)$ on a point $z \in \ell$, we use $\mathcal{A}(\ell)(z)$ (independent of the representation of ℓ).

restricted to planes and points inside u by $\tilde{\gamma}$. There are two crucial properties of planes in a cube:

1. Two different planes in a cube that intersect – must intersect in a line.
2. For any line in the cube, almost all planes in the cube ($1 - \frac{1}{|\mathbb{F}|}$ fraction) intersect the line.

The analysis of G_u follows from two propositions that in turn follow from the two crucial properties. The first proposition states that a pair of planes that disagree essentially divide the planes in the cube into three sets – those that agree with the first plane, those that agree with the second plane, and those that agree with neither. There are (almost) no planes in the cube that agree with both:

Proposition 5.1. *If $s_1, s_2 \subseteq u$ and $(s_1, s_2) \notin E$, then for $1 - \frac{d+1}{|\mathbb{F}|}$ fraction of the the planes $s_3 \subseteq u$, either $(s_1, s_3) \notin E$ or $(s_2, s_3) \notin E$.*

Proof. From the premise and the first crucial property, s_1 and s_2 intersect in a line. Moreover, using the recursive structure and the code property of polynomials, $\mathcal{A}(s_1)$, $\mathcal{A}(s_2)$ do not agree on $1 - \frac{d}{|\mathbb{F}|}$ fraction of the points in the line. Using the second crucial property and the symmetry between the points on the line, at least $1 - \frac{d+1}{|\mathbb{F}|}$ fraction of the planes in the cube intersect s_1 and s_2 on a point in which they disagree. □

This proposition allows us to partition the planes in the cube into cliques, such that there are very few edges between cliques. Specifically:

Corollary 5.2. *There is a partition of the planes in G_u into cliques, such that at most $3\sqrt{(d+1)/|\mathbb{F}|}$ fraction of the pairs of planes form edges that connect different cliques.*

The next proposition says that there are many edges in G_u. We will use it to deduce that G_u contains a large clique. A large clique in G_u corresponds to a low degree polynomial that agrees with the planes in the clique.

Proposition 5.3. *For at least $\tilde{\gamma}^2 - (d+1)/|\mathbb{F}|$ fraction of all pairs of different intersecting planes in G_u, there is an edge between the planes in G_u.*

Proof. For a plane $s \subseteq u$ and a point $x \in s$, Let $I_{s,x}$ be an indicator variable for whether the Plane-vs.-Point Test passes for s and x, i.e., $\mathcal{A}(s)(x) = f(x)$. We use a *squaring trick* that applies the inequality $\mathbf{E}\left[X^2\right] \geq \mathbf{E}\left[X\right]^2$:

$$\mathop{\mathbf{E}}_{s_1 \neq s_2, x \in s_1, s_2} [I_{s_1,x} \cdot I_{s_2,x}] \geq \mathop{\mathbf{E}}_{x} \left[\mathop{\mathbf{E}}_{s \ni x} [I_{s,x}]^2 \right] - \frac{1}{|\mathbb{F}|} \geq \mathop{\mathbf{E}}_{x, s \ni x} [I_{s,x}]^2 - \frac{1}{|\mathbb{F}|} = \tilde{\gamma}^2 - \frac{1}{|\mathbb{F}|}.$$

The $1/|\mathbb{F}|$ bounds (non-tightly) the probability to get $s_1 = s_2$ when picking independently $s_1 \ni x$ and $s_2 \ni x$. By a Markov inequality,

$$\Pr_{s_1 \neq s_2, s_1 \cap s_2 \neq \phi} \left[\mathop{\mathbf{E}}_{x \in s_1, s_2} [I_{s_1,x} \cdot I_{s_2,x}] > d/|\mathbb{F}| \right] \geq \tilde{\gamma}^2 - \frac{d+1}{|\mathbb{F}|}.$$

Hence,

$$\Pr_{s_1 \neq s_2, s_1 \cap s_2 \neq \phi} [(s_1, s_2) \in E] \geq \tilde{\gamma}^2 - \frac{d+1}{|\mathbb{F}|}.$$

□

From Corollary 5.2 that partitions G_u into cliques and from Proposition 5.3 that assures us that G_u has many edges, we can deduce that there is a large clique in G_u:

Corollary 5.4. *There exists a clique in G_u that contains a fraction of at least $\tilde{\gamma}^2 - 4\sqrt{(d+1)/|\mathbb{F}|}$ of G_u's vertices.*

Proof. Assume otherwise. Let $C_1, \ldots, C_l \subseteq V$ be the partition of G_u into cliques. The number of edges in G_u is the number of edges between cliques plus the number of edges inside cliques. Thus, the number of edges is less than:

$$3\sqrt{\frac{d+1}{|\mathbb{F}|}}\,|V|^2 + \sum_{i=1}^{l}|C_i|^2 \leq 3\sqrt{\frac{d+1}{|\mathbb{F}|}}\,|V|^2 + \left(\tilde{\gamma}^2 - 4\sqrt{\frac{d+1}{|\mathbb{F}|}}\right)|V| \cdot \sum_{i=1}^{l}|C_i| = \left(\tilde{\gamma}^2 - \sqrt{\frac{d+1}{|\mathbb{F}|}}\right)|V|^2 .$$

which contradicts Proposition 5.3 when $d \leq |\mathbb{F}| - 4$. □

A sufficiently large clique in the graph G_u (of fraction larger than $(2d+1)/|\mathbb{F}|$) corresponds to a low degree polynomial for u that agrees with all the planes in the clique. Therefore, Corollary 5.4 implies the existence of a polynomial of degree at most d to u that agrees with at least $\tilde{\gamma}^2 - 4\sqrt{(d+1)/|\mathbb{F}|}$ fraction of the planes in u. With further arguments, one can show that this polynomial must in fact agree with a larger fraction $\tilde{\gamma} - O(((d+1)/|\mathbb{F}|)^{\Omega(1)}) \approx \tilde{\gamma}$ of the points in u and the planes in u. Since this was true for a general cube u, we found an assignment of low degree polynomials to all the cubes in $\mathbb{F}^m$ on which a Cube-vs.-Point Test accepts with probability $\approx \mathbf{E}[\tilde{\gamma}] \approx \gamma$. In general, for every dimension $2 \leq k \leq m$, one can define:

k-Dim-Subspace-vs.-Point Test:

1. Pick uniformly at random a subspace s of dimension k and a point $x \in s$.
2. Test whether $\mathcal{A}(s)(x) = f(x)$.

Similarly to what we outlined above, one can show for every $2 \leq k < m$,

$$\Pr[k\text{-Dim-Subspace-vs.-Point Test passes}] \geq \gamma$$

$$\Downarrow$$

$$\Pr[(k+1)\text{-Dim-Subspace-vs.-Point Test passes}] \geq \gamma - O(((d+1)/|\mathbb{F}|)^{\Omega(1)}).$$

Since the probability that the m-Dim-Subspace-vs.-Point Test passes is precisely the fraction of points on which f agrees with an m-variate polynomial of degree at most d, the proof works by iteratively invoking the above implication.

5.3 Saving Randomness in Low Degree Testing

Since the number of planes in $\mathbb{F}^m$ is about $|\mathbb{F}^m|^3$, using the Plane-vs.-Point Test in PCP, raises the size of the of the proof to the third power. In contrast, we want the proof to be of almost-linear size, which corresponds to a test that considers only $|\mathbb{F}^m|^{1+o(1)}$ planes. For a set of planes $\mathcal{P}_m$ in $\mathbb{F}^m$, we define:

Plane-vs.-Point Test($\mathcal{P}_m$):

1. Pick uniformly at random a plane $s \in \mathcal{P}_m$ and a point $x \in s$.
2. Test whether $\mathcal{A}(s)(x) = f(x)$.

If $|\mathcal{P}_m| = |\mathbb{F}^m|^{1+o(1)}$, we say that Plane-vs.-Point Test($\mathcal{P}_m$) is *randomness efficient*, as it uses $(1+o(1))\log|\mathbb{F}^m|$ random bits instead of $3\log|\mathbb{F}^m|$ random bits. What should $\mathcal{P}_m$ satisfy for the Raz-Safra analysis to go through? Turns out that there are two requirements.

The first requirement is a sampling property of the family $\mathcal{P}_m$: We say that a family $\mathcal{H}$ of affine subspaces is *sampling*, if for every $S \subseteq \mathbb{F}^m$, with high probability, a uniformly random $s \in \mathcal{H}$ has about $|S| / |\mathbb{F}^m|$ fraction of its points in S. The family $\mathcal{P}_m$ needs to be sampling so we can argue that the clique polynomial agrees with $\approx \tilde{\gamma}$, rather than $\approx \tilde{\gamma}^2$, fraction of the points in the cube. This is a step we did not describe in detail, but is crucial for the analysis because of the inductive structure. A random family of $|\mathbb{F}^m|^{1+o(1)}$ affine subspaces of any positive dimension is sampling, and there are various explicit constructions of sampling families.

The second requirement, however, calls for a non-random structure in the family $\mathcal{P}_m$: There should be a family of cubes $\mathcal{H}$ that is sampling, such that in a cube from $\mathcal{H}$, there is a set of planes isomorphic to $\mathcal{P}_3$. In general: for every $m \geq 3$, $2 < k \leq m$, there should be a family $\mathcal{H}_{m,k}$ of affine subspaces of dimension k in $\mathbb{F}^m$ that satisfies the sampling property, and such that a subspace in $\mathcal{H}_{m,k}$ contains a family of affine subspaces of dimension $k-1$ that is isomorphic to $\mathcal{H}_{k,k-1}$. A random family of $|\mathbb{F}^m|^{1+o(1)}$ planes lacks such structure: if we limit our view to any specific cube, we do not expect many random planes to land inside the cube, whose fraction is merely $\left|\mathbb{F}^3\right| / |\mathbb{F}^m|$.

Yet, this *randomness vs. structure problem* can be solved, and families $\mathcal{P}_m$ and $\mathcal{H}_{m,k}$ that satisfy both requirements are constructed in [21]. We describe the construction next. Take $\mathbb{F}$ that has a subfield $\mathbb{K} \subseteq \mathbb{F}$ of size $|\mathbb{F}|^{o(1)}$. For $2 \leq k \leq m$, let $\mathcal{H}_{m,k}$ be the family of all affine subspaces of dimension k defined by directions over $\mathbb{K}$, i.e., of the form[4]

$$\{\, x + t_1 y_1 + \cdots + t_k y_k \mid t_1, \ldots, t_k \in \mathbb{F} \},$$

for $x \in \mathbb{F}^m$ and linearly independent $y_1, \ldots, y_k \in \mathbb{K}^m$. Set $\mathcal{P}_m \doteq \mathcal{H}_{m,2}$. The families $\mathcal{H}_{m,k}$ are sampling, and this can be argued using Fourier analysis; the interested reader is referred to [21]. Moreover, in every subspace $\{ x + t_1 y_1 + \cdots + t_k y_k \mid t_1, \ldots, t_k \in \mathbb{F}\}$ for $x \in \mathbb{F}^m$ and $y_1, \ldots, y_k \in \mathbb{K}^m$, there is an affine subspace that corresponds to every $(k-1)$-dimensional affine subspace $\{ z + t_1 h_1 + \cdots + t_{k-1} h_{k-1} \mid t_1, \ldots, t_{k-1} \in \mathbb{F}\}$ for $z \in \mathbb{F}^k$ and linearly independent $h_1, \ldots, h_{k-1} \in \mathbb{K}^k$: this is the subspace defined by the shift $x' \doteq x + \sum_{j=1}^{k} z_j y_j \in \mathbb{F}^m$, and the directions $y'_i \doteq \sum_{j=1}^{k} h_{i,j} y_j \in \mathbb{K}^m$ for $1 \leq i \leq k-1$, where we denote $h_i = (h_{i,1}, \ldots, h_{i,k})$.

6 Sum Check

In this section we describe the "sum check" protocol of Lund, Fortnow, Karloff and Nisan [16]. Using this protocol we prove a simplified PCP theorem (see bullets below). In Section 7 we use this simplified PCP to construct a projection PCP.

The sum-check protocol verifies probabilistically that a low degree multivariate polynomial with n monomials and variables over a finite field $\mathbb{F}$ is zeroed by an assignment to its variables, and does so by making substantially less than n queries to the proof. We consider degree-3 polynomials for coefficients $q_{ijl} \in \mathbb{F}$, $1 \leq i, j, l \leq n$:

$$q(x_1, \ldots, x_n) = \sum_{1 \leq i,j,l \leq n} q_{ijl} x_i x_j x_l.$$

[4]The construction in [21] is slightly different: x is used as a direction in a linear subspace, rather than as a shift in an affine subspace. The reason for this modification lies within details we skipped: it allows us to argue that the the degree does not decreases under most restrictions.

Since there are only n monomials, most of the n^3 possible coefficients are 0.

The reader can verify that the problem of testing whether a collection of such polynomials has a common root is an NP-complete problem. Moreover, even distinguishing between the case that the polynomials in the collection all have a common root, and the case where at most $2/|\mathbb{F}|$ fraction of the polynomials have a common root, is NP-hard. This is shown by a reduction from 3SAT that takes the polynomials to be different $\mathbb{F}$-linear combinations of the clauses (see [17]). It allows us to focus on verifying that a single polynomial is zeroed – the verifier will choose the polynomial uniformly at random from the collection.

We save in the number of queries by having the prover perform partial computations needed for the verification. Each such partial computation is defined so it is a low degree polynomial. The verifier checks the consistency between the partial computations. Since those are of low degree, the consistency check only requires evaluating the polynomials on a random point.

In the presentation we make three simplifications that we later get rid of:

- We construct a PCP verifier in the sense of Definition 4 (many queries), and not a projection PCP. We transform the PCP to a projection PCP in Section 7.
- We analyze the verifier assuming the proof actually contains low degree polynomials (not necessarily the designated ones!). This assumption will be replaced by careful low degree testing.
- The PCP we describe has a polynomial, but not an almost-linear, size. In Section 6.1 we explain what is needed to achieve almost linear size and give the appropriate references.

Our first step is to reformulate our verification problem in terms of low degree polynomials with fewer than n variables. We use the encoding of Section 4. Let $m = (\log n)^{1-\alpha}$ for a small constant $\alpha > 0$. Identify $\{1, \ldots, n\}$ with H^m, $H \subseteq \mathbb{F}$, $|H| = 2^{(\log n)^\alpha}$. Let:

- $\hat{a} : \mathbb{F}^m \to \mathbb{F}$ be the supposed low degree encoding of an assignment to q's variables that zeros q, so for $i \in H^m$, it holds that $\hat{a}(i)$ is the assignment to x_i.
- $\hat{q} : \mathbb{F}^m \times \mathbb{F}^m \times \mathbb{F}^m \to \mathbb{F}$ be the low degree encoding of the coefficients of q, so for $i, j, l \in H^m$, it holds that $\hat{q}(i, j, l) = q_{ijl}$.

Using this notation, we want to verify that:

$$\sum_{w,y,z \in H^m} \hat{q}(w, y, z)\hat{a}(w)\hat{a}(y)\hat{a}(z) = 0.$$

Let us denote $p(w, y, z) \doteq \hat{q}(w, y, z)\hat{a}(w)\hat{a}(y)\hat{a}(z)$.

We design a proof that allows probabilistic verification by making only $\Theta(|H|\, m) \ll n$ queries to the proof. The proof consists of $\hat{a}$ and of the "partial sums" of the sum $\sum_{u \in H^{3m}} p(u)$ we want to check. The i'th partial sum for $0 \leq i \leq 3m$ fixes the first i coordinates, and sums only over the remaining coordinates:

$$S_i(u_1, \ldots, u_i) \doteq \sum_{u_{i+1}, \ldots, u_{3m} \in H} p(u_1, \ldots, u_i, u_{i+1}, \ldots, x_{3m}), \quad i = 0, \ldots, 3m.$$

The proof stores the tables of $\hat{a}$ and of the S_i's, i.e., $\hat{a}(u_1, \ldots, u_m)$ for all $u_1, \ldots, u_m \in \mathbb{F}$ and $S_i(u_1, \ldots, u_{3m})$ for all $u_1, \ldots, u_{3m} \in \mathbb{F}$ for $i = 0, \ldots, 3m$. We assume that those tables are of low degree polynomials.

By definition, the partial sums are related to each other and to p. The exact equations they satisfy are these:

1. $S_0() = \sum_{u_1,\ldots,u_{3m} \in H} p(u_1, \ldots, u_{3m})$.
2. $S_{i-1}(u_1, \ldots, u_{i-1}) = \sum_{u_i \in H} S_i(u_1, \ldots, u_i)$ for $i = 1, \ldots, 3m+1$.
3. $S_{3m}(u_1, \ldots, u_{3m}) = p(u_1, \ldots, u_{3m})$.

Next we describe the verifier. As we mentioned above, equality between low degree polynomials can be checked probabilistically by comparing the polynomials on a random point. Thus, equality 3 can be checked probabilistically by making one query to S_{3m} and 3 queries to $\hat{a}$; the $3m+1$ equalities in 2 can be checked probabilistically by checking each on $|H|$ points from S_i and one point from S_{i-1}. By equality 1, this allows verification of the overall sum. The total number of queries is $\Theta(|H|\, m)$, instead of $|H|^m$. The large saving in the number of queries was made possible by the pre-computing done by the partial sums. The pre-computing can be checked using the recursive structure and the code property of multivariate polynomials.

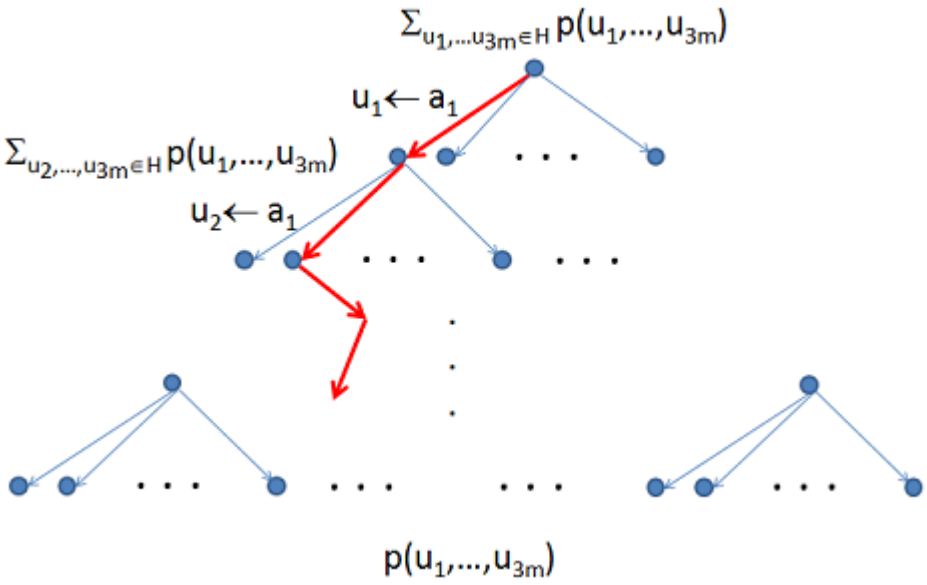

Figure 1: Sum Check.

6.1 Almost-Linear Size

The PCP we described has $O(m)$ tables for each polynomial q we want to check, and each table is of size at most $|\mathbb{F}|^{3m}$. This gives us a bound of $O(m\,|\mathbb{F}|^{4m})$ on the proof size, instead of $|\mathbb{F}^m|^{1+o(1)}$. To achieve almost-linear size, we need to index the coefficients and the variables in the different polynomials by elements in H^m, so for every monomial, the indices of the variables in the monomial can be expressed as polynomials of the index of the coefficient. This can be done, but requires a certain machinery; the interested reader is referred to [6].

7 Projection PCPs With Large Alphabet

In this section we describe how to combine the sum check protocol from Section 6 and the low degree testing from Section 5 to construct a projection PCP. This projection PCP has a large

alphabet, inherited from the low degree test. In the next section we describe how to decrease the alphabet, while keeping the projection and the low error.

Our first observation is that instead of storing many different low degree polynomials in the proof, we can combine them into a single polynomial by adding an extra variable that serves as an index: Assume that the sum-check polynomials are given by $T_i(u_1, \ldots, u_m)$ for $i = 1, \ldots, l$, where $u_1, \ldots, u_m \in \mathbb{F}$. Fix a set $I = \{v_1, \ldots, v_l\} \subseteq \mathbb{F}$, and define for $i = 1, \ldots, l$,

$$T(v_i, u_1, \ldots, u_m) \doteq T_i(u_1, \ldots, u_m).$$

The degree of T is at most l times the maximum degree of the T_i's. We choose the parameters so T's relative degree is polynomially small in ε.

Let k be the number of queries made by the sum-check verifier. The proof is supposed to consist of $T(v, u_1, \ldots, u_m)$ for every $v, u_1, \ldots, u_m \in \mathbb{F}$ and of the trivariate, degree-$k \deg T$, restrictions of T to various three-dimensional, degree-k, manifolds (see the preliminaries for the definition of manifolds). The verifier, referred to as the "Manifold vs. Point" verifier in [22], is as follows:

1. Pick randomness for the sum-check verifier. This determines queries $u_1, \ldots, u_k \in \mathbb{F}^{m+1}$ to T.
2. Pick a random plane $s_0 \subseteq \mathbb{F}^{m+1}$. Let $s \subseteq \mathbb{F}^{m+1}$ be a degree-k, three-dimensional manifold that contains $u_1, \ldots, u_k$, s_0.
3. Query $\mathcal{A}(s)$, a polynomial that is supposed to be the restriction of T to s. Pick a random point $x \in s$, and query the proof for $T(x)$. Check that $\mathcal{A}(s)(x) = T(x)$.
4. Use $\mathcal{A}(s)$ to get answers to $u_1, \ldots, u_k$, and perform the sum check test on those answers. Accept if all tests so far accepted; reject otherwise.

Note that the verifier performs a projection test: it makes one query to a manifold and one query to a point, and the answer to the manifold determines at most one satisfying answer to the point. Moreover, for a correct proof, the verifier always accepts. Using low degree testing, one can show that whenever the low degree test accepts, $\mathcal{A}(s)$ is the restriction to s of one of a "list decoding" of few low degree polynomials for T. Those low degree polynomials induce low degree polynomials for the sum-check polynomials, and the soundness of sum-check thus implies the soundness of the Manifold vs. Point verifier.

We remark that the Manifold vs. Point construction works with any PCP that has a relatively small number of queries k, and not just with the sum-check PCP. Indeed, [22] does not use the sum-check protocol, but rather a randomness-efficient repetition of Dinur's constant-error PCP [8, 7].

7.1 Randomness efficiency

To make the Manifold vs. Point verifier randomness-efficient, one has to use the randomness-efficient versions of both the sum-check and the low degree test. But even this is not enough. The natural bound on the number of random bits of the Manifold vs. Point verifier is the sum of the number of random bits of the sum-check protocol plus the number of random bits of the low degree test. This sums to more than $2 \log |\mathbb{F}^m|$ random bits, instead of $(1+o(1)) \log |\mathbb{F}^m|$ random bits. Luckily, the Manifold vs. Point verifier can be implemented using only $(1 + o(1)) \log |\mathbb{F}^m|$ random bits, by recycling the random bits used for sum-check in the low degree test. This

should be done carefully, as x, a uniform point in s, should be distributed uniformly (or close to uniformly) over $\mathbb{F}^m$ for the analysis to go through. In [20] there is a general technique for recycling randomness this way.

8 Composition

The projection PCP we constructed in the previous section has a sub-exponential alphabet $exp(exp((\log n)^{\alpha'}))$ for some constant $0 < \alpha' < 1$, instead of an alphabet whose size only depends on the designed error ε, and is at most polynomial in n. In this section we explain how to take a projection PCP whose alphabet grows with the input size n, and produce another projection PCP with about the same error, but with smaller alphabet. This operation, called composition for projection PCPs, is from [22], and was abstracted and simplified in [10]. It is purely combinatorial, and can be understood without knowing the details of the previous sections.

We continue with some intuition for composition of projection PCPs. The actual construction is described in the next sections. The basic idea of composition is to replace every vertex $a \in A$ with new vertices over a smaller alphabet, and similarly replace every vertex $b \in B$ with new vertices. Then, composition puts edges between new vertices that correspond to an edge $e = (a, b) \in E$ such that the new edges form a small PCP that verifies that $\pi_e(f_A(a)) = f_B(b)$. There is, however, a problem with this idea that makes it seem hopeless: the small PCP necessarily depends both on a and on b. On the other hand, in the scheme described above, the small PCP should consist of vertices whose assignments only depend on a, and vertices whose assignments only depend on b.

Surprisingly, the composition operation works by changing the original projection PCP, so the endpoints of each edge depend on both a and b. This seems like it should ruin the soundness of the PCP: knowing both a and b, it is easy to find an assignment that satisfies the edge (a, b). The reason the soundness is not ruined is that each vertex has many possible a's on which the test can be done, or many possible b's on which the test can be done, so it is impossible to satisfy all possible edges (a, b).

The composition operation consists of the following steps, which we review in the following sections:

1. Decreasing $|\Sigma_B|$ using an error correcting code.
2. Decreasing the graph degree of the B vertices using expanders.
3. Switching sides in the graph: viewing the projection as going from B to A (Here we use the low graph degree of the B vertices).
4. Decreasing $|\Sigma_A|$ using another PCP (Here we use the structure obtained in the previous steps).

8.1 Decreasing $|\Sigma_B|$

Given a projection PCP $G = (A, B, E)$, with alphabets Σ_A, Σ_B, and projections $\{\pi_e\}_{e \in E}$, as well as an error correcting code $E : \Sigma_B \to (\Sigma'_B)^t$ of relative distance $1 - \varepsilon$, alphabet of size $|\Sigma'_B| = poly(1/\varepsilon)$ and rate $poly(\varepsilon)$ (see Lemma 2.1), we construct a new projection PCP in which Σ_B is replaced by Σ'_B.

The changes in the new projection PCP compared to the original projection PCP are as follows:

- B is replaced with $B \times \{1, \ldots, t\}$; Σ_B is replaced with Σ'_B. A vertex $\langle b, i \rangle \in B \times \{1, \ldots, t\}$ is supposed the be assigned the i'th symbol of the E-encoding of the assignment to b.
- For every $e = (a, b) \in E$, for every $1 \leq i \leq t$, there is a new edge $e_i = (a, \langle b, i \rangle)$. The projection on this edge is the i'th symbol of the encoding of e's projection to b, i.e., $\pi_{e_i}(\sigma_a) \doteq E(\pi_e(\sigma_a))_i$.

An assignment that satisfies all the edges of the original PCP translates into an assignment that satisfies all the edges of the new PCP. We outline the soundness analysis next. Consider an assignment to the vertices of the new PCP that satisfies many of its edges, and let us show that it corresponds to an assignment to the original PCP that satisfies a similar fraction of edges. Let $b \in B$ be such that many of the edges that involve vertices $\langle b, \cdot \rangle$ in the new PCP are satisfied. Consider the assignment $w \in (\Sigma'_B)^t$ to the t vertices $\langle b, \cdot \rangle$. While w does not necessarily correspond to any codeword (which corresponds to an assignment to b in the original PCP), any neighbor $a \in A$ of b such that many of the edges $(a, \langle b, \cdot \rangle)$ are satisfied implies that w is close to some codeword. By our assumption on b, there should be many neighbors a for which this is the case. Moreover, no word can be close to too many codewords (see Proposition 2.2). Thus, many of those codewords are the same codeword, and we can construct an assignment to G that satisfies a fraction of edges similar to that satisfied in the new PCP.

The costs of this operation are modest: the size of the PCP increases by a factor $\log |\Sigma_B| \cdot poly(1/\varepsilon)$, and the error of the PCP increases by an added $O(\varepsilon^{\Omega(1)})$.

We emphasize that only Σ_B, the projected alphabet, and not Σ_A, the projecting alphabet, can be decreased the way we described.

8.2 Decreasing B's Graph Degree

We are given a projection PCP $G = (A, B, E)$, with alphabets Σ_A, Σ_B, and projections $\{\pi_e\}_{e \in E}$. For simplicity, we assume that the degree of all the B vertices is the same large number D. In the algebraic construction the degree is the number of manifolds that contain a point. We show how to decrease D to a smaller degree $D' = poly(1/\varepsilon)$. Assume an expander $H = (\{1, \ldots, D\}, E_H)$ with degree D' and second eigenvalue $O((D')^\alpha)$ for some constant $\alpha < 1$ (see Lemma 2.4). We replace each B vertex with D copies of it, and use the expander H to distribute the different edges that go into the vertex between its different copies. This is a bipartite version of an operation that originally appeared in [23].

The changes in the new projection PCP compared to the original projection PCP are as follows:

- B is replaced with $B \times \{1, \ldots, D\}$. Every vertex $\langle b, \cdot \rangle$ for $b \in B$ is supposed to be assigned the assignment to b.
- For every $e = (a, b) \in E$, if a is the i'th neighbor of b for $1 \leq i \leq D$, and $(i, j) \in E_H$, then we introduce an edge $(a, \langle b, j \rangle)$. The projection on this edge is the projection on e.

An assignment that satisfies all the edges of the original PCP translates into an assignment that satisfies all the edges of the new PCP. We outline the soundness analysis next. Consider an assignment to the vertices of the new PCP that satisfies many of its edges, and let us show that it corresponds to an assignment to the original PCP that satisfies a similar fraction of edges.

Consider $b \in B$ and the assignments to the vertices $\langle b, i \rangle$ for $i = 1, \ldots, D$. All those assignments are supposed to be equal. If they are, the fraction of satisfied edges involving b is the same in G and in the new PCP. The expander mixing lemma promises that if the assignments are substantially different, most neighbors a of b in G will encounter different assignments as well, and most of the edges $(a, \langle b, i \rangle)$ will not be satisfied.

The costs of the degree reduction operation are modest: the size of the PCP increases by a factor $poly(1/\varepsilon)$, and the error of the PCP increases by a term of $O(1/(D')^{\Omega(1)})$.

We emphasize that when the projection is from A to B, only the degree of the B vertices, and not the degree of the A vertices, can be decreased the way we described. Moreover, the degree cannot be decreased beyond $1/\varepsilon$, as one can show that $\varepsilon \geq 1/D$.

8.3 Switching Sides

Given a projection PCP $G = (A, B, E)$, where the B vertices have degree D, alphabets Σ_A, Σ_B, and projections $\{\pi_e\}_{e \in E}$, we can view the projection as going from the B side to the A side as follows:

- Σ_B is replaced with Σ_A^D, namely a B vertex stores assignments for all its neighbors.
- We reverse all edges $e = (a, b) \in E$ to $e' = (b, a)$.
- Assume that b's neighbors are $a_1, \ldots, a_D \in A$. Then, for every $1 \leq i \leq D$ we have the projection defined naturally $\pi_{(b,a_i)}(\sigma_1, \ldots, \sigma_D) = \sigma_i$ assuming $\sigma_1, \ldots, \sigma_D$ are assignments to b's neighbors that agree on a single assignment to b, i.e., $\pi_{(a_1,b)}(\sigma_1) = \pi_{(a_2,b)}(\sigma_2) = \cdots = \pi_{(a_D,b)}(\sigma_D)$. If the latter does not hold, then the projection is undefined.

The new assignments to the B vertices were referred to in [22] as "sunflowers", where a "sunflower" consists of D assignments with a common center. The degree D of the B vertices was decreased because of its influence on the alphabet size of the new PCP. The alphabet of the B vertices was decreased in order to decrease the alphabet of the center.

The switching sides operation completes the transformation we apply on the PCP, and yields a construction in which not only the assignments to the A vertices contain assignments to (old) B vertices, but also the (new) assignments to the B vertices contain assignments to A vertices.

8.4 Composition

Lastly, we utilize the structure formed in the previous section to decrease the alphabet, by composing our projection PCP with an additional projection PCP. The latter is a strengthening of the projection PCP of Definition 2: it locally decodes symbols of the NP witness it encodes. We discuss the new definition in Section 8.4.1, and in Section 8.4.2 we show how to decrease the alphabet using it.

8.4.1 Locally Decode or Reject

Usually, a predicate φ on n variables is translated to a Label-Cover instance; a satisfying assignment to φ translates into an assignment that satisfies all the edges in the instance; an assignment to the instance that satisfies a non-negligible fraction of the edges corresponds to a satisfying assignment to φ. In a "decode or reject" PCP, however, we associate an index $1 \leq i_a \leq n$ with every vertex $a \in A$. The vertex is supposed to decode the i'th symbol of a satisfying assignment x to φ; a satisfying assignment x to φ translates into an assignment to

the Label-Cover instance that satisfies all the edges e and makes all A vertices decode x_{i_a}; an assignment to the instance that satisfies a non-negligible fraction of the edges, should have almost all of the A vertices involved decode correctly x_{i_a} for a satisfying assignment x to φ.

Definition 6 (Locally Decode or Reject Label-Cover)**.** *Let φ be a predicate over n variables, each taking values from an alphabet Σ. An (ε, l)-*locally decode-or-reject Label-Cover *instance $G = (A, B, E)$, Σ_A, Σ_B, $\{\pi_e\}_{e \in E}$ for φ has:*

- *An index $1 \leq i_a \leq n$ associated with every vertex $a \in A$. For a uniformly random vertex $a \in A$, the index i_a is uniformly distributed over $\{1, \ldots, n\}$.*
- *A decoding function $dec_a : \Sigma_A \to \Sigma$ associated with every vertex $a \in A$.*

An assignment $f_A : A \to \Sigma_A$, $f_B : B \to \Sigma_B$ has edge $e = (a, b) \in E$ read $x \in \Sigma^n$ if $dec_a(f_A(a)) = x_{i_a}$.

- *For every $x \in \Sigma^n$ that satisfies φ, there is an assignment $f_A : A \to \Sigma_A$, $f_B : B \to \Sigma_B$, such that all edges $e \in E$ are satisfied and read x.*
- *For every assignment $f_A : A \to \Sigma_A$, $f_B : B \to \Sigma_B$, there is a short list $x^{(1)}, \ldots, x^{(l)} \in \Sigma^n$ of assignments that satisfy φ, such that when picking a uniformly random edge $e \in E$, the probability that e is satisfied but does not read any of $x^{(1)}, \ldots, x^{(l)}$, is at most ε.*

Note that there are assignments to the Label-Cover instance that satisfy a non-negligible fraction of the edges, and correspond to several different satisfying assignments $x^{(1)}, \ldots, x^{(l)} \in \Sigma^n$ to φ: one can partition the A vertices, as well as the B vertices, into l equal-sized parts, and assign the i'th part for $i = 1, \ldots, l$, the Label-Cover assignment that corresponds to $x^{(i)}$.

Decode or reject projection PCPs can be constructed similarly to standard projection PCPs. In the construction described in Section 7, one needs to make sure the manifold goes through an extra point that is uniformly distributed in H^m, and the verifier, if it does not reject, should decode the value to this point.

8.4.2 Composing

Let $G = (A, B, E)$, Σ_A, $(\Sigma_A)^D$, $\{\pi_e\}_{e \in E}$, be the projection PCP we constructed in Section 8.3. Assume that all the A vertices have degree s. Then, without loss of generality, we can assume that each element in Σ_A is a vector in $(\Sigma_B)^s$. For instance, in the Manifold vs. Point verifier, Σ_A consists of all trivariate degree-$k \deg S$ polynomials, and can be thought of as a subset of the set of vectors $\mathbb{F}^{|\mathbb{F}^3|}$.

Let φ be a predicate that given an element in $(\Sigma_B)^s$ checks whether it is in Σ_A. For φ we construct a $(\varepsilon^{\Theta(1)}, poly(1/\varepsilon))$-decode or reject Label-Cover instance: $G' = (A', B', E')$, $\Sigma_{A'}$, $\Sigma_{B'}$, $\{\pi_{e'}\}$, $\{i_{a'}\}_{a' \in A'}$, $\{dec_{a'}\}_{a' \in A'}$. Importantly: the size of G' is polynomial (or almost-linear) in $n' \doteq s \log |\Sigma_B| = n^{o(1)}$, and the alphabet sizes $|\Sigma_{A'}|$ and $|\Sigma_{B'}|$ are only super-polynomial in n', and not in n.

The new Label-Cover instance combines G and G':

- Replace A with $A \times B'$; replace Σ_A with $\Sigma_{B'}$. A vertex $\langle a, b' \rangle \in A \times B'$ is supposed to be assigned the assignment to b' in G' associated with the assignment in $(\Sigma_B)^s$ to a in G.

- Replace B with $B \times A'$; replace $(\Sigma_A)^D$ with $(\Sigma_{A'})^D$. A vertex $\langle b, a' \rangle \in B \times A'$ is supposed to be assigned D values in $\Sigma_{A'}$, one for each neighbor a of b in G: this value is the assignment to a' in G' associated with the assignment in $(\Sigma_B)^s$ to a in G.
- If there is an edge $e = (b, a) \in E$, where b is the j'th neighbor of a in G for $1 \leq j \leq s$, and $a' \in A'$ decodes a value for b, i.e., $i_{a'} = j$, then for every edge $e' = (a', b') \in E'$, introduce an edge $(\langle b, a' \rangle, \langle a, b' \rangle)$.
- Assume that the neighbors of b in G are $a_1, \ldots, a_D \in A$. Then, the projection on an edge $(\langle b, a' \rangle, \langle a_i, b' \rangle)$ maps $(\sigma_1, \ldots, \sigma_D) \in (\Sigma_{A'})^D$ to $\pi_{(a',b')}(\sigma_i)$, assuming $dec_{a'}(\sigma_1) = dec_{a'}(\sigma_2) = \cdots = dec_{a'}(\sigma_D)$.

The operation constructs a projection PCP that simulates G using G'. It uses the fact that in G the assignments to both endpoints of an edge $(a, b) \in E$ consist of assignments to a (b's assignment consists of D assignments, one of them is the assignment to a).

The size of the new PCP is the multiplication of the size of G and the size of G'. If the size of G is almost-linear $n^{1+o(1)}$ and the size of G' is sub-linear $n^{o(1)}$, we get an almost-linear size construction. The alphabet size $\left|(\Sigma_{A'})^D\right|$ can be much smaller than $|\Sigma_A|$ as it depends on n' and not on n. Note, however, that since $D = poly(1/\varepsilon)$, the alphabet size $\left|(\Sigma_{A'})^D\right|$ is at least exponential in $poly(1/\varepsilon)$. The error of the new PCP combines the errors of the two PCPs, and is low if both are low.

8.5 Completing The Construction

To complete the construction, we consider the instantiations of the Manifold vs. Point verifier for all three settings of parameters[5] in Table 1. We start by applying the composition operation on the almost-linear construction as G together with the polynomial size construction as G'. This yields a projection PCP with polynomial alphabet. Then, we apply the composition operation again with the composed construction as G and the exponential size construction as G'. This brings the alphabet to $exp(poly(1/\varepsilon))$.

9 Open Problems

Several intriguing open problems were mentioned throughout the main text. One of them is whether there are PCPs of linear size. Perhaps the main open problem is whether Theorem 3 can be obtained with alphabet size $poly(1/\varepsilon)$ rather than $exp(poly(1/\varepsilon))$. This would allow polynomially small error $1/n^\alpha$ for some $\alpha > 0$ together with polynomial alphabet size, and would open the way for new hardness of approximation results. Bellare et al conjectured that polynomially small error together with polynomial size and polynomial alphabet size should be achievable for some constant number of queries [5]. In [19], the author suggested the name "the projection games conjecture" for a similar conjecture concerning projection PCPs.

References

[1] 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.

[5] The exponential size construction does not require sum-check, but an adaptation of the Manifold vs. Point construction.

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

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

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

[5] M. Bellare, S. Goldwasser, C. Lund, and A. Russell. Efficient probabilistically checkable proofs and applications to approximations. In *Proc. 25th ACM Symp. on Theory of Computing*, pages 294–304, 1993.

[6] 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.

[7] E. Ben-Sasson and M. Sudan. Short PCPs with polylog query complexity. *SIAM Journal on Computing*, 38:551–607, 2008.

[8] I. Dinur. The PCP theorem by gap amplification. *Journal of the ACM*, 54(3):12, 2007.

[9] I. Dinur, E. Fischer, G. Kindler, R. Raz, and S. Safra. PCP characterizations of NP: Toward a polynomially-small error-probability. *Computational Complexity*, 20(3):413–504, 2011.

[10] I. Dinur and P. Harsha. Composition of low-error 2-query PCPs using decodable PCPs. In *Proc. 41st ACM Symp. on Theory of Computing*, 2009.

[11] 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.

[12] U. Feige and J. Kilian. Impossibility results for recycling random bits in two-prover proof systems. In *Proc. 27th ACM Symp. on Theory of Computing*, pages 457–468, 1995.

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

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

[15] S. Khot. Guest column: inapproximability results via long code based PCPs. *SIGACT News*, 36:25–42, June 2005.

[16] 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.

[17] D. Moshkovitz. Lecture notes in probabilistically checkable proofs, MIT. http://people.csail.mit.edu/dmoshkov/courses/pcp-mit/index.html.

[18] D. Moshkovitz. An alternative proof of the Schwartz-Zippel lemma. Technical report, ECCC TR10-096, 2010.

[19] D. Moshkovitz. The projection games conjecture and the NP-hardness of $\ln n$-approximating Set-Cover. Technical Report TR11-112, Electronic Colloquium on Computational Complexity (ECCC), 2011.

[20] D. Moshkovitz and R. Raz. Sub-constant error probabilistically checkable proof of almost-linear size. Technical Report TR07-026, Electronic Colloquium on Computational Complexity, 2007.

[21] D. Moshkovitz and R. Raz. Sub-constant error low degree test of almost-linear size. *SIAM Journal on Computing*, 38(1):140–180, 2008.

[22] D. Moshkovitz and R. Raz. Two query PCP with sub-constant error. *Journal of the ACM*, 57(5), 2010.

[23] C. Papadimitriou and M. Yannakakis. Optimization, approximation and complexity classes. *Journal of Computer and System Sciences*, 43:425–440, 1991.

[24] M. Plotkin. Binary codes with specified minimum distance. *IRE Transactions on Information Theory*, 6:445–450, 1960.

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

[26] 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.

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

[28] M. Sudan. Lecture notes in essential coding theory, MIT. http://people.csail.mit.edu/madhu/coding/course.html.

[29] M. A. Tsfaman, S. G. Vlăduţ, and T. Zink. Modular curves, shimura curves, and codes better than the Varshamov-Gilbert bound. *Math. Nachrichten*, 109:21–28, 1982.