Ruling Out PTAS for Graph Min-Bisection, Dense k-Subgraph and Bipartite Clique *

Subhash Khot
khot@cc.gatech.edu †

February 15, 2005

Abstract

Assuming that NP $\not\subseteq \cap_{\epsilon>0}$ BPTIME(2^{n^ϵ}), we show that Graph Min-Bisection, Dense k-Subgraph and Bipartite Clique have no Polynomial Time Approximation Scheme (PTAS).

We give a reduction from the Minimum Distance of Code Problem (MDC). Starting with an instance of MDC, we build a *Quasi-random PCP* that suffices to prove the desired inapproximability results. In a Quasi-random PCP, the query pattern of the verifier looks random in certain precise sense.

Among the several new techniques we introduce, the most interesting one gives a way of certifying that a given polynomial belongs to a given linear subspace of polynomials. As is important for our purpose, the certificate itself happens to be another polynomial and it can be checked probabilistically by reading a constant number of its values.

*A preliminary version of the paper appeared in FOCS 2004.

†Georgia Institute of Technology, Atlanta GA-30332. Email: `khot@cc.gatech.edu`. Work was done while at the Institute for Advanced Study.

This material is based upon work supported by the National Science Foundation under agreement No. DMS-0111298. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the National Science Foundation.

1 Introduction

Several optimization problems of theoretical and practical importance are NP-hard, meaning, there is no efficient (i.e. polynomial time) algorithm to obtain exact solutions to these problems unless P = NP. However for many of these problems, it is possible to obtain approximate solutions efficiently. An approximation algorithm with ratio C computes a solution that is guaranteed to be within factor C of the optimal solution (the approximation ratio for minimization and maximization problems is defined appropriately so that $C > 1$). In the best scenario, a problem may have an approximation algorithm with ratio $1 + \epsilon$ for arbitrarily small constant $\epsilon > 0$. Many packing and scheduling problems have this property (e.g. the Knapsack problem), and they are said to have a Polynomial Time Approximation Scheme (PTAS). For an excellent treatment of approximation algorithms, please refer to Vazirani's book [36].

Existence of a PTAS is a difficult question to answer in general. For example, it wasn't known till early 90's whether MAX-3SAT has a PTAS. The celebrated PCP Theorem ([6], [5]) settled this question in the negative. Such results that rule out the possibility of an approximation algorithm within certain factor (assuming of course a hypothesis like P $\neq$ NP) are known as inapproximability results or hardness results. After the discovery of the PCP Theorem, there has been a tremendous progress in proving hardness results for various problems. A very incomplete list includes [3], [9], [16] [25], [26], [19], [15], [23], [13], [14], [28]. For some problems like MAX-3SAT [26], Clique [25], and Set-Cover ([16]), optimal hardness results have been proved. A recent survey of Trevisan [35] gives a nice exposition to this area (see also [4]).

The PCP Theorem has an equivalent formulation in terms of probabilistic checking of proofs. The theorem states that every NP-statement has a short proof that can be checked very efficiently by a probabilistic verifier. The verifier reads only a constant number of bits from the proof and has the following completeness and soundness properties : every correct statement has a proof that the verifier accepts with probability 1 and *every* proof of an incorrect statement is rejected with high probability. The connection between the notion of probabilistic proof checking and inapproximability results for optimization problems has been one of the most exciting developments in theoretical computer science.

In spite of great success of PCP tools towards proving inapproximability results, there are some notorious problems which so far have resisted all attempts to prove *good* inapproximability results. Examples include Vertex Cover, Traveling Salesman Problem, Graph Coloring, and so on. In this paper, we make progress on one set of notorious problems, namely the Graph Min-Bisection, Dense k-Subgraph and Bipartite Clique problems. The best known algorithms for these problems have approximation ratios $O(\log^2 n)$ (see [22]), $O(n^{1/3})$ (see [21]) and $n/(\log n)^{O(1)}$ (folklore) respectively whereas no inapproximability results were known. Ruling out a PTAS for these problems has been considered a major open question in inapproximability theory.

Recently, Feige showed [17] that these problems have no PTAS assuming a certain hypothesis about average-case hardness of Random 3SAT (see Section 1.7). The main result in this paper rules out a PTAS for these problems assuming only a (fairly) standard assumption that NP $\not\subseteq \cap_{\epsilon>0}$ BPTIME(2^{n^ϵ}) (i.e. NP does not have randomized algorithms that run in sub-exponential time).

Typically, inapproximability results are proved by *reducing* a known NP-hard problem to the target problem. The PCP Theorem itself can be viewed as a (rather sophisticated) NP-hardness reduction. We too obtain our result via a reduction. We use a reduction from the Minimum Distance of Linear Code (MDC) problem. The inapproximability result for MDC has been obtained by Dumer, Micciancio and Sudan [15] (see Section 1.2).

1.1 Problem Definitions and Results

Graph Min-Bisection : Given a graph $G(V, E)$ with $|V|$ even, partition V into two equal parts V' and V'' so as to minimize the number of *crossing* edges, i.e. edges with one endpoint each in V' and V''.

Dense k-Subgraph : Given a graph $G(V, E)$ and a parameter k, find a subset $V' \subseteq V$ of size k so as to maximize the number of edges whose both endpoints are in V'.

Bipartite Clique : Given a bipartite graph $G(V, W, E)$, maximize k such that there exist $V' \subseteq V$ and $W' \subseteq W$, each of size k and the subgraph of G induced on the set of vertices $V' \cup W'$ is a complete bipartite graph.

Graph Min-Bisection is used as a sub-routine by many graph algorithms based on Divide-and-Conquer strategy. Such algorithms partition the given graph into two equal parts, solve the desired optimization problem recursively on the two parts and then *merge* the two solutions to obtain an overall solution (it usually suffices to have a partition of the graph into two pieces of comparable size instead of exactly equal size). Dense k-Subgraph and Bipartite Clique are natural graph theoretic problems which may also have algorithmic applications (e.g., Bipartite Clique problem is same as finding large monochromatic 1-squares in a 0-1 array. This may have applications to vision in processing arrays of pixels).

The main result in this paper is an inapproximability result for the above three problems. A formal statement of the result is :

Theorem 1.1 *Let $\epsilon > 0$ be an arbitrarily small constant. Assume that SAT does not have a probabilistic algorithm that runs in time 2^{n^ϵ} on an instance of size n. Then there is no polynomial time (possibly randomized) algorithm for Graph Min-Bisection, Dense k-Subgraph or Bipartite Clique that achieves an approximation ratio of $1 + \frac{1}{2^{O(1/\epsilon \log(1/\epsilon))}}$. In particular, assuming $\mathrm{NP} \not\subseteq \cap_{\epsilon>0}$ $\mathrm{BPTIME}(2^{n^\epsilon})$, these three problems have no PTAS.*

Note that the inapproximability factor $1 + \frac{1}{2^{O(1/\epsilon \log(1/\epsilon))}}$ approaches 1 as ϵ tends to 0. Thus we have a (strange) trade-off between the quality of the inapproximability result and the complexity assumption that SAT has no 2^{n^ϵ} time algorithm. The inapproximability factor for Bipartite Clique can be *boosted* via randomized graph products (this is done as in [10], [11]; a proof appears in Appendix D). This gives the following theorem [1] :

Theorem 1.2 *Let $\epsilon > 0$ be an arbitrarily small constant. Assume that SAT does not have a probabilistic algorithm that runs in time 2^{n^ϵ} on an instance of size n. Then there is no polynomial time (possibly randomized) algorithm for Bipartite Clique that achieves an approximation ratio of $N^{\epsilon'}$ on graphs of size N where $\epsilon' = \frac{1}{2^{O(1/\epsilon \log(1/\epsilon))}}$.*

We would like to mention that independent of our work, Feige and Kogan [20] obtained the following (weaker) results for the Bipartite Clique problem :

Theorem 1.3 *([20]) Bipartite Clique problem is hard to approximate within factor $2^{(\log n)^\delta}$ for some $\delta > 0$ assuming that 3-SAT $\notin \mathrm{DTIME}(2^{n^{3/4+\epsilon}})$ for some $\epsilon > 0$. Also, the problem is NP-hard to approximate within some constant factor if the Clique problem is NP-hard to approximate within factor $n/2^{c\sqrt{\log n}}$ for some $c > 0$.*

[1] The FOCS 2004 version of this paper contained a bug. We erroneously claimed that ϵ' could be made independent of ϵ in Theorem 1.2 .

1.2 Minimum Distance of Code Problem

As mentioned earlier, our hardness results are proved via a reduction from the Minimum Distance of (Linear) Code problem that we state next.

Minimum Distance of Code : Given a $N \times n'$ matrix $\mathbf{A}$ over a field $\mathbb{F}$, find a non-zero vector $\mathbf{z} \in \mathbb{F}^{n'}$ that minimizes the fraction of non-zero coordinates in the vector $\mathbf{Az}$. If one interprets the columns of $\mathbf{A}$ as the basis for a linear code, the problem is to find the (relative) minimum distance of the code. Let $OPT(\mathbf{A})$ denote the optimum value.

Remark 1.4 *All instances of MDC in this paper have the following properties : (1) The columns of* $\mathbf{A}$ *are independent and* $N \geq n'$*; otherwise the problem becomes trivial. (2) The field* $\mathbb{F}$ *has characteristic* 2*. This enables us to concatenate the code with a binary code. (3)* $N \leq |\mathbb{F}| \leq N^2$*.*

A result of Dumer, Micciancio and Sudan [15] says that MDC is NP-hard to approximate within any constant factor on binary asymptotically good codes. The following result can be obtained from Dumer et al.'s result (a proof appears in Appendix B). It is the starting point for our reduction.

Theorem 1.5 *There exists an absolute constant* C *such that the following holds. For every integer* K*, there is a reduction from SAT to MDC satisfying (think of* K *as a large constant) :*

1. *The MDC instance is over a field* $\mathbb{F}$ *of characteristic* 2*, has* N *rows and* $N \leq |\mathbb{F}| \leq N^2$*.*
2. *(Completeness :) YES instance of SAT is mapped to an MDC instance* $\mathbf{A}$ *with* $OPT(\mathbf{A}) \leq \frac{1}{2^K}$*.*
3. *(Soundness :) NO instance of SAT is mapped to an MDC instance* $\mathbf{A}$ *with* $OPT(\mathbf{A}) \geq 1 - \frac{1}{2^{2^K}}$*.*
4. *The reduction runs in time* n^{CK} *and in particular,* $N \leq n^{CK}$*.*

We would like to note that Dumer et al.'s reduction and therefore reduction in Theorem 1.5 is randomized (the completeness and soundness properties hold w.h.p.). The PCPs in this paper are constructed from Theorem 1.5 and their construction is randomized too, though we do not explicitly state so.

1.3 PCPs

We state the PCP Theorem for future reference. The theorem was proved by Arora, Safra [6] and Arora et al. [5]. See Arora's PhD thesis [2] for a comprehensive introduction to the subject. The PCP Theorem states that languages in NP have short membership proofs that can be efficiently checked by a probabilistic constant-query verifier.

Theorem 1.6 *(The PCP Theorem) For every language* $L \in \mathrm{NP}$*, there is a polynomial time probabilistic verifier* V *that has access to input* x *and a supposed proof* Π*. Size of the proof* Π *is polynomial in the size of the input* $|x| = n$*. The verifier uses* $O(\log n)$ *random bits, reads* $d = O(1)$ *queries from* Π *(choice of query locations depends on verifier's random bits), performs a pre-determined test on the query bits and then accepts or rejects. The verifier has the following completeness and soundness properties :*

- *(Completeness/YES Case :)* $x \in L \implies \exists\, \Pi \quad s.t. \quad \Pr[V(\Pi) = accept] = 1$
- *(Soundness/NO Case :)* $x \notin L \implies \forall\, \Pi \quad \Pr[V(\Pi) = accept] \leq \frac{1}{10}$

Here is an equivalent formulation of the PCP Theorem as an inapproximability result.

Theorem 1.7 *(The PCP Theorem) There exists an absolute constant $c < 1$ and a polynomial time reduction from 3SAT to MAX-3SAT mapping instance ϕ of 3SAT to instance ψ of MAX-3SAT with these properties :*

- *(YES Case :) If ϕ is satisfiable, then so is ψ, i.e. $OPT(\psi) = 1$.*
- *(NO Case :) If ϕ is unsatisfiable, then $OPT(\psi) \leq c$, i.e. no assignment to ψ satisfies more than a fraction c of its clauses.*

In particular, MAX-3SAT is hard to approximate within factor better than $1/c$ unless P = NP.

1.4 Quasi-random PCPs

In this section, we introduce the notion of Quasi-random PCPs. The main contribution of the paper is a construction of a quasi-random PCP that immediately implies the inapproximability results for problems in Theorem 1.1. In Section 1.5, we explain why the known PCPs are not quasi-random and in Section 1.6, we explain our new ideas for building Quasi-random PCPs.

Quasi-random PCPs focus on the distribution of queries made by the verifier. We require that the distribution depends on whether the input to the PCP verifier is a YES input or a NO input. In the NO case, the queries are required to be distributed *randomly* over the proof and in the YES case, the distribution is required to be *far for being random.* This is quite counter-intuitive and mystical at first sight. The verifier does not know whether the input is YES or NO; then how can he make his query pattern depend on the YES/NO case ? However, looking at some known PCPs makes this less mystical. For example, let us look at Holmerin's [27] 4-query PCP which has the following properties : (i) In the YES case, there exists a set of half the bits in the proof such that no PCP test accesses all 4 queries from this set. (ii) In the NO case, for any set of half the bits in the proof, at least $\frac{1}{16} - \epsilon$ fraction of the tests access all 4 queries from this set.

Thus it is conceivable (and necessary to prove inapproximability results) that the query pattern depends on the YES/NO case, without the verifier knowing which case it is. The Quasi-random PCPs deal with one specific random-like property of query pattern. Here is the definition :

Definition 1.8 *(**Quasi-random PCP**) A d-query PCP is called quasi-random if*

- *In the YES case, there exists a set of half the bits Π_0 in the proof Π such that at least $(1 - o(1))\frac{1}{2^{d-1}}$ fraction of tests access all d queries from Π_0.*
- *In the NO case, for any set Π_* of half the bits in the proof Π, the fraction of tests that access all d queries from Π_* is $(1 \pm o(1))\frac{1}{2^d}$.*

Thus the query pattern has random-like property in the NO case and in the YES case, the property is violated.

Note that the definition of quasi-random PCPs includes conditions in both the YES and NO cases. However, by abuse of terminology, we will often refer only to the condition in the NO case. The following theorem states the main PCP construction in the paper.

Theorem 1.9 *For every $\epsilon > 0$, there exists an integer $d = O(1/\epsilon \log(1/\epsilon))$ such that the following holds : there is a PCP verifier for SAT instance of size n satisfying :*

1. *The proof* Π *for the verifier is of size* 2^{n^ϵ}.

2. *The verifier queries* d *bits from the proof. Let* Q *denote the set of query bits.*

3. *Every query is uniformly distributed over* Π *(two different queries are of course correlated).*

4. *(YES Case/Completeness :) Suppose the SAT instance is a YES instance and* Π *is a correct proof. Let* Π_0 *be the set of* 0*-bits in the proof (it contains half the bits from the proof). Then*

$$\Pr_Q[Q \subseteq \Pi_0] \geq (1 - O(1/d))\ \frac{1}{2^{d-1}}$$

   The probability is taken over the random test of the verifier.

5. *(NO Case/Soundness:) Suppose the SAT instance is a NO instance, and let* Π_* *be any set of half the bits from* Π. *Then*

$$\frac{1}{2^d} - \frac{1}{2^{20d}} \ \leq\ \Pr_Q[Q \subseteq \Pi_*] \ \leq\ \frac{1}{2^d} + \frac{1}{2^{20d}}$$

Note that Theorem 1.9 says nothing about the test the verifier performs on the query bits or the acceptance probability in completeness/soundness case. When we actually construct the PCP, we will use a homogeneous linear test on query bits.

The inapproximability results for Graph Min-Bisection, Dense k-Subgraph and Bipartite Clique follow easily from the PCP in Theorem 1.9 (see Appendix C). The reductions follow the same outline as Feige's reductions [17].

In the next two sections, we explain why the known PCPs are not quasi-random and our new ideas for building Quasi-random PCPs.

1.5 Why Known PCPs are not Quasi-random

Consider Håstad's 3-query PCP [26] as an illustration. The PCP can be abstracted out in the following manner : the proof Π is partitioned into blocks

$$\Pi = (A_1, A_2, \ldots, A_M\ ; B_1, B_2, \ldots, B_{M'})$$

Every block A_i, B_j is of constant length and it is supposed to be a Long Code. This is however irrelevant for present discussion. The blocks B_j's are *bigger* than the blocks A_i's. The verifier picks a pair of blocks (A_i, B_j) and reads 3 bits from the proof : one from block A_i and two from block B_j.

It is easy to see why Håstad's PCP fails to be quasi-random (to be precise, it fails the NO Case of Definition 1.8). Construct a set Π_* containing half the bits from the proof Π as follows : for every block A_i or B_j, we include the entire block in Π_* with probability $\frac{1}{2}$ and exclude it from Π_* with probability $\frac{1}{2}$. Clearly, the probability that all 3 queries access bits from Π_* equals $\frac{1}{4}$ whereas quasi-randomness condition requires this probability to be close to $\frac{1}{8}$.

Why does Håstad's verifier access two bits from the bigger block B_j ? When one analyzes the PCP using Fourier analysis, it is necessary that the Fourier coefficient of the block B_j appears in squared form. Reading two bits from block B_j is a way to introduce redundancy and ensure that we have the Fourier coefficient $\widehat{B}_j^2$. However, as we observed, this form of redundancy leads to loss of quasi-randomness.

Though the above discussion applies only to one specific PCP (namely Håstad's PCP) and only one specific analysis technique (namely Fourier analysis), it turns out that all known PCPs and known techniques suffer from the same problem. On the high level, the proof is partitioned into blocks, the verifier reads q queries and reads more than one query from at least one block. This typically happens when the verifier, instead of reading a desired bit, reads two or more related bits and infers the value of the desired bit via error-correction (PCPs, almost by definition, must use error-correction in some form or the other. See for example, Hadamard Code Testing [12] and Long Code Testing [9]). In particular, the verifier accesses less than q blocks and therefore fails to be quasi-random. Let us formulate this as an observation :

Fact 1.10 *Any q-query PCP where the proof is partitioned into blocks and the verifier accesses less than q blocks is not quasi-random.*

Thus we do need redundant bits and some form of error-correction; but at the same time, we do not want to store all these bits at the same physical location (e.g. the same block B_j as in Håstad's PCP).

1.6 Basic Idea behind Our PCPs

Now we explain how we achieve the seemingly impossible task of constructing a Quasi-random PCP. Here we present only the basic idea that has to fit in a much bigger context.

We look at the Low Degree Test which is an important building block of the PCP Theorem ([34], [6]). Given a function $f : \mathbb{F}^m \mapsto \mathbb{F}$, the test verifies that f is a degree $d-1$ polynomial (think of d as a constant). The verifier has access to a table of values of f, call it *points table*. In addition, the verifier has access to a *lines table* which contains, for every line ℓ in $\mathbb{F}^m$, a degree $d-1$ univariate polynomial $f|_\ell$. The polynomial $f|_\ell$ is supposed to be the restriction of f to the line ℓ. The verifier works as follows : pick a random line ℓ and a random point $\overline{v}$ on the line. Read $f(\overline{v})$ from the points table and read the polynomial $f|_\ell$ from the lines table. Accept if and only if $f(\overline{v})$ and $f|_\ell$ are consistent, i.e. if and only if the value produced by $f|_\ell$ at point $\overline{v}$ is same as the value $f(\overline{v})$.

In our PCP construction however, the verifier has no access to the additional lines table (this feature is also used independently in a recent paper of Ben-Sasson et al. [8]). Instead, the verifier proceeds as follows :

- Pick a random line ℓ and $d+1$ random points $\overline{v}_1, \overline{v}_2, \ldots, \overline{v}_{d+1}$ on the line. Read the values $f(\overline{v}_1), f(\overline{v}_2), \ldots, f(\overline{v}_{d+1})$.
- Let $f|_\ell$ be the unique degree $d-1$ univariate polynomial that takes values $f(\overline{v}_2), f(\overline{v}_3), \ldots, f(\overline{v}_{d+1})$ at points $\overline{v}_2, \overline{v}_3, \ldots, \overline{v}_{d+1}$ respectively.
- Accept if and only if $f(\overline{v}_1)$ and $f|_\ell$ are consistent.

Thus the verifier reads $f(\overline{v}_1)$ and then reads additional d random points on the line and *reconstructs* the line polynomial. This is the so-called Outer Verifier.

Now let us see why this PCP is quasi-random (the queries are non-boolean though). Let Π_* be any set of half the points from $\mathbb{F}^m$. What is the probability that all $d+1$ queries of the verifier fall in Π_* ? It is easy to show that lines in $\mathbb{F}^m$ have excellent *mixing property*. For almost all lines ℓ, the fraction of points on ℓ that are in Π_* is very close to $\frac{1}{2}$. Once the line is chosen, the verifier picks $d+1$ random points on the line. Therefore, the probability that all the queries fall in Π_* is very close to $\frac{1}{2^{d+1}}$ and the PCP is quasi-random !

However the queries are elements of $\mathbb{F}$ whereas we want to have boolean queries. We assume that $\mathbb{F}$ has characteristic 2 and therefore the field elements can be represented as bit-strings of length $\log|\mathbb{F}|$. We use a standard technique for constructing a PCP with boolean queries. The verifier expects the proof to contain, instead of a value $f(\overline{v})$, the *Hadamard Code* of the bit-string $f(\overline{v})$. We can build an appropriate Inner Verifier that reads one bit from the (supposed) Hadamard Code of $f(\overline{v}_i)$ for $i = 1, 2, \ldots, d+1$. The test of the Outer Verifier is replaced by a meaningful test on the $d+1$ bits read.

The Inner Verifier is analyzed using Fourier analysis. Recall that in Håstad's PCP, one needs to have the squared Fourier coefficient $\widehat{B}_j^2$ and this is achieved by reading two bits from the block B_j. Our Fourier analysis also needs something similar. However, we cannot read more than one bit from any block (= Hadamard Code); otherwise we lose quasi-randomness property. We get around this problem by a clever trick. We modify the verifier as follows : as before, the verifier picks a random line ℓ and $d+1$ points on it $\overline{v}_1, \overline{v}_2, \ldots, \overline{v}_{d+1}$ and performs the test described. In addition, the verifier picks *another* set of $d+1$ random points on the line, say $\overline{u}_1, \overline{u}_2, \ldots, \overline{u}_{d+1}$. He verifies that the degree d polynomial reconstructed from values $f(\overline{u}_1), f(\overline{u}_2), \ldots, f(\overline{u}_{d+1})$ is identical to the polynomial reconstructed from $f(\overline{v}_1), f(\overline{v}_2), \ldots, f(\overline{v}_{d+1})$ (this polynomial actually happens to be of degree $d-1$). This is exactly the analogue of reading two bits from the block B_j in Håstad's PCP. The additional queries look wasteful, but this is precisely how we introduce redundancy (= error-correction) without losing quasi-randomness. Unfortunately, the exact significance of this trick will not be clear unless one looks at the actual Fourier analysis.

Remark 1.11 *Definition 1.8 requires that the query pattern of the PCP verifier is different in the YES and NO case. We would like to stress that the query pattern in the YES and NO cases is the same at the level of (modified) Outer Verifier since in both YES and NO case, the verifier picks random points on a randomly selected line. However, quite magically, the difference in the query pattern in the YES and NO cases manifests itself at the Inner Verifier level.*

1.7 Comparison with Feige's Hypothesis

Feige [17] makes the following hypothesis about average-case hardness of 3SAT.

Feige's Hypothesis : *Let ϕ be a 3SAT formula with n variables and Cn clauses where every clause is picked at random from the set of all possible clauses. Then for an arbitrarily large constant C, there is no polynomial time algorithm that (1) says YES if ϕ is satisfiable and (2) says NO at least* 50% *of the times for a random formula ϕ.*

Assuming this hypothesis, Feige shows that Graph Min-Bisection, Dense k-Subgraph and Bipartite Clique have no PTAS (Alekhnovich [1] later shows these results assuming random instances of 3-Linear-Equations are hard).

Let us see how we are able to bypass Feige's hypothesis and still prove these inapproximability results. Consider the following PCP constructed from an instance ϕ of Gap-3SAT. The proof supposedly consists of a satisfying assignment to the 3SAT formula ϕ. The verifier picks one clause at random, queries the three variables in the clause and checks if the clause is satisfied. Now what if the formula ϕ was a random instance of 3SAT ? Then the query pattern of the verifier would be *truly random.* Thus Feige's hypothesis, in some sense, says that there exists a PCP whose query pattern is *truly random.*

On the other hand, quasi-randomness is only one specific property which is satisfied by a truly random query pattern. Applied to 3SAT, the quasi-randomness property says that for any set of half the variables, $(1 \pm o(1))\frac{1}{8}$ fraction of the clauses have all their 3 variables from this set. We make the following observation : In order to prove inapproximability results for graph theoretic problems, it suffices to have hard instances of 3SAT (or any CSP with constraints on bounded number of variables) with quasi-randomness property. One does not really need the 3SAT instances to be truly random, as in Feige's paper. We are able to construct such Quasi-random PCPs, thereby obtaining the desired inapproximability results.

1.8 Overview of the Paper

The main result in this paper is Theorem 1.9. The proof of this result is quite lengthy, involving a sequence of reductions (see Fig. 1).

We introduce a new problem called Homogeneous Algebraic Constraint Satisfaction Problem (Algebraic CSPs have been used before, see [24] for example. Our CSPs have additional property that all constraints are homogeneous linear and we do not allow the trivial solution that is identically zero). Our proof can be divided into two parts : First we reduce MDC to HomAlgCSP. This reduction appears in Sections 3, 4, 5 and 6. This is perhaps the most interesting part of the paper. The modules used in this reduction, namely the Generalized Sum-Check protocol, Splitability Certificate and Subspace Membership Certificate could be of independent interest.

Then we construct a PCP verifier from the HomAlgCSP instance. The verifier consists of an Outer Verifier (Section 7 and 8) and the Inner Verifier (Section 10). We actually need to modify the Outer Verifier before constructing the Inner Verifier. This Modified Outer Verifier appears in Section 9.

We recommend that in the first reading, the paper be read in the following way (i) Browse through Section 2 that gives an overview of techniques and the high level view of the reduction. Also take a look at Appendix A that gives some of the facts/tools used. (ii) Read the beginning of Section 3 and statement of Theorem 3.3. Skip the rest of Section 3 as well as Sections 4, 5 and 6 which are devoted to proving Theorem 3.3. Though this is where the meat of the paper lies, it is rather lengthy and one might feel lost in the first reading. (iii) Read Section 7 and Theorem 7.5. Skip Section 8. (iv) Read Section 9 and 10.

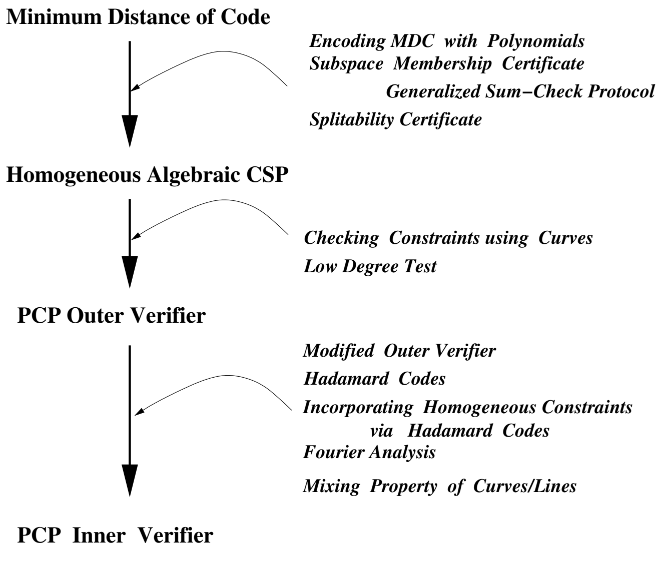

Figure 1: Overview of Full Reduction

2 Techniques and High Level View of the Reduction

In this section, we give an overview of techniques in the paper. This is a general overview that hides many subtleties involved in our reduction.

2.1 MDC to Homogeneous Algebraic CSP

HomAlgCSP is the following problem (see Definition 3.1) : Given space $\mathbb{F}^m$, the goal is to find a non-zero low degree m-variate polynomial f that satisfies maximum number of constraints. Every constraint is a homogeneous linear constraint on $O(1)$ values of f at specific points in $\mathbb{F}^m$. As explained in Remark 3.2, HomAlgCSP is a very similar problem to MDC. A reduction from MDC to HomAlgCSP transforms (see Theorem 3.3) the *global* constraints of MDC to *local* constraints of HomAlgCSP that depend only on $O(1)$ values of f at specific points. This global to local transformation is a major step in our reduction (as must be the case with any PCP).

The reduction from MDC to HomAlgCSP works as follows : It is easy to encode MDC as an instance of the following problem : Given a subspace of low degree polynomials $\mathcal{B}$ and a set of points $\mathcal{P} \subseteq \mathbb{F}^m$, find a non-zero polynomial $f \in \mathcal{B}$ that vanishes at maximum number of points in $\mathcal{P}$. Note that we now have local constraints, one for every point $\overline{p} \in \mathcal{P}$, saying that f vanishes at $\overline{p}$. However, the main trouble is, how do we enforce the condition that $f \in \mathcal{B}$?

We do this by constructing what we call a *Subspace Membership Certificate*. It is a low degree polynomial g that supposedly certifies that $f \in \mathcal{B}$. The polynomial g needs to be of a certain type and this is certified by constructing another polynomial h which we call the *Splitability Certificate*. We add a collection of local constraints on values of f, g and h that enforce the consistency between the three polynomials. The polynomials g and h are constructed as follows.

We observe that $f \in \mathcal{B}$ if and only if the value of f at any point $\mathbf{q} \in \mathbb{F}^m$ is a *weighted sum* of its values at some well-chosen points. Thus we can check whether this *sum-check* holds for a random point $\mathbf{q}$. We employ a generalization of the Sum-Check protocol in order to carry out the sum-check. The usual Sum-Check protocol sums over values of f over the boolean cube $\{0,1\}^m$ whereas our protocol sums over an arbitrary subset of the boolean cube. We choose the subset to consist of only low hamming-weight points of the boolean cube so that the *sum-check polynomials* remain of low degree.

The polynomial g (i.e. the Subspace Membership Certificate) is a linear combination of all the sum-check polynomials. For the Sum-Check protocol to work, the sum-check polynomials need to be independent in some sense. The polynomial h (i.e. the Splitability Certificate) certifies that this is indeed so. The polynomials g and h together form a certificate that $f \in \mathcal{B}$. Finally, we *club together* the three polynomials f, g and h into a single polynomial ϕ that serves as an instance of HomAlgCSP. The constraints are all homogeneous linear on $O(1)$ values of ϕ at specific points.

Let us illustrate the power of Subspace Membership Certificate construction by an example. Let us say we have a low degree m-variate polynomial f that is given as a table of values. We want to verify that f is multilinear (i.e. f has no monomial that looks like $x_1^2 x_2 x_3$). How many queries do we need to the table of values of f (and possibly to an additional proof) ?

The straightforward way is to pick a random point in $\mathbb{F}^m$ and check that on *every* axis-parallel line passing through this point, f behaves as a linear function (this is precisely what [18] does). However, this needs $O(m)$ queries. On the other hand, our construction gives a query-efficient way of certifying that f is multilinear. The certificate consists of a pair of new low degree polynomials (g, h) and we need only $O(1)$ queries to tables of values of f, g and h ! Moreover, the test is homogeneous linear. The fact that the certificate itself is a polynomial is crucial for our purpose.

2.2 HomAlgCSP to Outer Verifier

We construct a PCP Outer Verifier from the HomAlgCSP instance. Let f be the polynomial required in the HomAlgCSP instance. The Outer Verifier expects as a proof, a table of values of f. The verifier needs to verify the following : (1) f is indeed a non-zero low degree polynomial and (2) Values of f satisfy constraints of HomAlgCSP.

It is well-known how to handle the first task, namely, verifying that f is a degree d polynomial. The so-called Low Degree Test expects an additional proof : a table that gives for every line in $\mathbb{F}^m$, a univariate polynomial that is supposedly the restriction of f to the line. The verifier picks a random line and a point on the line, and checks consistency between the table of points and table of lines. For our purposes however, the verifier has no access to the *lines table*. The verifier, instead, reads $d+1$ values of f on a line ℓ and reconstructs by himself, the restriction of f to line ℓ. In this paper, the degree d of polynomials is a constant and therefore, the verifier still makes only a constant number of queries. We need the strong analysis of the LDT by Arora and Sudan [7].

The next task of the verifier is to verify that f satisfies certain constraints. The verifier picks one constraint at random that depends on k values of f at specific points, say $\{\overline{p}_i\}_{i=1}^{k} \in \mathbb{F}^m$. In principle, the verifier could just read off these values and verify the constraint. However, it is well-known in PCP literature that one needs to have some form of *error correction* and infer the values of f at points $\{\overline{p}_i\}_{i=1}^{k}$ from the global behavior of f. We use another well-known primitive to handle this. We pick a random *curve* L that passes through the points $\{\overline{p}_i\}_{i=1}^{k}$. Restriction of f to the curve L is again a low degree univariate polynomial that we reconstruct by reading sufficiently many values of f on the curve. After reconstructing this *curve-polynomial*, we get values of f at points $\{\overline{p}_i\}_{i=1}^{k}$ and verify that they indeed satisfy the constraint.

We in fact combine the two tasks of the verifier. We pick a random pair of a curve and a line that intersect at a point. The verification procedures described above are carried out both for the line and the curve. We in addition check that the line and the curve are consistent at the intersection point. This completes the description of the Outer Verifier.

Now let us see what exactly the Outer Verifier achieves. Let us restrict ourselves only to the first task (i.e. the variant of line-point LDT where the line polynomial is reconstructed by reading several values on the line). Recall that our eventual goal is to build a Quasi-random PCP. The definition of the Quasi-random PCP requires that if Π_* is any subset of the proof of half the size, and the number of queries is q, then the probability that all queries fall in Π_* is essentially $\frac{1}{2^q}$.

Let us illustrate that the Outer Verifier achieves the quasi-randomness property. Let Π_* be any set of half the points from $\mathbb{F}^m$. It is easy to show that lines in $\mathbb{F}^m$ have excellent *mixing property*, i.e. for any such set Π_*, for almost every line ℓ, the fraction of points on ℓ belonging to Π_* is very close to $\frac{1}{2}$. Now recall that the Outer Verifier picks a random line and q random points on the line. Thus the probability that all his queries fall into Π_* is essentially $\frac{1}{2^q}$. A similar mixing property holds for curves as well (see Appendix A.4) and therefore, the Outer Verifier is quasi-random.

2.3 Outer Verifier to Modified Outer Verifier

The purpose of the Modified Outer Verifier is exactly as explained in Section 1.5 and 1.6. It is the analogue of reading two bits from block B_j in Håstad's 3-query PCP. The Modified Outer Verifier introduces redundancy (= error-correction) and this enables us to bound the Fourier terms in the analysis of the Inner Verifier. This works in exactly the same way that the squared Fourier coefficient $\widehat{B}_j^2$ enables the analysis of Håstad's PCP.

Recall that the Outer Verifier reads values of f on $d+1$ points from a line and a curve and reconstructs the line and curve polynomials. The Modified Outer Verifier reads values of f on additional $d+1$ points from the line and the curve and checks that he again gets the *same* line and curve polynomials. Thus one introduces redundancy, but without losing quasi-randomness.

2.4 Modified Outer Verifier to Inner Verifier

The final step is to build the Inner Verifier and prove the quasi-randomness property. Note that the proof for the Modified Outer Verifier is a table of values of a polynomial $f : \mathbb{F}^m \mapsto \mathbb{F}$. We work only with fields of characteristic 2. Therefore denoting $|\mathbb{F}| = 2^l$, we can think of elements of $\mathbb{F}$ as l-bit strings. The Inner Verifier expects as a proof, the Hadamard Codes of these l-bit strings. Let us say the Modified Outer Verifier makes q queries. Then the Inner Verifier also makes q queries, reading one bit each from the corresponding Hadamard Code. As pointed out earlier, it is crucial that one reads only one bit from each code/block; otherwise quasi-randomness property fails.

The quasi-randomness of Inner Verifier is proved using Fourier analysis. We prove the desired two properties : (1) In completeness case, let Π_0 be the set of 0-bits in the proof. This consists of half the bits in the proof. The probability that all q queries fall in Π_0 is essentially $\frac{1}{2^{q-1}}$. (2) In soundness case, we assume on the contrary that there is a set Π_* of half the bits in the proof such that with probability $\frac{1}{2^q} \pm \delta$, all queries fall in Π_*. Then we show that one can *decode* the proof and construct a table $f : \mathbb{F}^m \mapsto \mathbb{F}$ that the Outer Verifier accepts with probability $\delta^{O(1)}$. This is a contradiction if the Outer Verifier is chosen to have sufficiently small soundness. This is ensured by choosing the MDC instance (and therefore the HomAlgCSP instance) with small soundness. The decoding is the standard probabilistic decoding in PCP literature. A supposed Hadamard Code is decoded to α with probability $\widehat{A}_\alpha^2$, i.e. proportional to the square of corresponding Fourier coefficient. We use several nice properties of Hadamard Codes that we list below (the popular Long Codes do not work; Hadamard Code based Inner Verifier was used earlier in [29]) :

1. The Modified Outer Verifier reads q values of f and performs homogeneous linear tests. There is a natural way of accessing Hadamard Codes so that only those Fourier coefficients survive which satisfy all the homogeneous linear constraints (this is where we need homogeneity). Thus, as far as the Inner Verifier is concerned, this testing comes for free.

2. If we have access to Hadamard Codes of x and y, then we essentially have access to Hadamard Code of the concatenated string $x \circ y$. The process of accessing a bit from Hadamard Code of $x \circ y$ can be simulated by reading one bit each from codes of x and y.

3. The zero Fourier coefficient of a supposed Hadamard Code equals the fraction of 1's in the supposed code.

There is a nice interpretation of the Fourier coefficients in the analysis. The zero coefficient corresponds to the term $\frac{1}{2^q}$ in the soundness probability. The non-zero coefficients are used to *decode* the proof and define a strategy for the Outer Verifier. This in turn gives a solution to the HomAlgCSP (and therefore the MDC) instance. Recall that the HomAlgCSP instance demands a *non-zero* polynomial f.

As mentioned in Remark 1.11, the difference in the query pattern of the verifier in the YES and NO cases is *created* at the Inner Verifier level. At the Outer Verifier level, the query pattern is the same, irrespective of the YES or NO case (in both cases, random points are chosen on a randomly selected line/curve).

3 MDC to Homogeneous Algebraic CSP : Encoding MDC with Polynomials

Now we begin our construction of the Quasi-random PCP. The first step is defining the HomAlgCSP problem :

Definition 3.1 *Let the HomAlgCSP $\mathcal{A}(f, k, d, m, \mathbb{F}, \mathcal{C})$ be the following problem (think of k as a fixed integer like 21, and d as a large constant integer) :*

1. *f is (supposed to be) m-variate degree d polynomial over field $\mathbb{F}$.*
2. *$\mathcal{C}$ is a system of constraints where every constraint is on k values of the polynomial f at certain points. This constraint is a conjunction of homogeneous linear constraints. Thus a typical constraint looks like*

$$\sum_{i=1}^{k} \gamma_{ij} f(\overline{p_i}) = 0 \;\; \textit{for } j = 1, 2, \ldots \qquad \textit{where } \overline{p}_i \in \mathbb{F}^m \textit{ and } \gamma_{ij} \in \mathbb{F}$$

   A constraint $C \in \mathcal{C}$ will be denoted by $C(\{\overline{p}_i\}_{i=1}^{k})$, thus indicating only the points on which the constraint is defined. The actual homogeneous constraints will be implicit.

3. *$\mathcal{C}$ has $|\mathbb{F}|^{O(m)}$ constraints.*

The goal is to find a polynomial f, not identically zero, so as to maximize the fraction of constraints satisfied. Let $OPT(\mathcal{A})$ denote the optimum.

Remark 3.2 *HomAlgCSP and MDC are similar problems. In MDC, we want to find a non-zero assignment to unknown variables that minimizes the number of unsatisfied homogeneous linear constraints. HomAlgCSP is essentially the same problem if we view the coefficients of the polynomial f as unknown variables (minor difference is that HomAlgCSP is a maximization problem and we allow a constraint to be an arbitrary conjunction of homogeneous linear constraints).*

However, the main difference is the following : for MDC, every constraint could depend on an arbitrary number of variables whereas for HomAlgCSP, every constraint has a compact/local representation in terms of values of f at a constant number of points. Thus a reduction from MDC to HomAlgCSP transforms global constraints to local ones. This is what PCPs are all about !.

The following Theorem states the main result in this and next three sections. It gives a gap-preserving reduction from MDC to HomAlgCSP.

Theorem 3.3 *Let $\mathbf{A}$ be an instance of MDC with size $N \times n'$ over field $\mathbb{F}$. Assume that $N \leq |\mathbb{F}| \leq N^2$. Let $\epsilon > 0$ be an arbitrarily small constant, $m = N^{\epsilon}$, and $d = \frac{100}{\epsilon}$ be an integer. There is a reduction from $\mathbf{A}$ to an instance of Homogeneous Algebraic CSP $\mathcal{A}(f, k = 21, d, m, \mathbb{F}, \mathcal{C})$ such that*

$$1 - OPT(\mathbf{A}) \;\leq\; OPT(\mathcal{A}) \;\leq\; \max\left\{1 - OPT(\mathbf{A}), O\left(\frac{d}{|\mathbb{F}|}\right)\right\} \tag{1}$$

We begin the proof of Theorem 3.3. The first step is to encode MDC as the following problem : Given a set of points $\mathcal{P}$ in $\mathbb{F}^m$, find a non-zero degree d polynomial f that is non-zero at minimum number of points from $\mathcal{P}$. In addition, f is required to be in a certain subspace of polynomials.

The next step is to define two more polynomials g and h that are supposed to serve as a *certificate* that f indeed belongs to the required subspace of polynomials. Then we define a collection of constraints all of which are homogeneous linear constraints on k' values of f, g and h. These constraints enforce the consistency between f, g and h as well as the condition that f takes non-zero values at a minimum number of points from $\mathcal{P}$. In the end, we will *club together* the three polynomials f, g and h into a single polynomial ϕ and all the constraints are homogeneous constraints on k values of ϕ. This would prove Theorem 3.3.

Remark 3.4 *Actually Theorem 3.3 holds for a more general setting of parameters. For any choice of integers* m *and* d *such that* $\binom{m}{d} \geq N$, *the reduction produces an instance of HomAlgCSP* $\mathcal{A}(f, k = 21, d' = 10d, m' = O(m^3 d), \mathbb{F}, \mathcal{C})$ *such that Equation (1) holds. However, we will stick to the setting of the parameters in Theorem 3.3 used in this paper.*

3.1 Encoding MDC with Polynomials

Let $\mathbf{A} = \{a_{ji}\}$ be an $N \times n'$ instance of MDC. We assume w.l.o.g. that the first n' rows of $\mathbf{A}$ form an identity matrix, i.e. $a_{ji} = 0$ if $1 \leq i \neq j \leq n'$ and $a_{ii} = 1$ for $1 \leq i \leq n'$. This can be achieved by rearranging rows so that the first n' rows are linearly independent and then multiplying $\mathbf{A}$ by a suitable $n' \times n'$ matrix. This transformation doesn't change $OPT(\mathbf{A})$.

Remark 3.5 *In the following, the polynomial* f *and the parameters* ϵ, m, d *are related to but not the same as in Theorem 3.3. They keep changing somewhat over the course of the reduction.*

Fix an arbitrarily small constant $\epsilon > 0$ and let $m = N^\epsilon$. Choose an integer d such that $N \leq \binom{m}{d}$. Thus $d = \frac{2}{\epsilon}$ suffices. Let $\mathcal{F}_0$ be a family of subsets of $[m]$ of size d such that $|\mathcal{F}_0| = N$. Let $\mathcal{F} \subseteq \mathcal{F}_0$ be a subfamily with $|\mathcal{F}| = n'$. The columns of the matrix $\mathbf{A}$ are indexed by $I \in \mathcal{F}$ and the rows are indexed by $J \in \mathcal{F}_0$ where the first n' rows are indexed by $I \in \mathcal{F}$.

For $J \in \mathcal{F}_0$, let $\overline{p}_J \in \mathbb{F}^m$ be the point with $\{0,1\}$-co-ordinates that is the characteristic vector of set J. Let $\mathbf{x}_J$ be the degree d monomial $\prod_{j \in J} x_j$. Let $\mathbf{x} = (x_1, x_2, \ldots, x_m)$ denote a vector of formal variables. The important property is that the monomial $\mathbf{x}_J$ evaluates to 1 at point $\overline{p}_J$ and evaluates to 0 at any point $\overline{p}_{J'}$ for $J' \neq J, |J'| = d$.

We will encode the columns of matrix $\mathbf{A}$ by polynomials in $\mathbb{F}[\mathbf{x}]$. For column I, define a degree d polynomial $f_I \in \mathbb{F}[\mathbf{x}]$ as

$$f_I := \sum_{J \in \mathcal{F}_0} a_{JI} \mathbf{x}_J \qquad (a_{JI} = a_{ji})$$

It is clear that $f_I(\overline{p}_J) = a_{JI}$ for $J \in \mathcal{F}_0, I \in \mathcal{F}$. In other words, the co-ordinates of the column indexed by I are given by $f_I(\overline{p}_J)\ \forall\ J \in \mathcal{F}_0$.

Let $span\{f_I | I \in \mathcal{F}\}$ denote the linear span of polynomials f_I's. Thus the MDC problem can be reformulated as :

MDC (Reformulation) : Find a non-zero polynomial $f \in span\{f_I | I \in \mathcal{F}\}$ so as to minimize the fraction of non-zero entries in the vector

$$\{\ f(\overline{p}_J)\ |J \in \mathcal{F}_0\ \}$$

How do we ensure that f belongs to the span of polynomials $\{f_I | I \in \mathcal{F}\}$? Next few sections are devoted to construction of two polynomials g and h that serve as a certificate that f indeed belongs to the span.

Overview of the Construction of Certificate Polynomials

We make a simple but important observation (Lemma 3.6) : f belongs to the span of $\{f_I | I \in \mathcal{F}\}$ if and only if for all points $\mathbf{q} \in \mathbb{F}^m$, the value $f(\mathbf{q})$ is a weighted sum of the values $\{f(\overline{p}_I) | I \in \mathcal{F}\}$. So we check whether this property holds for a randomly selected point $\mathbf{q}$. The checking procedure could be carried out by reading the values $f(\mathbf{q})$ and $\{f(\overline{p}_I) | I \in \mathcal{F}\}$. This however would take too many queries whereas we can afford only a constant number of queries. The idea is to use the Sum-Check Protocol used in the proof of the PCP Theorem. It turns out that the protocol needs values of linear combinations of *sum check polynomials.* The first *certificate polynomial* g is the Hadamard Code over the sum-check polynomials. However, for the sum-check protocol to work, the sum-check polynomials need to be *independent* of each other in some sense. The second *certificate polynomial* h serves as a proof of this independence. We define a notion of a *splitable polynomial* and h is actually a certificate that the polynomial g is splitable. We call g and h as *Subspace Membership Certificate* and *Splitability Certificate* respectively.

3.2 Basic Observation for Constructing Subspace Membership Certificate

Let $f(\mathbf{x}) \in \mathbb{F}[\mathbf{x}]$ be the desired polynomial. Let $\mathbf{q}$ be a vector of formal variables. Consider the polynomial

$$\psi(\mathbf{q}, \mathbf{x}) := \sum_{I \in \mathcal{F}} f_I(\mathbf{q}) \mathbf{x}_I \tag{2}$$

Note that f_I's are of degree d, therefore ψ is of degree $2d$. We make the following important observation :

Lemma 3.6 *$f \in span\{f_I | I \in \mathcal{F}\}$ if and only if this formal identity holds :*

$$f(\mathbf{q}) = \sum_{I \in \mathcal{F}} \psi(\mathbf{q}, \overline{p}_I) f(\overline{p}_I) \tag{3}$$

Proof: Assume that $f = \sum_{I' \in \mathcal{F}} \lambda_{I'} f_{I'}$ where $\lambda_{I'} \in \mathbb{F}$. Then

$$\begin{aligned}
\sum_{I \in \mathcal{F}} \psi(\mathbf{q}, \overline{p}_I) f(\overline{p}_I) &= \sum_{I \in \mathcal{F}} \psi(\mathbf{q}, \overline{p}_I) \sum_{I' \in \mathcal{F}} \lambda_{I'} f_{I'}(\overline{p}_I) \\
&= \sum_{I \in \mathcal{F}} \psi(\mathbf{q}, \overline{p}_I) \lambda_I \qquad f_{I'}(\overline{p}_I) = a_{II'} = 0 \text{ if } I \neq I' \text{ and } 1 \text{ o.w.} \\
&= \sum_{I \in \mathcal{F}} \lambda_I \sum_{I'' \in \mathcal{F}} f_{I''}(\mathbf{q}) \cdot \mathbf{x}_{I''}(\overline{p}_I) \\
&= \sum_{I \in \mathcal{F}} \lambda_I f_I(\mathbf{q}) \\
&= f(\mathbf{q})
\end{aligned}$$

as desired. For the other direction, note that if $f(\mathbf{q}) = \sum_{I \in \mathcal{F}} \psi(\mathbf{q}, \overline{p}_I) f(\overline{p}_I)$, then we have

$$\begin{aligned}
f(\mathbf{q}) &= \sum_{I \in \mathcal{F}} f(\overline{p}_I) \sum_{I' \in \mathcal{F}} f_{I'}(\mathbf{q}) \mathbf{x}_{I'}(\overline{p}_I) \\
&= \sum_{I \in \mathcal{F}} f(\overline{p}_I) f_I(\mathbf{q})
\end{aligned}$$

as desired. ∎

Thus, one can probabilistically check whether f belongs to $span\{f_I | I \in \mathcal{F}\}$ by checking whether Equation (3) holds for a random $\mathbf{q} \in \mathbb{F}^m$. Note that verifying Equation (3) would need accessing all the values $f(\overline{p}_I)$. We however want to read only a constant number of values of f. The idea is to use the Sum-Check Protocol used in the proof of PCP Theorem. This protocol lets us verify that sum of values of a function over $\{0,1\}^m$ equals some target value. We need a generalization of this protocol where the sum is taken over the restricted set of points $\{\overline{p}_I | I \in \mathcal{F}\}$. An important feature of this generalized protocol is that the points $\{\overline{p}_I | I \in \mathcal{F}\}$ are low hamming-weight points (i.e. only d non-zero co-ordinates). This ensures that all the sum-check polynomials are of degree $O(d)$.

Before we get into the sum-check protocol, let us introduce the notion of *Splitability* of polynomials and how to construct a certificate for splitability. This construction will be a useful primitive later and it will introduce some of the new ideas. Admittedly, the reader would have no clue at this point how/why it is useful, but we think the construction is also of independent interest.

4 Splitability Certificate

For $1 \leq j \leq M$, let $\mathbf{x}^{(j)}$ denote a vector of m_j formal variables where the sets of variables for different j are disjoint. Let $\mathbf{x} = \cup_{j=1}^{M} \mathbf{x}^{(j)}$.

Definition 4.1 *A degree d polynomial $g \in \mathbb{F}[\mathbf{x}]$ is called Splitable if it can be written as*

$$g(\mathbf{x}) := \sum_{j=1}^{M} g_j(\mathbf{x}^{(j)}) \tag{4}$$

where $g_j \in \mathbb{F}[\mathbf{x}^{(j)}]$ are degree d polynomials.

In other words, splitability means that the variables in different sets $\mathbf{x}^{(j)}$ do not *interfere* with each other. In this section, we show how to *certify* that a given polynomial g is splitable. The certificate consists of another polynomial h. The certificate can be verified w.h.p. with constant number of accesses given tables of values of g and h.

Theorem 4.2 *Let $\mathbf{x} = \cup_{j=1}^{M} \mathbf{x}^{(j)}$ be a partition of variables and $g \in \mathbb{F}[\mathbf{x}]$ be a polynomial of degree d. Assume that every $\mathbf{x}^{(j)}$ has at most m variables. Let $h \in \mathbb{F}[\mathbf{z}]$ be a degree $2d$ polynomial that is supposed to be a certificate that g is splitable. The number of variables in $\mathbf{z}$ is bounded by $O(m^2M + dM)$. There is a verifier with the following properties :*

1. *The verifier accesses one value of g and 4 values of h.*

2. *If g is splitable, then there exists a unique polynomial h such that the verifier accepts with probability 1 and for any other h, the verifier accepts with probability at most $O(\frac{d}{|\mathbb{F}|})$. If $g \equiv 0$, then the corresponding $h \equiv 0$.*

3. *If g is not splitable, then no matter what h is (of degree $2d$), the verifier accepts with probability at most $O(\frac{d}{|\mathbb{F}|})$.*

4.1 Basic Idea

Let us explain the basic idea behind our construction by a small example. Let us say we have a polynomial $g(x_1, x_2, \ldots, x_m; w_1, w_2, \ldots, w_m)$ and we want to certify that g is splitable, i.e. g can be written as sum of two polynomials, one in $x_1, \ldots, x_m$ and the other in $w_1, \ldots, w_m$. Assume for the moment that g is guaranteed to be multilinear.

Here is our first attempt. We check for every pair (i_0, j_0) that g has no monomial involving the product $x_{i_0} w_{j_0}$. This can be done as follows. Substitute random values for all variables except x_{i_0}, w_{j_0} and check that the resulting (restricted) function is linear in x_{i_0} and w_{j_0}. More precisely, pick random values for variables $x_i, w_j, i \neq i_0, j \neq j_0$. Pick two sets of random values x_{i_0}, w_{j_0} and x'_{i_0}, w'_{j_0}. Pick a random value ζ and check that

$$\zeta \cdot g(\ldots, x_{i_0}, \ldots; \ldots, w_{j_0}, \ldots) + (1-\zeta) \cdot g(\ldots, x'_{i_0}, \ldots; \ldots, w'_{j_0}, \ldots) = \\ g(\ldots, \zeta x_{i_0} + (1-\zeta) x'_{i_0}, \ldots; \ldots, \zeta w_{j_0} + (1-\zeta) w'_{j_0}, \ldots)$$

where the dots indicate the same set of values for $x_i, y_j, i \neq i_0, j \neq j_0$. Note that we read only 3 values of g and do not need any additional certificate. Clearly, if g is multilinear and splitable, the test always accepts. If g has a monomial involving the product $x_{i_0} w_{j_0}$, the test accepts with probability at most $O(\frac{d}{|\mathbb{F}|})$.

Now the trouble is that we have to do this for *every* pair i_0, j_0 which we cannot afford to do (we want to keep the number of queries constant). To overcome this trouble, we use the following trick. Make the following substitution :

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix} = \begin{bmatrix} a_{11} & \ldots & a_{1m} \\ & \ldots & \\ & \vdots & \\ a_{m1} & \ldots & a_{mm} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_m \end{bmatrix}; \qquad \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_m \end{bmatrix} = \begin{bmatrix} b_{11} & \ldots & b_{1m} \\ & \ldots & \\ & \vdots & \\ b_{m1} & \ldots & b_{mm} \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_m \end{bmatrix} \tag{5}$$

Here a_{ij}, b_{ij}, u_i, v_j are new formal variables. After substituting values for x_i, y_j in the polynomial g, we get a new polynomial h in a_{ij}, b_{ij}, u_i, v_j. Here is a simple observation : g is splitable if and only if h has no monomial involving the product $u_1 v_1$.

The polynomial h will be the certificate. Using the checking procedure described earlier, we can check that h has no monomial involving $u_1 v_1$. Also, we can verify that g and h are consistent by checking that they agree on points which are related via the transformation (5).

This is the basic idea behind the splitability certificate. An additional twist is that g is not necessarily multilinear, but this possibility is handled easily as well.

4.2 Defining the Splitability Certificate

Now we prove Theorem 4.2. For $1 \leq j \leq M$, let m_j be the number of variables in $\mathbf{x}^{(j)}$. Let $\mathbf{y}^{(j)}$ denote a vector of $m_j - 1$ variables and let Y_j denote a single special variable. Let $\mathbf{T}^{(j)}$ denote a matrix of m_j^2 variables. The intended meaning is that $\mathbf{x}^{(j)} = \mathbf{T}^{(j)}(\mathbf{y}^{(j)} \circ Y_j)$. To be precise, we intend that

$$\mathbf{x}_i^{(j)} = \sum_{s=1}^{m_j - 1} \mathbf{T}_{is}^{(j)} \mathbf{y}_s^{(j)} + \mathbf{T}_{im_j}^{(j)} Y_j \quad \text{for } 1 \leq i \leq m_j \tag{6}$$

Let $\mathbf{T} = \cup_{j=1}^{M} \mathbf{T}^{(j)}$, $\mathbf{y} = \cup_{j=1}^{M} \mathbf{y}^{(j)}$ and $\mathbf{Y} = \cup_{j=1}^{M} \{Y_j\}$. Substituting the values of $\mathbf{x}^{(j)}$ into Equation (4), we get

$$g = \sum_{j=1}^{M} h_j(\mathbf{T}^{(j)}, \mathbf{y}^{(j)}, Y_j)$$

where each h_j has degree $2d$. We write h_j in increasing powers of Y_j so that

$$h_j := h_{j0}(\mathbf{T}^{(j)}, \mathbf{y}^{(j)}) + \sum_{l=1}^{2d} h_{jl}(\mathbf{T}^{(j)}, \mathbf{y}^{(j)}) Y_j^l$$

Therefore

$$g = \sum_{j=1}^{M} h_{j0}(\mathbf{T}^{(j)}, \mathbf{y}^{(j)}) + \sum_{j=1}^{M} \sum_{l=1}^{2d} h_{jl}(\mathbf{T}^{(j)}, \mathbf{y}^{(j)}) Y_j^l$$

The *splitability certificate* is obtained by replacing powers of Y_j by new variables. To be precise, for every $1 \leq j \leq M$, $1 \leq l \leq 2d$, let U_{jl} be a single formal variable and let $\mathbf{U}$ denote the vector of all such variables. The certificate is the polynomial

$$h(\mathbf{T}, \mathbf{y}, \mathbf{U}) := \sum_{j=1}^{M} h_{j0}(\mathbf{T}^{(j)}, \mathbf{y}^{(j)}) + \sum_{j=1}^{M} \sum_{l=1}^{2d} h_{jl}(\mathbf{T}^{(j)}, \mathbf{y}^{(j)}) U_{jl}$$

4.3 Checking the Splitability Certificate

The checking procedure consists of two parts : first we check that the polynomial h is a linear function of variables $\mathbf{U}$. Then we check that g and h are consistent (this automatically checks that g is indeed splitable).

Checking Linearity

Pick values for $\mathbf{T}^{(j)}, \mathbf{y}^{(j)}$ at random. Let $\mathbf{U}$, $\mathbf{U}'$ be two sets of values for variables in $\mathbf{U}$ picked at random. Pick a scaler/field value ζ at random. Accept if and only if

$$h(\mathbf{T}, \mathbf{y}, \zeta\mathbf{U} + (1-\zeta)\mathbf{U}') = \zeta \cdot h(\mathbf{T}, \mathbf{y}, \mathbf{U}) + (1-\zeta) \cdot h(\mathbf{T}, \mathbf{y}, \mathbf{U}')$$

Clearly, the test reads 3 values of h. It accepts with probability 1 if h is indeed linear in $\mathbf{U}$ and accepts with negligible probability (i.e. at most $O(\frac{d}{|\mathbb{F}|})$) otherwise.

Checking Consistency

Pick values for $\mathbf{T}^{(j)}, \mathbf{y}^{(j)}$ and variables Y_j at random. Define $\mathbf{x}^{(j)} = \mathbf{T}^{(j)}(\mathbf{y}^{(j)} \circ Y_j)$. Let $U_{jl} = Y_j^l$. Accept if and only if

$$g(\mathbf{x}) = h(\mathbf{T}, \mathbf{y}, \mathbf{U})$$

It is clear that if g is splitable, and h is defined appropriately, then this test accepts with probability 1. So the main task is proving soundness. Assume that g is not splitable. We want to show that the test accepts with negligible probability, no matter what certificate h the adversary gives. Let $\tilde{h}$ be the polynomial given by the adversary. We can assume that it is linear in $\mathbf{U}$ and thus looks like

$$\tilde{h}(\mathbf{T}, \mathbf{y}, \mathbf{U}) = \tilde{h}_0(\mathbf{T}, \mathbf{y}) + \sum_{j=1}^{M} \sum_{l=1}^{2d} \tilde{h}_{jl}(\mathbf{T}, \mathbf{y}) U_{jl}$$

For the given polynomial $g[\mathbf{x}]$, let $\tilde{g}(\mathbf{T}, \mathbf{y}, \mathbf{Y})$ be the polynomial obtained by substituting $\mathbf{x}^{(j)} = \mathbf{T}^{(j)}(\mathbf{y}^{(j)} \circ Y_j)$. The test accepts with negligible probability unless we have a formal identity

$$\tilde{g}(\mathbf{T}, \mathbf{y}, \mathbf{Y}) = \tilde{h}_0(\mathbf{T}, \mathbf{y}) + \sum_{j=1}^{M} \sum_{l=1}^{2d} \tilde{h}_{jl}(\mathbf{T}, \mathbf{y}) Y_j^l$$

Now the crucial point is that on the RHS, the powers Y_j^l do not *interfere* with each other. In other words, there is no monomial that involves the product $Y_j Y_{j'}$ where $j \neq j'$. On the other hand, if $g[\mathbf{x}]$ is not splitable, then it will have a monomial that involves the product $\mathbf{x}_i^{(j)} \mathbf{x}_{i'}^{(j')}$ for $j \neq j'$. This will give a monomial involving the product $Y_j Y_{j'}$ on the LHS, a contradiction !

5 Generalized Sum-Check Protocol

In this section, we describe the Generalized Sum-Check (GSC) Protocol. This protocol along with the construction of the Splitability Certificate (in previous section) is used to construct the Subspace Membership Certificate in the next section. The protocol is much more complicated than the usual Sum-Check protocol that allows us to verify that the sum of values of a polynomial f over the cube $\{0,1\}^m$ equals a target value. Our protocol needs to work for a sum over an arbitrary set of weight-d inputs (inputs in $\{0,1\}^m$ with exactly d co-ordinates equal to 1). The protocol could be of independent interest.

$GSC(a, \mathbb{F}, \tilde{f}, m, D, d, \mathcal{F}, \{g_{b,r} | b \in \{0,1\}, 0 \leq r \leq m-1\})$ consists of :

1. $\mathbb{F}$ is a field and $a \in \mathbb{F}$ is a *target* value.
2. $\tilde{f} \in \mathbb{F}[\mathbf{x}]$ is a degree D polynomial where $\mathbf{x} = (x_1, x_2, \ldots, x_m)$ is a vector of m formal variables. We are given $\tilde{f}$ as a table of values.
3. $\mathcal{F} \subseteq \binom{m}{d}$, i.e. $\mathcal{F}$ is a family of size d subsets of $[m]$.
4. Every $g_{b,r}$ is a polynomial in $\mathbb{F}[\boldsymbol{\alpha}, \boldsymbol{\theta}, X]$ where $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_m)$, $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_m)$ are vectors of formal variables and X is a single formal variable. Degree of $g_{b,r}$ is $D + d$ and $g_{b,r}$ is guaranteed to depend only on $\alpha_1, \alpha_2, \ldots, \alpha_r, \theta_1, \ldots, \theta_r, X$. The polynomials $g_{b,r}$ are given as table of values.

The goal is to verify that

$$a = \sum_{I \in \mathcal{F}} \tilde{f}(\overline{p}_I) \tag{7}$$

by reading a constant number of values of $\tilde{f}$ and of each of the polynomials $g_{b,r}$. The *sum-check polynomials* $g_{b,r}$ are supposed to assist us in verifying that the summation is true. We have the following theorem :

Theorem 5.1 *There is a protocol (verifier) for*

$$GSC(a, \mathbb{F}, \tilde{f}, m, D, d, \mathcal{F}, \{g_{b,r} | b \in \{0,1\}, 0 \leq r \leq m-1\})$$

satisfying these properties :

1. *The protocol picks $\boldsymbol{\alpha} \in \mathbb{F}^m$, $\boldsymbol{\theta} \in \mathbb{F}^m$ at random.*
2. *The protocol reads one value of $\tilde{f}$, namely, $\tilde{f}(\boldsymbol{\theta})$. For every $b \in \{0,1\}, 0 \leq r \leq m-1$, the protocol reads two values of $g_{b,r}$, namely, $g_{b,r}(\boldsymbol{\alpha}, \boldsymbol{\theta}, X = b)$ and $g_{b,r}(\boldsymbol{\alpha}, \boldsymbol{\theta}, X = \theta_{r+1})$.*
3. *The protocol performs a single homogeneous linear test on the target value a and the values read.*

4. *(Completeness :) If* $a = \sum_{I\in\mathcal{F}} \tilde{f}(\overline{p}_I)$*, then there exists a choice of sum-check polynomials* $\{g_{b,r}\}$ *such that the protocol accepts with probability* 1.

5. *If* $a \neq \sum_{I\in\mathcal{F}} \tilde{f}(\overline{p}_I)$*, then no matter what the sum-check polynomials* $\{g_{b,r}\}$ *are, the protocol accepts with probability at most* $O(\frac{D+d}{|\mathbb{F}|})$.

Before we describe the protocol, let us see what the intended meaning of $g_{b,r}$ is. For a set $I \in \mathcal{F}$, let I_i denote the i^{th} co-ordinate of its characteristic vector.

For $\boldsymbol{\alpha} \in \mathbb{F}^m, I \in \mathcal{F}, 0 \leq s \leq m$, let

$$w(I, \boldsymbol{\alpha}, s) := \prod_{1\leq i\leq s,\ I_i=1} \alpha_i$$

Note that in particular, for $s = 0$, $w(I, \boldsymbol{\alpha}, 0) = 1$. Also, since $|I| = d$, each $w(\cdot, \cdot, \cdot)$ is a monomial of degree at most d.

For $\boldsymbol{\theta} \in \mathbb{F}^m, I \in \mathcal{F}, 0 \leq s \leq m$, let $p(I, \boldsymbol{\theta}, s) \in \mathbb{F}^m$ be a point whose i^{th} co-ordinate equals θ_i for $1 \leq i \leq s$ and equals I_i for $s < i \leq m$. In particular, $p(I, \boldsymbol{\theta}, 0) = \overline{p}_I$ and $p(I, \boldsymbol{\theta}, m) = \boldsymbol{\theta}$.

Consider the following *partial sum* for $0 \leq r \leq m$:

$$S_r := \sum_{I\in\mathcal{F}} w(I, \boldsymbol{\alpha}, r)\tilde{f}(p(I, \boldsymbol{\theta}, r))$$

In particular,

$$\begin{aligned} S_0 &= \sum_{I\in\mathcal{F}} w(I, \boldsymbol{\alpha}, 0)\tilde{f}(p(I, \boldsymbol{\theta}, 0)) \\ &= \sum_{I\in\mathcal{F}} \tilde{f}(\overline{p}_I) \end{aligned}$$

Thus the goal of the GSC Protocol is to verify that $S_0 = a$ where a is the target value. Note that

$$S_m = \left(\sum_{I\in\mathcal{F}} w(I, \boldsymbol{\alpha}, m)\right) \tilde{f}(\boldsymbol{\theta})$$

Here, the verifier can evaluate the quantity in the summation by himself and the value $\tilde{f}(\boldsymbol{\theta})$ can be read from the table of values of $\tilde{f}$. Thus the verifier *knows* the values S_0 and S_m and thus has to verify that the values $S_1, S_2, \ldots, S_{m-1}$ give a correct *interpolation* between S_0 and S_m.

For $b \in \{0, 1\}$ and $1 \leq s \leq m$, let

$$\mathcal{F}_s^b := \{I \mid I \in \mathcal{F},\ I_s = b\}$$

Thus $\mathcal{F}_s^0$ is the subfamily of $\mathcal{F}$ consisting of all sets that do not contain $s \in [m]$ and $\mathcal{F}_s^1$ consists of those sets that do contain s.

For $\boldsymbol{\theta} \in \mathbb{F}^m, I \in \mathcal{F}, 1 \leq s \leq m$, let $P(I, \boldsymbol{\theta}, s, X) \in \mathbb{F}^m$ be a point whose i^{th} co-ordinate equals θ_i for $1 \leq i < s$, equals I_i for $s < i \leq m$ and the s^{th} co-ordinate is a formal variable X. Note that if $I_{s+1} = b$, then $P(I, \boldsymbol{\theta}, s+1, b) = p(I, \boldsymbol{\theta}, s)$ where $P(I, \boldsymbol{\theta}, s+1, b)$ denotes the point obtained from $P(I, \boldsymbol{\theta}, s+1, X)$ by substituting $X = b$. Also, when one substitutes $X = \theta_{s+1}$, we get $P(I, \boldsymbol{\theta}, s+1, \theta_{s+1}) = p(I, \boldsymbol{\theta}, s+1)$.

With so much notation, we are ready to see what $g_{b,r}$ are. Fix $b \in \{0,1\}$ and $0 \le r \le m-1$. Consider the following sum

$$g_{b,r}(\boldsymbol{\alpha}, \boldsymbol{\theta}, X) := \sum_{I \in \mathcal{F}^b_{r+1}} w(I, \boldsymbol{\alpha}, r)\tilde{f}(P(I, \boldsymbol{\theta}, r+1, X)) \tag{8}$$

Note that $g_{b,r}$ is a polynomial in $\boldsymbol{\alpha}, \boldsymbol{\theta}$ and X. The degree in $\boldsymbol{\alpha}$ is at most d and the degree in $\boldsymbol{\theta}, X$ is at most D. Also, $g_{b,r}$ depends only on $\alpha_1, \alpha_2, \ldots, \alpha_r, \theta_1, \theta_2, \ldots, \theta_r$ and X.

Here is the link between the partial sums S_r and the polynomials $g_{b,r}$.

Lemma 5.2

$$S_r = g_{0,r}(\boldsymbol{\alpha}, \boldsymbol{\theta}, 0) + g_{1,r}(\boldsymbol{\alpha}, \boldsymbol{\theta}, 1)$$

$$g_{0,r}(\boldsymbol{\alpha}, \boldsymbol{\theta}, \theta_{r+1}) + \alpha_{r+1} g_{1,r}(\boldsymbol{\alpha}, \boldsymbol{\theta}, \theta_{r+1}) = S_{r+1}$$

Proof:

$$\begin{aligned}
S_r &= \sum_{I \in \mathcal{F}} w(I, \boldsymbol{\alpha}, r)\tilde{f}(p(I, \boldsymbol{\theta}, r)) \\
&= \sum_{I \in \mathcal{F}^0_{r+1}} w(I, \boldsymbol{\alpha}, r)\tilde{f}(p(I, \boldsymbol{\theta}, r)) + \sum_{I \in \mathcal{F}^1_{r+1}} w(I, \boldsymbol{\alpha}, r)\tilde{f}(p(I, \boldsymbol{\theta}, r)) \\
&= \sum_{I \in \mathcal{F}^0_{r+1}} w(I, \boldsymbol{\alpha}, r)\tilde{f}(P(I, \boldsymbol{\theta}, r+1, 0)) + \sum_{I \in \mathcal{F}^1_{r+1}} w(I, \boldsymbol{\alpha}, r)\tilde{f}(P(I, \boldsymbol{\theta}, r+1, 1)) \\
&= g_{0,r}(\boldsymbol{\alpha}, \boldsymbol{\theta}, 0) + g_{1,r}(\boldsymbol{\alpha}, \boldsymbol{\theta}, 1)
\end{aligned}$$

On the other hand,

$$\begin{aligned}
& g_{0,r}(\boldsymbol{\alpha}, \boldsymbol{\theta}, \theta_{r+1}) + \alpha_{r+1} g_{1,r}(\boldsymbol{\alpha}, \boldsymbol{\theta}, \theta_{r+1}) \\
= & \textstyle\sum_{I \in \mathcal{F}^0_{r+1}} w(I, \boldsymbol{\alpha}, r)\tilde{f}(P(I, \boldsymbol{\theta}, r+1, \theta_{r+1})) + \sum_{I \in \mathcal{F}^1_{r+1}} (\alpha_{r+1} w(I, \boldsymbol{\alpha}, r))\, \tilde{f}(P(I, \boldsymbol{\theta}, r+1, \theta_{r+1})) \\
= & \textstyle\sum_{I \in \mathcal{F}^0_{r+1}} w(I, \boldsymbol{\alpha}, r+1)\tilde{f}(p(I, \boldsymbol{\theta}, r+1)) + \sum_{I \in \mathcal{F}^1_{r+1}} w(I, \boldsymbol{\alpha}, r+1)\tilde{f}(p(I, \boldsymbol{\theta}, r+1)) \\
= & \textstyle\sum_{I \in \mathcal{F}} w(I, \boldsymbol{\alpha}, r+1)\tilde{f}(p(I, \boldsymbol{\theta}, r+1)) \\
= & S_{r+1}
\end{aligned}$$

∎

Corollary 5.3

$$g_{0,r}(\boldsymbol{\alpha}, \boldsymbol{\theta}, \theta_{r+1}) + \alpha_{r+1} g_{1,r}(\boldsymbol{\alpha}, \boldsymbol{\theta}, \theta_{r+1}) = g_{0,r+1}(\boldsymbol{\alpha}, \boldsymbol{\theta}, 0) + g_{1,r+1}(\boldsymbol{\alpha}, \boldsymbol{\theta}, 1)$$

5.1 The GSC Protocol

The protocol works as follows. The protocol accepts if and only if all of the following checks are satisfied (we later combine these checks into a single homogeneous linear test).

- Pick $\boldsymbol{\alpha}, \boldsymbol{\theta} \in \mathbb{F}^m$ at random.
- (Stage 0 :) Check if

$$a = g_{0,0}(\boldsymbol{\alpha}, \boldsymbol{\theta}, 0) + g_{1,0}(\boldsymbol{\alpha}, \boldsymbol{\theta}, 1) \tag{9}$$

- (Stage $r+1$ for $r = 0, 1, \ldots, m-2$:) check if

$$g_{0,r}(\boldsymbol{\alpha}, \boldsymbol{\theta}, \theta_{r+1}) + \alpha_{r+1} g_{1,r}(\boldsymbol{\alpha}, \boldsymbol{\theta}, \theta_{r+1}) = g_{0,r+1}(\boldsymbol{\alpha}, \boldsymbol{\theta}, 0) + g_{1,r+1}(\boldsymbol{\alpha}, \boldsymbol{\theta}, 1) \tag{10}$$

- (Stage m :) Check if

$$g_{0,m-1}(\boldsymbol{\alpha}, \boldsymbol{\theta}, \theta_m) + \alpha_m g_{1,m-1}(\boldsymbol{\alpha}, \boldsymbol{\theta}, \theta_m) = \left(\sum_{I \in \mathcal{F}} w(I, \boldsymbol{\alpha}, m) \right) \tilde{f}(\boldsymbol{\theta}) \tag{11}$$

Note that the values on lines (9), (10), (11) are (supposed to be) S_0, S_{r+1}, S_m respectively.

5.2 Completeness

Lemma 5.4 *If the polynomial $\tilde{f}$ satisfies (7), then it is possible to define polynomials $g_{b,r}(\boldsymbol{\alpha}, \boldsymbol{\theta}, X)$ so that the protocol accepts with probability* 1.

Proof: Obvious. Just define $g_{b,r}$ as they are supposed to be as in Equation (8). ■

5.3 Soundness

Lemma 5.5 *If the polynomial $\tilde{f}$ does not satisfy (7), then no matter what polynomials $g_{b,r}$ one takes, the protocol accepts with negligible probability, i.e. with probability at most $O(\frac{D+d}{|\mathbb{F}|})$. It is assumed that the polynomials $g_{b,r}$ have degree at most $D+d$ and $g_{b,r}$ depends only on $\alpha_1, \ldots, \alpha_r$, $\theta_1, \ldots, \theta_r$, X.*

Proof: Let $\tilde{g}_{b,r}$ be polynomials obtained from $\tilde{f}$ using Rule (8). We will argue in the reverse order (i.e. $r = m-1, m-2, \ldots, 0$) that unless $g_{b,r} \equiv \tilde{g}_{b,r}$, the protocol succeeds with negligible probability. The only fact we use is that two polynomials agree with non-negligible probability if and only if they are identical as formal polynomials.

(Case $r = m-1$) : Look at Stage m, Equation (11). The RHS is just S_m which using Lemma 5.2 can be written in terms of $\tilde{g}_{0,m-1}$ and $\tilde{g}_{1,m-1}$. Thus we get

$$g_{0,m-1}(\boldsymbol{\alpha}, \boldsymbol{\theta}, \theta_m) + \alpha_m g_{1,m-1}(\boldsymbol{\alpha}, \boldsymbol{\theta}, \theta_m) = \tilde{g}_{0,m-1}(\boldsymbol{\alpha}, \boldsymbol{\theta}, \theta_m) + \alpha_m \tilde{g}_{1,m-1}(\boldsymbol{\alpha}, \boldsymbol{\theta}, \theta_m)$$

Note that $g_{0,m-1}, g_{1,m-1}, \tilde{g}_{0,m-1}, \tilde{g}_{1,m-1}$ do not depend on α_m, θ_m. Therefore, if this is a formal identity, then we must have $g_{0,m-1} = \tilde{g}_{0,m-1}$ and $g_{1,m-1} = \tilde{g}_{1,m-1}$.

(Case $0 \le r \le m-2$) : Assume that we have already proved that $g_{0,r+1} = \tilde{g}_{0,r+1}$ and $g_{1,r+1} = \tilde{g}_{1,r+1}$. Therefore, the RHS of Equation (10) is

$$\begin{aligned} g_{0,r+1}(\boldsymbol{\alpha}, \boldsymbol{\theta}, 0) + g_{1,r+1}(\boldsymbol{\alpha}, \boldsymbol{\theta}, 1) &= \tilde{g}_{0,r+1}(\boldsymbol{\alpha}, \boldsymbol{\theta}, 0) + \tilde{g}_{1,r+1}(\boldsymbol{\alpha}, \boldsymbol{\theta}, 1) \\ &= \tilde{g}_{0,r}(\boldsymbol{\alpha}, \boldsymbol{\theta}, \theta_{r+1}) + \alpha_{r+1} \tilde{g}_{1,r}(\boldsymbol{\alpha}, \boldsymbol{\theta}, \theta_{r+1}) \end{aligned}$$

where we used Corollary 5.3. Now we equate this with the LHS of Equation (10) and note that $g_{0,r}, g_{1,r}, \tilde{g}_{0,r}, \tilde{g}_{1,r}$ do not depend on $\alpha_{r+1}, \theta_{r+1}$. Hence we conclude that $g_{0,r} = \tilde{g}_{0,r}$ and $g_{1,r} = \tilde{g}_{1,r}$

Now we show that the test fails in Stage 0. This is because, $g_{0,0} = \tilde{g}_{0,0}, g_{1,0} = \tilde{g}_{1,0}$ and

$$\begin{aligned} a &\neq \sum_{I \in \mathcal{F}} \tilde{f}(\overline{p}_I) \\ &= \tilde{g}_{0,0}(\boldsymbol{\alpha}, \boldsymbol{\theta}, 0) + \tilde{g}_{1,0}(\boldsymbol{\alpha}, \boldsymbol{\theta}, 1) \\ &= g_{0,0}(\boldsymbol{\alpha}, \boldsymbol{\theta}, 0) + g_{1,0}(\boldsymbol{\alpha}, \boldsymbol{\theta}, 1) \end{aligned}$$

■

5.4 Combining All Tests into a Single Homogeneous Test

Instead of verifying Equations (9), (10) and (11) separately, the protocol can multiply these equations with random values in $\mathbb{F}$, add them up and verify that the single combined equation is satisfied. It is easily seen that the completeness and soundness properties are still satisfied.

6 Constructing Subspace Membership Certificate

Now we return to the task we started with. We want polynomials g and h that certify that the polynomial f in Section 3.1 satisfies Equation (3). The equation says that $f(\mathbf{q})$ equals a sum of values of $\psi(\mathbf{q}, \cdot)f(\cdot)$. We verify this by running the generalized sum-check protocol on $\psi(\mathbf{q}, \cdot)f(\cdot)$. The polynomial g will be obtained by *clubbing together* all the sum-check polynomials. The polynomial h will certify that g is splitable, a property that makes sure that the sum-check protocol works (as explained below).

We want to verify Equation (3) for a random $\mathbf{q}$. Let $\tilde{f}[\mathbf{q}, \mathbf{x}] = \psi(\mathbf{q}, \mathbf{x})f(\mathbf{x})$ and $a = f(\mathbf{q})$. Thus we can run the protocol

$$GSC(a = f(\mathbf{q}), \mathbb{F}, \tilde{f}, m, D = 3d, d, \mathcal{F}, \{g_{b,r} | b \in \{0,1\}, 0 \leq r \leq m-1\})$$

Here, $\tilde{f}$ is a polynomial in $\mathbf{q}, \mathbf{x}$, but the sum-check protocol is run on the $\mathbf{x}$-part only. Usually, the sum-check polynomials $g_{b,r}$ are polynomials in $\boldsymbol{\alpha}, \boldsymbol{\theta}, X$. But now, examining Rule (8), they are polynomials also in $\mathbf{q}$. Since $g_{b,r}$ is polynomial of degree $D + d$, we can write it in increasing powers of X as :

$$g_{b,r}(\mathbf{q}, \boldsymbol{\alpha}, \boldsymbol{\theta}, X) := \sum_{t=0}^{D+d} g_{b,r,t}(\mathbf{q}, \boldsymbol{\alpha}, \boldsymbol{\theta}) X^t$$

where $g_{b,r,t}$ are polynomials in $\mathbf{q}, \alpha_1, \ldots, \alpha_r, \theta_1, \ldots, \theta_r$.

The Generalized Sum-Check Protocol (Section 5.1) needs values $a = f(\mathbf{q})$ and $\tilde{f}(\mathbf{q}, \boldsymbol{\theta})$. The protocol reads values $f(\mathbf{q})$ and $f(\boldsymbol{\theta})$ from the table of values of f. The value $\tilde{f}(\mathbf{q}, \boldsymbol{\theta}) = \psi(\mathbf{q}, \boldsymbol{\theta}) f(\boldsymbol{\theta})$ is computed from the value $f(\boldsymbol{\theta})$ and using Equation (2). In addition, the protocol needs the value of some homogeneous linear form on the values :

$$g_{b,r,t}(\mathbf{q}, \boldsymbol{\alpha}, \boldsymbol{\theta}) \;:\; b \in \{0,1\},\; 0 \leq r \leq m-1,\; 0 \leq t \leq D+d$$

However, we can afford only a constant number of queries. So reading all these values separately is ruled out. We do the following trick : we just *read off* the value of the linear form as a single query. This can be done if we have access to the Hadamard Code over the polynomials $g_{b,r,t}$. This is exactly what the *certificate polynomial* g will do. As a first attempt (which does not work), we expect to have the following polynomial g^* as a certificate that enables us to read off the linear form. Let $R_{b,r,t}$ be formal variables and $\mathbf{R}$ denote the vector of all these variables. The polynomial g^* is intended to be :

$$g^*(\mathbf{q}, \boldsymbol{\alpha}, \boldsymbol{\theta}, \mathbf{R}) := \sum_{b,r,t} g_{b,r,t}(\mathbf{q}, \boldsymbol{\alpha}, \boldsymbol{\theta}) R_{b,r,t} \tag{12}$$

We are not quite done yet. How do we ensure that the supposed polynomial g^* given by an adversary is of the form (12) ? First of all, note that g^* is intended to be a linear function when treated as a function of $\mathbf{R}$. This can be checked by substituting random values of $\mathbf{q}, \boldsymbol{\alpha}, \boldsymbol{\theta}$ and then using a linearity test. Assuming now that g^* is indeed a linear function of $\mathbf{R}$, let $g^*_{b,r,t}$ denote the coefficient of $R_{b,r,t}$.

The problem is the following : how do we ensure that $g^*_{b,r,t}$ depends only on $\mathbf{q}, \alpha_1, \ldots, \alpha_r, \theta_1, \ldots, \theta_r$ and is independent of $\alpha_{r+1}, \ldots, \alpha_m$, $\theta_{r+1}, \ldots, \theta_m$? This property is necessary for the Generalized Sum-Check Protocol to work.

In order to fix this problem, we need to redefine the polynomial g^*. We give a *fresh copy* of variables to every polynomial $g_{b,r,t}$. To be precise, for any (b, r, t), let $\mathbf{q}^{(b,r,t)}$ denote a vector of m variables which is supposed to be a *copy* of $\mathbf{q}$. Also, let $\boldsymbol{\alpha}^{(b,r,t)}$ denote a copy of $\alpha_1, \ldots, \alpha_r$ and $\boldsymbol{\theta}^{(b,r,t)}$ denote a copy of $\theta_1, \ldots, \theta_r$.

Let

$$\mathbf{q}' = \cup_{b,r,t}\mathbf{q}^{(b,r,t)}, \quad \boldsymbol{\alpha}' = \cup_{b,r,t}\boldsymbol{\alpha}^{(b,r,t)}, \quad \boldsymbol{\theta}' = \cup_{b,r,t}\boldsymbol{\theta}^{(b,r,t)}, \quad \mathbf{R} = \cup_{b,r,t}\{R_{b,r,t}\} \tag{13}$$

Define

$$g(\mathbf{q}', \boldsymbol{\alpha}', \boldsymbol{\theta}', \mathbf{R}) := \sum_{b,r,t} g_{b,r,t}(\mathbf{q}^{(b,r,t)}, \boldsymbol{\alpha}^{(b,r,t)}, \boldsymbol{\theta}^{(b,r,t)}) R_{b,r,t} \tag{14}$$

Note that g is splitable. The sets of variables on which $g_{b,r,t}$'s depend are disjoint for different (b, r, t)'s. Now we can attach a splitability certificate h to make sure that an adversary is forced to give a g that is splitable !

6.1 Proving Theorem 3.3

Now we prove Theorem 3.3. First we prove a version of the theorem where instead of a single polynomial f in the HomAlgCSP instance, we have three polynomials f, g, h and the constraints are homogeneous constraints on $O(1)$ values of these polynomials.

Given tables of values of polynomials $f(\mathbf{q})$ and $g(\mathbf{q}', \boldsymbol{\alpha}', \boldsymbol{\theta}', \mathbf{R})$ and a splitability certificate h, we run the following verification procedure :

Checking that g is Linear in $\mathbf{R}$

Pick random values for $\mathbf{q}', \boldsymbol{\alpha}', \boldsymbol{\theta}'$ and two sets of random values for $\mathbf{R}$ and $\mathbf{R}'$. Pick two random scalar/field values ζ, ζ' and check if

$$g(\mathbf{q}', \boldsymbol{\alpha}', \boldsymbol{\theta}', \zeta\mathbf{R} + \zeta'\mathbf{R}') = \zeta \cdot g(\mathbf{q}', \boldsymbol{\alpha}', \boldsymbol{\theta}', \mathbf{R}) + \zeta' \cdot g(\mathbf{q}', \boldsymbol{\alpha}', \boldsymbol{\theta}', \mathbf{R}')$$

Checking that h is a Splitability Certificate

This can be done as in Section 4.3 by reading one value of g and 4 values of h.

Running the Generalized Sum-Check Protocol

Pick random values for $\mathbf{q}, \boldsymbol{\alpha}, \boldsymbol{\theta}$. The values $f(\mathbf{q}), f(\boldsymbol{\theta})$ needed by the protocol are read from the table of values for f. The protocol also needs a value of a linear form of type

$$\sum_{b,r,t} \zeta_{b,r,t}\ g_{b,r,t}(\mathbf{q}, \boldsymbol{\alpha}, \boldsymbol{\theta}) \qquad \zeta_{b,r,t} \in \mathbb{F} \tag{15}$$

For all (b, r, t), we set $\mathbf{q}^{(b,r,t)} = \mathbf{q}$, $\boldsymbol{\alpha}^{(b,r,t)} = (\alpha_1, \alpha_2, \ldots, \alpha_r)$, $\boldsymbol{\theta}^{(b,r,t)} = (\theta_1, \theta_2, \ldots, \theta_r)$ (this makes sense because $\mathbf{q}^{(b,r,t)}, \boldsymbol{\alpha}^{(b,r,t)}, \boldsymbol{\theta}^{(b,r,t)}$ are supposed to be copies of the same $\mathbf{q}, \boldsymbol{\alpha}, \boldsymbol{\theta}$ respectively). Set $\mathbf{R}_{b,r,t} = \zeta_{b,rt}$. Define $\mathbf{q}', \boldsymbol{\alpha}', \boldsymbol{\theta}', \mathbf{R}$ as in (13).

Finally, take the value of the linear form in (15) needed by the protocol to be the value $g(\mathbf{q}', \boldsymbol{\alpha}', \boldsymbol{\theta}', \mathbf{R})$.

Checking that f is a Solution to MDC Instance

Pick a random point from set $\{\overline{p}_J \mid J \in \mathcal{F}_0\}$ and check whether $f(\overline{p}_J) = 0$.

Number of Queries

Note that the verification procedure reads 3 values of f (namely, $f(\overline{p}_J), f(\mathbf{q}), f(\boldsymbol{\theta})$). It reads 5 values of g (three for checking linearity in $\mathbf{R}$, one for checking splitability, one for reading linear form needed in sum-check). It reads 4 values of h (as in splitability check). Thus we have total of 12 queries.

Relating Optimum of MDC with Maximum Acceptance Probability

It is clear that if the MDC instance $\mathbf{A}$ has a solution with value $OPT(\mathbf{A})$ then we can take this solution, encode it as a non-zero polynomial f, and construct g, h as they should be. The verification procedure accepts whenever $f(\overline{p}_J) = 0$ which happens with probability $1 - OPT(\mathbf{A})$.

On the other hand, note that the verification process accepts with negligible property unless $f \in span\{f_I | I \in \mathcal{F}\}$. If $f \equiv 0$, then the corresponding polynomials g, h must also be identically zero (otherwise probability of acceptance is negligible). Now if f is a nonzero polynomial in the span, we can have $f(\overline{p}_J) = 0$ only for $1 - OPT(\mathbf{A})$ fraction of $J \in \mathcal{F}_0$. Hence the verification procedure can accept with probability at most $1 - OPT(\mathbf{A})$.

6.2 Clubbing Together f, g, h

Now we show how to combine f, g, h into a single polynomial ϕ and thus prove Theorem 3.3. Assume that $f \in \mathbb{F}[\mathbf{x}]$, $g \in \mathbb{F}[\mathbf{y}]$, $h \in \mathbb{F}[\mathbf{z}]$ where $\mathbf{x}, \mathbf{y}, \mathbf{z}$ are disjoint sets of variables. Let X, Y, Z be formal variables and let ϕ (as intended to) be :

$$\phi(\mathbf{x}, \mathbf{y}, \mathbf{z}, X, Y, Z) := Xf(\mathbf{x}) + Yg(\mathbf{y}) + Zh(\mathbf{z})$$

Again the trouble is that the adversary might give an arbitrary polynomial ϕ and not necessarily of this form. We give the following verification procedure.

Checking Linearity in X, Y, Z

Pick random values for $\mathbf{x}, \mathbf{y}, \mathbf{z}$ and random values X, X', Y, Y', Z, Z'. Pick random values ζ, ζ'. Check if

$$\phi(\mathbf{x}, \mathbf{y}, \mathbf{z}, \zeta X + \zeta' X', \zeta Y + \zeta' Y', \zeta Z + \zeta' Z') = \zeta \cdot \phi(\mathbf{x}, \mathbf{y}, \mathbf{z}, X, Y, Z) + \zeta' \cdot \phi(\mathbf{x}, \mathbf{y}, \mathbf{z}, X', Y', Z')$$

Checking Splitability

Thus we can assume that ϕ is linear and hence of the form $X\tilde{f} + Y\tilde{g} + Z\tilde{h}$. We need to ensure that $\tilde{f}$ depends only on $\mathbf{x}$ and not on $\mathbf{y}, \mathbf{z}$. Also that $\tilde{g}$ depends only on $\mathbf{y}$ and $\tilde{h}$ depends only on $\mathbf{z}$. This can be done by *brute-force* as follows. Note that setting $X = 1, Y = 0, Z = 0$ gives us *access* to $\tilde{f}$ and similarly for $\tilde{g}, \tilde{h}$.

- Pick random values $\mathbf{x}, \mathbf{y}, \mathbf{z}$. Pick another set of random values $\mathbf{x}', \mathbf{y}', \mathbf{z}'$.

- Check that

$$\phi(\mathbf{x},\mathbf{y},\mathbf{z},1,0,0) = \phi(\mathbf{x},\mathbf{y}',\mathbf{z}',1,0,0)$$
$$\phi(\mathbf{x},\mathbf{y},\mathbf{z},0,1,0) = \phi(\mathbf{x}',\mathbf{y},\mathbf{z}',0,1,0)$$
$$\phi(\mathbf{x},\mathbf{y},\mathbf{z},0,0,1) = \phi(\mathbf{x}',\mathbf{y}',\mathbf{z},0,0,1)$$

Simulating Previous Verification Procedure

Now we can assume that ϕ is of the form $Xf(\mathbf{x})+Yg(\mathbf{y})+Zh(\mathbf{z})$. We run the verification procedure described in Section 6.1. Values for $f[\mathbf{x}], g[\mathbf{y}], h[\mathbf{z}]$ can be accessed by reading $\phi(\mathbf{x},\mathbf{0},\mathbf{0},1,0,0)$, $\phi(\mathbf{0},\mathbf{y},\mathbf{0},0,1,0)$ and $\phi(\mathbf{0},\mathbf{0},\mathbf{z},0,0,1)$ respectively.

Note that the new protocol reads $3+6+12=21$ queries from table of values of ϕ. The maximum probability of acceptance (for any non-zero polynomial ϕ) equals $1-OPT(\mathbf{A})$ where $\mathbf{A}$ is the MDC instance one starts with. The constraints are all homogeneous linear. This proves Theorem 3.3.

7 The Low Degree Test and Outer Verifier

In this section, we describe our Outer Verifier. Later we modify the Outer Verifier and build a Hadamard Code based Inner Verifier. This gives a Quasi-random PCP.

The starting point for the Outer Verifier is the gap-instance of HomAlgCSP obtained by combining Theorem 1.5 and Theorem 3.3. Let $\mathcal{A}(f,k,d^*,m,\mathbb{F},\mathcal{C})$ denote this HomAlgCSP instance. We have

$$\text{Either } \ OPT(\mathcal{A}) \geq 1-1/2^K \qquad \text{Or } \ OPT(\mathcal{A}) \leq 1/2^{2^K}$$

Remark 7.1 *It is helpful to keep in mind the quantitative parameters. If the size of the SAT instance is n, then the size of the MDC instance is $N \leq n^{CK}$, and $N \leq |\mathbb{F}| \leq N^2$, $m = N^\epsilon$, $d^* = 100/\epsilon$. In particular, the degree d^* is constant. The size of the HomAlgCSP instance is $|\mathbb{F}|^{O(m)} \leq 2^{O(N^\epsilon)}$.*

The Outer Verifier is given the polynomial f as a table of values. He wants to pick a random constraint $C(\{\overline{p}_i\}_{i=1}^k) \in \mathcal{C}$ and check that it is satisfied. However, an adversary could cheat by giving a table which is *not* a low degree polynomial. The *Low Degree Test* allows the verifier to guard against such cheating. LDT was first proposed by Rubinfeld and Sudan [34] and it has been a crucial ingredient in the proof of the PCP Theorem ([6], [5]).

7.1 Outer Verifier

Definition 7.2 *A curve L in $\mathbb{F}^m$ is a function $L : \mathbb{F} \mapsto \mathbb{F}^m$. In other words :*

$$L(t) = \overline{a}(t) = (a_1(t), a_2(t), \ldots, a_m(t))$$

It is a degree d curve if each of the co-ordinate functions $a_i(t)$ is a degree d (univariate) polynomial.

We fix a constraint $C(\{\overline{p_i}\}_{i=1}^k) \in \mathcal{C}$ for this section and the next.

Let $t_1, t_2, \ldots, t_{k+3}$ be fixed distinct field elements in $\mathbb{F}$. For $\overline{a}, \overline{b}, \overline{c} \in \mathbb{F}^m$, let $L = L_{\overline{a},\overline{b},\overline{c}}$ be the unique degree $k+2$ curve that passes through points $\{\overline{p}_i\}_{i=1}^k, \overline{a}, \overline{b}, \overline{c}$. More precisely,

$$L(t_i) = \overline{p}_i, \ 1 \leq i \leq k, \quad L(t_{k+1}) = \overline{a}, \ L(t_{k+2}) = \overline{b}, \ L(t_{k+3}) = \overline{c} \tag{16}$$

If f is a degree d^* polynomial, then its restriction to the curve $L = L_{\overline{a},\overline{b},\overline{c}}$ is a degree $d-1 := (k+2)d^*$ univariate polynomial. Call this polynomial $f|_L$. In particular, $f|_L$ can be *reconstructed* from *any* d values of f on the curve. The Outer Verifier does precisely this. He picks $d+1$ random points from the curve, say $\overline{v}_1, \overline{v}_2, \ldots, \overline{v}_d, \overline{v}$. He reconstructs $f|_L$ using values of f at first d points, and verifies that the value of $f|_L$ at point $\overline{v}$ is same as the value $f(\overline{v})$. In addition, *knowing* $f|_L$, the verifier *calculates* the values of f at points $\{\overline{p}_i\}_{i=1}^k$ and verifies that these values indeed satisfy the constraint C.

However, we need to combine the above task with the line-point Low Degree Test. We will use the strong analysis of LDT by Arora and Sudan [6]. The Outer Verifier becomes somewhat complicated. Note that restriction of f to a line is a degree d^* univariate polynomial. We will be generous and allow the degree to be $d-2$.

The verifier picks a random pair of a line ℓ and a curve L intersecting at a point $\overline{v}$. The univariate polynomials $f|_L$ and $f|_\ell$ are reconstructed by reading values of f at d random points on L and $d-1$ random points on ℓ respectively. Then the verifier checks that $f|_L$ and $f|_\ell$ both have the value $f(\overline{v})$ at point $\overline{v}$. In addition, the verifier checks that values of $f|_L$ at points $\{\overline{p}_i\}_{i=1}^k$ satisfy the constraint C.

To prove the correctness of the Outer Verifier, we need to be careful about how the intersecting line-curve pair is chosen. Formally, the action of the verifier is as follows (see Fig. 2) :

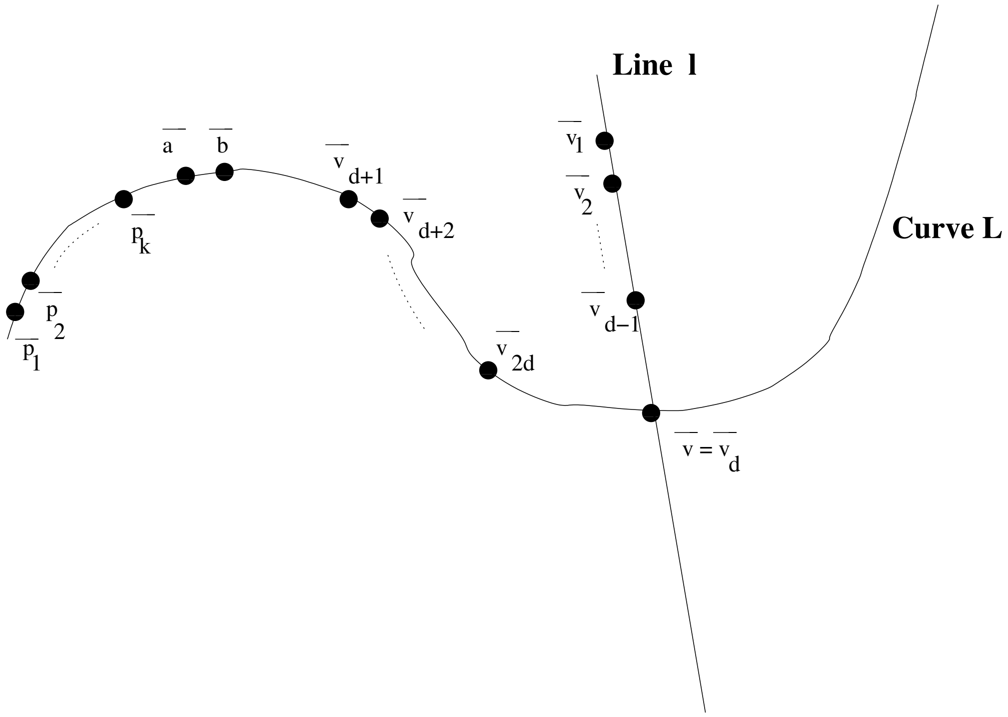

Figure 2: Outer Verifier

Outer Verifier

1. Pick a random line ℓ and pick random points $\overline{v}_1, \ldots, \overline{v}_{d-1}, \overline{v}_d$ on the line (assume that these points are all distinct; this happens w.h.p.). Let $\overline{v} = \overline{v_d}$.

2. Pick $t \in \mathbb{F} \setminus \{t_1, \ldots, t_{k+3}\}$ at random, points $\overline{a}, \overline{b}$ at random and let L be the unique degree $k+2$ curve $L = L_{\overline{a},\overline{b},\overline{c}}$ such that

$$L(t_i) = \overline{p}_i, 1 \leq i \leq k, \quad L(t_{k+1}) = \overline{a}, \quad L(t_{k+2}) = \overline{b}, \quad L(t) = \overline{v}$$

   (and $\overline{c} = L(t_{k+3})$ gets defined automatically).

3. Pick random points $\overline{v}_{d+1}, \ldots, \overline{v}_{2d}$ on curve L (we assume that these points are distinct; this happens w.h.p.).

4. Read the values $f(\overline{v}_i)$ for $1 \leq i \leq 2d$.

5. Let $f|_\ell$ be the unique degree $d-2$ polynomial *interpolated* using the values $\{f(\overline{v}_i)\}_{i=1}^{d-1}$.

6. Let $f|_L$ be the unique degree $d-1$ polynomial *interpolated* using the values $\{f(\overline{v}_i)\}_{i=d+1}^{2d}$.

7. Check if
$$f|_L(\overline{v}) = f(\overline{v}) = f|_\ell(\overline{v})$$

8. Check if the values of $f|_L$ at points $\{\overline{p}_i\}_{i=1}^{k}$ satisfy the constraint C.

9. Check that not all the values read are zero, i.e. Not-All-Zero($f(\overline{v}_i), 1 \leq i \leq 2d$).

Remark 7.3 *Note that the polynomials $f|_\ell$ and $f|_L$ depend on the specific choice of points $\{\overline{v}_i\}_{i=1}^{d-1}$ and $\{\overline{v}_i\}_{i=d+1}^{2d}$ respectively. We hide this fact for notational convenience. Also, a main step in the proof of Theorem 7.5 is to get rid of this dependence.*

Remark 7.4 *The verifier's action is straightforward except the last step where he checks that not all values read are zero. This is needed to make sure that the adversary does not cheat by giving a function f that is identically zero.*

Given a table $f : \mathbb{F}^m \mapsto \mathbb{F}$ and a m-variate polynomial P, the *agreement* between f and P is defined to be the fraction of points $\overline{u} \in \mathbb{F}^m$ such that $f(\overline{u}) = P(\overline{u})$. Now we state the main theorem in this section.

Theorem 7.5 *There are constants c_2, c_3 such that the following holds. Let $P_1, P_2, \ldots, P_t$ be all degree d polynomials that have agreement at least $(\delta/2)^{c_2}/c_3$ with f. Then, if the Outer Verifier accepts with probability δ, then for some $1 \leq j \leq t$, the polynomial P_j is a non-zero polynomial and its values at points $\{\overline{p}_i\}_{i=1}^{k}$ satisfy the constraint C. Also, $t \leq 2c_3/(\delta/2)^{c_2}$.*

The next section is devoted to proving this theorem.

8 Proof of Theorem 7.5

Theorem 7.5 is obtained by applying the strong analysis of the Low Degree Test as in Arora and Sudan's paper [7]. Theorem 8.1 states the result of Arora and Sudan. However we need to go through a rather unpleasant sequence of arguments before we can apply their theorem.

We first observe that there are two equivalent ways in which the Outer Verifier can make his probabilistic choices. The first way is as in Steps (1)-(3). An alternate way is :

- Pick $\overline{a}, \overline{b}, \overline{c}$ at random and let $L = L_{\overline{a},\overline{b},\overline{c}}$ be the curve as in Equation (16). Pick random points $\overline{v}_{d+1}, \ldots, \overline{v}_{2d}$ on L.
- Pick a random point $\overline{v} = L(t), t \in \mathbb{F} \setminus \{t_1, \ldots, t_{k+3}\}$ on the curve. Let $\overline{v}_d = \overline{v}$.
- Pick a random line ℓ through $\overline{v}$ and pick random points $\overline{v}_1, \ldots, \overline{v}_{d-1}$ on line ℓ.

Let NZ be the event that not all values read by the verifier are zero. Let SAT^C be the event that the values of the polynomial $f|_L$ at points $\{\overline{p}_i\}_{i=1}^k$ satisfy the constraint C. The probability of acceptance of the Outer Verifier is

$$\Pr\left[f|_L(\overline{v}) = f(\overline{v}) = f|_\ell(\overline{v}), \quad NZ, \quad SAT^C\right] \tag{17}$$

First we construct a table f_{lines} that gives a single univariate polynomial of degree $d-2$ (call it $f_{lines}(\ell)$) for every line ℓ. Then we show that the probability of acceptance of the following verifier (call it $\mathcal{V}$) is at least equal to the probability of acceptance of the Outer Verifier : the verifier $\mathcal{V}$, instead of reconstructing $f|_\ell$ from the values $\{f(\overline{v}_i)\}_{i=1}^{d-1}$, directly uses the polynomial $f_{lines}(\ell)$. Also, instead of checking for the event NZ, it checks that not all of these are zero : the polynomial $f_{lines}(\ell)$ and the value $f(\overline{v})$, $\{f(\overline{v}_i)\}_{i=d+1}^{2d}$. Let NZ' denote this event.

It is easy to construct f_{lines}. Think of verifier's probabilistic choices in the order : $\ell, \{\overline{v}_i\}_{i=1}^{d-1}$, $\overline{v}, L, \{\overline{v}_i\}_{i=d+1}^{2d}$. For any line ℓ, fix a choice of $\{\overline{v}_i\}_{i=1}^{d-1}$ so as to maximize the probability in (17). Define $f_{lines}(\ell)$ to be the polynomial obtained by interpolating from this *optimal* choice of $\{\overline{v}_i\}_{i=1}^{d-1}$. Thus probability of acceptance of $\mathcal{V}$ is

$$\Pr\left[f|_L(\overline{v}) = f(\overline{v}) = f_{lines}(\ell)(\overline{v}), \quad NZ', \quad SAT^C\right] \tag{18}$$

Now we do a similar trick again. We construct a table f_{curves} that gives a single univariate polynomial of degree $d-1$ (call it $f_{curves}(L)$) for every curve L. Then we show that the probability of acceptance of the following verifier (call it $\tilde{\mathcal{V}}$) is at least equal to the probability of acceptance of $\mathcal{V}$: this verifier $\tilde{\mathcal{V}}$, instead of reconstructing $f|_L$ from the values $\{f(\overline{v}_i)\}_{i=d+1}^{2d}$, directly uses the polynomial $f_{curves}(L)$. Also, instead of checking for the event NZ', it checks that not both the polynomials $f_{lines}(\ell)$ and $f_{curves}(L)$ are zero. Let NZ'' denote this event.

It is again easy to construct f_{curves}. Think of verifier's probabilistic choices in the alternative (but equivalent) order : $L, \{\overline{v}_i\}_{i=d+1}^{2d}, \overline{v}, \ell, \{\overline{v}_i\}_{i=1}^{d-1}$. For any curve L, fix a choice of $\{\overline{v}_i\}_{i=d+1}^{2d}$ so as to maximize the probability in (18). Define $f_{curves}(L)$ to be the polynomial obtained by interpolating from this *optimal* choice of $\{\overline{v}_i\}_{i=d+1}^{2d}$. Thus probability of acceptance of $\tilde{\mathcal{V}}$ is

$$\Pr\left[f_{curves}(L)(\overline{v}) = f(\overline{v}) = f_{lines}(\ell)(\overline{v}), \quad NZ'', \quad SAT^C\right]$$

If the Outer Verifier accepts with probability δ, then we have

$$\Pr\left[f_{curves}(L)(\overline{v}) = f(\overline{v}) = f_{lines}(\ell)(\overline{v}), \quad NZ'', \quad SAT^C\right] \geq \delta \tag{19}$$

Let $P_1, P_2, \ldots, P_t$ be all degree d polynomials that agree with f at $(\delta/2)^{c_2}/c_3$ fraction of points. Let $GOOD$ be the event that $f(\overline{v}) \in \{P_1(\overline{v}), \ldots, P_t(\overline{v})\}$. Using Theorem 8.1 (see below), we see that

$$\Pr_{\overline{v},\ell}\left[\neg GOOD, \quad f_{lines}(\ell)(\overline{v}) = f(\overline{v})\ \right] \leq \delta/2 \tag{20}$$

From (19) and (20), we get

$$\Pr\left[f_{curves}(L)(\overline{v}) = f(\overline{v}) = f_{lines}(\ell)(\overline{v}), \ GOOD, \quad NZ'', \quad SAT^C\right] \geq \delta/2 \tag{21}$$

Now, let $GOOD_{curves}$ be the event that

$$f_{curves}(L) \in \{P_1(L), \ldots, P_t(L)\}$$

and $GOOD_{lines}$ be the event that

$$f_{lines}(\ell) \in \{P_1(\ell), \ldots, P_t(\ell)\}$$

Here $P_j(L), P_j(\ell)$ denote the univariate polynomials that are restrictions of the polynomial P_j to L and ℓ respectively.

We observe that unless the events $GOOD_{curves}$ and $GOOD_{lines}$ happen, the probability of acceptance in (21) is negligible. This is because $\overline{v}$ is a random point on the line ℓ. If it happens that $f_{lines}(\ell)(\overline{v}) = f(\overline{v}) \in \{P_1(\overline{v}), \ldots, P_t(\overline{v})\}$ with non-negligible probability, then it must be the case that $f_{lines}(\ell)$ is identically equal to one of $P_1(\ell), \ldots, P_t(\ell)$. A similar argument holds for curves. Hence, we conclude

$$\Pr\left[f_{curves}(L)(\overline{v}) = f_{lines}(\ell)(\overline{v}),\ GOOD_{curves},\ GOOD_{lines},\ \ NZ'',\ \ SAT^C\right] \ \geq \delta/4$$

Now for any $i \neq j$, the probability $\Pr_{\overline{v}}[P_i(\overline{v}) = P_j(\overline{v})]$ is negligible. Hence if both events $GOOD_{curves}$ and $GOOD_{lines}$ happen, they must happen w.r.t. the same polynomial P_{j_0}. Therefore, there exists $1 \leq j_0 \leq t$ such that

$$P_{j_0}(L) = P_{j_0}(\ell), \quad NZ'', \quad SAT^C$$

Thus P_{j_0} is a non-zero polynomial and its values at points $\{\overline{p}_i\}_{i=1}^k$ satisfy the constraint. This proves Theorem 7.5.

We used the following theorem that appears as Theorem 3 and Proposition 2 in [7]. In our application, d is constant and $|\mathbb{F}|$ is a large polynomial in m (recall that $N \leq |\mathbb{F}|$ and $m = N^\epsilon$ where N is the size of the MDC instance used by our reduction).

Theorem 8.1 *There are constants c_0, c_1, c_2, c_3 such that the following holds : Let δ, d, m be parameters and $\mathbb{F}$ be a field satisfying $|\mathbb{F}| \geq c_0(dm/\delta)^{c_1}$. Then the following holds : Fix $f : \mathbb{F}^m \mapsto \mathbb{F}$ and f_{lines}. Let $\{P_1, P_2, \ldots, P_t\}$ be all polynomials that have agreement at least δ^{c_2}/c_3 with f. Then*

$$\Pr_{\overline{v},\ell}\left[f(\overline{v}) \notin \{P_1(\overline{v}), \ldots, P_t(\overline{v})\} \ \ \textit{and} \ \ f_{lines}(\ell)(\overline{v}) = f(\overline{v})\ \right] \ \leq \ \delta$$

Also, it follows from coding theoretic bounds that $t \leq 2c_3/\delta^{c_2}$.

9 Modified Outer Verifier

We use a modification of the Outer Verifier. This modification is necessary for the analysis of the inner verifier to go through.

Remark 9.1 *Note that the Outer Verifier reads $2d$ values. The verifier's checks are all homogeneous linear. This is because all operations including interpolation, evaluating a polynomial at a specific value, and the constraint C are homogeneous linear.*

We modify the Outer Verifier as follows (see Fig. 3) :

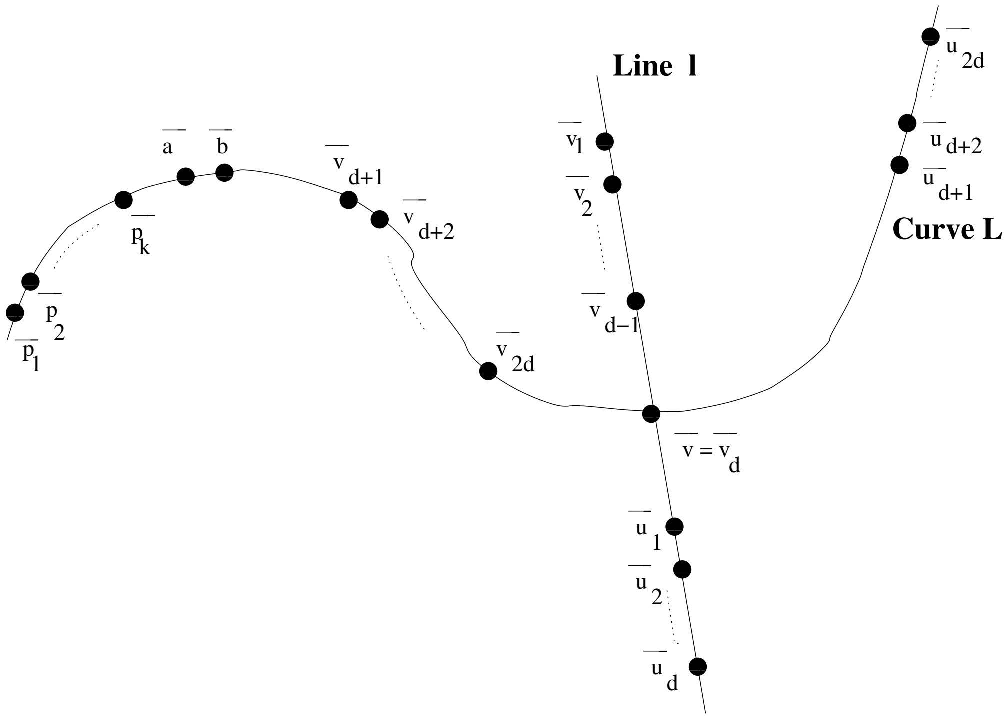

Figure 3: Modified Outer Verifier

Modified Outer Verifier

1. Follow Steps (1)-(9) as in the Outer Verifier.
2. Pick additional random points $\overline{u}_1, \ldots, \overline{u}_d$ on the line ℓ and random points $\overline{u}_{d+1}, \ldots, \overline{u}_{2d}$ on the curve L.
3. Check that degree $d-1$ univariate polynomial interpolated from values $\{f(\overline{v}_i)\}_{i=1}^{d}$ equals the polynomial interpolated from values $\{f(\overline{u}_i)\}_{i=1}^{d}$ (this actually happens to be a degree $d-2$ polynomial because of earlier steps).
4. Check that degree $d-1$ univariate polynomial interpolated from values $\{f(\overline{v}_i)\}_{i=d+1}^{2d}$ equals the polynomial interpolated from values $\{f(\overline{u}_i)\}_{i=d+1}^{2d}$.

Remark 9.2 *Note that the Modified Outer Verifier reads $4d$ values. The verifier's checks are all homogeneous linear.*

Remark 9.3 *The purpose of this modification is exactly as explained in Sections 1.5, 1.6 and 2.3. This modification adds redundancy to the Outer Verifier. It ensures that we get a squared Fourier coefficient when we do the Fourier analysis of the Inner Verifier. This is exactly the analogue of Håstad's 3-query PCP that reads 2 queries from the larger table so that the Fourier coefficient for the larger table appears in squared form.*

10 Inner Verifier

Notation : We will let $\mathbb{F} = 2^l$ so that the field elements are represented by bit strings of length l. Note that addition over $\mathbb{F}$ and multiplication by a constant $\alpha \in \mathbb{F}$ are homogeneous linear operations on these l-bit strings. To keep the description of the Inner Verifier conceptually simpler, we abstract out the Modified Outer Verifier as follows (change of notation : redenote $2d$ by d).

Abstract View of Modified Outer Verifier

1. Pick an intersecting line-curve pair. Pick points $\{\overline{v}_i\}_{i=1}^d$ ($d/2$ of them from the line and $d/2$ from the curve, counting the intersection point as a point on the line. This is not of much relevance at Inner Verifier's level).

2. Pick another set of points $\{\overline{u}_i\}_{i=1}^d$ (again, $d/2$ from the line and $d/2$ from the curve).

3. Let $T_{ld\times ld}$ be an invertible matrix over $\mathbb{F}_2$ chosen appropriately and H be a subspace of $\mathbb{F}_2^{ld}$ chosen appropriately (see the remark below on how T, H are chosen).

4. Read the values $f(\overline{v}_1), \ldots, f(\overline{v}_d)$ and values $f(\overline{u}_1), \ldots, f(\overline{u}_d)$. Let ($\circ$ denotes concatenation of strings)

$$\begin{aligned} x &= f(\overline{v}_1) \circ f(\overline{v}_2) \circ \ldots \circ f(\overline{v}_d) \\ y &= f(\overline{u}_1) \circ f(\overline{u}_2) \circ \ldots \circ f(\overline{u}_d) \end{aligned} \tag{22}$$

5. Accept if and only if

$$x \neq 0, \quad x = Ty \qquad \text{and} \qquad h \cdot x = 0 \ \forall\ h \in H \quad (i.e.\ x \perp H) \tag{23}$$

Remark 10.1 *1. The condition $x = Ty$ abstracts out Modified Outer Verifier's check that the two polynomials interpolated from values $\{f(\overline{v}_i)\}_{i=1}^{d/2}$ and $\{f(\overline{v}_i)\}_{i=d/2+1}^{d}$ are equal to the polynomials interpolated from values $\{f(\overline{u}_i)\}_{i=1}^{d/2}$ and $\{f(\overline{u}_i)\}_{i=d/2+1}^{d}$ respectively. This can be checked by multiplying one set of values by an appropriate invertible matrix and equating the result with the second set of values.*

2. The condition $h \cdot x = 0\ \forall\ h \in H$ abstracts out the check that the set of values $\{f(\overline{v}_i)\}_{i=1}^d$ satisfy all the homogeneous constraints of the (Modified) Outer Verifier.

3. The condition $x \neq 0$ says that Not-All-Zero($f(\overline{v}_i), 1 \leq i \leq d$), as in the (Modified) Outer Verifier.

Now we are ready to describe the Inner Verifier. The Inner Verifier expects, for every point $\overline{v} \in \mathbb{F}^m$, the Hadamard Code of the string $f(\overline{v}) \in \{0,1\}^l$ (see Appendix A.3 for an overview of Hadamard Codes). The action of the Inner Verifier is as follows (remember that the following action is carried out after randomly picking a constraint $C(\{\overline{p}_i\}_{i=1}^k)$ of the HomAlgCSP instance) :

Inner Verifier

1. Pick an intersecting line-curve pair, points $\{\overline{v}_i\}_{i=1}^d$ and points $\{\overline{u}_i\}_{i=1}^d$ exactly as in the Modified Outer Verifier.

2. Let $T_{ld \times ld}$ be an invertible matrix over $\mathbb{F}_2$ chosen appropriately and H be a subspace of $\mathbb{F}_2^{ld}$ chosen appropriately.

3. Let $A_1, A_2, \ldots, A_d$ be the (supposed) Hadamard Codes of $f(\overline{v}_1), \ldots, f(\overline{v}_d)$ respectively. Let $B_1, \ldots, B_d$ be the (supposed) Hadamard Codes of $f(\overline{u}_1), \ldots, f(\overline{u}_d)$ respectively.

4. Pick a random string $z \in \mathbb{F}_2^{ld}$ and a random $h \in H$. Write

$$z = z_1 \circ z_2 \circ \ldots \circ z_d$$

$$h = h_1 \circ h_2 \circ \ldots \circ h_d$$

$$zT = w_1 \circ w_2 \circ \ldots \circ w_d$$

5. Accept if and only if

$$\oplus_{i=1}^{d} A_i(z_i \oplus h_i) = \oplus_{j=1}^{d} B_j(w_j)$$

10.1 Sanity Check

Let us see why the test makes sense. Suppose each A_i is indeed Hadamard Code of $f(\overline{v}_i)$ and each B_j is indeed Hadamard Code of $f(\overline{u}_j)$. Therefore

$$A_i(z_i \oplus h_i) = (z_i \oplus h_i) \cdot f(\overline{v}_i) \quad \text{and} \quad B_j(w_j) = w_j \cdot f(\overline{u}_j)$$

Moreover assume that x and y are as in (22), and they satisfy $x = Ty$ and $h \cdot x = 0 \; \forall \; h \in H$. Now we will show that the Inner Verifier's test accepts. Indeed,

$$\begin{aligned}
\oplus_{i=1}^{d} A_i(z_i \oplus h_i) &= \oplus_{i=1}^{d} (z_i \oplus h_i) \cdot f(\overline{v}_i) \\
&= (z \oplus h) \cdot (f(\overline{v}_1) \circ f(\overline{v}_2) \circ \ldots \circ f(\overline{v}_d)) \\
&= (z \oplus h) \cdot x \\
&= z \cdot x \oplus h \cdot x \\
&= z \cdot Ty \oplus 0 \\
&= (zT) \cdot y \\
&= (w_1 \circ w_2 \circ \ldots \circ w_d) \cdot (f(\overline{u}_1) \circ f(\overline{u}_2) \circ \ldots \circ f(\overline{u}_d)) \\
&= \oplus_{j=1}^{d} w_j \cdot f(\overline{u}_j) \\
&= \oplus_{j=1}^{d} B_j(w_j)
\end{aligned}$$

10.2 Proof of Theorem 1.9

Note that the Inner Verifier's test reads $2d$ bits in total. Let Π be the proof given to the Inner Verifier. Let us first prove that every query of the Inner Verifier is uniform over Π.

Uniformity of Queries

Note that any particular query has the following distribution : pick a point $\overline{v} \in \mathbb{F}^m$ at random and then pick a random bit from the supposed Hadamard Code of $f(\overline{v})$. Thus the query is uniformly distributed over the proof Π.

Now the following two lemmas are sufficient to prove Theorem 1.9. These are the completeness and the soundness lemmas respectively.

Lemma 10.2 *Suppose that the instance $\mathcal{A}$ of Homogeneous Algebraic CSP has optimum $OPT(\mathcal{A})$. Let f be the polynomial that achieves this optimum and let a proof Π be constructed by taking Hadamard Code for every value of f. Let Π_0 be the set of 0-bits in the proof (this constitutes half of the bits in the whole proof). Let Q be the set of $2d$ query bits by the Inner Verifier. Then*

$$\Pr_Q[Q \subseteq \Pi_0] \geq OPT(\mathcal{A}) \cdot \frac{1}{2^{2d-1}}$$

The probability is taken over the random test of the Inner Verifier.

Lemma 10.3 *If Π_* is any set of half the bits from Π, and*

$$\left| \Pr_Q[Q \subseteq \Pi_*] - \frac{1}{2^{2d}} \right| \geq \delta$$

then the instance $\mathcal{A}$ of Homogeneous Algebraic CSP has $OPT(\mathcal{A}) \geq \delta^{C''}$ where C'' is an absolute constant.

Let us see how these lemmas prove Theorem 1.9. Let $N = n^{CK}$ be the instance of the MDC given by Theorem 1.5. Theorem 3.3 gives an instance $\mathcal{A}$ of HomAlgCSP with parameters ϵ^*, the degree $d^* = 100/\epsilon^*$, $m = N^{\epsilon^*} = N^{100/d^*}$, and $k = 21$. We have :

- $|\mathbb{F}|^m \leq (N^2)^{N^{100/d^*}} \leq 2^{N^{200/d^*}}$
- $OPT(\mathcal{A}) \geq 1 - 1/2^K$ in completeness case and $OPT(\mathcal{A}) \leq 1/2^{2^K}$ in soundness case.
- The Inner Verifier makes $2d$ queries where $d = 2((k+2)d^* + 1) \leq 50d^*$.

Soundness/Quasi-randomness

We show that if Π_* is any set of half the bits from the proof Π, then

$$\left| \Pr_Q[Q \subseteq \Pi_*] - \frac{1}{2^{2d}} \right| \leq \frac{1}{2^{40d}}$$

If on the contrary this is not true, then Lemma 10.3 implies that

$$OPT(\mathcal{A}) \geq 1/2^{40C''d} \geq 1/2^{2000C''d^*} > 1/2^{2^K}$$

where we choose $d^* = 2^K/(3000C'')$. This is a contradiction.

Now note that the size of the proof Π is essentially $|\mathbb{F}|^m$ and it is at most

$$|\mathbb{F}|^m \leq 2^{N^{200/d^*}} \leq 2^{n^{200CK \cdot 3000C''/2^K}} \leq 2^{n^\epsilon}$$

We can make ϵ arbitrarily small by choosing K large enough. Note that $\epsilon = \Omega(K/2^K)$ whereas $d = O(2^K)$. Hence $d = O(1/\epsilon \log(1/\epsilon))$.

Completeness

Completeness follows directly from Lemma 10.2 by noting that $OPT(\mathcal{A}) \geq 1 - 1/2^K = 1 - O(1/d)$.

10.3 Proofs of Lemmas 10.2, 10.3

We will prove both Lemma 10.2 and 10.3 simultaneously. Let Π be a proof and let Π_* be a set of half the bits in the proof. Imagine setting bits in Π_* to 1 and the rest to 0. This defines the tables A_i, B_j in the proof. Let us estimate the probability that for a random test of the Inner Verifier, all the queries fall in Π_*, i.e.

$$\Pr_Q[Q \subseteq \Pi_*]$$

This probability can be arithmetized as

$$E_{\substack{C,\ell,L,\overline{v_1},\ldots,\overline{v_d},\\ \overline{u_1},\ldots,\overline{u_d},z,h}} \left[\prod_{i=1}^{d} A_i(z_i \oplus h_i) \prod_{j=1}^{d} B_j(w_j)\right]$$

Here C is the choice of a random constraint of the HomAlgCSP instance. The rest of the choices are as in the Inner Verifier's action. Note that the bits $A_i(z_i \oplus h_i)$ and $B_j(w_j)$ are exactly the query bits. Thus we count a test only if each of the query bits falls in Π_*. We analyze this expression.

Fix everything else and consider only expectation over z, h. We drop the expectation sign for convenience. Plugging in Fourier expansions of A_i's and B_j's we get

$$\sum_{\alpha_1,\ldots,\alpha_d,\beta_1,\ldots,\beta_d} \left[\prod_{i=1}^{d} \widehat{A}_{i,\alpha_i} \prod_{j=1}^{d} \widehat{B}_{j,\beta_j} \cdot \prod_{i=1}^{d} \chi_{\alpha_i}(z_i \oplus h_i) \cdot \prod_{j=1}^{d} \chi_{\beta_j}(w_j)\right]$$

which simplifies to

$$\sum_{\substack{\alpha=\alpha_1\circ\alpha_2\ldots\circ\alpha_d,\\ \beta=\beta_1\circ\beta_2\ldots\circ\beta_d}} \left[\prod_{i=1}^{d} \widehat{A}_{i,\alpha_i} \prod_{j=1}^{d} \widehat{B}_{j,\beta_j} \cdot (-1)^{\alpha\cdot z\oplus\beta\cdot w} \cdot (-1)^{\alpha\cdot h}\right]$$

Now note that h is chosen randomly from the linear subspace H and thus the expectation over h is zero unless $\alpha \perp H$. Moreover,

$$\begin{aligned} z\cdot\alpha \oplus w\cdot\beta &= z\cdot\alpha \oplus zT\cdot\beta \\ &= z\cdot(\alpha\oplus T\beta) \end{aligned}$$

Therefore, the expectation over z is zero unless $\alpha = T\beta$. Thus we get

$$\sum_{\substack{\alpha=\alpha_1\circ\alpha_2\ldots\circ\alpha_d,\\ \alpha\perp H,\ \beta=T^{-1}\alpha}} \left[\prod_{i=1}^{d} \widehat{A}_{i,\alpha_i} \prod_{j=1}^{d} \widehat{B}_{j,\beta_j}\right] \tag{24}$$

10.4 Proof of Lemma 10.2

Let f be the solution for the HomAlgCSP instance with optimum $OPT(\mathcal{A})$. Thus for any $\overline{v} \in \mathbb{F}^m$, the corresponding Hadamard Code $A_{\overline{v}}$ is precisely (see Fact A.7)

$$\frac{1}{2} - \frac{1}{2}\chi_{f(\overline{v})}$$

Thus the table $A_{\overline{v}}$ has exactly two non-zero Fourier coefficients, $\widehat{A}_{\overline{v},0} = \frac{1}{2}$ and $\widehat{A}_{\overline{v},f(\overline{v})} = -\frac{1}{2}$.

Let us prove that

$$\Pr_Q[Q \subseteq \Pi_0] \geq OPT(\mathcal{A}) \cdot \frac{1}{2^{2d-1}}$$

Set $\Pi_* = \Pi_0$. As before, we imagine the bits in Π_* as set to 1 and rest of the bits as set to 0. This corresponds to flipping the bits in the Hadamard Codes. Hence the resulting tables A_i, B_j have Fourier representation (see Fact A.7)

$$A_i = \frac{1}{2} + \frac{1}{2}\chi_{f(\overline{v}_i)} \qquad B_j = \frac{1}{2} + \frac{1}{2}\chi_{f(\overline{u}_j)}$$

Note that f satisfies $OPT(\mathcal{A})$ fraction of the constraints of the HomAlgCSP. For any satisfied constraint, we get a contribution of $\frac{1}{2^{2d}}$ in Term (24) in each of these cases :

$$\begin{aligned} \alpha_1 = \alpha_2 = \ldots = \alpha_d = \beta_1 = \ldots = \beta_d = 0 \\ \alpha_i = f(\overline{v}_i), \ \beta_j = f(\overline{u}_j) \text{ for } i, j = 1, 2, \ldots, d \end{aligned} \tag{25}$$

All other terms are non-negative. Thus

$$\Pr_Q[Q \subseteq \Pi_0] \geq OPT(\mathcal{A}) \cdot (\frac{1}{2^{2d}} + \frac{1}{2^{2d}}) = OPT(\mathcal{A}) \cdot \frac{1}{2^{2d-1}}$$

10.5 Proof of Lemma 10.3

Assume that the quantity in (24) is δ away from $\frac{1}{2^{2d}}$. First, we show that the contribution of the term with $\alpha = 0$ is essentially $\frac{1}{2^{2d}}$. This implies that the contribution of terms with non-zero α is δ. Using this, we define a strategy for the Outer Verifier with success probability δ^2 which in turn gives a solution to the HomAlgCSP instance with value $\delta^{O(1)}$.

Contribution of the term with $\alpha = 0$

This contribution is

$$E_{\substack{\ell, L, \overline{v}_1, \ldots, \overline{v}_d, \\ \overline{u}_1, \ldots, \overline{u}_d}} \left[\prod_{i=1}^{d} \widehat{A}_{i,0} \widehat{B}_{i,0} \right]$$

where the expectation is taken over all line-curve pairs and choice of d random points each on the line and the curve.

Note that the zero Fourier coefficient equals the fraction of 1's in the table. Overall the proof contains half 1's.

We use the mixing property of curves and lines (see Appendix A.4). Therefore for a random choice of intersecting curve L and line ℓ, except with probability $O(1/|\mathbb{F}|^{1/3})$, we have

$$E_{\overline{v} \in L}[\widehat{A}_{\overline{v},0}] \approx \frac{1}{2}, \qquad E_{\overline{v} \in \ell}[\widehat{A}_{\overline{v},0}] \approx \frac{1}{2}$$

where the error in the approximation is bounded by $O(1/|\mathbb{F}|^{1/3})$. Once the curve and the line are chosen, $\overline{v}_1, \ldots, \overline{v}_d, \overline{u}_1, \ldots, \overline{u}_d$ are $2d$ random points on the curve and the line. Thus,

$$E_{\substack{\overline{v}_1, \ldots, \overline{v}_d, \\ \overline{u}_1, \ldots, \overline{u}_d}} \left[\prod_{i=1}^{d} \widehat{A}_{i,0} \widehat{B}_{i,0} \right] = \left(E_{\overline{v} \in L}[\widehat{A}_{\overline{v},0}] \right)^d \cdot \left(E_{\overline{v} \in \ell}[\widehat{A}_{\overline{v},0}] \right)^d \approx \frac{1}{2^{2d}}$$

upto error $O(1/|\mathbb{F}|^{1/3})$. This proves that the contribution of the term with $\alpha = 0$ is essentially $\frac{1}{2^{2d}}$.

Contribution of the terms with non-zero α

Using Cauchy-Schwartz, contribution of the terms with non-zero α in (24) is at most

$$\sqrt{E\left[\sum_{\alpha\neq 0,\alpha\perp H}\prod_{i=1}^{d}\widehat{A}^2_{i,\alpha_i}\right]}\sqrt{E\left[\sum_{\alpha\neq 0,\alpha\perp H,\beta=T^{-1}\alpha}\prod_{j=1}^{d}\widehat{B}^2_{j,\beta_j}\right]}$$

which is at most

$$\sqrt{E\left[\sum_{\alpha\neq 0,\alpha\perp H}\prod_{i=1}^{d}\widehat{A}^2_{i,\alpha_i}\right]}$$

Observe that the expectation naturally gives a strategy for Outer Verifier ! For any point $\overline{v}\in\mathbb{F}^m$, define $f(\overline{v})=\alpha_i$ with probability $\widehat{A}^2_{\overline{v},\alpha_i}$. This gives a (probabilistic) construction of a table of values $f:\mathbb{F}^m\mapsto\mathbb{F}$. The condition $\alpha\neq 0$ ensures that not all values read by the Outer Verifier are zero. The condition $\alpha\perp H$ ensures that the homogeneous linear constraint checked by the Outer Verifier is satisfied. These are exactly the two conditions when the Outer Verifier accepts. Note how the redundancy introduced by the Modified Outer Verifier enabled us to apply Cauchy-Schwartz.

Thus, if the contribution of the terms with non-zero α is δ, then we can construct a table $f:\mathbb{F}^m\mapsto\mathbb{F}$ which the Outer Verifier accepts with probability δ^2. This probability is the average of the Outer Verifier's success probability over the choice of the HomAlgCSP constraint. Thus for at least $\delta^2/2$ fraction of the HomAlgCSP constraints, the Outer Verifier's success probability is $\delta^2/4$. Apply Theorem 7.5 with $\delta^2/4$ in place of δ. Let $P_1,P_2,\ldots,P_t$ be the non-zero polynomials as in there. It follows that for at least $\delta^2/4$ fraction of the HomAlgCSP constraints, at least one of the polynomials P_j satisfies this constraint. Thus there is one non-zero polynomial P_{j_0} that satisfies $(\delta^2/4)/t\geq\delta^{C''}$ fraction of the HomAlgCSP constraints. Note that $t\leq 2c_3/(\delta^2/8)^{c_2}$. This proves Lemma 10.3.

11 Conclusion

We have ruled out PTAS for three notorious problems : Graph Min-Bisection, Dense k-Subgraph and Bipartite Clique. Our results hold under the assumption that NP does not have subexponential time algorithms. This could be a new avenue for showing inapproximability results instead of the more common assumption that NP does not have polynomial time or quasi-polynomial time algorithms.

We have made several new contributions in terms of conceptual ideas and techniques. We introduced the notion of Quasi-random PCPs that should be very useful to prove inapproximability results for graph theoretic problems, e.g. Graph Expansion. Some of the ideas in this paper have roots in [30]. To the best of our knowledge, it was the first paper that employed a combination of the algebraic techniques (those used to prove the PCP Theorem) and more recent Fourier analysis methods. All recent results, on the other hand, use the PCP Theorem as a black-box and then use Parallel Repetition [33] and Fourier analysis. In this paper, we make an even heavier use of algebraic techniques, using generalizations and variants of the Polynomial Encoding method, Low Degree Test and the Sum-Check Protocol. This new direction in inapproximability theory looks very promising.

The connection between Minimum Distance of Code problem and the graph theoretic problems is very surprising. It could be possible to prove inapproximability results assuming sufficiently strong hardness results for the Shortest Vector Problem, the analogue of MDC for integer lattices.

12 Acknowledgments

I would like to thank Guy Kindler, Muli Safra, Misha Alekhnovich, Venkatesan Guruswami, Noga Alon, Sanjeev Arora, Madhu Sudan, Avi Wigderson, Moses Charikar and Howard Karloff for many helpful discussions. Thanks to anonymous referees for pointing out a bug in FOCS 2004 version of Theorem 1.2.

References

[1] M. Alekhnovich. More on average case vs approximation complexity. In *Proc. 44th IEEE Symposium on Foundations of Computer Science*, pages 298–307, 2003.

[2] S. Arora. Probabilistic checking of proofs and the hardness of approximation problems. *Ph.D. thesis*, UC Berkeley, 1994.

[3] S. Arora, L. Babai, J. Stern, and E.Z. Sweedyk. The hardness of approximate optima in lattices, codes and systems of linear equations. *Journal of Computer and Systems Sciences* (54), pages 317–331, 1997.

[4] S. Arora and C. Lund. The approximability of NP-hard problems. Chapter in *Approximation Algorithms for NP-hard Problems*, D. Hochbaum ed, PWS Publishing, Boston, 1996.

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

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

[7] S. Arora and M. Sudan. Improved low-degree testing and its applications. In *Proc. 29th ACM Symposium on Theory of Computing*, 1997.

[8] E. Ben-Sasson, O. Goldreich, P. Harsha, M. Sudan, and S. Vadhan. Robust PCPs of proximity, shorter PCPs and applications to coding. In *Proc. 36th ACM Symposium on Theory of Computing*, pages 1–10, 2004.

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

[10] P. Berman and G. Schnitger. On the complexity of approximating the independent set problem. *Information and Computation*, Vol 96, pages 77–94, 1992.

[11] A. Blum. *Ph.D. Thesis*, MIT, MIT/LCS/TR-506, 1991.

[12] M. Blum, M. Luby, and R. Rubinfeld. Self-testing/correcting with applications to numerical problems. *Journal of Computer and Systems Sciences*, 47(3), pages 549–595, 1993.

[13] I. Dinur and S. Safra. On the importance of being biased. In *Proc. 34th ACM Symposium on Theory of Computing*, pages 33–42, 2002.

[14] I. Dinur, V. Guruswami, S. Khot, and O. Regev. A new multilayered PCP and the hardness of hypergraph vertex cover. In *Proc. 35th ACM Symposium on Theory of Computing*, pages 595-601, 2003.

[15] I. Dumer, D. Micciancio and M. Sudan. Hardness of approximating the minimum distance of a linear code. *IEEE Transactions on Information Theory*, 49(1), pages 22–37, 2003.

[16] U. Feige. A threshold of ln n for approximating set cover. Journal of the ACM, 45(4), pages 634–652, 1998.

[17] U. Feige. Relations between average case complexity and approximation complexity. In *Proc. 34th ACM Symposium on Theory of Computing*, pages 534–543, 2002.

[18] U. Feige, S. Goldwasser, L. Lovász, S. Safra, and M. Szegedy. Interactive proofs and the hardness of approximating cliques. *Journal of the ACM*, 43(2), pages 268–292, 1996.

[19] U. Feige and J. Kilian. Zero knowledge and the chromatic number. *Journal of Computer and Systems Sciences*, 57(2), pages 187–199, 1998.

[20] U. Feige and S. Kogan. Hardness of approximation of the Balanced Complete Bipartite Subgraph problem. *Technical Report MCS04-04*, Department of Computer Science and Applied Math., The Weizmann Institute of Science, 2004.

[21] U. Feige, G. Kortsarz, and D. Peleg. The dense k-subgraph problem. *Algorithmica*, 29(3):410–421, 2001.

[22] U. Feige and R. Krauthgamer. A polylogarithmic approximation of the minimum bisection. *SIAM Journal of Computing*, 31(4), pages 1090–1118, 2002.

[23] V. Guruswami, J. Håstad, and M. Sudan. Hardness of approximate hypergraph coloring. In *Proc. 41st IEEE Symposium on Foundations of Computer Science*, pages 149–158, 2000.

[24] P. Harsha and M. Sudan. Small PCPs with low query complexity. *Computational Complexity*, 9(3-4), pages 157–201, Dec 2000.

[25] J. Hastad. Clique is hard to approximate within $n^{1-\epsilon}$. *Acta Mathematica*, Vol 182, pp 105-142, 1999.

[26] J. Håstad. Some optimal inapproximability results. *Journal of ACM*, Vol. 48, pp 798–859, 2001.

[27] J. Holmerin. Vertex cover on 4-regular hyper-graphs is hard to approximate within 2-epsilon. In *Proc. 34th ACM Symposium on Theory of Computing*, pages 544–552, 2002.

[28] S. Khot. Hardness of approximating the shortest vector problem in lattices. In *Proc. 45th IEEE Symposium on Foundations of Computer Science*, pages 126–135, 2004.

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

[30] S. Khot and J. Holmerin. A new PCP outer verifier with applications to homogeneous linear equations and Max-Bisection. In *Proc. 36th ACM Symposium on Theory of Computing*, 2004.

[31] A. Lubotzky, R. Phillips, P. Sarnak. Ramanujan graphs. *Combinatorica*, Vol. 8, pages 261–277, 1988.

[32] M. Luby, A. Wigderson. Pairwise independence and derandomization. Tech Report TR-95-035, International Computer Science Institute, 1995.

[33] R. Raz. A parallel repetition theorem. *SIAM Journal on Computing*, 27(3), pages 763–803, 1998.

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

[35] L. Trevisan. Inapproximability of combinatorial optimization problems. Survey paper, 2004.

[36] V. Vazirani. Approximation Algorithms. Springer-Verlag, Berlin, 2001.

A Useful Facts and Tools

A.1 Schwartz Lemma

We use the following fact (called Schwartz Lemma) again and again :

Fact A.1 *Let $f(x_1, \ldots, x_m)$ be a degree d polynomial in m variables over field $\mathbb{F}$. Assume that when $x_1, \ldots, x_m$ are chosen at random from $\mathbb{F}$,*

$$\Pr[f(x_1, \ldots, x_m) = 0] \geq \frac{2d}{|\mathbb{F}|}$$

Then f must identically be zero. Consequently, for two degree d polynomials f and g in m variables, if $f(x_1, \ldots, x_m) = g(x_1, \ldots, x_m)$ with probability $\frac{2d}{|\mathbb{F}|}$, then they must be identical polynomials.

A.2 Addition and Multiplication over Fields of Characteristic 2

Here we summarize simple facts about addition and multiplication over the Field $\mathbb{F} = \mathbb{F}_{2^l}$. Elements of $\mathbb{F}$ can be viewed as l-bit strings, i.e. as vectors in $\mathbb{F}_2^l$. For $x \in \mathbb{F}$, let x^* denote the corresponding vector in $\mathbb{F}_2^l$.

Fact A.2

$$(x + y)^* = x^* \oplus y^* \ \forall\ x, y \in \mathbb{F}$$

where on LHS we have addition in the field and on RHS, we have addition in $\mathbb{F}_2^l$.

Fact A.3 *If $\gamma \in \mathbb{F}$, then there exists a $l \times l$ matrix A_γ with entries in $\mathbb{F}_2$ such that*

$$(\gamma x)^* = A_\gamma x^* \quad \forall\ x \in \mathbb{F}$$

where on LHS we have multiplication in field and on RHS we have multiplication of a matrix and a vector.

A.3 Hadamard Codes and Fourier Analysis

Definition A.4 *Let $x \in \mathbb{F}_2^l$ be an l-bit string. Its Hadamard Code A_x is a 2^l-bit string whose co-ordinates are indexed by $s \in \mathbb{F}_2^l$ and*

$$A_x(s) = s \cdot x \quad \forall s \in \mathbb{F}_2^l$$

where $s \cdot x$ denotes the inner product over $\mathbb{F}_2^l$.

Definition A.5 *For $\alpha \in \mathbb{F}_2^l$, let $\chi_\alpha : \mathbb{F}_2^l \mapsto \{-1, 1\}$ be a real-valued function defined as :*

$$\chi_\alpha(x) = (-1)^{\alpha \cdot x}$$

Fact A.6 *Any real-valued function $A : \mathbb{F}_2^l \mapsto \mathbb{R}$ can be uniquely written as*

$$A = \sum_{\alpha \in \mathbb{F}_2^l} \widehat{A}_\alpha\ \chi_\alpha$$

where $\widehat{A}_\alpha$ are real numbers called Fourier Coefficients. We have Parseval's identity :

$$\sum_{\alpha \in \mathbb{F}_2^l} \widehat{A}_\alpha^2 = \|A\|_2^2 = \frac{1}{2^l} \sum_{s \in \mathbb{F}_2^l} A(s)^2$$

Fact A.7 *Let A_x be Hadamard Code of x. It can be regarded as a real-valued function $A_x : \mathbb{F}_2^l \mapsto \{0,1\}$. Then its Fourier representation is :*

$$A_x = \frac{1}{2} - \frac{1}{2}\chi_x$$

The first term is $\frac{1}{2}\chi_0$ where $\chi_0 \equiv 1$. If A'_x is a function obtained by flipping values of A_x (i.e. 0 flipped to 1 and vice versa), then Fourier representation of A'_x is :

$$A'_x = \frac{1}{2} + \frac{1}{2}\chi_x$$

A.4 Mixing Property of Lines and Curves

Lemma A.8 *Let $\psi : \mathbb{F}^m \mapsto [0,1]$ be any function with $E_{\,\overline{u}\in\mathbb{F}^m}[\psi(\overline{u})] = \frac{1}{2}$. Let ℓ be a line chosen at random. Then w.h.p. the average of ψ-values of points on ℓ is very close to $\frac{1}{2}$. To be precise*

$$\Pr_\ell\left[\left|E_{\overline{u}\in\ell}[\psi(\overline{u})] - \frac{1}{2}\right| \geq \frac{1}{|\mathbb{F}|^{1/3}}\right] \leq \frac{1}{|\mathbb{F}|^{1/3}}$$

A similar statement holds for curves. Let $\{\overline{p}_i\}_{i=1}^k$ and $\overline{v}$ be points in $\mathbb{F}^m$. Let $\{t_1, \ldots, t_{k+3}, t\}$ be distinct field elements. A random curve L passing through $\overline{v}$ is obtained as follows :

- Pick random points $\overline{a}, \overline{b} \in \mathbb{F}^m$.
- Let L be the unique degree $k+2$ curve such that

$$L(t_i) = \overline{p}_i,\ 1 \leq i \leq k,\quad L(t_{k+1}) = \overline{a},\ L(t_{k+2}) = \overline{b},\quad L(t) = \overline{v}$$

Lemma A.9 *Let $\psi : \mathbb{F}^m \mapsto [0,1]$ be any function with $E_{\,\overline{u}\in\mathbb{F}^m}[\psi(\overline{u})] = \frac{1}{2}$. Let L be a curve chosen at random as above. Then w.h.p. the average of ψ-values of points on L is very close to $\frac{1}{2}$. To be precise*

$$\Pr_L\left[\left|E_{\overline{u}\in L}[\psi(\overline{u})] - \frac{1}{2}\right| \geq \frac{1}{|\mathbb{F}|^{1/3}}\right] \leq \frac{1}{|\mathbb{F}|^{1/3}}$$

Both lemmas are proved using pairwise independence and Chebyshev's inequality.

B Proof of Theorem 1.5

We start with a result of Dumer, Micciancio and Sudan [15] that it is NP-hard to approximate the Minimum Distance of Code within any constant factor (say 5) and the result holds on binary asymptotically good codes. Formally,

Theorem B.1 *([15]) There is an absolute constant $\zeta > 0$ and a reduction from SAT to MDC where*

1. *The MDC instance is over $\mathbb{F}_2$, i.e. the code is binary.*
2. *(Completeness :) YES instance of SAT is mapped to an MDC instance $\mathbf{A}$ with $OPT(\mathbf{A}) \leq \zeta$.*
3. *(Soundness :) NO instance of SAT is mapped to an MDC instance $\mathbf{A}$ with $OPT(\mathbf{A}) \geq 5\zeta$.*

4. *The reduction runs in time n^C where n is the size of SAT instance and C is an absolute constant.*

The hardness of MDC can be boosted via Tensor Product giving this theorem :

Theorem B.2 *([15]) There is an absolute constant $\zeta > 0$ such that for every integer K, there is a reduction from SAT to MDC satisfying :*

1. *The MDC instance is over $\mathbb{F}_2$, i.e. the code is binary. It has N rows and n' columns.*
2. *(Completeness :) YES instance of SAT is mapped to an MDC instance $\mathbf{A}$ with $OPT(\mathbf{A}) \leq \zeta^K$.*
3. *(Soundness :) NO instance of SAT is mapped to an MDC instance $\mathbf{A}$ with $OPT(\mathbf{A}) \geq 5^K\zeta^K$.*
4. *The reduction runs in time n^{CK} where n is the size of SAT instance and C is an absolute constant. In particular, $N \leq n^{CK}$.*

Let $\mathbf{A} = \{a_{ij}\}$ be an instance of MDC and let n, n', N, K, C, ζ be parameters as in Theorem B.2. We interpret the rows of $\mathbf{A}$ as linear forms

$$L_i = \sum_{j=1}^{n'} a_{ij} z_j \quad 1 \leq i \leq N$$

Thus the goal of the MDC problem is to find a non-zero solution $\mathbf{z} = (z_1, z_2, \ldots, z_{n'})$ so as to minimize the number of non-zero linear forms. By taking an extension field of $\mathbb{F}_2$, we can assume that Theorem B.2 holds for codes over a field $\mathbb{F}$ of characteristic 2 with size $\approx N$.

Now the idea is the following. Create a new instance $\mathbf{A}'$ of MDC whose linear forms are linear combinations of the linear forms $\{L_i\}_{i=1}^N$. Thus a typical linear form looks like :

$$\sum_{s=1}^{t} \alpha_s L_{i_s} \quad \text{where} \quad \alpha_s \in \mathbb{F}, \;\; i_s \in [N]$$

There are $|\mathbb{F}|^t N^t$ such linear forms. However, we want to take only $O(N)$ of them and we achieve this using walks on expanders. Choose $t = 1/(\zeta^K 2^K)$.

Let G be a degree D expander on N vertices with second largest eigenvalue λ. It is possible to explicitly (and efficiently) construct such expanders with $\lambda \leq D^{9/10}$ (in fact one can get $\lambda \approx 2\sqrt{D}$ using the Ramanujan Graphs [31]). Choose $D = \left(\frac{4}{5^K\zeta^K}\right)^{10}$ so that $\lambda/D \leq 5^K\zeta^K/4$. Let $S \subseteq \mathbb{F}$ with $|S| = 2^{2\cdot 2^K}$.

Let the new instance $\mathbf{A}'$ of MDC consist of these linear forms :

$$\sum_{s=1}^{t} \alpha_s L_{i_s} \quad \text{where} \quad \alpha_s \in S, \;\; \text{and} \;\; [i_1, i_2, \ldots, i_t] \text{ is a walk on graph } G \tag{26}$$

The number of such linear forms is $|S|^t N \cdot D^{t-1}$ which is $O(N)$ since ζ, K are constants. Now we prove the completeness and soundness properties :

Completeness

If the SAT instance is a YES instance, we know that $OPT(\mathbf{A}) \leq \zeta^K$. Thus there is a non-zero solution $\mathbf{z}$ such that at most ζ^K fraction of the linear forms $\{L_i\}_{i=1}^N$ are non-zero.

For a random walk $[i_1, i_2, \ldots, i_t]$ on G, the probability that there exists $L_{i_s} \neq 0$ is at most $t\zeta^K$. This implies that the fraction of linear forms in (26) that are non-zero is at most $t\zeta^K$. Thus $OPT(\mathbf{A}') \leq t\zeta^K = \frac{1}{2^K}$.

Soundness

If the SAT instance is a NO instance, we know that $OPT(\mathbf{A}) \geq 5^K \zeta^K$. Fix any non-zero $\mathbf{z}$. We know that at least $5^K \zeta^K$ fraction of linear forms $\{L_i\}_{i=1}^N$ are non-zero. Let

$$W := \{i \mid i \in [N],\ L_i \neq 0\}$$

so that $\mu(W) = |W|/N \geq 5^K \zeta^K$. Using [32, Section 15], we can bound the probability that all vertices of the random walk $[i_1, i_2, \dots, i_t]$ fall outside W.

$$\begin{aligned}
\Pr[i_1 \in \overline{W} \wedge i_2 \in \overline{W} \dots \wedge i_t \in \overline{W}] &\leq \left(\sqrt{\mu(\overline{W})} + \frac{\lambda}{D}\right)^t \\
&\leq \left(\sqrt{1 - 5^K\zeta^K} + 5^K\zeta^K/4\right)^t \\
&\leq \left(1 - 5^K\zeta^K/4\right)^{1/(\zeta^K 2^K)} \\
&\leq e^{-(5/2)^K/4} \leq 2^{-2\cdot 2^K}
\end{aligned}$$

Therefore at least a fraction $1 - 2^{-2\cdot 2^K}$ of the linear forms in (26) contain a linear form $L_{i_{s^*}} \neq 0$. For any fixed tuple $(L_{i_1}, \dots, L_{i_t})$ that contains at least one $L_{i_{s^*}} \neq 0$, after choosing the coefficients $\alpha_1, \dots, \alpha_t \in S$ at random, the probability that (26) is non-zero is at least $1 - \frac{1}{|S|}$. Therefore, the fraction of non-zero linear forms among (26) is at least

$$(1 - 2^{-2\cdot 2^K}) \cdot (1 - \frac{1}{|S|}) \geq (1 - 2^{-2\cdot 2^K}) \cdot (1 - 1/2^{2\cdot 2^K}) \geq 1 - 1/2^{2^K}$$

Thus $OPT(\mathbf{A}') \geq 1 - \frac{1}{2^{2^K}}$. This proves Theorem 1.5.

C Reductions from Quasi-random PCP to Graph Min-Bisection, Dense k-Subgraph and Bipartite Clique

In this section, we prove that the PCP construction given by Theorem 1.9 implies inapproximability results for Graph Min-Bisection, Dense k-Subgraph and Bipartite Clique. This proves Theorem 1.1. The reductions given here are similar to the reductions given by Feige [17] (correctness of his reduction however relies on his hypothesis about hardness of Random 3SAT).

C.1 Graph Min-Bisection

Let N be the proof size and M be the total number of tests of the PCP verifier in Theorem 1.9. Both N and M are bounded by $2^{O(n^\epsilon)}$. Let d be the integer as in that theorem.

Construct a graph G as follows : it is disjoint union of a bipartite graph and a clique. The clique has size $(1 - \frac{3}{2}\frac{1}{2^{d-1}})dM^2 - \frac{N}{2}$. The bipartite graph is constructed as follows : the LHS has N vertices corresponding to the N bits in the PCP proof. The RHS has M clusters, one for every test of the PCP verifier. Every cluster is a clique of size dM. One vertex in every cluster is designated as a *connection vertex*. The bipartite edges are only between the LHS vertices and the connection vertices on RHS. There is an edge between a vertex (= bit) on LHS and a connection vertex (= PCP test) on RHS if and only if the bit is accessed by the PCP test. This completes the description of the graph G. Note that every connection vertex has d neighbors on LHS since this is a d-query PCP. Since every query is uniformly distributed, every bit in the proof appears in the same number of tests. Therefore, the LHS vertices all have the same degree, say Δ, so that $\Delta N = dM$.

Completeness

We claim that the graph G can be partitioned into two equal parts so that the number of crossing edges is at most $(\frac{1}{2} - \frac{3}{4}\frac{1}{2^{d-1}})dM$.

Theorem 1.9 gives a set Π_0 of half the bits from the proof such that at least $(1 - O(1/d))\frac{1}{2^{d-1}} \geq \frac{3}{4}\frac{1}{2^{d-1}}$ fraction of the tests access all the d queries from Π_0. Partition the graph G into two equal parts as follows : one part consists of

1. $\frac{N}{2}$ vertices from LHS corresponding to bits in Π_0. Abusing notation, denote the set of these vertices also by Π_0.
2. $\frac{3}{4}\frac{1}{2^{d-1}}M$ clusters from RHS corresponding to the tests that access all d queries from Π_0. Call the set of these tests (or connection vertices) as $\mathcal{T}_0$.
3. The clique of size $(1 - \frac{3}{2}\frac{1}{2^{d-1}})dM^2 - \frac{N}{2}$.

The second part consists of all the remaining $(1 - \frac{3}{4}\frac{1}{2^{d-1}})M$ clusters on RHS corresponding to the tests in $\overline{\mathcal{T}}_0$. The only crossing edges are between Π_0 and $\overline{\mathcal{T}}_0$. The total number of edges incident on Π_0 is $\Delta|\Pi_0| = \Delta\frac{N}{2}$. The number of edges between Π_0 and $\mathcal{T}_0$ is $d|\mathcal{T}_0| = d(\frac{3}{4}\frac{1}{2^{d-1}}M)$ since every connection vertex in $\mathcal{T}_0$ has degree d and all its neighbors (queries) are in Π_0. Therefore, the number of crossing edges is at most

$$\Delta\frac{N}{2} - \frac{3}{4}\frac{1}{2^{d-1}}dM = \frac{dM}{2} - \frac{3}{4}\frac{1}{2^{d-1}}dM = \left(\frac{1}{2} - \frac{3}{4}\frac{1}{2^{d-1}}\right)dM$$

Soundness

We show that for any bisection of G, the number of crossing edges is at least the expression (27). Note that the clusters and the clique have size at least dM, so any bisection that cuts either a cluster or the clique has at least $dM - 1$ crossing edges. Hence we need to consider only those partitions for which every cluster and the clique are completely contained inside one of the parts.

This implies that one of the parts consists of :

1. $\frac{N}{2}$ vertices from LHS. Call the set of these vertices and the set of corresponding bits in the PCP proof as Π_*.
2. $\frac{3}{4}\frac{1}{2^{d-1}}M$ clusters from RHS. Call the set of corresponding connection vertices and the set of corresponding PCP tests as $\mathcal{T}_*$.
3. The clique of size $(1 - \frac{3}{2}\frac{1}{2^{d-1}})dM^2 - \frac{N}{2}$.

By Theorem 1.9, at most $\frac{1}{2^d} + \frac{1}{2^{20d}}$ fraction of the tests have all their queries in Π_*. Hence, the number of edges between Π_* and $\mathcal{T}_*$ is at most

$$d \cdot \left(\frac{1}{2^d} + \frac{1}{2^{20d}}\right) M + (d-1) \cdot \left(\frac{3}{4}\frac{1}{2^{d-1}} - (\frac{1}{2^d} + \frac{1}{2^{20d}})\right) M = dM\left(\frac{3}{4}\frac{1}{2^{d-1}} - \frac{1}{d}(\frac{1}{2^{d+1}} - \frac{1}{2^{20d}})\right)$$

The number of crossing edges is at least equal to the number of edges between Π_* and $\overline{\mathcal{T}}_*$. This is equal to the number of edges incident on Π_* (which is $\Delta\frac{N}{2} = \frac{dM}{2}$) minus the number of edges between Π_* and $\mathcal{T}_*$. Thus the number of crossing edges is at least

$$dM\left(\frac{1}{2} - \frac{3}{4}\frac{1}{2^{d-1}} + \frac{1}{d}(\frac{1}{2^{d+1}} - \frac{1}{2^{20d}})\right) \tag{27}$$

Inapproximability Factor

Clearly, the inapproximability factor is

$$\frac{\frac{1}{2} - \frac{3}{4}\frac{1}{2^{d-1}} + \frac{1}{d}\left(\frac{1}{2^{d+1}} - \frac{1}{2^{20d}}\right)}{\frac{1}{2} - \frac{3}{4}\frac{1}{2^{d-1}}}$$

This is $\approx 1 + \frac{2}{d2^{d+1}} = 1 + \frac{1}{2^{O(1/\epsilon \log(1/\epsilon))}}$ since $d = O(1/\epsilon \log(1/\epsilon))$. Recall that the reduction runs in time $2^{O(n^\epsilon)}$ on a SAT instance of size n.

C.2 Dense k-Subgraph

Construct a bipartite graph G as follows : the LHS consists of N vertices corresponding to the bits in the PCP proof. The RHS consists of M vertices corresponding to the tests of the PCP. Connect a LHS vertex (= bit) to a RHS vertex (= PCP test) if and only if the bit is accessed by the test. Assume that $N = \frac{M}{2^{5d}}$ (this can be achieved by duplicating vertices. The Dense Subgraph must take either *all* copies of a vertex or *none*). Set the size of the subgraph to $k = \left(\frac{1}{2^{5d+1}} + \frac{3}{4}\frac{1}{2^{d-1}}\right) M$.

Completeness

By Theorem 1.9, there is a set Π_0 of $\frac{N}{2}$ bits such that at least $(1 - O(1/d))\frac{1}{2^{d-1}} \geq \frac{3}{4}\frac{1}{2^{d-1}}$ fraction of the tests have all their d queries in Π_0. This gives an induced subgraph of G on

$$\frac{N}{2} + \frac{3}{4}\frac{1}{2^{d-1}}M = \left(\frac{1}{2^{5d+1}} + \frac{3}{4}\frac{1}{2^{d-1}}\right) M$$

vertices that has at least $(\frac{3}{4}\frac{1}{2^{d-1}})dM$ edges.

Soundness

Take any induced subgraph of G on $(\frac{1}{2^{5d+1}} + \frac{3}{4}\frac{1}{2^{d-1}})M$ vertices.

Case i : If the number of LHS vertices is $\geq (\frac{1}{2} + \frac{1}{2^{5d}})N$, then the number of RHS vertices is at most

$$\left(\frac{1}{2^{5d+1}} + \frac{3}{4}\frac{1}{2^{d-1}}\right) M - \left(\frac{1}{2} + \frac{1}{2^{5d}}\right) N = \left(\frac{3}{4}\frac{1}{2^{d-1}} - \frac{1}{2^{10d}}\right) M$$

Hence, the number of edges in the induced subgraph is at most $(\frac{3}{4}\frac{1}{2^{d-1}} - \frac{1}{2^{10d}})dM$.

Case ii : So assume that the number of LHS vertices is $\leq (\frac{1}{2} + \frac{1}{2^{5d}})N$ (assume w.l.o.g. that the number is exactly this). Call these vertices Π'_*. The number of RHS vertices is at most $(\frac{1}{2^{5d+1}} + \frac{3}{4}\frac{1}{2^{d-1}})M$.

Now we bound the fraction of tests all of whose queries are in Π'_*. Let Π_* be any subset of Π'_* of size $\frac{N}{2}$. By Theorem 1.9, the fraction of tests with all queries in Π_* is bounded by $\frac{1}{2^d} + \frac{1}{2^{20d}}$. The fraction of tests with at least one query in $\Pi'_* \setminus \Pi_*$ is bounded by $\frac{d}{2^{5d}}$ since the queries are uniformly distributed and the probability that any single query is in $\Pi'_* \setminus \Pi_*$ equals the fractional size of this set which is at most $\frac{1}{2^{5d}}$.

Therefore, the number of tests all of whose queries are in Π'_* is at most $(\frac{1}{2^d} + \frac{1}{2^{20d}} + \frac{d}{2^{5d}})M$. Hence the number of edges in the induced subgraph is at most

$$d\left(\frac{1}{2^d} + \frac{1}{2^{20d}} + \frac{d}{2^{5d}}\right)M + (d-1)\left((\frac{1}{2^{5d+1}} + \frac{3}{4}\frac{1}{2^{d-1}}) - (\frac{1}{2^d} + \frac{1}{2^{20d}} + \frac{d}{2^{5d}})\right)M =$$

$$dM\left((\frac{3}{4}\frac{1}{2^{d-1}} + \frac{1}{2^{5d+1}}) - \frac{1}{d}(\frac{1}{2^{5d+1}} + \frac{3}{4}\frac{1}{2^{d-1}}) + \frac{1}{d}(\frac{1}{2^d} + \frac{1}{2^{20d}} + \frac{d}{2^{5d}})\right) =$$

$$dM\left(\frac{3}{4}\frac{1}{2^{d-1}} - \frac{1}{d \cdot 2^{d+1}} + (\text{low order terms})\right)$$

In either case, the number of edges in the induced subgraph is at most $(\frac{3}{4}\frac{1}{2^{d-1}} - \frac{1}{2^{10d}})dM$.

Inapproximability Factor

Clearly, the inapproximability factor is

$$\frac{\frac{3}{4}\frac{1}{2^{d-1}}}{\frac{3}{4}\frac{1}{2^{d-1}} - \frac{1}{2^{10d}}}$$

which is $\approx 1 + \frac{4 \cdot 2^{d-1}}{3 \cdot 2^{10d}} = 1 + \frac{1}{2^{O(1/\epsilon \log(1/\epsilon))}}$ since $d = O(1/\epsilon \log(1/\epsilon))$.

C.3 Bipartite Clique

Construct a bipartite graph G as follows : the LHS consists of N vertices corresponding to the bits in the PCP proof. The RHS consists of M vertices corresponding to the tests of the PCP. Connect a LHS vertex (= bit) to a RHS vertex (= PCP test) if and only if the bit is *not* accessed by the test. Assume that $\frac{N}{2} = (\frac{3}{4}\frac{1}{2^{d-1}})M$ (this can be achieved by duplicating vertices. Any maximal Bipartite Clique must take either *all* copies of a vertex or *none*).

Completeness

We will show that there is a bipartite clique of size $\frac{N}{2}$. Theorem 1.9 shows that the set of 1-bits Π_1 in the proof is such that a fraction $(1 - O(1/d))\frac{1}{2^{d-1}} \geq \frac{3}{4}\frac{1}{2^{d-1}}$ of tests *do not* access any query from Π_1. Let $\mathcal{T}_1$ denote the set of all such tests with $|\mathcal{T}_1| = (\frac{3}{4}\frac{1}{2^{d-1}})M = \frac{N}{2}$. Clearly, Π_1 and $\mathcal{T}_1$ give a $\frac{N}{2}$ sized bipartite clique.

Soundness

We will show that there is no bipartite clique of size $(1 - \frac{1}{2^{5d}})\frac{N}{2}$. Assume on the contrary that there is such a clique and let Π_* be the LHS and $\mathcal{T}_*$ be the RHS of this clique. By hypothesis,

$$|\Pi_*| = (1 - \frac{1}{2^{5d}})\frac{N}{2} \qquad \text{and} \qquad |\mathcal{T}_*| = (1 - \frac{1}{2^{5d}})\frac{N}{2} = (1 - \frac{1}{2^{5d}})\frac{3}{4}\frac{1}{2^{d-1}}M$$

Clearly, $|\overline{\Pi}_*| = (1 + \frac{1}{2^{5d}})\frac{N}{2}$ and $\mathcal{T}_*$ is a set of tests all of whose queries fall in $\overline{\Pi}_*$. But this is a contradiction since by Theorem 1.9, the number of such tests is bounded by $(\frac{1}{2^d} + \frac{1}{2^{20d}} + \frac{d}{2^{5d}})M$. The calculation is exactly the same as in the Case (ii) of the soundness analysis for the Dense k-Subgraph problem.

Inapproximability Factor

Clearly, the inapproximability factor is

$$\frac{1}{1-\frac{1}{2^{5d}}} \approx 1+\frac{1}{2^{5d}} = 1+\frac{1}{2^{O(1/\epsilon\log(1/\epsilon))}}$$

D Boosting Hardness Factor for Bipartite Clique

In this section, we prove Theorem 1.2 showing that Bipartite Clique is hard to approximate within a polynomial factor. We mimic the proof of Berman and Schnitger [10] who prove a similar boosting result for the Clique problem. The boosting consists of taking a random induced subgraph of a product graph.

Definition D.1 *For a bipartite graph $G(V,W,E)$ with $|V|=|W|=n$, let $OPT_{BC}(G)$ denote a maximum integer t such that the complete bipartite graph $K_{t,t}$ is a subgraph of G.*

Definition D.2 *For a bipartite graph $G(V,W,E)$ with $|V|=|W|=n$, and an integer k, let $G^k(V',W',E')$ denote the product graph defined as follows :*

- $V'=V^k, W'=W^k$. *Thus* $|V'|=|W'|=n^k$.
- $((v_1,v_2\ldots,v_k),(w_1,w_2,\ldots,w_k))\in E' \iff \forall i,j,\ \ 1\le i,j\le k,\ (v_i,w_j)\in E$

Lemma D.3 *Let $G(V,W,E)$ be a bipartite graph with $|V|=|W|=n$ such that $OPT_{BC}(G)\ge \alpha n$. Let $H(V_*,W_*,E_*)$ be a random induced subgraph of $G^k(V',W',E')$ where $|V_*|=|W_*|=O(n^2/\alpha^k)$. Then w.h.p., we have $OPT_{BC}(H)\ge \frac{1}{2}\alpha^k|V_*|$.*

Proof: Let $V_1\subseteq V,\ W_1\subseteq W$ be such that $|V_1|=|W_1|=\alpha n$ and the subgraph of G induced on V_1,W_1 is a complete bipartite graph. Clearly, the subgraph of G^k induced on V_1^k,W_1^k is a complete bipartite graph.

Now V_* is a random subset of V^k of size $O(n^2/\alpha^k)$ and $|V^k|=n^k$, $|V_1^k|=\alpha^k n^k$. We will show that w.h.p. $|V_*\cap V_1^k|\ge\frac{1}{2}\alpha^k|V_*|$. We may assume that vertices in V_* are picked randomly and independently from V^k. For every vertex in V_*, the probability that it belongs to V_1^k is α^k. Thus the expected size of $|V_*\cap V_1^k|$ is $\alpha^k|V_*|$. Using Chernoff bounds,

$$\Pr\left[|V_*\cap V_1^k|\le\frac{1}{2}\alpha^k|V_*|\right]\ \le\ 2^{-\Omega(\alpha^k|V_*|)}\le 2^{-\Omega(n^2)}$$

Similarly w.h.p., $|W_*\cap W_1^k|\ge\frac{1}{2}\alpha^k|W_*|$. Clearly, the induced subgraph of H on $V_*\cap V_1^k, W_*\cap W_1^k$ is a complete bipartite graph and this proves the lemma. ∎

Lemma D.4 *Let $G(V,W,E)$ be a bipartite graph with $|V|=|W|=n$ such that $OPT_{BC}(G)\le\beta n$. Let $H(V_*,W_*,E_*)$ be a random induced subgraph of $G^k(V',W',E')$ where $|V_*|=|W_*|=O(n^2/\beta^k)$. Then w.h.p., we have $OPT_{BC}(H)\le 2\beta^k|V_*|$.*

Proof: It suffices to prove that w.h.p., for *every* complete bipartite subgraph $(\tilde V,\tilde W)$ of G^k, we have either $|V_*\cap\tilde V|\le 2\beta^k|V_*|$ or $|W_*\cap\tilde W|\le 2\beta^k|W_*|$.

In fact, it suffices to show this statement for every *maximal* complete bipartite subgraph of G^k. It is easily seen that a maximal complete bipartite subgraph of G^k must be of the form (V_1^k,W_1^k)

where (V_1, W_1) is a complete bipartite subgraph of G. In particular, the number of maximal complete bipartite subgraphs of G^k is bounded by 2^{2n}, and for any such subgraph (V_1^k, W_1^k), either $|V_1^k| \leq \beta^k n^k$ or $|W_1^k| \leq \beta^k n^k$.

Fix any maximal complete bipartite subgraph of G^k, induced on (V_1^k, W_1^k) and assume w.l.o.g. that $|V_1^k| \leq \beta^k n^k$. We may assume that vertices in V_* are picked randomly and independently from V'. For every vertex in V_*, the probability that it belongs to V_1^k is at most β^k. Thus the expected size of $|V_* \cap V_1^k|$ is at most $\beta^k |V_*|$. Using Chernoff bounds,

$$\Pr\left[|V_* \cap V_1^k| \geq 2\beta^k |V_*|\right] \;\leq\; 2^{-\Omega(\beta^k |V_*|)} \leq 2^{-\Omega(n^2)}$$

Taking union bound over all possible (at most 2^{2n} many) maximal complete bipartite subgraphs of G^k proves the lemma. ∎

D.1 Proof of Theorem 1.2

Now we prove Theorem 1.2. Fix $\epsilon > 0$ to be arbitrarily small constant. Let $G(V, W, E)$ be the bipartite graph given by the reduction in Appendix C.3. By adding dummy vertices to LHS, we may assume that $|V| = |W| = M$. If the size of the original SAT instance is n, then note that $M = 2^{n^\epsilon}$ and for integer $d = O(1/\epsilon \log(1/\epsilon))$, the graph G satisfies :

1. If the SAT instance is a YES instance, then $OPT_{BC}(G) \geq \alpha M$ where $\alpha = \frac{3}{4}\frac{1}{2^{d-1}}$
2. If the SAT instance is a NO instance, then $OPT_{BC}(G) \leq \beta M$ where $\beta = (1 - \frac{1}{2^{5d}})\alpha$.

Let $k = 2^{5d} \log M$ and let H be a random induced subgraph of G^k of size $S = O(M^2/\beta^k)$. Applying Lemma D.3 and D.4, we see that w.h.p.,

1. If the SAT instance is a YES instance, then $OPT_{BC}(H) \geq \frac{1}{2}\alpha^k S$.
2. If the SAT instance is a NO instance, then $OPT_{BC}(H) \leq 2\beta^k S$.

Thus the hardness factor is

$$\frac{1}{4}(\alpha/\beta)^k \geq \frac{1}{4}\left(1 + \frac{1}{2^{5d}}\right)^{2^{5d} \log M} \geq 2^{\log M} = M$$

. On the other hand,

$$S = O(M^2/\beta^k) \leq M^2 2^{dk} = M^{2+d2^{5d}} = 2^{(2+d2^{5d})n^\epsilon}$$

Thus the hardness factor can be expressed as $S^{\frac{1}{2+d2^{5d}}}$. Hence, assuming that SAT $\notin$ BPTIME(2^{n^ϵ}), we conclude that it is hard to approximate Bipartite Clique on graph of size S within factor $S^{\epsilon'}$ for $\epsilon' = \frac{1}{2^{O(1/\epsilon \log(1/\epsilon))}}$.