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

Subhash Khot
khot@cc.gatech.edu †

Abstract

Assuming that NP $\not\subseteq \cap_{\epsilon>0}$ BPTIME($2^{n^{\epsilon}}$), we show that Graph Min-Bisection, Densest Subgraph and Bipartite Clique have no 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 some precise sense.*

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

1. Introduction

In this paper, we study the following three notorious problems : Graph Min-Bisection, Densest Subgraph and Bipartite Clique. The best known algorithms for these problems have approximation ratios $O(\log^2 n)$ (see [13]), $O(n^{1/3})$ (see [12]) and $n/(\log n)^{O(1)}$ (folklore) respectively whereas no inapproximability results were known. Ruling out a PTAS (i.e. $(1+\delta)$-approximation for all $\delta > 0$) for these problems is considered a major open question in inapproximability theory.

Recently, Feige showed [9] that these problems have no PTAS assuming a hypothesis about average-case hardness of Random 3SAT. In this paper, we rule out a PTAS for these problems assuming only a standard assumption that NP $\not\subseteq \cap_{\epsilon>0}$ BPTIME($2^{n^{\epsilon}}$). We give a (quite complicated) reduction from the Minimum Distance of Linear Code (MDC) problem. The inapproximability result for MDC has been obtained by Dumer, Micciancio and Sudan [8].

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''.

Densest Subgraph : Given a graph $G(V,E)$ and a parameter t, find a subset $V' \subseteq V$ of size t so as to maximize the number of edges whose both endpoints are in V'.

Bipartite Clique : Given a bipartite graph $G(U,V,E)$, maximize t such that there exist $U' \subseteq U$ and $V' \subseteq V$, each of size t and the subgraph of G induced on the set of vertices $U' \cup V'$ is a complete bipartite graph.

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

---

∗ Full version of the paper is available at http://www.cs.princeton.edu/∼khot/publications.html.

† Georgia Institute of Technology, Atlanta GA-30332 and Institute for Advanced Study, Princeton, NJ-08544. Email: khot@math.ias.edu. *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.*

a linear code, the problem is to find the (relative) minimum distance of the code. Let $OPT(\mathbf{A})$ denote the optimum value.

Remark 1.1 *All instances of MDC in this paper have the following properties : (1) The columns of* $\mathbf{A}$ *are linearly 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 [8] 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 (see the full version for a proof). It is the starting point for our reduction.

Theorem 1.2 *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,* $N \leq n^{CK}$*.*

Remark 1.3 *Dumer et al.'s reduction and therefore reduction in Theorem 1.2 is randomized (the completeness and soundness properties hold w.h.p.). The PCPs in this paper are constructed from Theorem 1.2 and their construction is randomized too, though we do not explicitly state so.*

The main result in this paper is :

Theorem 1.4 *Let* $\epsilon > 0$ *be an arbitrarily small constant. If there is a PTAS for Graph Min-Bisection, Densest Subgraph or Bipartite Clique, then SAT has a (probabilistic) algorithm that runs in time* 2^{n^ϵ} *on an instance of size* n*.*

Remark 1.5 *The inapproximability factor is* $1 + \frac{1}{2^{O(1/\epsilon \log(1/\epsilon))}}$ *which approaches* 1 *as* ϵ *tends to* 0*. Thus we have a (strange) trade-off between the quality of the inapproximability result and the intractability assumption that SAT has no* 2^{n^ϵ} *time algorithm.*

The inapproximability factor for Bipartite Clique can be *boosted* via derandomized graph products (as in [2]). This gives the following theorem (see Feige and Kogan's [11] paper for weaker results, with no need to design new PCP) :

Theorem 1.6 *There exists an absolute constant* $\delta_{BC} > 0$ *such that the following holds : assuming* NP $\not\subseteq \cap_{\epsilon>0}$ BPTIME(2^{n^ϵ})*, there is no polynomial time algorithm that approximates Bipartite Clique on an instance of size* N *within factor* $N^{\delta_{BC}}$*.*

2. 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 1.4. In Section 2.1, we explain why the known PCPs are not quasi-random and in Section 2.2, 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. See for instance, Holmerin's [16] 4-query PCP (or Håstad's PCP for MAX-CUT [15]). This PCP 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 2.1 *(***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 properties of both the YES and NO cases. However, by abuse of terminology, we will often refer only to the property in the NO case.

The following theorem states the main PCP construction in the paper.

Theorem 2.2 *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*
$$\mathrm{Pr}_Q[Q \subseteq \Pi_0] \geq (1 - O(1/d))\,\frac{1}{2^{d-1}}$$
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*
$$\frac{1}{2^d} - \frac{1}{2^{20d}} \;\leq\; \mathrm{Pr}_Q[Q \subseteq \Pi_*] \;\leq\; \frac{1}{2^d} + \frac{1}{2^{20d}}$$

Note that Theorem 2.2 says nothing about the test the verifier performs on the query bits or the completeness/soundness probabilities. When we actually construct the PCP, we will use a homogeneous linear test on query bits.

The inapproximability results for Graph Min-Bisection, Densest Subgraph and Bipartite Clique follow easily from the PCP in Theorem 2.2 (see Section B). The reductions follow the same outline as Feige's reductions [9].

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

2.1. Why Known PCPs are not Quasi-random

Consider Håstad's 3-query PCP [15] 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 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 2.1). 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 ? The reason is the following : 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 stability.

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 [3] and Long Code Testing [6]). In particular, the verifier accesses less than q blocks and therefore fails to be quasi-random. Let us formulate this as an observation :

Fact 2.3 *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).

2.2. 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 ([4], [3]). 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. [7]). 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 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 2.4 *Definition 2.1 requires that the query pattern of the PCP verifier is different in 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 Outer and Modified Outer Verifiers (in both cases, the verifier picks random points on randomly selected line). However, (quite magically) the difference manifests itself at the Inner Verifier level.*

3. Comparison with Feige's Hypothesis

Feige [9] 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, Densest 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 implying the desired inapproximability results.

4. Overview of the Paper

The main result in this paper is Theorem 2.2. The proof of this result is quite lengthy, involving a sequence of reductions (see Fig. 1).

We introduce a new problem called Homogeneous Algebraic CSP. The proof can be divided into two parts. First we reduce MDC to HomAlgCSP. 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. As usual, the verifier consists of an Outer Verifier and the Inner Verifier. We actually need to modify the Outer Verifier before constructing the Inner Verifier.

For the lack of space, we omit all proofs and give only a high level outline of our reduction (see next section). Appendix A defines the HomAlgCSP problem and states the theorem describing the reduction from MDC to HomAlgCSP. Appendix B describes the reduction from Quasi-random PCP to Graph Min-Bisection, ruling out PTAS for the problem.

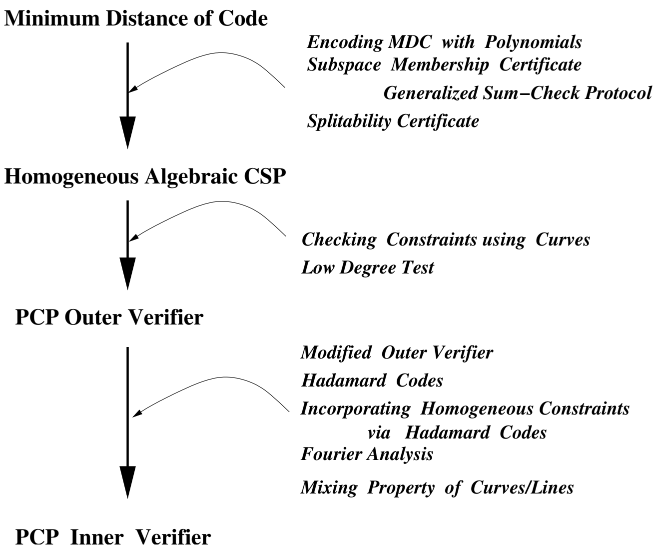

Figure 1. Overview of Full Reduction

5. 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.

5.1. MDC to Homogeneous Algebraic CSP

HomAlgCSP is the following problem (see Definition A.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 A.3, HomAlgCSP is a very similar problem to MDC. A reduction from MDC to HomAlgCSP transforms (see Theorem A.4) 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 $\overline{p} \in \mathcal{P}$, 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 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 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 [10] 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.

5.2. 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 [5].

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*. 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 the curves as well and therefore, the Outer Verifier is quasi-random.

5.3. Outer Verifier to Modified Outer Verifier

The purpose of the Modified Outer Verifier is exactly as explained in Section 2.1 and 2.2. It is the analogue of reading two bits from block B_j in Håstad's 3-query PCP. The Modified Outer Verifier introduces redundancy 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 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.

5.4. 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 there is a way of *decoding* the proof so that the Outer Verifier accepts the decoded proof 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) :

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. Hadamard Code is *splitable*, meaning 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 give a strategy for the Outer Verifier which 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 2.4, 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).

6. Conclusion

We have ruled out PTAS for three notorious problems, namely, Graph Min-Bisection, Densest 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 [17]. To the best of our knowledge, it was the first paper that employed a combination of the algebraic techniques 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 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 (and interesting) to prove inapproximability results assuming sufficiently strong hardness results for the Shortest Vector Problem, the analogue of MDC for integer lattices.

7. 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. Many thanks to anonymous referees for their excellent comments.

References

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

[2] N. Alon, U. Feige, A. Wigderson, D. Zuckerman. Derandomized graph products. *Computational Complexity*, 5, pages 60–75, 1995.

[3] 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.

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

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

[6] M. Bellare, O. Goldreich and M. Sudan. Free bits, PCPs and non-Approximability – towards tight results. *Siam Journal of Computing*, Vol. 27, No. 3, pages 804–915, 1998.

[7] E. Ben-Sasson, O. Goldreich, P. Harsha, M. Sudan and S. Vadhan. Robust PCPs of proximity, shorter PCPs and applications to coding. To appear in *Proc. 36th ACM Symposium on Theory of Computing*, 2004.

[8] I. Dumer, D. Micciancio and M. Sudan. Hardness of Approximating the Minimum Distance of a Linear Code. In *Proc. 40th IEEE Symposium on Foundations of Computer Science*, 1999.

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

[10] 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):268–292, 1996.

[11] 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.

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

[13] U. Feige and R. Krauthgamer. A polylogarithmic approximation of the minimum bisection. In *Proc. 41st IEEE Symposium on Foundations of Computer Science*, pages 105–115, 2000.

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

[15] J. Håstad. Some optimal inapproximability results. In *Proc. 29th ACM Symposium on Theory of Computing*, pages 1–10, 1997.

[16] J. Holmerin. Vertex cover on 4-uniform hypergraphs is hard to approximate within $2 - \epsilon$. In *Proc. 34th ACM Symposium on Theory of Computing*, 2002.

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

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

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

A. Homogeneous Algebraic CSP

Definition A.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*

$$\sum_{i=1}^{k} \gamma_{ij} f(\overline{p_i}) = 0 \ \text{for } j = 1, 2, \ldots$$

$$\text{where } \overline{p_i} \in \mathbb{F}^m \text{ and } \gamma_{ij} \in \mathbb{F}$$

   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. *(Clearly) $\mathcal{C}$ 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 A.2 *The HomAlgCSP problem is similar to the Algebraic CSPs defined by Harsha and Sudan [14]. However, we require the constraints in HomAlgCSP to be homogeneous linear (and forbid the zero polynomial as a candidate solution). Harsha and Sudan allow arbitrary algebraic constraints.*

Remark A.3 *HomAlgCSP and MDC are similar problems (the former is a maximization problem and the latter is a minimization problem). 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.*

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

The following theorem states the reduction from MDC to HomAlgCSP.

Theorem A.4 *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 $d = \frac{20}{\epsilon}$ 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*

$$1 - OPT(\mathbf{A}) \leq OPT(\mathcal{A}) \leq 1 - OPT(\mathbf{A}) + O(\frac{dm}{|\mathbb{F}|})$$

B. Reduction from PCP to Graph Min-Bisection

In this section, we prove that the PCP construction given by Theorem 2.2 implies inapproximability result for Graph Min-Bisection. Results for Densest Subgraph and Bipartite Clique follow similarly. The reduction is similar to the one given by Feige [9] (correctness of his reduction however relies on his hypothesis about hardness of Random 3SAT).

Let N be the proof size and M be the total number of tests of the PCP verifier in Theorem 2.2. 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 2.2 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

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

Soundness We show that for any bisection of G, the number of crossing edges is at least the expression (1). 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 2.2, 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

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

Inapproximability Factor Clearly, the inapproximability factor is

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

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.