Recycling Queries in PCPs and in Linearity Tests

[Extended Abstract]

LUCA TREVISAN*

Abstract

We study query-efficient Probabilistically Checkable Proofs (PCPs) and linearity tests. We focus on the number of *amortized query bits*. A testing algorithm uses $\bar{q}$ amortized query bits if, for some constant k, it reads $\bar{q}k$ bits and has error probability at most 2^{-k}. The best known PCP construction for NP in this respect uses 3 amortized query bits [13]; at least one amortized query bit is necessary, unless $\mathsf{P} = \mathsf{NP}$ [5]. This parameter is a fairly natural one and has applications to proving non-approximability results for constraint satisfaction problems. Furthermore, a PCP characterization of NP with less than 2 amortized query bits implies a separation of the PCP model from the 2-Prover 1-Round model.

Our approach is to take an atomic verification procedure and then iterate it several times, saving queries by recycling them between different iterations of the atomic test.

We first apply this idea in order to develop query-efficient *linearity tests*. Linearity testing is a problem closely related to testing the *Long Code* and making PCP constructions. It is also a significant combinatorial problem still lacking tight characterizations, except for the case of three queries [4]. The best known linearity test uses 3 amortized query bits [4]; a different one achieves 1 amortized free bit (a different parameter related to the Max Clique problem) but uses an unbounded number of amortized query bits [5]. We develop a general analysis technique and a linearity test achieving simultaneously amortized query complexity 1.5 and amortized free bit complexity $.5$. This test answers an open question raised by Bellare, Goldreich and Sudan.

We then show how to adapt a weaker result to the PCP setting, and we obtain a PCP for NP that makes 5 queries and has error probability $1/4$, so that its amortized query complexity is 2.5.

*MIT Laboratory for Computer Science, Room NE43-371, 545 Technology Square, Cambridge, MA 02139, USA. luca@theory.lcs.mit.edu.

1 Introduction

The PCP characterization of NP [2, 1], or *the PCP Theorem*, is a major achievement of complexity theory and a powerful tool for proving hardness of approximation for optimization problems. Informally stated, the PCP Theorem says that there is a way of encoding proofs of NP statements so that such encoded proofs can be probabilistically tested using logarithmic randomness and looking only at a *constant* number of bits of the encoding. The encoding of a correct proof will pass the test with probability one[1] while, if the statement is wrong, any adversarially chosen string submitted to the test will be accepted only with small probability, say at most $1/2$. The latter probability is the *error probability*, or *soundness* of the test. The algorithm implementing the test is usually called the *verifier*. After the appearance of the PCP Theorem, there has been a lot of effort devoted to finding *quantitative* strengthenings of it, with improved tradeoffs between the different parameters arising in the proof-checking procedure. One of the main motivations for this line of research has been the goal of getting improved, and eventually tight, non-approximability results for certain optimization problems.

PCP PARAMETERS. Four main parameters have been considered for their applications to proving hardness of approximation: the number of *query bits*, the number of *free bits* and their *amortized* versions (see also [3] for a survey on recent PCP results and for the role of different parameters.) The number of query bits is the number of bits of the proof that are accessed by the verifier. We say that a verifier uses f free bits if there is a subset of f queries such that for any possible outcome to these queries there is only one possible answer to the other queries that would make the verifier accept.[2] We

[1] In some cases, including the construction presented in this paper, there is the less restrictive assumption that correct proofs are accepted with probability at least $1-\varepsilon$ where ε can be made arbitrarily small independently of the other parameters of interest.

[2] A more general definition is given in [5].

say that a verifier uses $\bar{q}$ amortized query bits if, for some constant k, it makes $\bar{q}k$ queries and has error probability at most 2^{-k}; amortized free bits are defined similarly. The notion of amortization is due to [7, 5].

PCPs with small query complexity (and, subject to that, with low error) have applications to prove hardness of approximation for the Max 3SAT problem; a tight result, due to Håstad [13] is known in this context (the tightness is due to [14, 26].) Protocols with small free bit complexity (and, subject to that, with low error) imply hardness of approximation results for the Vertex Cover problem; a tight result is not known and is a major open question. The amortized free bit complexity parameter is related to the hardness of approximating Max Clique; in this case a tight result is known, due to Håstad [12], showing that NP can be characterized with ε amortized free bits for any $\varepsilon > 0$. The amortized query complexity parameter is related to the approximability of constraint satisfaction problems described below. A tight result is not known.

Max kCSP (for k-ary Constraint Satisfaction Problem) is the generalization of Max kSAT where each clause is allowed to be an arbitrary predicate over (at most) k boolean variables. Max k CSP was implicit in [18], has been named in [15] and then studied e.g. in [10, 22, 23, 25, 16, 26]. This problem is known to be 2-approximable for $k = 3$ [26] and 2^{k-1}-approximable for $k \geq 4$ [22]; on the negative side it is NP-hard to approximate within a factor $2^{\lfloor k/3 \rfloor} - \varepsilon$ for any k [13]. A PCP for NP using $\bar{q}$ amortized query bits implies the NP-hardness of approximation of Max kCSP within $2^{\lfloor k/\bar{q} \rfloor} - \varepsilon$ for sufficiently large k.

An additional motivation for the study of PCPs with low amortized query complexity is the fact that NP cannot be characterized with 2-Provers 1-Round having amortized query complexity smaller than 2 (we take the query complexity of a 2-Prover 1-Round protocol as the sum of the answer sizes) unless P = NP [20]. Thus, a characterization of NP in terms of a PCP with 1.99 (say) amortized query bits implies a separation between the PCP and the MIP model, a question asked in [6] and [5]. The best previous result was a characterization of NP with $3 + \varepsilon$ amortized query bits [13]. There cannot be a characterization with 1 amortized query bit unless P = NP [5, 22]. We believe that $1 + \varepsilon$ is the right answer: our goal is to make progress in this direction.

TESTING PROOFS. The standard method of getting good PCPs is by *composition*, a paradigm introduced in [2] and then used e.g. in [1, 6, 7, 5, 11, 12, 13]. In the most recent constructions [5, 12, 13] one starts with an *outer* protocol given by Raz [19], a 2-Prover 1-Round protocol with perfect completeness, constant answer size and small soundness, and then one tries to improve on the parameter of interest, in our case the number of amortized query bits. To this aim, one defines an appropriate error correcting code and a testing procedure, called the *inner* protocol. Given two strings, the inner protocol checks whether or not they are the encodings of two possible consistent answers of the provers of the outer protocol. There is a way to compose the two protocols so that the "composed verifier" V inherits the query complexity and the error probability of the inner verifier V^{inner}, and even if V^{inner} required a polynomial or even exponential amount of random bits, the composed verifier V will only use logarithmic randomness. Note that since the outer verifier given in Raz [19] is essentially the best possible, we only have to care about the inner one, i.e. on the definition of the code and on checking codewords and consistency.

THE LONG CODE. A first series of works [1, 6, 7] developed inner verifiers using the Hadamard Code, and the essential testing problem was the "linearity test" problem, first investigated by Blum, Luby and Rubinfeld [8]. More recently, the Long Code was introduced in [5] and used in [5, 11, 13]. The Long Code of a string $a \in \{0,1\}^n$ is a string A of length 2^{2^n} indexed by the functions $f : \{0,1\}^n \to \{0,1\}$, and such that $A(f) = f(a)$ for any function f (we give an alternate equivalent definition in Section 2.) It should be noted that the linearity test is still relevant to the analysis of the Long Code: in [5] the linearity testing was a subroutine of the Long Code test; in [13] the analysis of the Long Code is based on a "perturbed" linearity test and its analysis follows the lines (alas with several complications) of the analysis of the linearity test given in [4]. Thus, it appears that techniques and ideas developed to test linearity are, with appropriate extensions, useful in testing the Long Code and eventually getting PCP constructions. This motivates us to get, for starters, a good linearity tester with low amortized query complexity.

TESTING LINEARITY: THE PROBLEM AND PREVIOUS RESULTS. In the linearity test problem we are given oracle access to a function $f : \{0,1\}^n \to \{0,1\}$ and we want to determine whether f is linear, i.e. $f(a) \oplus f(b) = f(a \oplus b)$ for any $a, b \in \{0,1\}^n$, or if f is very far from every linear function, where the distance $\mathbf{Dist}(f, g)$ between two functions is the fraction of points where they disagree (for a function f and a family $\mathcal{F}$ we also use the notation $\mathbf{Dist}(f, \mathcal{F}) = \min_{g \in \mathcal{F}} \mathbf{Dist}(f, g)$.) Observe that the set of linear n-ary functions, denoted LIN, is a Hadamard Code[3] with codewords of length 2^n; recall that this code has minimum distance $1/2$. We want a randomized test that always accepts linear functions and that accepts with very small probability functions that are far from the set of linear functions. The error probability of the test is the probability that it accepts a function that is far; clearly this definition depends on what do we mean by "far". Bellare, Goldreich and Sudan [5] call "codeword test" the problem where "far" functions are at distance at least $1/4$ from the code. This definition is motivated by the fact that "non-far" functions have unique decoding. Defining "far" to be the set of functions at distance at least $1/2 - \varepsilon$ from the set of linear functions is still good (this is called "relaxed codeword testing" in [5]) since a function that is at distance at most $1/2 - \varepsilon$ from some linear func-

[3] In fact, it is a first-order Reed-Muller Code.

tion is at this distance from at most $1/4\varepsilon^2$ other linear functions, so we can still use bounded distance decoding. We obtain the same bound $1/4\varepsilon^2$ if we say that a function is not "far" whenever it is at distance at most $1/2 - \varepsilon$ from either LIN or from the set of functions that are the bitwise complement of linear functions, denoted $\overline{\mathsf{LIN}}$. This is the notion of "relaxed codeword test" that we adopt in this paper. The linearity test of Blum, Luby and Rubinfeld [8] (henceforth called the *BLR test*) makes three queries: it picks random $a, b \in \{0,1\}^n$ and accepts iff $f(a) \oplus f(b) = f(a \oplus b)$. The BLR analysis has been improved in [9, 4, 17]; in [4] it has been shown that the acceptance probability of this test is at most $1 - \mathbf{Dist}(f, \mathsf{LIN})$, that is $1/2 + \varepsilon$ when f is at distance $1/2 - \varepsilon$ from the set of linear functions. Thus, the BLR test has amortized query complexity essentially 3. No better test is known with respect to amortized query complexity. With respect to amortized free bit complexity, in [5] there is a test that has amortized free bit complexity essentially 1 and that distinguishes linear functions from functions at distance $1/4$ from the set of linear functions. There is a lower bound showing that this result is tight. Bellare et al. asked whether this lower bound can be beaten using relaxed codeword tests.

TESTING LINEARITY: OUR RESULTS. We develop a general framework to define and analyse linearity tests that execute several instances of the BLR test and recycle query bits between different executions.

In order to analyse the acceptance probability of such tests, we describe the way bits are recycled in terms of graphs. To every graph G with k vertices and m edges we associate a test that we call $\mathsf{LinTestGraph}(G, \cdot)$. Such a test queries $k + m$ bits, uses k free bits, and runs m instances of the BLR test. We show that the acceptance probability of $\mathsf{LinTestGraph}(G, \cdot)$ can be formulated in a way that is very convenient for Fourier analysis (Theorem 5). We then use Fourier analysis to obtain tight bounds on the error probability of three infinite families of tests. The use of Fourier analysis to study linearity tests was introduced in [4]. Håstad elected it to a fine art in his papers on testing the Long Code and on improved PCP constructions [11, 13]. Our results are as follows ($k \geq 1$ is any fixed positive integer):

1. The test associated with a path of length k has acceptance probability at most
$$(1 - \mathbf{Dist}(f, \mathsf{LIN} \cup \overline{\mathsf{LIN}}))^k,$$
with $2k + 1$ query bits and $k + 1$ free bits. Thus it uses roughly 2 amortized query bits and 1 amortized free bit.

2. The test associated with a star with k rays has acceptance probability at most
$$\frac{1}{2^k} + \frac{(2^k - 1)}{2^k}(1 - 2\mathbf{Dist}(f, \mathsf{LIN} \cup \overline{\mathsf{LIN}})).$$
Its (amortized) query and free bit complexity are as before.

3. The test associated with the graph $K_{2,k}$ has acceptance probability at most
$$\frac{1}{2^{2k}} + \frac{(2^{2k} - 1)}{2^{2k}}(1 - 2\mathbf{Dist}(f, \mathsf{LIN} \cup \overline{\mathsf{LIN}})),$$
it reads $3k + 2$ bits, and uses $2k + 2$ free bits. The amortized query complexity is roughly 1.5 and the amortized free bit complexity is roughly $.5$, beating the lower bound of [5].

The only tool of Fourier analysis that we use is Parseval's equality. In a sense that can be made formal, 1.5 amortized query bits are a natural limit to the techniques developed in this paper (this is discussed in Section 6.)

PCP: OUR RESULTS. All the results that we have for the linearity test extend to the analysis of *one* Long Codeword, using the random perturbation idea of [13]. Unfortunately, it is necessary to look at *two* alleged Long Codewords in order to build an inner verifier, and there is also a consistency test to be implemented. We are able to extend to the PCP setting the analysis of the linearity test based on the star with three vertices: this gives a PCP that makes five queries and has soundness $1/4$. The amortized query complexity of our PCP is thus 2.5. This also implies that Max 5CSP is hard to approximate within $4 - \varepsilon$ and that Max kCSP is hard to approximate within $2^{\lfloor .4k \rfloor} - \varepsilon$. (This result has been recently improved, see Section 6.)

OVERVIEW OF THE PAPER. We introduce basic definitions in Section 2. We describe graph-based linearity tests in Section 3 and present a first analysis of their acceptance probability. In Section 4 we present a Fourier analysis of three graph-based families of tests. Section 5 is devoted to our PCP construction. In Section 6 we discuss a recent improvement and some open questions.

2 Linear Functions, Fourier Analysis, and PCP

From now on boolean functions will be defined with values in $\{1, -1\}$ rather than $\{0, 1\}$. The association is that -1 stands for 1 (or true) and 1 stands for 0 (or false). Observe that multiplication in $\{1, -1\}$ acts as boolean xor in $\{0, 1\}$. For an integer k, we denote by $[k]$ the set $\{1, \ldots, k\}$. For two sets α and β we denote by $\alpha \Delta \beta = (\alpha \cup \beta) - (\alpha \cap \beta)$ their symmetric difference. Recall that Δ is commutative and associative.

We will often blur the difference between vectors in $\{1, -1\}^n$ and functions from $[n]$ to $\{1, -1\}$; for a vector $x \in \{1, -1\}^n$, its a-th entry ($a \in [n]$) is denoted by $x(a)$ and can be thought of as the evaluation of a function $x : [n] \to \{1, -1\}$ in the point a. If $\pi : [m] \to [n]$ and $x \in \{1, -1\}^n$, the vector $x \circ \pi \in \{1, -1\}^m$ is defined as $x \circ \pi(b) = x(\pi(b))$ for any $b \in [m]$.

Given two vectors x and y, their bit-wise product, denoted xy, is defined as $xy(a) = x(a)y(a)$.

Given two functions $f, g : \{1,-1\}^n \to \{1,-1\}$, their distance is the fraction of points where they disagree, that is

$$\mathbf{Dist}(f,g) = \frac{|\{x : f(x) \neq g(x)\}|}{2^n} = \mathbf{Pr}_x[f(x) \neq g(x)] \ .$$

We say that a function $f : \{1,-1\}^n \to \{1,-1\}$ is linear iff $f(x)f(y) = f(xy)$ for all $x, y \in \{1,-1\}^n$. There are 2^n linear functions. There is a linear function l_α for any set $\alpha \subseteq \{1, \ldots, n\}$; it is defined as

$$l_\alpha(x) = \prod_{a \in \alpha} x(a) \ .$$

By convention, we say that a product ranging over the empty set equals 1. We will be using three standard properties of linear functions, the fact that they are linear in α, linear in x and that they are equally often 1 and -1, except for the case of the function that is identically 1:

$$l_\alpha(x) l_\beta(x) = l_{\alpha \Delta \beta}(x) \ , \ l_\alpha(x) l_\alpha(y) = l_\alpha(xy) \ , \quad (1)$$

$$\mathop{\mathbf{E}}_x l_\alpha(x) = \begin{cases} 1 & \text{If } \alpha = \emptyset \\ 0 & \text{otherwise.} \end{cases} \quad (2)$$

We denote by LIN_n the set of linear functions of arity n: this set is a Hadamard Code with codewords of length 2^n. We also use the notation $\overline{\mathsf{LIN}}_n = \{f : -f \in \mathsf{LIN}\}$. We usually drop the subscript. It is useful to see a function $f : \{1,-1\}^n \to \{1,-1\}$ as a real-valued functions $f : \{1,-1\}^n \to \mathbf{R}$. The set of functions $f : \{1,-1\}^n \to \mathbf{R}$ is a vector space over the reals of dimension 2^n. We define a scalar product between functions.

$$f \cdot g = \frac{1}{2^n} \sum_{x \in \{1,-1\}^n} f(x) g(x) = \mathop{\mathbf{E}}_x [f(x) g(x)] \ .$$

For functions $f : \{1,-1\}^n \to \{1,-1\}$, the scalar product has a couple of alternative characterizations.

$$\begin{aligned} f \cdot g &= \mathbf{Pr}_x[f(x) = g(x)] - \mathbf{Pr}_x[f(x) \neq g(x)] \\ &= 1 - 2\mathbf{Dist}(f,g) \ . \end{aligned}$$

The set of linear functions is easily seen to form an orthonormal basis for the set of functions $f : \{1,-1\}^n \to \mathbf{R}$. This implies that for any function $f : \{1,-1\}^n \to \mathbf{R}$ we have

$$f(x) = \sum_\alpha \hat{f}_\alpha l_\alpha(x) \text{ where } \hat{f}_\alpha = f \cdot l_\alpha \ .$$

For a function $f : \{1,-1\}^n \to \{1,-1\}$, we also have several useful properties of the coefficients $\hat{f}_\alpha$, namely

- $-1 \leq \hat{f}_\alpha \leq 1$,
- $\hat{f}_\alpha = 1 - 2\mathbf{Dist}(f, l_\alpha)$,
- $|\hat{f}_\alpha| = 1 - 2\min\{\mathbf{Dist}(f, l_\alpha), \mathbf{Dist}(f, -l_\alpha)\}$,
- $\max_\alpha \hat{f}_\alpha = 1 - 2\mathbf{Dist}(f, \mathsf{LIN})$,
- $\max_\alpha |\hat{f}_\alpha| = 1 - 2\mathbf{Dist}(f, \mathsf{LIN} \cup \overline{\mathsf{LIN}})$.

We now state the only result from Fourier analysis that will be used in the rest of the paper.

Lemma 1 (Parseval's equality) *For any function* $f : \{1,-1\}^n \to \{1,-1\}$, $\sum_\alpha \hat{f}_\alpha^2 = 1$.

The Long Code is the set of linear functions whose support is a singleton, i.e. $\mathsf{LONG}_n = \{l_{\{a\}} : a \in [n]\}$. We say that $l_{\{a\}}$ is the Long Code of a. Thus, the Long Code is formed by n codewords of length 2^n. An alternate but equivalent definition was used in [5]. We think that our definition makes more explicit the relation between linear functions and the Long Code.

We give the definition of PCP parameters and classes. We follow the notation of [6, 5].

Definition 2 (Verifier) *A* verifier *V for a language L is a probabilistic polynomial time algorithm that receives an input x and has oracle access to a string P, that is supposed to represent a proof of the statement "$x \in L$". For a verifier V, an input x and a proof P, we denote by $\mathbf{Acc}(V, x, P)$ the probability over the random choices of V that V accepts x having oracle access to P.*

Definition 3 (PCP Classes) *For constants $0 \leq s \leq c \leq 1$ and for an integer q, we say that $L \in \mathbf{na}\mathsf{PCP}_{c,s}[\log, q]$ if there exists a verifier V for L that satisfies the following properties:*

- [NUMBER OF RANDOM BITS AND QUERIES.] *For any input x and any proof P, V tosses $O(\log |x|)$ random coins, where $|x|$ is the length of x, and makes at most q* non-adaptive *queries to P;*
- [COMPLETENESS.] *For any $x \in L$, there exists a P such that $\mathbf{Acc}(V, x, P) \geq c$;*
- [SOUNDNESS.] *For any $x \notin L$, for all P, $\mathbf{Acc}(V, x, P) \leq s$.*

We have the following connection with the Max kCSP problem.

Theorem 4 ([1]) *If $\mathsf{NP} = \mathbf{na}\mathsf{PCP}_{c,s}[\log, k]$ then it is NP-hard to approximate Max kCSP within $\frac{c}{s} - \varepsilon$ for any $\varepsilon > 0$.*

In [22] Theorem 4 was extended to the case of adaptive queries, we will not need such extension in this paper.

3 Graph-Based Tests and Their Acceptance Probability

The BLR test picks x_1 and x_2 independently and uniformly at random and tests whether $f(x_1)f(x_2) = f(x_1x_2)$ (equivalently, whether $f(x_1)f(x_2)f(x_1x_2) = 1$). In order to run several instances of the BLR test we will pick at random $x_1, \ldots, x_k$, and then, for some pairs (i,j), test whether $f(x_i)f(x_j)f(x_ix_j) = 1$; the total number of queries will be k plus the number m of tests that we performed. If we are using the same x_i more than once, then the number of queries is smaller than $3m$ and this may lead to an improvement in the amortized query complexity. The problem is that we have to show that re-using queries does not increase the error probability. A convenient way to represent a test that recycles queries is to think of it as a graph $G = ([k], E)$ where the k vertices are associated with the k elements $x_1, \ldots, x_k$ that we pick at random and the edges $(i,j) \in E$ are associated with the tests $f(x_i)f(x_j)f(x_ix_j)$ that we perform.

Formally, for a graph $G = ([k], E)$ we define the following test $\mathsf{LinTestGraph}(G, f)$.

$\mathsf{LinTestGraph}(G, f)$ Choose uniformly at random $x_1, \ldots, x_k \in \{1,-1\}^n$ **if** $f(x_i)f(x_j)f(x_ix_j) = 1$ for all $(i,j) \in E$ **then accept** **else reject**

$\mathsf{LinTestGraph}(([k], E), f)$ reads $k + |E|$ bits, k of which are free bits. We denote by $\mathbf{AccGraph}(G, f)$ the probability that $\mathsf{LinTestGraph}(G, f)$ accepts. The acceptance probability of the test is expressed by the following formula.

Theorem 5

$$\mathbf{AccGraph}(([k], E), f) =$$

$$\frac{1}{2^{|E|}} \sum_{S \subseteq E} \mathop{\mathbf{E}}_{x_1,\ldots,x_k} \left[\prod_{(i,j)\in S} f(x_i)f(x_j)f(x_ix_j) \right] . \quad (3)$$

PROOF: We introduce the following operator $\wedge : \{1,-1\}^* \to \{0,1\}$:

$$x_1 \wedge \cdots \wedge x_k = \begin{cases} 1 & \text{If } x_1 = \cdots = x_k = 1 \\ 0 & \text{Otherwise.} \end{cases}$$

It is not hard to see that it holds

$$x_1 \wedge \cdots \wedge x_k = \left(\frac{1+x_1}{2}\right) \cdots \left(\frac{1+x_k}{2}\right) = \frac{1}{2^k} \sum_{S\subseteq[k]} \prod_{i\in S} x_i . \quad (4)$$

where the latter equality may e.g. be proved by induction on k. The reader may want to notice that the last term in Equation (4) is the Fourier expansion of the function $f(x_1, \ldots, x_k) = x_1 \wedge \cdots \wedge x_k$. It remains to observe that when $\mathsf{LinTestGraph}(([k], E), f)$ picks $x_1, \ldots, x_k$, it accepts if and only if

$$\bigwedge_{i,j} f(x_i)f(x_j)f(x_ix_j) = 1 ,$$

and thus

$$\begin{aligned} &\mathbf{AccGraph}(([k], E), f) \\ &= \mathop{\mathbf{E}}_{x_1,\ldots,x_k} \left[\bigwedge_{i,j} f(x_i)f(x_j)f(x_ix_j) \right] \\ &= \mathop{\mathbf{E}}_{x_1,\ldots,x_k} \left[\frac{1}{2^{|E|}} \sum_{S\subseteq E} \prod_{(i,j)\in S} f(x_i)f(x_j)f(x_ix_j) \right] \\ &= \frac{1}{2^{|E|}} \sum_{S\subseteq E} \mathop{\mathbf{E}}_{x_1,\ldots,x_k} \left[\prod_{(i,j)\in S} f(x_i)f(x_j)f(x_ix_j) \right] . \end{aligned}$$

□

4 Fourier Analysis of the Path-Test, the Star-Test and the $K_{2,k}$-Test

For any k, let $([k+1], P_k)$ be a path of length k where $P_k = \{(i, i+1) : i = 1, \ldots, k\}$. In order to study the test induced by such graph we have to study the expectation of products like the ones appearing in the left-hand side of Eq. (3).

Lemma 6 *For any function $f : \{1,-1\}^n \to \{1,-1\}$,*

$$\mathop{\mathbf{E}}_{x_1,\ldots,x_k} \left[\prod_{(i,j)\in P_k} f(x_i)f(x_j)f(x_ix_j) \right] = \sum_\alpha \hat{f}_\alpha^{k+2} .$$

PROOF: We first note that all the terms of the form $f(x_i)$ in the product cancel except for $f(x_1)$ and $f(x_{k+1})$. Thus the product can be rewritten as

$$f(x_1)f(x_1x_2)\cdots f(x_k, x_{k+1})f(x_{k+1}) . \quad (5)$$

We expand each function and distribute the products. We use the following expansions:

- $f(x_1) = \sum_{\alpha_0} \hat{f}_{\alpha_0} l_{\alpha_0}(x_1)$;
- $f(x_jx_{j+1}) = \sum_{\alpha_j} \hat{f}_{\alpha_j} l_{\alpha_j}(x_jx_{j+1})$;
- $f(x_{k+1}) = \sum_{\alpha_{k+1}} \hat{f}_{\alpha_{k+1}} l_{\alpha_{k+1}}(x_{k+1})$.

Using Equations (1), Expression (5) becames

$$\sum_{\alpha_0,\ldots,\alpha_{k+1}} \hat{f}_{\alpha_0} \cdots \hat{f}_{\alpha_{k+1}} l_{\alpha_0\Delta\alpha_1}(x_1) \cdots l_{\alpha_k\Delta\alpha_{k+1}}(x_{k+1}) . \quad (6)$$

When we take the average over the choices of $x_1, \ldots, x_{k+1}$, all the expressions $l_{\alpha_i\Delta\alpha_{i+1}}(x_{i+1})$ are independent random variables, whose average is zero unless $\alpha_i\Delta\alpha_{i+1} = \emptyset$. Thus, everything cancels except when $\alpha_0 = \alpha_1, \ldots, \alpha_k = \alpha_{k+1}$ so that the expression reduces to $\sum_{\alpha_0} \hat{f}_{\alpha_0}^{k+2}$. □

Lemma 7 *For any graph $([k+1], S)$, where $S \subseteq P_k$,*

$$\mathop{\mathbf{E}}_{x_1,\ldots,x_k} \left[\prod_{(i,j)\in S} f(x_i)f(x_j)f(x_ix_j) \right] \leq (\max_\alpha |\hat{f}_\alpha|)^{|S|} .$$

PROOF: The lemma is certainly true for $S = \emptyset$. Otherwise, S is a non-empty subset of a path, and so it is a collection of paths. Let us say that S is a collection of h paths, of length $l_1, \ldots, l_h$ respectively. Note that $|S| = l_1 + \ldots + l_h$. The outcomes of the BLR test in different subpaths are clearly independent random variables, so that the expectation of their product is equal to the product of the expectations. Applying Lemma 6 to each subpath we get

$$\mathop{\mathbf{E}}_{x_1,\ldots,x_k}\left[\prod_{(i,j)\in S} f(x_i)f(x_j)f(x_ix_j)\right] \le \prod_{i=1}^{h}\sum_{\alpha} \hat{f}^{l_i+2} .$$

Now, observe that, using Parseval's equality,

$$\sum_{\alpha} \hat{f}_{\alpha}^{l_i+2} \le (\max_{\alpha}|\hat{f}_{\alpha}|)^{l_i}\sum_{\alpha}\hat{f}_{\alpha}^2 = (\max_{\alpha}|\hat{f}_{\alpha}|)^{l_i} .$$

We thus have

$$\begin{aligned} &\mathop{\mathbf{E}}_{x_1,\ldots,x_k,x_{k+1}}\left[\prod_{(i,j)\in S} f(x_i)f(x_j)f(x_ix_j)\right] \\ \le\ & \prod_{i=1}^{h}(\max_{\alpha}|\hat{f}_{\alpha}|)^{l_i} = (\max_{\alpha}|\hat{f}_{\alpha}|)^{|S|} . \end{aligned}$$

□

The analysis of the Path-Test is now straightforward.

Theorem 8 $\mathbf{AccGraph}(([k+1], P_k), f) \le (1 - \mathbf{Dist}(f, \mathsf{LIN} \cup \overline{\mathsf{LIN}}))^k$.

PROOF: To save notation, let us define $\hat{f}_{\max} = \max_{\alpha}|\hat{f}_{\alpha}|$. Recall that $1 - \mathbf{Dist}(f, \mathsf{LIN} \cup \overline{\mathsf{LIN}}) = (1 + \hat{f}_{\max})/2$. Using Theorem 5, Lemma 7 and the Binomial Theorem, we get

$$\begin{aligned} &\mathbf{AccGraph}(([k+1], P_k), f) \\ \le\ & \frac{1}{2^k}\sum_{S\subseteq P_k}\hat{f}_{\max}^{|S|} \\ =\ & \frac{1}{2^k}\sum_{h=0}^{k}\binom{k}{h}\hat{f}_{\max}^{h} \\ =\ & \frac{1}{2^k}(1+\hat{f}_{\max})^k \\ =\ & (1 - \mathbf{Dist}(f, \mathsf{LIN} \cup \overline{\mathsf{LIN}}))^k . \end{aligned}$$

□

For any k, let $([k+1], S_k)$ be the star with k rays, i.e. the graph whose set of edges is $S_k = \{(i, k+1) : i = 1, \ldots, k\}$. We now analyse the associated test.

Lemma 9 *For any $f : \{1,-1\}^n \to \{1,-1\}$, and any $k \ge 1$*

$$\mathop{\mathbf{E}}_{x_1,\ldots,x_k,x_{k+1}}\left[\prod_{(i,j)\in S_k} f(x_i)f(x_j)f(x_ix_j)\right] \le \max_{\alpha}|\hat{f}_{\alpha}| .$$

PROOF: Omitted from this extended abstract. □

Theorem 10 $\mathbf{AccGraph}(([k+1], S_k), f) \le \frac{1}{2^k} + \frac{2^k-1}{2^k}(1 - 2\mathbf{Dist}(f, \mathsf{LIN}))$.

PROOF: From Theorem 5, Lemma 9, and the fact that a subgraph of a star is a star we have

$$\begin{aligned} &\mathbf{AccGraph}(([k+1], S_k), f) \\ =\ & \frac{1}{2^k} + \frac{1}{2^k}\sum_{S\subseteq S_k, S\neq\emptyset}\max_{\alpha}|\hat{f}_{\alpha}| \\ =\ & \frac{1}{2^k} + \frac{2^k-1}{2^k}(1 - 2\mathbf{Dist}(f, \mathsf{LIN} \cup \overline{\mathsf{LIN}})) . \end{aligned}$$

□

We finally consider the bipartite complete graph with components of size 2 and k, $K_{2,k} = ([k+2], B_k)$, where $B_k = \{(i, k+1) : i \in [k]\} \cup \{(i, k+2) : i \in [k]\}$.

Lemma 11 *For any graph $([k+2], S)$ where $S \subseteq B_k$ and $S \neq \emptyset$, it holds*

$$\mathop{\mathbf{E}}_{x_1,\ldots,x_{k+2}}\prod_{(i,j)\in S} f(x_i)f(x_j)f(x_ix_j) \le \max_{\alpha}|\hat{f}_{\alpha}| .$$

PROOF: We can assume without loss of generality that all the vertices $1, \ldots, k$ in $([k+2], S)$ have degree at least one, otherwise the analysis reduces to that of a smaller graph with such property. We can also assume that both $k+1$ and $k+2$ have degree at least one, otherwise the analysis reduces to the analysis of the star. Let $A \subseteq [k]$ be the set of vertices connected with $k+1$, B the set of vertices connected with $k+2$, and C the set of vertices connected with both $k+1$ and $k+2$. We have to consider the three cases arising when $k+1$ and $k+2$ have odd and even degree (there are four cases, but two of them are symmetric).

If both have odd degree, then we have to estimate the expression of Figure 1.
If $k+i$ has even degree ($i = 1, 2$), then $f(x_{k+i})$ does not appear at the beginning of the expression. We will give a complete analysis only for the expression of Figure 1. For the other cases we will just give the final result. We call α_1 and α_2 the Fourier coefficients used to expand $f(x_{k+1})$ and $f(x_{k+2})$ respectively. For any $j \in [k]$, x_j occurs in two functions; we call β_j and γ_j the sets used in the Fourier expansion of the two functions (in the order they appear in the expression of Figure 1). We get the expression of Figure 2.

When we take the average, everything cancels except when $\beta_i = \gamma_i$ for all i, $\alpha_1 = \Delta_{i\in A}\beta_i \Delta_{i\in C}\beta_i$, and $\alpha_2 = \Delta_{i\in B}\beta_i \Delta_{i\in C}\gamma_i$. So the average is

$$\begin{aligned} &\sum_{\beta_1,\ldots,\beta_k}\hat{f}_{(\Delta_{i\in A\cup B}\beta_i)}\hat{f}_{(\Delta_{i\in B\cup C}\beta_i)}\hat{f}_{\beta_1}^2\cdots\hat{f}_{\beta_k}^2 \\ \le\ & (\max_{\alpha}|\hat{f}_{\alpha}|)^2\sum_{\beta_1,\ldots,\beta_k}\hat{f}_{\beta_1}^2\cdots\hat{f}_{\beta_k}^2 \\ \le\ & \max_{\alpha}|\hat{f}_{\alpha}| . \end{aligned}$$

$$\mathop{\mathbf{E}}_{x_1,\ldots,x_{k+2}} f(x_{k+1})f(x_{k+2}) \left(\prod_{j\in A} f(x_{k+1}x_j)f(x_j)\right) \left(\prod_{j\in B} f(x_{k+2}x_j)f(x_j)\right) \left(\prod_{j\in C} f(x_{k+1}x_j)f(x_{k+2}x_j)\right) .$$

Figure 1: An expression arising in the proof of Lemma 11.

$$\sum_{\substack{\alpha_1,\alpha_2,\beta_1,\\ \gamma_1,\ldots,\beta_k,\gamma_k}} \hat{f}_{\alpha_1}\cdots\hat{f}_{\gamma_k} \mathop{\mathbf{E}}_{x_1,\ldots,x_{k+2}} \left[l_{\left(\alpha_1 \mathop{\Delta}_{i\in A} \beta_i \mathop{\Delta}_{i\in C} \beta_i\right)}(x_{k+1}) l_{\left(\alpha_2 \mathop{\Delta}_{i\in B} \beta_i \mathop{\Delta}_{i\in C} \gamma_i\right)}(x_{k+2}) \prod_{i=1}^{k} l_{\beta_i\Delta\gamma_i}(x_i) \right] .$$

Figure 2: Another expression arising in the proof of Lemma 11.

The reader can verify that when the degree of $k+1$ is even and the degree of $k+2$ is odd (or vice-versa), the the expectation is bounded by $(\max_\alpha |\hat{f}_\alpha|)^3$ (the path of length three is a special case of this analysis), and when the degree of both $k+1$ and $k+2$ is even the expectation is at most $(\max_\alpha |\hat{f}_\alpha|)^4$, or $(\max_\alpha |\hat{f}_\alpha|)^2$ in the special case occurring when $A = B = \emptyset$ (that is, when $S = B_k$ and k is even). $\square$

Theorem 12 $\mathbf{AccGraph}(([k+2], B_k), f) \leq \frac{1}{2^{2k}} + \frac{2^{2k}-1}{2^{2k}}(1 - 2\mathbf{Dist}(f, \mathsf{LIN} \cup \overline{\mathsf{LIN}}))$.

5 Application to PCP Constructions

In this Section we present our PCP construction and its analysis. To this aim, we first extend the framework of proof-composition of [5]. Our definition of inner verifier and of decoding procedure is general enough to capture (some of) the recent results of Håstad, is fairly compact and conceptually simpler than previous similar definitions.

5.1 The General Framework

We first need a definition analogous to that of *folding* from [5]. Observe that if $A = l_{\{a\}}$ is a Long Codeword, then $A(x) = -A(-x)$ for any x; for any function $A : \{1,-1\}^n \to \{1,-1\}$ we will define a new function A' that satisfies such a property. The definition of A' is as follows:

$$A'(x) = \begin{cases} A(x) & \text{If } x(1) = 1 \\ -A(-x) & \text{If } x(1) = -1. \end{cases}$$

We stress that, for any x, $A'(x)$ can be evaluated with one query to A, moreover A' is equal to A if A is a Long Codeword. From the fact that $A'(x) = -A'(-x)$ for any x it follows that $\hat{A}'_\alpha = 0$ for any α of even size, in particular for $\alpha = \emptyset$. This property is useful in applying Lemma 17 below.

Definition 13 (Inner Verifier) *An* inner verifier *is a randomized algorithm V that given a function $\pi : [m] \to [n]$ and given oracle access to two functions $A : \{1,-1\}^n \to \{1,-1\}$ and $B : \{1,-1\}^m \to \{1,-1\}$ decides whether to accept or reject.*

Definition 14 (Decoding Procedure) *A* decoding procedure *is a pair of randomized algorithms (D_1, D_2) such that for any $A : \{1,-1\}^n \to \{1,-1\}$ and any $B : \{1,-1\}^m \to \{1,-1\}$, $D_1(A)$ returns an element of $[n]$ and $D_2(B)$ returns an element of $[m]$.*

Definition 15 (Good Inner Verifier) *An inner verifier V is (c,s,q)-good with respect to a decoding procedure (D_1, D_2) if for any $\pi : [m] \to [n]$, any $A : \{1,-1\}^n \to \{1,-1\}$, and any $B : \{1,-1\}^m \to \{1,-1\}$, the following properties hold.*

- [NUMBER OF QUERIES.] *V makes at total number of at most q non-adaptive oracle queries.*
- [COMPLETENESS.] *If A is the Long Code of a, B is the long code of b, and $\pi[b] = a$, then*
$$\mathbf{Pr}[V(A', B', \pi)\text{accepts}] \geq c\ .$$
- [SOUNDNESS.] *For any constant $\delta > 0$, there is a positive constant $\delta' > 0$ independent of m and n such that if*
$$\mathbf{Pr}[V(A', B', \pi)\text{accepts}] \geq s + \delta\ ,$$
then
$$\mathbf{Pr}[D_1(A') = \pi(D_2(B'))] \geq \delta'\ .$$

Theorem 16 *If there exists a (c,s,q)-good inner verifier then, for any $\delta > 0$, $\mathsf{NP} = \mathbf{na}\mathsf{PCP}_{c,s+\delta}[\log, q]$.*

A proof of Theorem 16 can be found in a preliminary full version of this paper [24]. It differs from a similar proof in [5] since, in our definition of inner verifier, we allow randomized decoding procedures (which is easy to deal with) and we do not have a "circuit test" (this requires some more care).

Using Theorem 16, the goal of developing a PCP construction reduces to find an inner verifier and a decoding procedure. We will use a decoding procedure from [13] and its Fourier analysis. Before quoting this result, we need one more piece of notation: let m and n be two integers and $\pi : [m] \to [n]$; for a set $\beta \subseteq [m]$ we denote by $\pi_2(\beta)$ the set of elements $a \in [n]$ such that there is an odd number of bs in β that are mapped into a by π. In symbols,

$$\pi_2(\{b_1, \ldots, b_h\}) = \pi(b_1)\Delta\pi(b_2)\Delta \cdots \Delta\pi(b_h) .$$

The reason why we introduce this operator is that $l_\beta(x \circ \pi) = l_{\pi_2(\beta)}(x)$, and, in turn, evaluating a linear function in $x \circ \pi$ is a problem that arises in the Fourier analysis of inner verifiers.

Lemma 17 ([13]) *A decoding procedure (D_1^H, D_2^H) exists with the following property. For every $\varepsilon, \delta > 0$ there exists a δ' such that, for any A and B such that $\hat{A}_\emptyset = \hat{B}_\emptyset = 0$, if*

$$\sum_{\beta \subseteq [m]} |\hat{A}_{\pi_2(\beta)}| \hat{B}_\beta^2 (1 - 2\varepsilon)^{|\beta|} \geq \delta ,$$

then

$$\mathbf{Pr}[D_1^H(A) = \pi(D_2^H(B))] \geq \delta' .$$

A proof of Lemma 17 is given in [13, Lemma 2.2]. In [24] we present a slight simplification of the proof of [13] (our decoding procedure is simpler.) Our alternative proof is omitted from this extended abstract.

5.2 Our PCP Construction

We now define our inner verifier and analyse it. It performs two executions of the protocol of Håstad [13]. One query is recycled between the two executions. The reader can compare our analysis with the analysis of the linearity test associated with a path of length two (or a star with two rays). The inner verifier is described in Figure 3.

Lemma 18 *For any $A : \{-1,1\}^n \to \{-1,1\}$, $B : \{-1,1\}^m \to \{-1,1\}$, $\pi : [m] \to [n]$, $\varepsilon > 0$,*

$$\begin{aligned} &\mathbf{Pr}[\mathsf{Inner}_\varepsilon(A, B, \pi) \textit{ accepts}] \\ \leq\ & \frac{1}{4} + \frac{3}{4} \sum_\beta |\hat{A}_{\pi_2(\beta)}| \hat{B}_\beta^2 (1 - 2\varepsilon)^{|\beta|} . \end{aligned}$$

PROOF: It is immediate to see that

$$\begin{aligned} &\mathbf{Pr}[\mathsf{Inner}_\varepsilon(A, B, \pi) \text{ accepts }] = \\ &\tfrac{1}{4} + \\ &\tfrac{1}{4} \mathop{\mathbf{E}}_{x_1, y, e_1} A(x_1)B(y)B((x_1 \circ \pi)ye_1) + \\ &\tfrac{1}{4} \mathop{\mathbf{E}}_{x_2, y, e_2} A(x_2)B(y)B((x_2 \circ \pi)ye_2) + \\ &\tfrac{1}{4} \mathop{\mathbf{E}}_{\substack{x_1, x_2 \\ y, e_1, e_2}} A(x_1)A(x_2)B((x_1 \circ \pi)ye_1)B((x_2 \circ \pi)ye_2) . \end{aligned}$$

It has been proved in [13] that

$$\begin{aligned} & \mathop{\mathbf{E}}_{x_i, y, e_i} A(x_i)B(y)B((x_i \circ \pi)ye_i) \\ = & \sum_\beta \hat{A}_{\pi_2(\beta)} \hat{B}_\beta^2 (1 - 2\varepsilon)^{|\beta|} \\ \leq & \sum_\beta |\hat{A}_{\pi_2(\beta)}| \hat{B}_\beta^2 (1 - 2\varepsilon)^{|\beta|} . \end{aligned}$$

It remains to evaluate the last term. We use the following expansions:

- $A(x_1) = \sum_{\alpha_1} \hat{A}_{\alpha_1} l_{\alpha_1}(x_1)$,
- $A(x_2) = \sum_{\alpha_2} \hat{A}_{\alpha_2} l_{\alpha_2}(x_2)$,
- $B(x_1 \circ \pi y e_1) = \sum_{\beta_1} \hat{B}_{\beta_1} l_{\beta_1}(x_1 \circ \pi) l_{\beta_1}(y) l_{\beta_1}(e_1)$,
- $B(x_2 \circ \pi y e_2) = \sum_{\beta_2} \hat{B}_{\beta_2} l_{\beta_2}(x_2 \circ \pi) l_{\beta_2}(y) l_{\beta_2}(e_2)$.

The expectation of the product is given in Figure 4.
Let us first estimate the last two terms. Following [13] we have

$$\mathop{\mathbf{E}}_{e_i} l_{\beta_i}(e_i) = \prod_{b \in \beta_i} \mathop{\mathbf{E}}_{e_i} e_i(b) = (1 - 2\varepsilon)^{|\beta_i|} .$$

For the other terms, everything cancels except when $\beta_1 = \beta_2 = \beta$ and $\alpha_1 = \alpha_2 = \pi_2(\beta)$. We thus have that (7) is equal to

$$\sum_\beta \hat{A}_{\pi_2(\beta)}^2 \hat{B}_\beta^2 (1 - 2\varepsilon)^{2|\beta|} \leq \sum_\beta |\hat{A}_{\pi_2(\beta)}| \hat{B}_\beta^2 (1 - 2\varepsilon)^{|\beta|} .$$

□

Lemma 19 *For any $0 < \varepsilon \leq 1/2$, $\mathsf{Inner}_\varepsilon$ is a $((1 - \varepsilon)^2, 1/4, 5)$-good inner verifier with respect to (D_1^H, D_2^H).*

PROOF: Of course $\mathsf{Inner}_\varepsilon(A', B', \pi)$ makes 5 queries, two to A' and three to B'. If A is the long code of a and B is the long code of b and $\pi(b) = a$, then the test performed by $\mathsf{Inner}_\varepsilon$ reduces to check whether $e_1(b) = e_2(b) = 1$. This test is satisfied with probability $(1 - \varepsilon)^2$. It remains to check the third property of the definition of inner verifier. Assume that

$$\mathbf{Pr}[\mathsf{Inner}_\varepsilon(A', B', \pi)\text{accepts}] \geq \frac{1}{4} + \delta .$$

Then we have that

$$\sum_\beta |\hat{A}'_{\pi_2(\beta)}| \hat{B}'^2_\beta (1 - 2\varepsilon)^\beta \geq \frac{4}{3}\delta > \delta$$

and by applying Lemma 17 to A' and B' we deduce that there exists a δ' (depending only on δ and on ε) such that

$$\mathbf{Pr}[D_1(A') = \pi(D_2(B'))] \geq \delta' .$$

□

Theorem 20 *For any $\varepsilon > 0$, $\mathsf{NP} = \mathbf{na}\mathsf{PCP}_{1-\varepsilon, \frac{1}{4}+\varepsilon}[\log, 5]$.*

$\mathsf{Inner}_\varepsilon(A, B, \pi)$ Choose uniformly at random $x_1, x_2 \in \{1,-1\}^n$ and $y \in \{1,-1\}^m$ Choose at random $e_1, e_2 \in \{1,-1\}^m$ such that $\forall b \in [m].\mathbf{Pr}[e_i(b) = 1] = 1 - \varepsilon$ **if** $A(x_1)B(y)B((x_1 \circ \pi)ye_1) = 1$ and $A(x_2)B(y)B((x_2 \circ \pi)ye_2) = 1$ **then accept** **else reject**

Figure 3: The inner verifier.

$$\sum_{\alpha_1,\alpha_2,\beta_1,\beta_2} \hat{A}_{\alpha_1}\hat{A}_{\alpha_2}\hat{B}_{\beta_1}\hat{B}_{\beta_2}\left(\mathop{\mathbf{E}}_{x_1} l_{\alpha_1\Delta\pi_2(\beta_1)}(x_1)\right)\left(\mathop{\mathbf{E}}_{x_2} l_{\alpha_2\Delta\pi_2(\beta_2)}(x_2)\right)\left(\mathop{\mathbf{E}}_{y} l_{\beta_1\Delta\beta_2}(y)\right)\left(\mathop{\mathbf{E}}_{e_1} l_{\beta_1}(e_1)\right)\left(\mathop{\mathbf{E}}_{e_2} l_{\beta_2}(e_2)\right). \quad (7)$$

Figure 4: An expression arising in the proof of Lemma 18.

As observed by Oded Goldreich, one can also derive $\mathsf{NP} = \mathbf{na}\mathsf{PCP}_{1-\varepsilon,1/4}[\log, 5]$ by letting the verifier fail unconditionally with probability $1 - (1/4)/\left(\frac{1}{4} + \varepsilon\right)$ and proceed normally otherwise. In this way, the soundness becames $1/4$ and the completeness becomes $(1-\varepsilon)/(1+4\varepsilon)$, that is arbitrarily close to 1 since ε was arbitrary.

6 Related Results and Open Questions

We believe that NP can be characterized using $1+\varepsilon$ amortized query bits and that there exists a linearity tester working with $1+\varepsilon$ amortized query bits.

A significant progress in the first direction has been recently achieved by Sudan and Trevisan [21] who adapted our analysis of the Bipartite-Graph Linearity Tester to the PCP setting, obtaining a PCP characterization of NP with $3k+2$ queries, completeness $(1-\varepsilon)$ and soundness 2^{-2k}, for any k and ε (such a verifier uses, asymptotically, essentially 1.5 amortized query bits, and thus shows a separation between the PCP model and the 2-Prover 1-Round model.)

The main difficulty in obtaining such result is the necessity to depart from the definition of inner verifier given in this paper. It is indeed possible to show that whenever an inner verifier is limited to look at two tables and has to check consistency between them, then it must use at least 2 amortized query bits. In order to do better, it is necessary to consider inner verifiers that look at a larger number of tables, and that try to check some weak form of consistency between the tables. This approach was originally used by Håstad [12] to obtain his tight result for the Max Clique problem.

It remains an open question whether it is possible to use less than 1.5 amortized query bits either for the Linearity Test problem or in PCP constructions. Regarding Linearity Testing, a reasonable conjecture is that the test associated with a bipartite complete graph $K_{k,h}$ has acceptance probability at most $\frac{1}{2^{kh}} + \frac{2^{kh}-1}{2^{kh}}(\max_\alpha|\hat{f}_\alpha|)$. If this is the case, then the test associated with the graph $K_{k,k}$ has amortized query complexity $1+\varepsilon$ (for any ε and sufficiently large k). In order to prove that, we have to bound expectations of products over all the subgraphs of $K_{k,h}$. The expectations always simplify to a sum of coefficients, and Parseval's equality is the only tool that we know in order to bound such summations in a convenient way. In order to have squares in the summation (and to apply Parseval) we need vertices with degree two in the graph. Thus, our approach only works for graphs where most vertices have degree two, and $K_{k,2}$ is the densest one with this property. In order to beat amortized query complexity 1.5 we have to bound summations without using Parseval's equation. At a higher level, we hope that the following statement holds.

Conjecture 21 *For any* $f : \{1,-1\}^n \to \{1,-1\}$, *for any non-empty graph* $G = ([k], E)$,

$$\mathop{\mathbf{E}}_{x_1,\ldots,x_k} \prod_{(i,j)\in E} f(x_i)f(x_j)f(x_ix_j) \le \max_\alpha |\hat{f}_\alpha|. \quad (8)$$

The technical lemmas of Section 4 prove some special cases of this conjecture (namely, for the case where G is a collection of paths, or a edge-induced subgraph of $K_{2,k}$.) An inspection of the proofs of the lemmas of Section 4 shows that in fact, whenever we can prove the inequality of Equation (8), we can prove it for any function f such that $\sum_\alpha \hat{f}_\alpha^2 = 1$. This motivates the following (unfortunately false) claim.

False Conjecture 22 *For any* $f : \{1,-1\}^n \to \mathbf{R}$, *such that* $\sum_\alpha \hat{f}_\alpha^2 = 1$, *for any non-empty graph* $G = ([k], E)$,

$$\mathop{\mathbf{E}}_{x_1,\ldots,x_k} \prod_{(i,j)\in E} f(x_i)f(x_j)f(x_ix_j) \le \max_\alpha |\hat{f}_\alpha|. \quad (9)$$

In fact the above statement fails quite dramatically. Consider the function $f : \{1,-1\}^n \to \mathbf{R}$ such that $\hat{f}_\alpha = 2^{-n/2}$ for any α. Then for a graph with k vertices and at least $2k+1$

edges the left-hand side of (9) tends to infinity with n, while the right-hand side tends to zero. A proof of Conjecture 21 needs to use the fact that f is a Boolean function, and not just a function of unit ℓ_2 norm.

Acknowledgements

I thank Madhu Sudan for several valuable suggestions and comments. An insightful conversation with Dan Spielman on probabilistically checkable codes was the starting point of this research.

References

[1] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and hardness of approximation problems. In *Proc. of the 33rd IEEE FOCS*, pages 14–23, 1992.

[2] S. Arora and S. Safra. Probabilistic checking of proofs; a new characterization of NP. In *Proc. of the 33rd IEEE FOCS*, pages 2–13, 1992.

[3] M. Bellare. Proof checking and approximation: Towards tight results. *Sigact News*, 27(1), 1996.

[4] M. Bellare, D. Coppersmith, J. Håstad, M. Kiwi, and M. Sudan. Linearity testing over characteristic two. *IEEE Trans. on Information Theory*, 42(6):1781–1795, 1996.

[5] M. Bellare, O. Goldreich, and M. Sudan. Free bits, PCP's and non-approximability – towards tight results (5th version). Technical Report TR95-24, Electronic Colloquium on Computational Complexity, 1997. Preliminary version in *Proc. of FOCS'95*.

[6] M Bellare, S. Goldwasser, C. Lund, and A. Russell. Efficient probabilistically checkable proofs and applications to approximation. In *Proc. of the 25th ACM STOC*, pages 294–304, 1993. See also the errata sheet in *Proc of STOC'94*.

[7] M. Bellare and M. Sudan. Improved non-approximability results. In *Proc. of the 26th ACM STOC*, pages 184–193, 1994.

[8] M. Blum, M. Luby, and R. Rubinfeld. Self-testing/correcting with applications to numerical problems. In *Proc. of the 22nd ACM STOC*, pages 73–83, 1990.

[9] D. Coppersmith. Unpublished notes, 1990.

[10] N. Creignou. A dichotomy theorem for maximum generalized satisfiability problems. *JCSS*, 51(3):511–522, 1995.

[11] J. Håstad. Testing of the long code and hardness for clique. In *Proc. of the 28th ACM STOC*, pages 11–19, 1996.

[12] J. Håstad. Clique is hard to approximate within $n^{1-\varepsilon}$. Technical Report TR97-38, Electronic Colloquium on Computational Complexity, 1997. Preliminary version in *Proc. of FOCS'96*.

[13] J. Håstad. Some optimal inapproximability results. In *Proc. of the 29th ACM STOC*, pages 1–10, 1997.

[14] B. Karloff and U. Zwick. A $(7/8 - \varepsilon)$-approximation algorithm for MAX 3SAT? In *Proc. of the 38th IEEE FOCS*, 1997.

[15] S. Khanna, R. Motwani, M. Sudan, and U. Vazirani. On syntactic versus computational views of approximability. In *Proc. of the 35th IEEE FOCS*, pages 819–830, 1994.

[16] S. Khanna, M. Sudan, and D.P. Williamson. A complete classification of the approximability of maximization problems derived from boolean constraint satisfaction. In *Proc. of the 29th ACM STOC*, pages 11–20, 1997.

[17] M. Kiwi. *Probabilistically Checkable Proofs and the Testing of Hadamard-like Codes*. PhD thesis, MIT, 1996.

[18] C. H. Papadimitriou and M. Yannakakis. Optimization, approximation, and complexity classes. *JCSS*, 43:425–440, 1991. Preliminary version in *Proc. of STOC'88*.

[19] R. Raz. A parallel repetition theorem. In *Proc. of the 27th ACM STOC*, pages 447–456, 1995.

[20] M. Serna, L. Trevisan, and F. Xhafa. The parallel approximability of non-boolean constraint satisfaction and restricted integer linear programming. In *Proc. of the 15th STACS*, 1998. To appear.

[21] M. Sudan and L. Trevisan. Unpublished Notes, February 1998.

[22] L. Trevisan. Positive linear programming, parallel approximation, and PCP's. In *Proc. of the 4th ESA*, pages 62–75. LNCS 1136, Springer-Verlag, 1996.

[23] L. Trevisan. Approximating satisfiable satisfiability problems. In *Proc. of the 5th ESA*, pages 472–485. LNCS 1284, Springer-Verlag, 1997.

[24] L. Trevisan. Recycling queries in PCPs and in linearity tests. Technical Report TR98-07, Electronic Colloquium on Computational Complexity, January 1998.

[25] L. Trevisan, G.B. Sorkin, M. Sudan, and D.P. Williamson. Gadgets, approximation, and linear programming. In *Proc. of the 37th IEEE FOCS*, pages 617–626, 1996.

[26] U. Zwick. Approximation algorithms for constraint satisfaction problems involving at most three variables per constraint. In *Proc. of the 9th ACM-SIAM SODA*, 1998.