Free Bits and Non-Approximability
(Extended Abstract)

MIHIR BELLARE[*] ODED GOLDREICH[†] MADHU SUDAN[‡]

Abstract

This paper investigates the possibility that tight hardness of approximation results may be derived for several combinatorial optimization problems via the "pcp-connection" (a.k.a., the FGLSS-reduction [FGLSS]). We study the amortized free bit-complexity of probabilistic verifiers and "invert the FGLSS-reduction" by showing that an NP-hardness result for the approximation of MaxClique to within a factor of $N^{1/(g+1)}$ would imply a probabilistic verifier for NP with logarithmic randomness and amortized free-bit complexity g. In addition, we present a new proof system for NP in which the amortized free-bit complexity is ≈ 2 which yields (via the FGLSS-reduction) that approximating the clique to within a factor of $N^{1/3}$ (in an N-vertex graph) is NP-hard. The new proof system is based on a new code and means for checking codewords. This machinery is used to obtain improved non-approximability results for Max-SNP problems such as Max-2SAT, Max-3SAT, Max-CUT and Min-VC.

This extended abstract provides less technical detail than customary. The interested reader is referred to our (124-pages) technical report [BGS].

[*] Advanced Networking Laboratory, IBM T.J. Watson Research Center, P.O. Box 704, Yorktown Heights, NY 10598, USA. e-mail: `mihir@watson.ibm.com`.

[†] Department of Computer Science and Applied Mathematics, Weizmann Institute of Sciences, Rehovot, Israel. e-mail: `oded@wisdom.weizmann.ac.il`. Partially supported by grant No. 92-00226 from the US–Israel Binational Science Foundation (BSF), Jerusalem, Israel.

[‡] Research Division, IBM T.J. Watson Research Center, P.O. Box 218, Yorktown Heights, NY 10598, USA. e-mail: `madhu@watson.ibm.com`.

1 Introduction

The success of the interactive proof based approach to deriving non-approximability results seems beyond question — not only has problem after problem fallen, but results grow succesively stronger. Today, it even seems possible that this approach may lead to *tight* non-approximability results for several popular optimization problems.

Max Clique is a canonical example. Known non-approximability results are strong; but they are not tight, and improvement seems desirable and feasible. Given the absence of any algorithm which approximates this problem to within $N^{1-\epsilon}$, for any $\epsilon > 0$, one wonders if it is the case that such an approximation may not be achievable in polynomial time, and if so how can we show this[1]? In particular, are interactive proofs still the right approach, and, if so, what kinds of theorems about them should we be proving?

1.1 Max Clique and amortized free bits

The first part of our paper is a result that we feel answers the above question. Roughly, it reverses the connection between proofs and approximation, to say that certain kinds of theorems about proof systems are *necessary* to prove non-approximability results. That is, the use and success of proofs was no co-incidence: they are inherent.

But the connection goes further. It identifies a specific parameter of proof systems that must be minimized to get tight non-approximability results for Max Clique. This parameter is the number of *amortized free bits* (to be defined very soon). Thus, as an answer to the question we asked above (about what kind of theorems we should be proving) it says that to get tight non-approximability results for Max Clique, we must construct proofs which minimize the number of amortized free bits. To describe this result, as well as others, we first need some definitions.

PROOF SYSTEM PARAMETERS. The key word in proof checking is *parameters*– there are many of them. First, recall that a probabilistically checkable proof system [FGLSS][2] is described by a probabilistic, polynomial time verifier V who has, in addition to the input x, oracle access to a proof string. While the task is typically language recognition, we will, more generally, consider promise problems (A, B) consisting of positive and negative instances; we can define a completeness probability c and a soundness probability s in the usual way, the former applying to positive instances and the latter to negative ones. In case $c = 1$ we say that the verifier is of *perfect completeness*. The *gap* is $g = c/s$. Now here are the parameters:

- *query complexity* – the maximum, over all possible coin tosses of V, of the number of bits of the oracle accessed by the verifier for a fixed input.
- *free bit complexity* – the maximum, over all possible coin tosses of V, of the logarithm of the number of possible accepting configurations of V.[3] For example a verifier which makes 3 queries and accepts iff the parity of the answers is odd has 4 accepting configuration and thus free bit complexity 2.
- *amortized free bit complexity* – the free bit complexity divided by the logarithm of gap. That is, the number of free bits needed per factor of 2 increase in the gap.
- *FGLSS-reduction* – a reduction of a promise problem (A, B) or rather a pcp system for (A, B), which maps an input x to a graph G_x so that the max-clique in G_x reflects the maximum accepting probability of the pcp verifier on input $x \in A \cup B$ [FGLSS].

We use the notation $\mathrm{PCP}_{c,s}[\,r, q\,]$ to denote the class of promise problems recognized by verifiers tossing r coins, having query complexity q, and achieving completeness probability c and soundness probability s. $\mathrm{FPCP}_{c,s}[r, f]$ is defined analogously with f being the free bit complexity; $\overline{\mathrm{FPCP}}_c[r, f]$ is defined analogously with f being the amortized free bit complexity and c being the completeness parameter. (Note that in the last case there is no need to specify s since we measure the "rate" of free bit usage!) Amortized query complexity can be defined analogously to amortized free bit complexity, and we let $\overline{\mathrm{PCP}}_c[\,r,\, q\,]$ be the corresponding class. In case the verifier is of perfect completeness (i.e., $c = 1$) we may omit the subsript from the latter two notations. In general, unless stated otherwise, the verifier is of perfect completeness and its error refers to the soundness error.

TOWARDS TIGHT RESULTS: THE ROLE OF THE PARAMETERS. The key factor in obtaining tight non-approximability results via the pcp-connection (for any problem), is the distilling of appropriate pcp-parameters that need to be minimized and the use of a good-reduction. For Max Clique, the basic reduction is that of [FGLSS], used today in slightly tighter randomized form due to [BeSc, Zu]. The sequence of works [FGLSS, ArSa, ALMSS, BGLR, FeKi, BeSu] lead

[1] An additional motivation for looking for such "weak" approximation algorithms was suggested by Blum [Bl]: such an algorithm would imply an algorithm for coloring a 3-colorable graph with $O(\log n)$ colors, which is significantly fewer colors than known algorithms use.

[2] An analogous discussion can be carried out also using the model of transparent proofs due to Babai et. al. [BFLS].

[3] This is a simplification which omits discussion of an additional constructivity property.

us through a sequence of parameters: query complexity, free bit complexity and, finally, for the best known results, amortized free bit complexity. The relation here is that if NP is in $\overline{\mathrm{FPCP}}[O(\log n), f]$ then the Max Clique size of an N-vertex graph is NP-hard to approximate (under randomized Karp reductions) within a factor of $N^{1/(1+f+\epsilon)}$, for any $\epsilon > 0$.

A REVERSE CONNECTION. Our result inverts the above mentioned connection to establish an equivalence.

Theorem 1.1 Let $f : \mathcal{Z}^+ \to \mathcal{R}^+$ be polynomial-time computable. Then the following two statements are equivalent.

(1) For all $\epsilon > 0$, approximating the Max Clique size of an N-vertex graph to within a factor of $N^{1/(1+f+\epsilon)}$ is NP-hard via a randomized Karp (resp., Cook) reduction.

(2) For all $\epsilon > 0$, the class NP is random Karp (resp., Cook) reducible to $\overline{\mathrm{FPCP}}[O(\log n), f + \epsilon]$.

Thus if there is some (any) way to show hardness of Max Clique approximaiton to within $N^{1/(1+f+\epsilon)}$ then NP has (via a randomized reduction) a proof system with logarithmic randomness and amortized free bit complexity $f + \epsilon$. We stress both the "qualitative" and the "quantitative" aspects of this result.

Qualitatively, it provides an answer to the following kind of a question: "What do proofs have to do with approximating clique size, and can we not prove the non-approximability result without using proof checking?" The result indicates that proofs are inherent, and explains, perhaps, why hardness results avoiding the proof connection have not appeared.

However, at this stage it is the quantitative aspect that interests us more. It says that to get tighter results on Max Clique hardness, we must construct proof systems to minimize the amortized free bit complexity.

Can we hope to do so? It is a feature of the measure that so far this seems entirely possible. In particular it seems possible that the amortized free bit complexity of a pcp verifier for NP can be ϵ for any $\epsilon > 0$. Indeed, the theorem says that if Max Clique is indeed hard to within $N^{1-\epsilon}$ as we believe, then such a system will exist.[4] It may be worth noting that both the freeness and the amortization are key to this: we can rule out such efficiency under query complexity, amortized query complexity, and (non-amortized) free bit complexity. This is because of the following:

Theorem 1.2 The following containments hold:

(1) $\mathrm{PCP}_{1,1/2}[\,\log,\, 2\,] \subseteq P$.

(2) $\overline{\mathrm{PCP}}[\,\log,\, 1\,] \subseteq P$.

(3) $\mathrm{FPCP}_{1,1/2}[\,\log,\, 1] \subseteq P$.

For further results regarding the structure of the PCP and FPCP hierarchies see Section 5. Also see Section 6 for various useful transformations between proof systems.

MINIMIZING AMORTIZED FREE BITS. Theorem 1.1 thus further motivates the search for the best possible proof systems for NP in the sense of amortized free-bit complexity. Bellare and Sudan [BeSu] have constructed a pcp verifier for NP that uses $3+\epsilon$ amortized free-bits. We improve upon this, presenting a new proof system with amortized free-bit complexity $2 + \epsilon$. This implies that Max Clique is hard to approximate to within $N^{\frac{1}{3+\epsilon}}$.

Theorem 1.3 For every $\epsilon > 0$, NP $\subseteq \overline{\mathrm{FPCP}}[\,\log,\, 2 + \epsilon\,]$. Thus the Max Clique size of an N vertex graph is NP-hard (under randomized Karp-reductions) to approximate within $N^{1/3+\epsilon}$.

Combined with a recent reduction by Furer [Fu], which in turn builds upon the reductions presented in [LuYa, KLS, BeSu], we get:

Theorem 1.4 For any $\epsilon > 0$, it is NP-Hard (under randomized Karp reductions) to approximate the chromatic number of an N-vertex graph to within $N^{\frac{1}{5}-\epsilon}$.

Previous improvements in the efficiency of proof systems [BFL, BFLS, FGLSS, ArSa, ALMSS, BGLR, FeKi, BeSu] had required non-trivial technical advances. Our improvement has the same feature. In particular, underlying our proof systems is a new (and simple) error-correcting code, called the long code, and means to check it. The corresponding tests and analysis are the main technical contribution of our paper.

Furthermore under the framework used within this and previous papers on this subject, the $2 + \epsilon$ threshold seems to be a natural barrier. We show that any proof system that is constructed via the paradigm used here must use $2 - \epsilon$ amortized free-bits. The result, including a definition of what we mean by "our framework," is in Section 4. We stress that this last result makes various assumptions about methods, and is intended to show that significantly novel techniques are required to go further. But it does not suggest an *inherent* limitation, and, in particular, it does not rule out the hope of getting a verifier with amortized free-bit complexity of $\epsilon > 0$ for all NP languages.

[4] Note that these statements do not take into account limitations imposed by current technology as to what we can prove. The discussion on this aspect is the focus of Section 4 as well as of a subsequent paper of Arora [Ar].

Problem	**Approx**		**Non-Approx**		
	Factor	Due to	New Factor	Previous Factor	Assumption
Max-3-SAT	1.319	[Ya, GoWi1, GoWi2]	1.027	$1+\frac{1}{72}$ [BeSu]	$\mathrm{P} \neq \mathrm{NP}$
Max-E3-SAT	$1+\frac{1}{7}$	folklore	$1+\frac{1}{37}$	unspecified [ALMSS]	$\mathrm{P} \neq \mathrm{NP}$
Max-2-SAT	1.075	[GoWi2, FeGo]	1.010	$1+\frac{1}{504}$ (implied [BeSu])	$\mathrm{P} \neq \mathrm{NP}$
MAX CUT	1.139	[GoWi2]	1.012	unspecified [ALMSS]	$\mathrm{P} \neq \mathrm{NP}$
Min-VC	$2-o(1)$	[BaEv, MoSp]	$1+\frac{1}{26}$	unspecified [ALMSS]	$\mathrm{P} \neq \mathrm{NP}$
Max-Clique	$N^{1-o(1)}$	[BoHa]		$N^{\frac{1}{4+\epsilon}}$ [BeSu]	$\mathrm{NP} \not\subseteq \mathrm{coR}\tilde{\mathrm{P}}$
			$N^{\frac{1}{3+\epsilon}}$	?	$\mathrm{coRP} \neq \mathrm{NP}$
			$N^{\frac{1}{4+\epsilon}}$	$N^{\frac{1}{5+\epsilon}}$ [BeSu]	$\mathrm{P} \neq \mathrm{NP}$
Chromatic Num[5]	$N^{1-o(1)}$	[BoHa]		$N^{\frac{1}{10+\epsilon}}$ [BeSu]	$\mathrm{NP} \not\subseteq \mathrm{coR}\tilde{\mathrm{P}}$
			$N^{\frac{1}{5+\epsilon}}$	?	$\mathrm{coRP} \neq \mathrm{NP}$
			$N^{\frac{1}{7+\epsilon}}$	$N^{\frac{1}{13+\epsilon}}$ [BeSu]	$\mathrm{P} \neq \mathrm{NP}$

Figure 1: *Approximation factors attainable by polynomial-time algorithms (Approx) versus factors we show are hard to achieve (Non-Approx). Here $\epsilon > 0$ is an arbitrary positive constant.*

1.2 Results for MaxSNP

In the rest of the paper we find more uses for the long code based machinery. We apply it systematically to yield improved non-approximability results for a variety of MaxSNP problems. The improvements here are perhaps more significant than the Max Clique one: for the first time, we see hardness results for MaxSNP problems which are comparable to the factors achieved by known polynomial time approximation algorithms. The following theorem summarizes our results. Fig. 1 present a broader picture, which depicts, for each problem, the best known factor achievable by a polynomial time algorithm, our lower bound, and the best previous lower bound.

Theorem 1.5 The following indicate factors shown NP-hard to achieve in polynomial time:
(1) The minimum vertex cover size of a graph is NP-hard to approximate within $27/26$.
(2) Max3SAT and Max Exact 3SAT are NP-hard to approximate within $38/37$.
(3) Max CUT is NP-hard to approximate within $82/81$.
(4) Max2SAT is NP-hard to approximate within $94/93$.

We note that we are obtaining the first explicity and reasonable non-approximability factor for the minimum vertex cover. Recall that it is approximable within 2-o(1) [BaEv, MoSp]. Our results for Max CUT and Max 2SAT show that it is not possible to find a solution with value which is only 1% away from being optimal. This may be contrasted with the recent results of [GoWi2, FeGo] which shows that solutions which are within 14% and 7.5%, respectively, of the optimum are obtainable in polynomial time. Thus even though, we do not know if the "pcp approach" allows to get the best possible non-approximability results for these problems, we feel that the current results are not ridiculously far from the known upper bounds. Consider, for example, the ratio $\frac{u-1}{l-1}$, where u and l are the currently known upper and lower bounds, respectively. Then, the ratios for the abovementioned MaxSNP problems are 5.3 for Max-Exact-3SAT, 7 for Max-2SAT, 11.3 for Max-CUT, 11.8 for Max-3SAT, and 26 for MinVC (Minimum Vertex Cover).

1.3 Techniques

As in all recent constructions of efficient pcp's our construction also relies on the use of recursive construction of verifiers. We have the advantage of being able to use, at the outer level, the the verifier of Raz [Raz] which appeared only recently and was not available to previous works. The inner level verifier relies on the use of a "good" encoding scheme. Since [ALMSS], constructions of inner verifiers have used the Hadamard Code. In this paper we change this

[5]This result is obtained by combining our non-approximability result for clique with a recent (improved) reduction of [Fu].

aspect of the protocol and use instead a much more redundant code which we call the long code. This code encodes an n-bit string as a 2^{2^n} bit string which consists of the value of every boolean function on the n bit string. It is easy to see such codes have large Hamming distance. What is important is that this code is also easily "testable" and "correctable". This is shown in Section 3, where we show how this code and its machinery translates into the theorem described above. We also benefit from a recent improved analysis of the Linearity Test due to Bellare et. al. [BCHKS].

A second aspect of the improved hardness results for MaxSNP is the fact that we use direct reductions from verifiers to the problems of interest. This follows and extends [BGLR], prior to which results had used "generic" reductions, which did not take advantage of the nature of the tests performed by the verifier. In particular, in our case it turns out that the verifier only performs two kinds of tests — (1) verify that $a + b + c = 0 (\mod 2)$; and (2) verify that $a \cdot b = c + d (\mod 2)$, where a, b, c, d are all elements of $\{0, 1\}$. By constructing local gadgets (i.e., one gadget per random coin toss sequence) to verify each of the verifier's tests, we achieve better non-approximability results than using more general reductions. In particular our work seems to suggest that optimizing for gadgets which "check" the two conditions listed above will lead to reasonably good lower bounds for many Max SNP problems.

2 FPCP and Clique Approximation

Denote by $\mathcal{G}(V, x)$ the graph constructed from verifier V and input x by the FGLSS reduction. The vertices correspond to possible accepting transcripts in V's computation and edges corresponding to consistent/non-conflicting computations. The maximum clique size in the constructed graph is proportional to the accepting probability of V. Here we "reverse" the process; given a graph we constract a verifier such that the same proportion holds. Furthermore, applying the FGLSS-construction to our verifier retreives the original graph. We stress that by the term *graph* we mean an undirected simple graph (i.e., no self-loops or parallel edges).

Theorem 2.1 (clique verifier of ordinary graphs): There exists a verifier, denoted W, of logarithmic randomness-complexity, logarithmic query-length and zero free-bit complexity, that, on input a N-vertex graph G, satisfies $\max_\pi \Pr[W^\pi$ accepts $w(G)/N$. Furthermore, $\mathcal{G}(W, G)$ is isomorphic to G where the isomorphism is easily computable. Lastly, given a proof/oracle π we can construct in polynomial-time a clique of size pN in G, where p is the probability that W accepts G with oracle access to π.

The construction of W. On input a graph G on N nodes, the verifier W works with proofs of length $\binom{N}{2} - |E(G)|$. The proof π is indexed by the edges in $\overline{G}$ (i.e., non-edges in G). For clarity of the proof we assume that the binary value $\pi(\{u, v\})$ is either u or v. On input G and access to oracle π, the verifier W picks uniformly a vertex u in the vertex set of G and queries the oracle at each $\{u, v\} \in E(\overline{G})$. The verifier accepts if and only if all answers were u. Clearly, W tosses $\log_2 N$ coins. Also, once W picks a vertex u, the only pattern it may accepts is $(u, u, \ldots, u)$. Thus the free-bit complexity of W is 0. To analyze the probability that W accepts the input G, when given the best oracle access, one may merely prove that the graphs $\mathcal{G}(W, G)$ and G are isomorphic.

We now generalize the above construction to get verifiers which indicate the existence of large cliques in layered graphs. An (L, M, N)*-layered graph* is an N-vertex graph in which the vertices are arranged in L layers so that there are no edges between vertices in the same layer and each layer has at most M vertices. We use a convention by which, whenever a layered graph is given to some algorithm, a partition into layers is given along with it.

Theorem 2.2 (clique verifier for layered graphs): There exists a verifier, denoted W, of logarithmic randomness-complexity and logarithmic query-length so that, on input an (L, M, N)-layered graph G, the free-bit complexity of W is $\log_2 M$ and it satisfies $\max_\pi \Pr[W^\pi \text{ accepts } G] = w(G)/L$. Furthermore, the additional properties in Theorem 2.1 hold here as well.

The generalized construction of W. On input a (L, M, N)-layered graph G, the verifier W works with proofs consisting of two parts. The first part assigns every layer (i.e., every integer $i \in [L]$) a vertex in the layer (i.e., again we use a redundant encoding by which the answers are vertex names rather then an index between 1 and the number of vertices in the layer). The second part assigns pairs of non-adjacent (in G) vertices, a binary value, which again is represented as one of the two vertices. On input G and access to oracle π, the verifier W picks uniformly a layer i in $\{1, ..., L\}$ and queries π at i obtaining as answer a vertex u. If u is not in the i^{th} layer of G then the verifier rejects. Otherwise, it continues as in Theorem 2.1 (i.e., queries the oracle at each $\{u, v\} \in E(\overline{G})$ and accepts iff all answers equal u). (Actually, it is not needed to query the oracle on pairs of vertices belonging to the same layer.) The properties of W are established as before; in particular, observe that once a vertex u is specified, the only accepting pattern is $(u, u, \ldots, u)$. The free-bit complexity is determined by the number of vertices in layer i, which is upper bounded by M (and on the average equals N/L).

Main Consequences

We are interested in problems exhibiting a gap in Max-Clique size between positive and negative instances. It is convenient to let $\overline{\mathsf{MaxClique}}(G) = \mathsf{MaxClique}(G)/N$ be the fraction of nodes in a maximum clique of G.

Definition 2.3 For any $0 \le s(\cdot) \le c(\cdot) \le 1$ we let the promise problem $\mathsf{Gap}\text{-}\mathsf{Clique}_{c,s}$ be the pair (A, B), where–
(1) A is the set of all graphs G with $\overline{\mathsf{MaxClique}}(G) \ge c(N)$, and
(2) B is the set of all graphs G with $\overline{\mathsf{MaxClique}}(G) \le s(N)$.
The *gap* of this problem is defined to be c/s.

As a direct consequence of Theorem 2.1, we get

Corollary 2.4 For functions c, s: $\mathcal{Z}^+ \to [0, 1]$ we have $\mathsf{Gap}\text{-}\mathsf{Clique}_{c,s} \in \mathrm{FPCP}_{c,s}[\log, 0]$.

This corollary transforms the gap in the promise problem into a gap in a pcp system. However, the accepting probabilities in this pcp system are very low (also on yes-instances). Below, we use Theorem 2.2 to obtain a pcp system with almost perfect completeness for this promise problem. We start by presenting a randomized reduction of the promise problem to a layer version. An alternative method is presented in Section 6 (cf., Proposition 6.5).

Proposition 2.5 (layering the clique promise problem): There exists a polynomial-time randomized transformation, T, of graphs into layered graphs so that, on input a graph G, integers C and $L \le C/(3 \log_2 N)$, outputs a subgraph $H = T(G, C, L)$ of G in L layers such that with high probability H has at most $2N/L$ vertices per layer and if $w(G) \ge C$ then $w(H) = L$.

We remark that there exist an alternative transformation,[6] using only logarithmically many coins, and guranteeing only $\Pr[\, w(H) \le (1 - \epsilon) \cdot L \,] < L/(\epsilon C)$, for every $\epsilon \in [0, 1]$, provided $w(G) \ge C$. Combining Theorem 2.2 and Proposition 2.5, we obtain

Proposition 2.6 (inverse of FGLSS-reduction): For any polynomial-time computable functions c, s, ϵ: $\mathcal{Z}^+ \to [0, 1]$, the promise problem $\mathsf{Gap}\text{-}\mathsf{Clique}_{c,s}$ is randomly (Karp-) reducible to $\mathrm{FPCP}_{1,s'}[\log, f']$, where $f'(N) \stackrel{\text{def}}{=} \log_2(1/c(N)) + \log_2 \log_2(N) + 2$ and $s'(N) \stackrel{\text{def}}{=} 2 \log_2(N) \cdot \frac{s(N)}{c(N)}$.

Proposition 2.6 shows that the well-known method of obtaining clique-approximation results from efficient pcp systems (cf., [FGLSS, FeKi, BeSu]) is "complete" in the sense that if clique-approximation to within some factor can be shown NP-hard then this can be done via the "pcp method". This concludes the sketch of the proof of Theorem 1.1.

3 New proof systems and non-approximability results

The starting point for all our proof systems is a two-prover proof system achieving arbitrarily small constant error with logarithmic randomness and constant answer size. Furthermore, this proof system has the property that the answer of the second prover is supposed to be a predetermined function of the answer of the first prover. (A proof system satisfying these properties can be easily obtained by combining [ALMSS] and [Raz].) Thus, verification in it amounts to checking that the first answer satisfies some predicate and that the second answer equals the value obtained from the first answer. Following the "proof composition" paradigm of Arora and Safra [ArSa], we will "encode" the answers of the two provers under a suitable code and then, "recursively", check these encodings. This is captured formally by the construction of appropriate "inner verifiers." As usual, we will check both that these encodings are valid and that they correspond to answer which would have been accepted by the original verifier. The brunt of our constructions is thus the construction of inner verifiers.

Our main technical contribution is a new code, called the *long code*, and means to check it. The long code of an n-bit information word w is the sequence of 2^{2^n} bits representing the values of all possible boolean functions at w. The long code is certainly a disaster in terms of coding theory, but it has big advantages in the context of proof verification, arising from the fact that it carries enormous amounts of data about w.

We show that checking that the oracle is "close" to a codeword amounts to two tests – a *linearity* test and a *multiplication* test. The tests themselves are simple, but their analysis is not. Our analysis is based on a characterization of the code in terms of monomial series. The linearity test is also used as self-corrector when testing that the second answer indeed matches the first one. Finally, an additional idea is used to eliminate the need for a "circuit" test.[7]

[6] This alternative transformation is used for presenting an alternative inverse of the FGLSS-reduction. See our technical report [BGS].
[7] This observation is inessential for the construction of a pcp system with amortized free-bit complexity 2.

After establishing some basic notation in Section 3.1, we present the the long code and some basic properties in Section 3.2. We describe some "atomic" tests in Section 3.4. By performing suitable combinations of these tests, a verifier will be able to test that a given word is close to a codeword (of the long code) and that the underlying message word satisfies the acceptance condition of the outer level verifier. All verifiers we use perform different mixes of these tests. We present four major implementations of the above ideas yielding different types of pcp systems for NP (in Sections 3.5, 3.6, 3.7, and 3.8). The latter yield the hardness results for the Max SNP problems, Min-VC and Max Clique, respectively.

3.1 Preliminaries

$\Sigma = \{0,1\}$ will be identified with the finite field of two elements, the field operations being addition and multiplication modulo two. For any m we regard Σ^m as a vector space over Σ, so that strings and vectors are identified. A map f from a group G to a group H is *linear* if $f(x+y) = f(x) + f(y)$ for all $x, y \in G$. Let $\textsc{Lin}(G,H)$ denote the set of all linear maps of G to H. The *distance* between functions f_1, f_2 defined over a common finite domain D is $\text{Dist}(f_1, f_2) = \Pr_{x \overset{R}{\leftarrow} D}[f_1(x) \neq f_2(x)]$. If $f\colon G \to H$ we denote by $\text{Dist}(f, \textsc{Lin})$ the minimum, over all $g \in \textsc{Lin}(G,H)$, of $\text{Dist}(f,g)$. We are concerned with the case of G being a vector space V over Σ and H being Σ. In this case we have $\text{Dist}(f, \textsc{Lin}) \leq 1/2$ for all $f\colon V \to \Sigma$. Specifically, for an integer l, we consider the set of all maps of Σ^l to Σ, denoted $\mathcal{F}_l$. We regard $\mathcal{F}_l$ as a vector space (of dimension 2^l) over Σ. Addition and multiplication of functions are defined in the natural way.

THE MONOMIAL BASIS. For each $S \subseteq [l]$ we let $\chi_S \in \mathcal{F}_l$ be the monomial corresponding to S, defined for $x \in \Sigma^l$ by $\chi_S(x) = \prod_{i \in S} x^{(i)}$, where $x^{(i)}$ is the i^{th} bit of x, and $\chi_\emptyset$ is defined as identically 1. The functions $\{\chi_S\}_{S \subseteq [l]}$ form a basis for the vector space $\mathcal{F}_l$ which we call the *monomial basis*. This means that for each $f \in \mathcal{F}_l$, there exists a unique vector $\mathcal{C}(f) = (\, C_f(S)\,)_{S \subseteq [l]} \in \Sigma^{2^l}$ such that $f = \sum_{S \subseteq [l]} C_f(S) \cdot \chi_S$.

FOLDING.[8] Fix $\prec$ to be some canonical, polynomial time computable total order on the set $\mathcal{F}_l$. For $h \in \mathcal{F}_l$, define the *h-span* of $f \in \mathcal{F}_l$ to be $\{f + \sigma \cdot h : \sigma \in \Sigma\}$. Let $h \in \mathcal{F}_l$ and $b \in \Sigma$, the (h,b)*-folding* of $A\colon \mathcal{F}_l \to \Sigma$, denoted $A_{(h,b)}$, is a function given by $A_{(h,b)}(f) = A(g) + \sigma \cdot b$, where $g = f + \sigma \cdot h$ is the smallest function (by $\prec$) in the h-span of f. Folding over several (function,bit)-pairs is defined analogously. In the paper we use folding over both $(h,0)$ and $(\overline{1},1)$, where h is an arbitrary function (i.e., the first-query acceptance predicate) and $\overline{1}$ is the identically 1 function. Folding over $(h,0)$ allowes to get rid of the "circuit test" (see Proposition 3.2) and folding over $(\overline{1},1)$ simplifies and improves the analysis of the RMB-test (see Section 3.4).

3.2 Evaluation operators and the long code

Let $a \in \Sigma^l$. We define the map $E_a\colon \mathcal{F}_l \to \Sigma$ by $E_a(f) = f(a)$ for all $f \in \mathcal{F}_l$. We say that a map $A\colon \mathcal{F}_l \to \Sigma$ is an *evaluation operator* if there exists some $a \in \Sigma^l$ such that $A = E_a$. We now provide a useful characterization of evaluation operators. First we need a definition. Namely, a map $A\colon \mathcal{F}_l \to \Sigma$ is said to respect the monomial basis if $A(\chi_\emptyset) = 1$ and $A(\chi_S) \cdot A(\chi_T) = A(\chi_{S \cup T})$, for all $S, T \subseteq [l]$.

Proposition 3.1 A map $\tilde{A}\colon \mathcal{F}_l \to \Sigma$ is an evaluation operator if and only if it is linear and respects the monomial basis.

The *long code* E is defined for any $a \in \Sigma^l$ by $E(a) = E_a$. Thus, formally, a codeword is a map of $\mathcal{F}_l$ to Σ. Intuitively, think of the codeword $E(a)$ as the 2^{2^l} bit string which in position $f \in \mathcal{F}_l$ stores the bit $f(a)$. It is thus an extremely "redundant" code, encoding an l-bit string by the values, at a, of *all* functions in $\mathcal{F}_l$.

We let $\text{Dist}(A, \textsc{Eval}) = \min_{a \in \Sigma^l} \text{Dist}(A, E_a)$ be the distance from A to a closest codeword of E. It is convenient to define $E^{-1}(A) \in \Sigma^l$ as the lexicographically least $a \in \Sigma^l$ such that $\text{Dist}(A, E_a) = \text{Dist}(A, \textsc{Eval})$. Notice[9] that if $\text{Dist}(A, \textsc{Eval}) < 1/4$ then there is exactly one $a \in \Sigma^l$ such that $\text{Dist}(A, E_a) = \text{Dist}(A, \textsc{Eval})$, and so $E^{-1}(A)$ is this a. The following is useful in relating folding to the long code.

Proposition 3.2 Let $A\colon \mathcal{F}_l \to \Sigma$, $h \neq h' \in \mathcal{F}_l$, $b, b' \in \Sigma$ and $a \in \Sigma^l$. Then–

(1) For any $f \in \mathcal{F}_l$ it is the case that $A_{(h,b),(h',b')}(f + h) = A_{(h,b),(h',b')}(f) + b$.

(2) If $\text{Dist}(A_{(h,b),(h',b')}, E_a) < 1/2$ then $h(a) = b$.

[8] Folding is not essential to the Max Clique result but it plays an important role in deriving the best results for the MaxSNP problems.

[9] Actually, this observation only simlifies the analysis of the composition of verifiers (below). It can be used only for the verifier constructed for the Max Clique result.

3.3 Recursive proof verification

We briefly outline the recursive proof checking scenario. We start with a two-prover proof system for NP, having an arbitrary small constant soundness error, and whose verifier V_{out} behaves as follows: On input x of length n, V_{out} computes (using internal coin tosses) questions q and q_1 to ask the two provers. V_{out} also constructs a predicate $h : \Sigma^l \to \Sigma$ and a function $\sigma : \Sigma^l \to \Sigma^{l_1}$ which will be used to evaluate the answers of the two provers in the following way. Let $a \in \Sigma^l$ and $a_1 \in \Sigma^{l_1}$ be the answers of the two provers, then V_{out} accepts only if $h(a) = 0$ and $\sigma(a) = a_1$.

To simulate the above computations of V_{out} we will create inner verifiers V_{in} which access two oracles A and A_1. These oracles are supposed to represent encodings according to the long code of the strings $a \in \Sigma^l$ and $a_1 \in \Sigma^{l_1}$ respectively. The inner verifier is also given h and σ and should verify the following (by few probes into A and A_1):

– *Codeword Test:* Verify there exists a string a such that $\mathrm{Dist}(A_{(h,0)}, E(a)) \leq 1/2 - \delta$, where $\delta > 0$.
– *Projection Test:* Verify that $\mathrm{Dist}(A_1, E(\sigma(a))) \leq 1/2 - \delta$.

Note that the first condition refers to the folded version of A. and recall that, by Proposition 3.2, the codeword test makes testing $h(a) = 0$ (the "circuit test") superflous.[10] If a inner verifier performs the above task with soundness error ρ, then it is considered (ρ, δ)-*good*. Any such inner verifier (i.e., for any $\rho < 1$ and $\delta > 0$) can be composed with V_{out} and yield a meaningful result (i.e., the soundness error of the composed verifier will be $O(\epsilon/\delta^4) + \rho$, where ϵ is the soundness error of V_{out}). In what follows we attempt to construct inner verifiers that minimizes the soundness ρ as we let $\delta \to 0$. We first start with some elementary tests which help solve the above tasks — then in the following subsections we show how to combine them appropriately so as to achieve the different hardness results.

3.4 The atomic tests

The aim here is to test various properties of functions A: $\mathcal{F}_l \to \Sigma$ and A_1: $\mathcal{F}_{l_1} \to \Sigma$ to which the tester has oracle access. These tests will be the basis for several forthcoming proof systems. In all our applications, A is supposed to be (a folding of) an encoding of the answer of the first prover (in a two-prover proof system) and A_1 is supposed to be the encoding of the answer of the second prover. However, the exposition and analysis of these tests, in this subsection, is independent of the usage of the codes in our proof systems. We remark that for the applications to Max-SNP problems it is important to have the best possible analysis of our atomic tests, and what follows strives to this end.

The first task that concerns us is to design a test which, with high probability, passes if and only if A is close to an evaluation operator (i.e., a valid codeword). The idea is to exploit the characterization of Proposition 3.1. Accordingly, the first step is to check that A is almost linear.

ATOMIC LINEARITY TEST. The *atomic linearity test* with parameters f_1 and f_2, denoted $\mathbf{LinTest}(A; f_1, f_2)$, consists of checking whether $A(f_1) + A(f_2) = A(f_1 + f_2)$. The analysis of this simple test, initiated by Blum et. al. [BLR] and continued by [BGLR], turned out to be very involved and was resolved recently by Bellare et. al. [BCHKS].

Lemma 3.3 [BLR, BGLR, BCHKS]: Let A: $\mathcal{F}_l \to \Sigma$ and let $x = \mathrm{Dist}(A, \textsc{Lin})$. Then,

$$\Pr_{f_1,f_2 \overset{R}{\leftarrow} \mathcal{F}_l} [\, \mathbf{LinTest}(A; f_1, f_2) \quad \text{rejects}\,] \geq \Gamma_{\mathrm{lin}}(x)$$

where $\Gamma_{\mathrm{lin}}(x) \overset{\mathrm{def}}{=} 3x - 6x^2$ if $x \leq \frac{5}{16}$, $\Gamma_{\mathrm{lin}}(x) \overset{\mathrm{def}}{=} \frac{45}{128}$ for $\frac{5}{16} \leq x \leq \frac{45}{128}$, and $\Gamma_{\mathrm{lin}}(x) \overset{\mathrm{def}}{=} x$ otherwise.

ATOMIC RESPECT OF MONOMIAL BASIS TEST (RMB-TEST). Having determined that A is close to linear, the *atomic respect of monomial basis* test makes sure that the linear function close to A respects the monomial basis. The test, $\mathbf{MBTest}(A; f_1, f_2, f_3)$, taking parameters f_1, f_2, f_3 consists of checking whether $A(f_1) \cdot A(f_2) = A(f_1 \cdot f_2 + f_3) - A(f_3)$. Actually, the test performs self-correction for a test of multiplication, yet it can be shown that this establishes the desired property of respecting the monomial basis. Namely,

Lemma 3.4 Let $A, \tilde{A}$: $\mathcal{F}_l \to \Sigma$ with $\tilde{A}$ linear but NOT respecting the monomial basis. Suppose that A satisfies $A(f + \overline{1}) = A(f) + 1$, for all $f \in \mathcal{F}_l$, and let $x = \mathrm{Dist}(A, \tilde{A})$. Then

$$\Pr_{f_1,f_2,f_3 \overset{R}{\leftarrow} \mathcal{F}_l} [\, \mathbf{MBTest}(A; f_1, f_2, f_3) \quad \text{rejects}\,] \geq \Gamma_{\mathrm{rmb}}(x)$$

where $\Gamma_{\mathrm{rmb}}(x) \overset{\mathrm{def}}{=} \frac{3}{8} - \frac{7}{4}x + \frac{5}{2}x^2 - x^3$.

[10] This observation (as well as letting $\delta \to 0$) is important for obtaining the best results for MaxSNP problems. For establishing the Max Clique result, we may use a circuit test (as well as $\delta > 1/4$) and yet obtain the same result we have.

Remark: Functions that are folded over the constant function $\overline{1}$ satisfy the condition of the hypothesis. (This is a case in our applications.) For other functions, one needs augment the RMB-test so that it checks $A(f+\overline{1}) = A(f)+1$ for a random f.

Having established that A is close to some evaluation operator E_a, we now want to test two things. The first is that $h(a) = 0$ for some predetermined function h. This test which would normally be implemented by "self-correction" (i.e., evaluating $h(a)$ by uniformly selecting $f \in \mathcal{F}_l$ and computing $A(f+h) - A(f)$) is not needed here, since in our applications we will use an $(h, 0)$-folding of A instead of A (see Prop. 3.2). Thus, it is left to test that the two oracles are consistent in the sense that A_1 encodes the projection of the word encoded in A.

ATOMIC PROJECTION TEST. This test checks that the second function A_1 is not too far from the evaluation operator E_{a_1} where $a_1 = \sigma(a)$ is a function of the string a whose evaluation operator is close to A. Here, unlike previous works (for instance [BeSu]), σ may be an arbitrary mapping from Σ^l to Σ^{l_1} rather than being a projection. Thus, the name "projection test" is adopted merely for historical reasons. The test, $\mathbf{ProjTest}_\sigma(A, A_1; f, g)$, takes parameters f and g and checks whether $A_1(g) = A(g \circ \sigma + f) - A(f)$.

Lemma 3.5 Let $A\colon \mathcal{F}_l \to \Sigma$ and $\sigma\colon \Sigma^l \to \Sigma^{l_1}$ be a function. Let $a \in \Sigma^l$, $x = \mathrm{Dist}(A, E_a)$ and $y = \mathrm{Dist}(A_1, E_{\sigma(a)})$. Then

$$\Pr_{f \overset{R}{\leftarrow} \mathcal{F}_l\,;\, g \overset{R}{\leftarrow} \mathcal{F}_{l_1}} [\, \mathbf{ProjTest}_\sigma(A, A_1; f, g) \quad \text{rejects} \,] \geq (1-2x) \cdot y$$

3.5 Minimizing the number of queries by recycling

Performing each of the atomic tests sufficiently many times yields an inner verifier with error smaller than $1/2$ and thus a pcp system for NP follows. To minimize the query complexity, we adopt the observation of Bellare et. al. [BGLR] by which the same queries can be used for different types of tests without effecting the error analysis.[11] Namely, we select $m \overset{\text{def}}{=} \max\{2N_1, 3N_2, N_3\}$ random functions $f_1, ..., f_m \in \mathcal{F}_l$, and N_3 random functions g_i's in $\mathcal{F}_{l_1}$. We perform the linear test N_1 times, in the i^{th} time with parameters f_{2i-1}, f_{2i}. Likewise, the MBtest is performed N_2 times (in the i^{th} time with $f_{3i-2}, f_{3i-1}, f_{3i}$), and the projection tests is performed N_3 times (in the i^{th} time with f_i and g_i). Setting $N_1 = 3$, $N_2 = 3$ and $N_3 = 2$ suffices for proving the worst case bound (below) and for the average case bound we pick N_2 and N_3 at random in $\{2, 3\}$ and $\{1, 2\}$, respectively.

Theorem 3.6 There exists a pcp system for NP with logarithmic randomness, error probability 1/2 and query complexity 19 (and 15.6 on the average). Furthermore, the free-bit complexity of this system is 11.

3.6 Selecting a random test – the MaxSNP inner verifier

Here we build an inner verifier which we later use for the MaxSNP hardness results. We are going to select one of the three atomic tests at random (not necessarily uniformly) and perform it only once (for uniformly chosen parameters). Specifically, with some probability p_1 we perform a Linear Test, with probability p_2 a MBtest and otherwise (with probability $p_3 = 1 - p_1 - p_2$) we perform a projection test. We stress that these tests are performed on a folded version of A and on A_1. (The folding is over $(h, 0)$ and $(\overline{1}, 1)$ and virtual access to the folded version is provided by accessing the actual oracle according to the definition of folding.) This defines an inner verifier, U_{SNP}, which we analyse to show has the following properties:

Lemma 3.7 Suppose $\delta > 0$. Suppose $p_1, p_2, p_3 \in [0, 1]$ satisfy $p_1 + p_2 + p_3 = 1$. Then the inner verifier U_{SNP} is $(\max_i\{\rho_i\}, \delta)$-good, where $\rho_1 \overset{\text{def}}{=} 1 - (\frac{1}{2} - \delta) \cdot p_1$, $\rho_2 \overset{\text{def}}{=} 1 - \min_{x \leq \frac{1}{2} - \delta} [\, \Gamma_{\text{lin}}(x) \cdot p_1 + \Gamma_{\text{rmb}}(x) \cdot p_2 \,]$ and $\rho_3 \overset{\text{def}}{=} 1 - \min_{x \leq \frac{1}{2} - \delta} [\, \Gamma_{\text{lin}}(x) \cdot p_1 + (\frac{1}{2} - \delta)(1 - 2x) \cdot p_3 \,]$.

We optimize the above expression to find the best possible values of p_1, p_2, p_3 and obtain $p_2 \approx 0.364$ and soundness error smaller than 0.864. Next, we implement the computation of the tests using gadgets of the MaxSNP problem at hand, which in turn induces a reduction of NP to the problem at hand. We observe that we need to implement only two types of gadgets – a *parity check* (*PC*), which given three variables checks that their parity equals 0, and a *Monomial Basis check* (*MBC*), which given a, b, c, d checks that $a \cdot b = c + d$. (Observe that the Projection Test also amounts to a parity check.) In case of the MaxSAT problems the gadgets are small formulii (of the appropriate type) whereas in the MaxCUT case the gadgets are graphs and Boolean values are associated to the vertices by any potential cut. The hardness results mentioned in Theorem 1.5, parts (2–4), follows.

[11] This is the case since the error analysis breaks into cases and in each case only one of the test types is used.

3.7 Two free-bits and inapproximability of vertex cover

It is known that approximating the minimum vertex cover of a graph to within a $1+\epsilon$ factor is hard, for some $\epsilon > 0$ [PaYa, ALMSS]. Our initial attempt to obtain a bound on ϵ uses VC-gadgets that implement the various tests in the inner verifier $U_{\rm SNP}$, analogously to the way it was done in the previous section. This yields a lower bound of $\epsilon > \frac{1}{54} > 0.018$. However, a stronger result is obtained via free-bit complexity. Specifically, we apply the FGLSS-reduction to a proof system (for NP) in which the free-bit complexity is the lowest one possible: which, by the results of Section 5, is 2 free-bits. Consequently, the clique size, in case the original input is in the language, is at least one fourth (1/4) of the size of the graph which means that translating clique-approximation factors to VC-approximation factors yields only a loss of one third. Since the FGLSS-transformation translates the completeness/soundness ratio to the gap-factor for clique approximation, our first goal is to construct for NP a proof systems which uses two free-bits and has soundness error as low as possible. The proof system of subsection 3.5 uses 11 free-bits and achieves soundness error less than $1/2$. The reader may observe that, using the clique to VC reduction, it is not worthwhile to use the proof system of subsection 3.5 or any proof systems which achieves a soundness error of $1/2$ at the cost of 5 free-bits or more. The 2-free-bit pcp system for NP is obtained by splitting the MBtest into its two natural parts: a multiplication test (i.e., $A(f_1)\cdot A(f_2) = A(f_1 \cdot f_2)$) and a corresponding self-corrector (i.e., $A(f_1\cdot f_2) = A(f_1\cdot f_2 + f_3) - A(f_3)$). The inner verifier decides at random which of the four tests to perform; actually, their are three cases as in one case the verifier performs both the linear test and the pure multiplication test (using the same paramters/functions and so maintaining a count of 2 free-bits in each case). This yields the following theorem and the hardness of Vertex Cover mentioned in Theorem 1.5, part (1), follows.

Theorem 3.8 There exists a pcp system for NP with logarithmic randomness, free-bit complexity 2, and error probability 0.884464. Furthermore, the query complexity is 4 (and less than 3.5 on the average).

Using the above mentioned 4 tests, so that only a single test is performed in each case, we obtain

Theorem 3.9 There exists a pcp system for NP with logarithmic randomness, query complexity 3, and error probability 0.8999. Furthermore, the free-bit complexity is 2.

3.8 Using related queries – two amortized free-bits

Here the inner verifier invokes each atomic test many times but with related parameters/functions. Specifically, we select uniformly m functions in $\mathcal{F}_l$ and m functions in $\mathcal{F}_{l_1}$ and invoke each test ${\rm poly}(2^m)$ times. In a typical invokation we use as parameters functions obtained by linear combinations of the corresponding m functions. For example, suppose we have selected $f_1, ..., f_m \in \mathcal{F}_l$. Then for every two (different and non-trivial) linear combinations, $f' = \sum_{i=1}^m c'_i \cdot f_i$ and $f'' = \sum_{i=1}^m c''_i \cdot f_i$, we invoke the linear test with parameters f' and f''. Clearly, the free-bit count in this system is $2m$, as all queries are determined by the value of the $2m$ functions selected above. On the other hand, the various invokations are sufficiently random so that a Law of Large Numbers can be invoked[12] and consequently the error probability of the resulting inner verifier can be bounded by $O(2^{-m})$. Thus,

Theorem 3.10 There is a constant c such that, for every $m \geq 3$, there exists a pcp system for NP with logarithmic randomness, free-bit complexity $2m$, and error probability $c \cdot 2^{-m}$. Thus, for any $\epsilon > 0$, there exists a pcp system for NP with logarithmic randomness and amortized free-bit complexity $2+\epsilon$.

This yields Theorem 1.3.

4 On the Limitations of Some Common Approaches

Recall that in Section 3.3 we outlined the basic tasks of an inner verifier. They are central to all existing ("low-complexity") pcps. In this section we provide lower bounds on the free-bit complexity of these tasks. Specifically, we consider the task of checking that a string (given by oracle access) is close[13] to a valid codeword and the task of checking that one oracle is an encoding of a projection of a string encoded by a second oracle. Loosely speaking, we show that each of these tasks has amortized free-bit complexity of at least one (and this is tight by the codes and tests presented in Section 3). Furthermore, we show that the amortized free-bit complexity of performing both tasks (with

[12]Specifically, we can isolate a set of $\Omega(2^m)$ invokations that are pairwise independent and apply Chebyshev's Inequality.

[13] Here 'close' means closer than half the distance of the code. Indeed, our Max Clique result may use a verifier that makes such a test, but other verifiers in the paper perform a weaker form of a codeword test which is required to detect only strings which are at distance almost equal to the distance of the code.

respect to the same given oracles) is at least two (which is also tight). We consider these bounds as an indication that one will have to depart significantly from the known techniques in order to obtain lower (than two) amortized free-bit complexity for $\mathcal{NP}$. One possible avenue which may lead to a amortized free-bit complexity of 1 is to perform a relaxed form of the codeword test (see footnote above) at free-bit complexity 0.

Definition 4.1 The *absolute distance* between two words $w, u \in \{0,1\}^n$, denoted $d(w,u)$, is the number of bits on which w and u disagree. We say that the code $E : \{0,1\}^m \to \{0,1\}^n$ has *absolute distance* d if for every $x \neq y \in \{0,1\}^m$ the absolute distance between $E(x)$ and $E(y)$ is at least d. The absolute distance between a word w and a code E, denoted $d_E(w)$, is defined as the minimum absolute distance between w and a codeword of E. A *codeword test (with respect to E)* is an oracle machine, T, such that $T^{E(a)}(R)$ accepts for all a, R. The error probability of T is defined to be $\max_{A\in\{0,1\}^n : d_E(A)\geq \lfloor d/2 \rfloor}\{\Pr_R\left[T^A(R) \text{ accepts}\right]\}$.

Lemma 4.2 Let $E : \{0,1\}^m \mapsto \{0,1\}^n$ be a code of absolute distance $d > 1$, and let T be a codeword test with respect to E which uses $f_{\rm av}$ free-bits on the average. Then, T has error probability at least $\frac{1}{F} - \frac{1}{M}$, where $F = 2^{f_{\rm av}}$ and $M = 2^m$, and its amortized free bits complexity is at least $1 - F/M$.

Definition 4.3 Let E_1: $\{0,1\}^m \to \{0,1\}^n$ and E_2: $\{0,1\}^k \to \{0,1\}^{n'}$ be two codes and let $\pi : \{0,1\}^m \to \{0,1\}^k$ be a function. A *projection test (with respect to the above)* is a two-oracle machine, T, such that $T^{E_1(x),E_2(\pi(x))}(R)$ accepts for all x, R. The error probability of T is defined to be

$$\max_{a,b:\pi(a)\neq b}\{\Pr_R\left[T^{E_1(a),E_2(b)}(R) \text{ accepts}\right]\}.$$

Lemma 4.4 Let E_1, E_2 and π be as above, and T be a projection test with respect to them, which uses $f_{\rm av}$ bits on the average. Then, T has error probability at least $\frac{1}{F} - \frac{1}{K}$, where $K = |\{\pi(a) | a \in \{0,1\}^m\}|$ and $F = 2^{f_{\rm av}}$, and thus its amortized free-bit complexity is at least $1 - \frac{F}{K}$.

Finally, we define a tester which combines the two tests above: i.e., T takes two oracles A and B and performs a codeword test on A and a projection test on the pair A, B.

Definition 4.5 Given a code $E_1 : \{0,1\}^m \to \{0,1\}^n$ of absolute distance d a code $E_2 : \{0,1\}^k \to \{0,1\}^{n'}$ and a function $\pi : \{0,1\}^m \to \{0,1\}^k$, a *Combined Test* for (E_1, E_2, π) is a two-oracle machine T such that $T^{E_1(a),E_2(\pi(a))}(R)$ accepts on all a, R. The error of the test is defined to be

$$\max_{\{(A,B)\in S\}}\left\{\Pr_R\left[T^{A,B}(R) \text{ accepts}\right]\right\}.$$

where S contains (A, B) iff either $d_{E_1}(A) \geq \lfloor d/2 \rfloor$ or $A = E_1(a)$ and $B = E_2(b)$ for some $b \neq \sigma(a)$.

Lemma 4.6 Let E_1 (of distance > 1), E_2 and π be as above and T be a combined codeword and projection test with respect to them. Suppose that, given access to a pair of oracles, of length n and n' respectively, T accepts at most $F^2 = 2^{2f}$ possible patterns for each possible sequence of coin tosses. Then, T has error probability at least $\frac{1}{64F} - \frac{1}{4K} - \frac{1}{2M}$, where $K = 2^k$ and M is the minimum, over all $b \in \{0,1\}^k$, of the number of $a \in \{0,1\}^m$ projected by π to b. Thus, the amortized free-bit complexity of T is at least $2 - O(\frac{1}{f} + \frac{F}{\min\{K,M\}})$.

5 The True Complexity of PCP and FPCP

Proposition 5.1 (pcp systems with at most 3 queries):
(1) $\forall\, c, s : \mathcal{Z}^+ \to [0,1]$ so that s is strictly less than c, $\mathrm{PCP}_{c,s}[\log, 1] = \mathrm{P}$.
(2) $\forall\, s : \mathcal{Z}^+ \to [0,1]$ strictly less than 1, $\mathrm{PCP}_{1,s}[\log, 2] = \mathrm{P}$. In contrast for some $0 < s < c < 1$, $\mathrm{PCP}_{c,s}[\log, 2] = \mathrm{NP}$.
(3) $\mathrm{PCP}_{1,0.9}[\log, 3] = \mathrm{NP}$. In contrast, $\forall s \leq 0.299$, $\mathbf{na}\mathrm{PCP}_{1,s}[\log, 3] = \mathrm{P}$ where $\mathbf{na}\mathrm{PCP}$ is a restriction of PCP in which the verifier is required to be non-adaptive.

Item (2) is folklore. The bound obtained in the second part of Item (3), let alone that it is restricted to the non-adaptive case, is weaker than what can be proven for MIP proof systems (see below). This contrast may perhaps provide a testing ground to separate PCP from MIP, a question raised by [BGLR]. Below, $\mathrm{MIP}_{\cdot,\cdot}[\cdot, p]$ denotes the class of languages accepted by a one-round p-prover protocol in which is prover answer is a single bit.

Proposition 5.2 (mip systems with at most 3 queries):
(1) $\mathrm{MIP}_{c,s}[r,p] \subseteq \mathrm{MIP}_{c,2s}[r,p-1]$.
(2) For $s < 1/2$, $\mathrm{MIP}_{1,s}[\log, 3] = \mathrm{P}$.

Lemma 5.3 For all functions c, s, q, r such that $\frac{c}{s} > 2^q$,

$$\mathrm{PCP}_{c,s}[r,q] \subseteq \mathrm{RTIME}(\mathrm{poly}(n,r,q,(c-2^q s)^{-1})$$

Furthermore, if r and q are both logarithmically bounded then $\mathrm{PCP}_{c,s}[r,q] = \mathrm{P}$.

Corollary 5.4 $\forall\, c>0$, $\overline{\mathrm{PCP}}_c[\,\log n,\ 1\,] = \mathrm{P}$.

Before turning to free-bit complexity, we comment that results analogous to Proposition 5.1, where PSPACE plays the role of P and NEXP the role of NP, can be derived for the classes $\mathrm{PCP}_{\cdot,\cdot}(\mathrm{poly},\cdot)$. Furthermore, for functions c, s so that $c(n) > s(n) + 1/\mathrm{poly}(n)$, we obtain $\mathrm{PCP}_{c,s}[\mathrm{poly}, 1] \subseteq \mathrm{AM}$. In light of the last item of Proposition 5.5, this result may be hard to strenthen.

Proposition 5.5 Let $s : \mathcal{Z}^+ \to [0,1]$ be a function strictly smaller than 1.
(1) $\mathrm{FPCP}_{1,s}[\log, 1] = \mathrm{P}$. In contrast, $\mathrm{FPCP}_{0.5,0.45}[\log,1] = \mathrm{NP}$, $\mathrm{FPCP}_{0.25,0.22}[\log,0] = \mathrm{NP}$, and $\mathrm{FPCP}_{1,0.95}[\log, f] = \mathrm{NP}$ for $f = \log_2 3$.
(2) $\mathrm{FPCP}_{1,s}[\mathrm{poly}, 0] \subseteq \mathrm{coNP}$ and $\mathrm{FPCP}_{1,s}[\mathrm{poly}, 1] \subseteq \mathrm{PSPACE}$.
(3) Graph Non-Isomorphism and Quadratic Non-Residousity have pcp systems, with perfect completeness and soundness bound $1/2$, in which the verifier makes a single query and this query is free.

The last item follows from [GMR, GMW].

6 Transformations of FPCP Systems

In this section we show several useful transformations which can be applied to pcp systems. We concentrate on the free bit complexity, and introduce an additional parameter into the notation – the proof length (i.e., $\mathrm{FPCP}_{c,s}[r,f,l]$ refers to randomness r, free bit f and proof length l). We start by stating the simple fact that the ratio between the completeness and soundness bounds (also referred to as gap) is amplified (i.e., raise to the power k) when one repeats the pcp system (k times). Note, however, that if the original system is not perfectly complete then the completeness bound in the resulting system gets decreased.

Proposition 6.1 (gap amplification):

$$\forall k, c, s \quad \mathrm{FPCP}_{c,s}[r,f,l] \subseteq \mathrm{FPCP}_{c^k,s^k}[kr, kf, l].$$

Next, we show that in some sense the randomness-complexity of a proof system need not be higher than logarithmic in the length of the proofs/oracles employed. The symbol $\leq_K^R$ denotes a randomized Karp reduction.

Proposition 6.2 (reducing randomness): There exists a constant $\gamma > 0$ so that for every two functions $s, \epsilon : \mathcal{Z}^+ \to [0,1]$, $\mathrm{FPCP}_{1,s}[r,f,l] \leq_K^R \mathrm{FPCP}_{1,s'}[r', f, l]$, where $s' = (1+\epsilon)s$ and $r' = \gamma + \log_2(l/\epsilon^2 s)$.

An analogous statement for two-sided error pcp is omitted. Combining Propositions 6.1 and 6.2, we obtain the following corollary which plays a central role in deriving clique approximation results via the FGLSS method [14]–

Corollary 6.3 For every $k : \mathcal{Z}^+ \to \mathcal{Z}^+$, $\overline{\mathrm{FPCP}}[\,r,\ f\,] \leq_K^R \mathrm{FPCP}_{1,2^{-k}}[r + k + \log, kf]$.

An alternative error reduction procedure, which allows to obtain inapproximability results under $\mathrm{P} \neq \mathrm{NP}$, follows (stated here only for the one-sided error case)

Proposition 6.4 $\forall \epsilon, s > 0,\ \forall k : \mathcal{Z}^+ \to \mathcal{Z}^+ \quad \mathrm{FPCP}_{1,s}[r,f] \subseteq \mathrm{FPCP}_{1,s^k}[r + (2+\epsilon)\cdot k + \log, kf]$.

The following transformation is analogous to the randomized layering procedure for the clique promise problem (i.e., Proposition 2.5). In view of the relation between FPCP and the clique promise problem (shown in Section 2), this analogy is hardly surprising. In this transformation the acceptance probability bounds are pushed higher at the expense of increasing the free-bit complexity.

[14]In a typical application, $r = O(\log n)$ and one sets k to be a large multiple of r. The FGLSS-graph corresponding to the resulting pcp system will have size $N = 2^{(r+k+O(1))+kf}$ and a gap in clique size of factor 2^k, which can be rewritten as $N^{1/(1+f+\epsilon)}$ where $\epsilon = r/k$.

Proposition 6.5 (increasing acceptance probabilities): For all functions $c, s : \mathcal{Z}^+ \to [0,1]$, and $r, f, m : \mathcal{Z}^+ \to \mathcal{Z}^+$, $\mathrm{FPCP}_{c,s}[r,f] \leq_K^R \mathrm{FPCP}_{c',s'}[r, f + \log_2 m]$, where $c' = 1 - 4(1-c)^m$ and $s' = m \cdot s$.

The following transformation has an opposite effect than the previous one, reducing the free-bit complexity at the expense of lowering the bounds on acceptance probability.

Proposition 6.6 (decreasing acceptance probabilities): For all functions $c, s : \mathcal{Z}^+ \to [0,1]$, and $r, f, k : \mathcal{Z}^+ \to \mathcal{Z}^+$, if $L \in \mathrm{FPCP}_{c,s}[r,f]$ with a verifier for which every random-pad has at least 2^k accepting configurations, then $L \in \mathrm{FPCP}_{\frac{c}{2^k}, \frac{s}{2^k}}[r+k, f-k]$.

References

[Ar] S. Arora. Reductions, Codes, PCPs and Inapproximability. These proceedings.

[ALMSS] S. Arora, C. Lund, R. Motwani, M. Sudan and M. Szegedy. Proof verification and intractability of approximation problems. *FOCS*, 1992.

[ArSa] S. Arora and S. Safra. Probabilistic checking of proofs: a new characterization of NP. *FOCS*, 1992.

[BFL] L. Babai, L. Fortnow and C. Lund. Non-deterministic Exponential time has two-prover interactive protocols. *FOCS*, 1990.

[BFLS] L. Babai, L. Fortnow, L. Levin, and M. Szegedy. Checking computations in polylogarithmic time. *STOC*, 1991.

[BaEv] R. Bar-Yehuda and S. Even. A local ratio theorem for approximating the weighted vertex cover problem. In *Analysis and Design of Algorithms for Combinatorial Problems* Vol. 25 of Annals of Discrete Math, Elsevier, 1985.

[BCHKS] M. Bellare, D. Coppersmith, J. Håstad, M. Kiwi and M. Sudan. Linearity testing in characteristic two. These proceedings.

[BGLR] M. Bellare, S. Goldwasser, C. Lund and A. Russell. Efficient probabilistically checkable proofs and applications to approximation. *STOC*, 1993. (See also Errata sheet in *STOC*, 1994).

[BGS] M. Bellare, O. Goldreich and M. Sudan. Free Bits, PCPs and Non-Approximability – Towards Tight Results. Auguat 1995 (replacing previous version of May 1995). Available from *ECCC*, *Electronic Colloquium on Computational Complexity*, via WWW using `http://www.eccc.uni-trier.de/eccc/`.

[BeSu] M. Bellare and M. Sudan. Improved non-approximability results. *STOC*, 1994.

[BGKW] M. Ben-Or, S. Goldwasser, J. Kilian and A. Wigderson. Multi-Prover interactive proofs: How to remove intractability assumptions. *STOC*, 1988.

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

[Bl] A. Blum. Algorithms for approximate graph coloring. Ph. D Thesis, MIT, 1991.

[BLR] M. Blum, M. Luby and R. Rubinfeld. Self-testing/correcting with applications to numerical problems. *Journal of Computer and System Sciences* Vol. 47, pp. 549–595, 1993.

[BoHa] R. Boppana and M. Haldórsson. Approximating maximum independent sets by excluding subgraphs. BIT, Vol. 32, No. 2, 1992.

[FRS] L. Fortnow, J. Rompel and M. Sipser. On the power of multiprover interactive protocols. Proceedings of the 3rd Structures, IEEE, 1988.

[FeGo] U. Feige and M. Goemans. Approximating the value of two prover proof systems, with application to Max-2SAT and Max-DICUT. *ISTCS*, 1995.

[FGLSS] U. Feige, S. Goldwasser, L. Lovász, S. Safra, and M. Szegedy. Approximating clique is almost NP-complete. *FOCS*, 1991.

[FeKi] U. Feige and J. Kilian. Two prover protocols – Low error at affordable rates. *STOC*, 1994.

[Fu] M. Furer. Improved hardness results for approximating the chromatic number. These proceedings.

[GJS] M. Garey, D. Johnson and L. Stockmeyer. Some simplified NP-complete graph problems. *Theoretical Computer Science* **1**, pp. 237–267, 1976.

[GoWi1] M. Goemans and D. Williamson. New 3/4-approximation algorithm for MAX SAT. *Proceedings of the 3rd Mathematical Programming Society Conference on Integer Programming and Combinatorial Optimization*, 1993.

[GoWi2] M. Goemans and D. Williamson. .878 approximation algorithms for Max-CUT and Max-2SAT. *STOC*, 1994.

[GMW] O. Goldreich, S. Micali, and A. Wigderson. Proofs that yield nothing but their validity and a methodology of cryptographic protocol design. *FOCS*, 1986.

[GMR] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proofs. *SIAM J. Computing* Vol 18, No. 1, 186–208, 1989.

[KLS] S. Khanna, N. Linial and S. Safra. On the hardness of approximating the chromatic number. *ISTCS*, 1993.

[LuYa] C. Lund and M. Yannakakis. On the hardness of approximating minimization problems. *STOC*, 1993.

[MoSp] Monien and Speckenmeyer. Some further approximation algorithms for the vertex cover problem. *Proceedings of CAAP 83*, Lecure Notes in Computer Science Vol. 159, Springer-Verlag, 1983.

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

[Raz] R. Raz. A parallel repetition theorem. *STOC*, 1995.

[Ya] M. Yannakakis. On the approximation of maximum satisfiability. *SODA*, 1992.

[Zu] D. Zuckerman. NP-Complete Problems have a version that is hard to Approximate. *Structures*, 1993.