Appears in *Proceedings of the Twenty Sixth Annual Symposium on the Theory of Computing*, ACM, 1994.

Improved Non-Approximability Results

Mihir Bellare[*] Madhu Sudan[†]

March 15, 1994

Abstract

We indicate strong non-approximability factors for central problems: $N^{1/4}$ for Max Clique; $N^{1/10}$ for Chromatic Number; and 66/65 for Max 3SAT.

Underlying the Max Clique result is a proof system in which the verifier examines only three "free bits" to attain an error of 1/2. Underlying the Chromatic Number result is a reduction from Max Clique which is more efficient than previous ones.

[*]Department of Computer Science & Engineering, Mail Code 0114, University of California at San Diego, 9500 Gilman Drive, La Jolla, CA 92093. E-mail: `mihir@cs.ucsd.edu`

[†]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

Max Clique is amongst the most important combinatorial optimization problems. Unfortunately it is NP-hard [16], and attention since this discovery has thus focused on approximation algorithms. Yet the best known ones can approximate the max clique size of an N node graph only to within a factor of $N^{1-o(1)}$ [11], scarcely better than the trivial factor of N.

The first twenty years following the NP-hardness discovery brought little understanding of why this is so. Today we know a lot more. The results of [12, 3, 2] indicated the existence of a constant $\alpha > 0$ for which N^{α} factor approximations are unlikely to be achievable. Recent work [7, 13] has been able to show that $\alpha \geq 1/15$.

One of the most basic goals in computational complexity theory is to find the *exact* complexity of problems. With the above advances, the time seems ripe to attempt something which four years ago may have seemed unthinkable; namely, to find the exact complexity of approximating Max Clique. That is, let α_{true} be the value such that Max Clique can be approximated within $N^{\alpha_{\mathrm{true}}}$ but no better. We know that $1/15 \leq \alpha_{\mathrm{true}} \leq 1 - o(1)$. But where in this range does α_{true} lie?

Our approach has been to try to improve the lower bound. That is, continuing the work of [7, 13], we wanted to increase the value of the constant $\alpha > 0$ for which a factor N^{α} non-approximability result could be shown. We were faced with the difficulty that probabilistic proof checking techniques seemed to have been pushed to their limits. How could one go further? As we will see, the key was the build proof systems which are efficient not under "standard" proof checking measures, but under a new one suggested by [13].

Our proof systems have enabled us to obtain results strong enough to be surprising. For the first time, we feel that the gap between upper and lower bounds may be bridged. Indeed, we feel that our techniques have the potential to show hardness of approximation within $N^{1-o(1)}$, leading us to conjecture —perhaps somewhat pessimistically and certainly in opposition to other conjectures![1]— that $\alpha_{\mathrm{true}} = 1 - o(1)$ is where the truth lies; and, furthermore, that probabilistic proof checking techniques will be able to show this.

We obtain also strong results for the chromatic number problem, and some improvements for Max 3SAT.

To simplify the presentation, we have elected to first describe the results and techniques and then give detailed credits in Section 1.3.

1.1 Non-approximability results

Denote by $\omega(G)$ the size of a maximum clique in G. Recall that an algorithm approximates Max Clique within factor $g(N) \geq 1$ if on input an N-node graph G it outputs a number which is at least $\omega(G)/g(N)$ but at most $\omega(G)$. Recall also that a function of $T(n)$ is "quasi-polynomial" if there is a constant c such that $T(n) \leq n^{\log^c n}$ for all large enough n. We denote by $\widetilde{\mathrm{P}}, \mathrm{N}\widetilde{\mathrm{P}}, \mathrm{R}\widetilde{\mathrm{P}}$ etc. the analogues of the usual complexity classes in the quasi-polynomial time domain. Our main result can be stated right away. It indicates that we should not hope to be able to approximate Max Clique to a factor better than $N^{1/4}$.

[1] It has been conjectured that the Lovász theta function approximates Max Clique within $N^{1/2}$.

Theorem 1.1 Suppose $\widetilde{\mathrm{NP}} \neq \mathrm{co}\widetilde{\mathrm{RP}}$. Then there is no polynomial time algorithm to approximate Max Clique within $N^{\frac{1}{4}-o(1)}$.

The assumption $\widetilde{\mathrm{NP}} \neq \mathrm{co}\widetilde{\mathrm{RP}}$ can be decreased at the cost of a decrease in the factor shown hard. For example, for any constant $\delta > 0$, assuming NP $\neq$ coRP we can show that $N^{\frac{1}{5}-\delta}$ factor approximation is impossible; and assuming just P $\neq$ NP we can use the constructions of [15] to show that $N^{\frac{1}{6}-\delta}$ factor approximation is impossible.

The technical underpinning of the above result is a new proof system which will be discussed in Section 1.2. We turn now to the chromatic number problem.

Denote by $\chi(H)$ the chromatic number of a graph H. Recall that an algorithm approximates the chromatic number within factor $g(N) \geq 1$ if on input an N-node graph H it outputs a number which is at most $\chi(H) \cdot g(N)$ but at least $\chi(H)$. The above mentioned proof systems combined with known results would directly imply that $N^{\frac{1}{16}-o(1)}$ factor approximation of the chromatic number is impossible unless $\widetilde{\mathrm{NP}} = \mathrm{co}\widetilde{\mathrm{RP}}$. To do even better, we provide a new reduction of Max Clique to chromatic number. Its features are discussed in Section 1.2. It enables us to show the following.

Theorem 1.2 Suppose $\widetilde{\mathrm{NP}} \neq \mathrm{co}\widetilde{\mathrm{RP}}$. Then there is no polynomial time algorithm to approximate the chromatic number within $N^{\frac{1}{10}-o(1)}$.

Again, the assumption can be traded off with the factor shown hard. For any constant $\delta > 0$, assuming NP $\neq$ coRP we can show that $N^{\frac{1}{13}-\delta}$ factor approximation is impossible; and assuming P $\neq$ NP we can show that $N^{\frac{1}{14}-\delta}$ factor approximation is impossible.

Max 3SAT is the problem of determining the maximum number of simultaneously satisfiable clauses in a 3cnf-formula. It is a canonical Max SNP complete problem [22]. An algorithm approximates it within $1+\gamma$ if it produces a value which is at least $1/(1+\gamma)$ times optimal, and at most optimal. An algorithm due to [24] achieves $\gamma = 1/3$.

Theorem 1.3 Suppose $\widetilde{\mathrm{NP}} \neq \widetilde{\mathrm{P}}$. Then there is no polynomial time algorithm to approximate Max 3SAT within $1 + 1/65$.

The factor decreases to $1 + 1/73$ if the assumption is just P $\neq$ NP. See Section 5 for a discussion of the proof.

1.2 Underlying results and techniques

We discuss here two things: first, the new proof systems underlying our non-approximability results; second, our chromatic number reduction.

Our new proof systems

We view the new proof systems as the most important contribution of this paper. To discuss them we first need some definitions.

We are in the probabilistically checkable proof (PCP) setting. A verifier V has access to an input x of length n, a string of $r = r(n)$ random bits, and a proof string σ which he queries non-adaptively.

He runs in time $\text{poly}(2^{r(n)})$. One can define in the usual way what it means for this system to have error-probability $\epsilon = \epsilon(n)$ with respect to some underlying language L.

The efficiency measure that has been most important to previous work is the number $q = q(n)$ of bits of the proof examined by the verifier. We focus on a different measure which we call the number of "free bits." Roughly speaking, we say that the verifier uses (or queries) $f = f(n) \leq q(n)$ free bits if, from the answers returned to his first f queries, he is able to compute bits $b_1, \ldots, b_{q-f}$ such that he only accepts if the sequence of answers to his remaining $q - f$ queries is exactly this sequence of bits. (For a more formal definition see Section 2. For history and an explanation of why this is the right measure in our context, see Section 1.3).

The complexity class $\text{FPCP}[\,r,\, f\,]$ that we now define should be viewed, informally, as the free bit analogue of $\text{PCP}[\,r,\, q\,]$; thus think of it as the class of languages which can be recognized with error 1/2 using $r(n)$ random bits and $f(n)$ free bits. This rough idea should suffice for what follows. However the actual definition is a different. Rather than say it takes $f(n)$ free bits to get error 1/2, we will require that the "rate" at which the error decreases for every additional $f(n)$ bits examined is roughly 1/2. Specifically a language L is in $\text{FPCP}[\,r,\, f\,]$ if for any function $k(n) = \omega(1)$ there is verifier V_k which recognizes L with error $2^{-k(n)}$ using $r(n)[k(n) + o(k(n))]$ random bits, $f(n)[k(n) + o(k(n))]$ free bits, and a proof size of $2^{r(n)}$. For more information see Section 2.

The theorem that follows thus says, roughly, that free bits and randomness need to be expended at rates of 3 and $\text{polylog}(n)$, respectively, for every 1/2 factor reduction in error.

Theorem 1.4 $\widetilde{\text{NP}} \subseteq \text{FPCP}[\,\text{polylog}(n),\, 3\,]$.

If the randomness is required to be logarithmic, we will use slightly more than one more free bit. Specifically we can show that for any constant $\delta > 0$ it is the case that NP is contained in $\text{FPCP}[\,O(\log n),\, 4{+}\delta\,]$.

The actual statements we prove are stronger: see Theorems 3.1 and 3.2.

The connection with Max Clique, which we state without proof, is as follows. Suppose $\text{coR}\widetilde{\text{P}}$ is not equal to $\text{FPCP}[\,\text{polylog}(n),\, f\,]$. Then there is no polynomial time algorithm to approximate the Max Clique within $N^{\alpha - o(1)}$ where $\alpha = 1/(1{+}f)$. Thus we get Theorem 1.1.

We can show that that many aspects of our constructions and analysis are tight. We feel that proof systems of a different nature are required to do better.

Our chromatic number reduction

Following [20], non-approximability results for Chromatic Number have been obtained by "reduction" from the Max Clique problem on the specific set of graphs constructed by the Max Clique reduction of [12]. For the purpose of discussing our contribution the problem can be abstracted as follows.

We say that G is a (R, Q)-clique graph ($Q > R$ and Q, R both powers of 2) if its nodes are arranged in a Q by R matrix with each column an independent set. We say that a polynomial time map $\mu(\cdot)$ is a *chromatic number reduction* if there is a polynomial time map $A(\cdot)$, always returning a positive integer, such that the following is true. For any (R, Q)-clique graph G and any integer $g \geq 1$ the graph $H = \mu(G)$ computed by the reduction has the property that if $\omega(G) = R$ then $\chi(H) \leq A(G)$

and if $\omega(G) \leq R/g$ then $\chi(H) \geq A(G) \cdot g$. We say that μ is a (a, b)-chromatic number reduction if the size of H equals $R^a Q^b$.

One can show the following, which we state without proof. Suppose FPCP[polylog(n), f] is not equal to co$\widetilde{\text{RP}}$. Then there is no polynomial time algorithm to approximate the Chromatic Number within $N^{\beta - o(1)}$ for $\beta = 1/(a+bf)$. Thus the problem is to design (a, b)-chromatic number reductions with a, b as small as possible.

The reduction of [20] achieves $a = 1$ and $b = 5$. (A simple reduction supplied later by [17] is slightly less efficient, achieving $a = 6$ and $b = 5$.) Applying this and Theorem 1.4 we can conclude that approximating Chromatic Number within $N^{\frac{1}{16} - o(1)}$ is hard, which is already better than the best previous hardness factor of $N^{1/71}$ due to [13]. However, we have the following improvement in the reduction.

Theorem 1.5 There is a (a, b)-chromatic number reduction achieving $a = 1$ and $b = 3$.

Theorem 1.2 follows.

Our reduction is an extension of the one of [17], staying within the same framework but using linear algebra and coding theory techniques to implement "shifts" in a different way. See Section 4.

1.3 History and explanations

Non-approximability results based on PCPs begin with [12]. They have since been proved under many different assumptions. These include: $\widetilde{\text{P}} \neq \text{N}\widetilde{\text{P}}$ [12]; P $\neq$ NP [3, 2]; NP $\neq$ coRP [25, 7, 13]; and $\text{N}\widetilde{\text{P}} \neq \text{coR}\widetilde{\text{P}}$ [7].[2] Since our focus is on the factor $g(N)$ shown hard rather than on the assumption, we will talk of a problem being hard to approximate within a certain factor; it is understood that we mean that no polynomial time approximation algorithm achieving this factor exists under one of the above assumptions.

The basic connection of PCPs to Max Clique is due to [12]. Until the work of [13], it has been expressed in terms of query bits, and we'll begin by discussing this part of the literature. Our discussion is in terms of $\text{PCP}^{\text{av}}[\,r, q\,]$, the average case version of PCP defined by [7] to equal the class of languages which can be recognized with error $1/2$ by a PCP which uses $r = r(n)$ random bits and makes a number of queries whose expectation is a constant strictly less than q. The connection, which we state without proof, is that if $\text{N}\widetilde{\text{P}}$ is in $\text{PCP}^{\text{av}}[\,\text{polylog}(n), q\,]$ then approximating Max Clique within N^{α} is hard for $\alpha = 1/(1+q)$. This is a slightly tighter version of the original connection of [12] which can be viewed as derived in two alternative ways: either apply the randomized graph products of [9] to the construction of [12], or apply [25].

The connection described above led researchers to focus on reducing the (expected) number of bits queried in the PCP. Extending [4, 5, 12, 3] it was shown by [2] that $\text{N}\widetilde{\text{P}}$ is in $\text{PCP}^{\text{av}}[\,\text{polylog}(n), q\,]$ for a value q which, although not specified in the paper, has been said by the authors to be around 10^4. Thus they could show that approximating Max Clique within $N^{0.0001}$ is hard. The problem of reducing q was first tackled in earnest in [7], and they showed that $\text{N}\widetilde{\text{P}}$ is in $\text{PCP}^{\text{av}}[\,\text{polylog}(n), 24\,]$.

[2] Note NP $\neq$ coRP (resp. $\text{N}\widetilde{\text{P}} \neq \text{coR}\widetilde{\text{P}}$) is a slight improvement on the assumption actually stated in the works in question, which is NP $\not\subseteq$ BPP (resp. NEXP $\not\subseteq$ BPEXP). For instance, NP = coRP implies that the polynomial hierarchy collapses to its first level. Also note NP $\neq$ ZPP (resp. $\text{N}\widetilde{\text{P}} \neq \text{Z}\widetilde{\text{PP}}$) is equivalent to NP $\neq$ coRP (resp. $\text{N}\widetilde{\text{P}} \neq \text{coR}\widetilde{\text{P}}$).

It followed that approximating Max Clique within $N^{1/25}$ is hard, a pretty substantial improvement. Amongst their technical contributions are a simplified framework for proof checking and the idea of reusing query bits. We use both.

To reduce below 24 the number of queried bits in a PCP seemed hard. The door to better results was opened by Feige and Kilian [13]. They suggested the notion of free bits (they didn't name it so; that was our doing) and observed that the Max Clique connection actually achieved what in our language would be the following: if $\widetilde{\mathrm{NP}}$ is in $\mathrm{FPCP}^{\mathrm{av}}[\,\mathrm{polylog}(n),\, f\,]$ then approximating Max Clique within N^{α} is hard for $\alpha = 1/(1{+}f)$. They then observed that of the 24 query bits used in the proof system of [7] only 14 were free; that is, $\widetilde{\mathrm{NP}}$ is in $\mathrm{FPCP}^{\mathrm{av}}[\,\mathrm{polylog}(n),\, 14\,]$. It followed that approximating Max Clique within $N^{1/15}$ is hard.[3] It was indicated in [13] that better results could probably be obtained by optimizing proof checking systems with the new parameter in mind. Our results show that they were right.

The first non-approximability result for Chromatic Number was that of Lund and Yannakakis [20]. They showed that there is a constant $\beta > 0$ such that approximating the chromatic number of a graph within N^{β} is hard. A value of $\beta = 1/121$ was obtained in [7]. This was improved to $\beta = 1/71$ in [13].

The first non-approximability result for Max 3SAT was that of [2]. Assuming $\mathrm{P} \neq \mathrm{NP}$ they showed that there exists a constant $\gamma > 0$ such that no polynomial time algorithm can approximate Max 3SAT within $1+\gamma$. Next it was shown by [7] that assuming $\widetilde{\mathrm{P}} \neq \widetilde{\mathrm{NP}}$, no polynomial time algorithm can approximate Max 3SAT within $1 + 1/93$. The assumption was improved to $\mathrm{P} \neq \mathrm{NP}$ in [13].

2 Definitions

A PCP is defined by a verifier V who has access to the input x of length n, a random string R of length $r = r(n)$, and a proof string σ which has a length $l = l(n)$ assumed wlog to be a power of 2. The verifier is specified by a $\mathrm{poly}(2^{r(n)})$ time computable query function $\mathcal{Q}_{x,R}(\cdot)$ and an answer checking function $\mathcal{C}_{x,R}(\cdot)$. He computes queries $q_i = \mathcal{Q}_{x,R}(i)$ for $i = 1, \ldots, q(n)$ which are $\lg l(n)$ bit strings, each indicating an "address" in the proof string; the total number of queries $q(n)$ is $\mathrm{poly}(2^{r(n)})$. The corresponding bits $a_1, \ldots, a_{q(n)}$ of the proof are returned. He now computes his decision according to $\mathcal{C}_{x,R}(a_1 \ldots a_{q(n)})$. We ask that $\mathcal{C}_{x,R}$ be computed by a circuit which can be generated in $\mathrm{poly}(2^{r(n)})$ time and has size $\mathrm{poly}(q(n))$.

The probability that V accepts x, σ is taken over the choice of R and is denoted $\mathtt{Acc}\,[\,V^{\sigma}(x)\,]$. As usual, the verifier defines a $\epsilon = \epsilon(n)$ error proof system for a language L if for every $x \in L$ there is a σ such that $\mathtt{Acc}\,[\,V^{\sigma}(x)\,] = 1$, and for every $x \notin L$ and every σ it is the case that $\mathtt{Acc}\,[\,V^{\sigma}(x)\,] \leq \epsilon(n)$.

We say that V uses $f = f(n)$ free bits if there is a function $\mathcal{G}_{x,R}(\cdot)$ which, given answers $a_1, \ldots, a_{f(n)}$ to the first $f(n)$ queries, returns $q(n) - f(n)$ bits $b_{f(n)+1}, \ldots, b_{q(n)}$ such that the following is true: if there is some $i \in \{f(n)+1, \ldots, q(n)\}$ such that $b_i \neq a_i$, then $\mathcal{C}_{x,R}(a_1 \ldots a_{q(n)}) = 0$. In other words, the verifier can accept only if $b_i = a_i$ for all $i = f(n)+1, \ldots, q(n)$. As with $\mathcal{C}_{x,R}$ we also ask that $\mathcal{G}_{x,R}$ be computed by a circuit which can be generated in $\mathrm{poly}(2^{r(n)})$ time and has size $\mathrm{poly}(q(n))$. We call $\mathcal{G}_{x,R}$ the *guessing* function.

[3] The observation of [13], and the reason it is the free bits rather than the query bits that are relevant, can be explained another way to a reader familiar with the construction of [12]. Recall that each node of the graph built by the latter encodes a computation of the verifier, and only nodes corresponding to accepting computations need be in the graph. The number of nodes is 2^{r+q} but the number of accepting ones is only 2^{r+f}.

Whenever n is understood we drop it as an argument to the proof system parameters.

Traditionally the important efficiency measures have been the number of queries q and the amount of randomness r. Instead of q, we focus on the number of free bits f. We continue to consider the randomness, but we consider also the proof size l which is actually more relevant in the kinds of reductions we discuss. Accordingly, $\mathrm{FPCP}[\,r,\ f,\ \epsilon,\ l\,]$ is the class of languages recognizable with error ϵ using r random bits, f free bits and proofs of size l. A language is in $\mathrm{FPCP}[\,r,\ f\,]$ if for every function $k(n) = \omega(1)$ there is a function $\delta_k(n) = o(k(n))$ such that L is in $\mathrm{FPCP}[\,(k+\delta_k)r,\ (k+\delta_k)f,\ 2^{-k},\ 2^r\,]$.

We'll also discuss verifiers who talk to a collection of $p = p(n)$ provers rather than to a proof string [8]. Definitions for such things are standard.

If Ω is a probability space then $\omega \overset{R}{\leftarrow} \Omega$ denotes the operation of selecting an element at random according to Ω, and $\Pr_{\omega \overset{R}{\leftarrow} \Omega}[\cdot]$ is the corresponding probability. If S is a set then $\omega \overset{R}{\leftarrow} S$ is the operation of selecting ω uniformly at random from S. When we say X is a random variable over a probability space Ω we mean that it is a map whose domain is the support of Ω; the probability $\Pr[X = b]$ that X attains a value b is hence by definition $\Pr_{\omega \overset{R}{\leftarrow} \Omega}[X(\omega) = b]$.

3 Efficient FPCPs

We prove Theorem 1.4 and the statement directly following it. In fact we will establish something stronger.

Theorem 3.1 There exists a constant c and a function $r(n) = \mathrm{polylog}(n)$ such that for all $m \geq 1$ the language SAT is in $\mathrm{FPCP}[\,mr,\ 3m,\ c2^{-m} + 1/\log n,\ 2^r\,]$.

Given any function $k(n) = \omega(1)$ a careful error reduction shows that SAT is in $\mathrm{FPCP}[\,r(k+\delta_k),\ 3(k+\delta_k),\ 2^{-k},\ 2^r\,]$ where

$$\delta_k(n) \;=\; O(1) + \left\lceil \frac{k \log\log\log n}{\log\log n} \right\rceil \;.$$

Theorem 1.4 follows.

Theorem 3.2 There exists a constant c and a polynomial $\kappa(\cdot)$ such that for all $m \geq 1$ and all constants $\epsilon > 0$ the language SAT is in $\mathrm{FPCP}[\,mr_\epsilon,\ 4m,\ c2^{-m} + \epsilon,\ 2^{r_\epsilon}\,]$, where $r_\epsilon(n) = \kappa(1/\epsilon) \cdot \log n$.

The statement following Theorem 1.4 is again obtained by a careful error reduction.

We now simultaneously prove the two theorems above.

We look at l bit strings as l-dimensional vectors over $\mathcal{Z}_2$. Let $a^{(i)}$ denote the i-th coordinate of a string $a \in \mathcal{Z}_2^l$. For $a, b \in \mathcal{Z}_2^l$, we use $a \cdot b$ to represent their *inner product*, i.e. the scalar $\sum_{i=1}^{l} a^{(i)} b^{(i)}$. The *outer product* of a and b, denoted $a \circ b$, is the l^2 bit string c with $c^{(ij)} = a^{(i)} b^{(j)}$. We often view c as an $l \times l$ matrix.

A function $\pi \colon \mathcal{Z}_2^l \mapsto \mathcal{Z}_2^t$ is a *projection* function if there exist $1 \leq i_1 < \ldots < i_t \leq l$ such that $\forall a \in \mathcal{Z}_2^l$ and all $j = 1, \ldots, t$ it is the case that $\pi^{(j)}(a) = a^{(i_j)}$. The canonical inverse $\pi^{-1} \colon \mathcal{Z}_2^t \to \mathcal{Z}_2^l$ of this projection is the function which maps $b \in \mathcal{Z}_2^t$ to the string $a \in \mathcal{Z}_2^l$ satisfying $a^{(i_j)} = b^{(j)}$ for $j = 1, \ldots, t$ and $a^{(i)} = 0$ for $i \notin \{i_1, \ldots, i_t\}$. For $a_1, \ldots, a_m \in \mathcal{Z}_2^l$, the set of vectors

$\{ \sum_{i=1}^{m} b_i a_i \; : \; b_1, \ldots, b_m \in \mathcal{Z}_2 \}$

is called the *span* of $a_1, \ldots, a_m$ and is denoted $\text{Span}(a_1, \ldots, a_m)$.

The *distance* between strings $a, b \in \mathcal{Z}_2^l$, denoted $d(a,b)$, is $|\{ i : a^{(i)} \neq b^{(i)} \}|/l$. Similarly the distance $d(f,g)$ between functions f, g is the fraction of points of their (common) domain on which they differ. Function f is said to be *ϵ-close* to function g if $d(f,g) \leq \epsilon$. The *Hadamard* (robust) encoding of a string $a \in \mathcal{Z}_2^l$ is the 2^l-bit string which may be described as the function E_a: $\mathcal{Z}_2^l \to \mathcal{Z}_2$ given by $E_a(x) = a \cdot x$. Notice that E_a is a linear function; i.e. it satisfies $E_a(x) + E_a(y) = E_a(x+y)$. Observe further that every linear function f from $\mathcal{Z}_2^l$ to $\mathcal{Z}_2$ is the Hadamard encoding of some unique string $a \in \mathcal{Z}_2^l$. This string a is denoted $E^{-1}(f)$. Finally, observe that $d(E_a, E_b) = 1/2$ for $a \neq b$.

3.1 Canonical verifiers

We begin by defining a class of verifiers which we call *canonical* and which can be used as the starting point for our construction. A canonical verifier V_1 has access to p provers and the following features.

Preprocessing. The verifier V_1 reads the input x and tosses $r_1(n)$ random coins. Let R be the outcome of the coin tosses. Based on R and x the verifier generates questions $q_0, \ldots, q_{p-1}$, a circuit C_{q_0} which depends only on q_0 (and x) but not on R given q_0, the lengths $l_0, \ldots, l_{p-1}$ expected of the answers, and projection functions $\pi_1, \ldots, \pi_{p-1}$, where π_k: $\mathcal{Z}_2^{l_0} \to \mathcal{Z}_2^{l_k}$. Quite often we will omit the subscript of C if it is clear from the context.

Interaction. For $k = 0, \ldots, p-1$ the verifier asks the question q_k of the prover P_k and receives response a_k from him.

Postprocessing. The verifier checks that $C(a_0) = 1$ and $\pi_k(a_0) = a_k$ for all $i = 1, \ldots, p-1$ and accepts if all the checks work out.

Guarantees. If $x \in L$, then there exist provers $P_0, \ldots, P_{p-1}$ such that V_1 always accepts. If $x \notin L$ then for all provers $P_0', \ldots, P_{p-1}'$, the probability that the verifier accepts is at most ϵ.

Parameters. The parameters of interest are p, ϵ, r_1 and $l = ||C|| + l_0$.

Our transformation applies to any canonical proof system. The above theorems are obtained by plugging in specific canonical proof systems as we now describe.

The only property missing to make the proof system of [19, 14] canonical is that the second prover's answers are not expressible as a projection of those of the first. This is simple to fix. To see how, recall that the first prover sends a sequence of polynomials $A_1, \ldots, A_d$: $F \to F$ where $d = O(|F| \cdot \log^2 n)$ and F is the underlying field. These polynomials are specified by their coefficients. The verifier evaluates $A_1, \ldots, A_d$ at points $t_1, \ldots, t_d$, respectively, to obtain $x_i = A_i(t_i)$ for $i = 1, \ldots, d$. Meanwhile the second prover has sent points $y_1, \ldots, y_d$. The verifier checks that $x_i = y_i$ for all $i = 1, \ldots, d$. The modification is that for each i instead of having the first prover specify A_i by its coefficients, have him send the list $\langle A_i(t) \; : \; t \in F \rangle$ of the values of the polynomial on all points in the field. Now the values x_i can be obtained by an appropriate projection. If we set the size of the field to $O(\log n)$ we end up with a canonical proof system having $p = 2$, $r_1(n) = O(\log^3 n)$, $\epsilon = 1/\log n$ and $l = \text{polylog}(n)$. Applying the transformation that follows to this system will yield Theorem 3.1.

In the proof system of [13] the answers of the second prover are not determined by those of the first. However, we observe that any p-prover proof system can be converted into a $p+1$ prover canonical proof system. Applying this to the system of [13] we can conclude that there is a polynomial $\kappa(\cdot)$ such that for any constant $\epsilon > 0$ there is a canonical proof system with $p = 3$, $r_1(n) = \kappa(1/\epsilon) \cdot \log n$, error ϵ, and $l = O(r_1(n))$. Theorem 3.2 is derived by applying what follows to this system.

3.2 Protocol

We now extend the task of the provers $P_0, \ldots, P_{p-1}$ so as to make the verifier's task easier, using the idea of recursive proof checking [3]. Notice that the circuit C may be assumed wlog to be an algebraic circuit over $\mathcal{Z}_2$. Hence there exists an augmented input set z (of length $||C||$) and an efficiently constructible degree 2 polynomial P_C of a_0 and z, such that for all a_0 there exists a unique z such that $C(a_0) = 1$ iff $P_C(a_0, z) = 0$. We will refer to the string $a_0 z$ as the C-augmented representation of a_0, denoted $C^{\text{aug}}(a_0)$.

We now describe how the "honest" extended provers behave. For each question q_0 the extended prover P_0 writes down the 2^l bit string $E_{C^{\text{aug}}(a_o)}$ and the 2^{l^2} bit string $E_{C^{\text{aug}}(a_0) \circ C^{\text{aug}}(a_0)}$. For $k \in \{1, \ldots, p-1\}$, the extended prover P_k writes down the 2^{l_k} bit string E_{a_k}. Notice that if P_0 is expected to provide the encodings described above then it is important that C be a function of q_0 alone, and this is indeed a property of the canonical verifier.

We now describe the extended verifier.

Simulating V_1. The verifier first simulates the preprocessing phase of the verifier V_1 and generates the questions q_k, the circuit C and the projection functions π_k. Based on these questions the extended verifier now decides to focus its attention on a small subset of the provers' responses, namely the responses to $q_0, \ldots, q_{p-1}$. Let P_0's response to q_0 be functions $f : \mathcal{Z}_2^l \mapsto \mathcal{Z}_2$ and $g : \mathcal{Z}_2^{l^2} \mapsto \mathcal{Z}_2$ respectively. Let the other provers' responses to their questions be the functions $f_1, \ldots, f_{p-1}$ respectively, where $f_k \colon \mathcal{Z}_2^{l_k} \to \mathcal{Z}_2$.

Random choices. The verifier picks

- $x_1, \ldots, x_m \overset{R}{\leftarrow} \mathcal{Z}_2^l$.
- $y_1, \ldots, y_m \overset{R}{\leftarrow} \mathcal{Z}_2^{l^2}$.
- $z_{1k}, \ldots, z_{mk} \overset{R}{\leftarrow} \mathcal{Z}_2^{l_k}$ for $k = 1, \ldots, p-1$.

Let $\mathcal{X}$ denote the span of $x_1, \ldots, x_m$. Let $\mathcal{Y}$ denote the span of $y_1, \ldots, y_m$. Let $\mathcal{Z}_k$ denote the span of $z_{1k}, \ldots, z_{mk}$.

Linearity tests. The verifier verifies that f restricted to the span of $x_1, \ldots, x_m$ is linear; i.e., for all $x, x' \in \mathcal{X}$

$$f(x) + f(x') \;=\; f(x + x') \; .$$

Similarly it verifies that g restricted to $\mathcal{Y}$ is linear and for each k the restriction of f_k to $\mathcal{Z}_k$ is linear.

Quadraticity tests. The verifier verifies that for all $x, x' \in \mathcal{X}$ and for all $y \in \mathcal{Y}$,

$$f(x) f(x') \;=\; g(y + x \circ x') - g(y) \; .$$

Projection Tests. The verifier verifies that for all $k = 1, \ldots, p-1$, all $z \in \mathcal{Z}_k$ and $x \in \mathcal{X}$,

$$f_k(z) \;=\; f(\pi_k^{-1}(z) + x) - f(x) \; .$$

Circuit Test. The verifier verifies that for all $x \in \mathcal{X}$,

$$g(\alpha + x) - g(x) = 0 ,$$

where $\alpha \in \mathcal{Z}_2^{l^2}$ is the vector of coefficients of the polynomial P_C.

3.3 Analysis

Observe that in the case that the verifier accepts, the values of $f(x_1), \ldots, f(x_m)$, $g(y_1), \ldots, g(y_m)$ and $f_k(z_{1k}), \ldots, f_k(z_{mk})$, for $k \in \{1, \ldots, p-1\}$, completely specify all the other bits that are read. Thus the number of free bits examined by this protocol are $(p+1)m$.

We now analyse the error of this protocol. It is clear that if $x \in L$ then the provers can follow the rules set for the "honest" provers and thus ensure that the verifier never rejects.

In what follows we show that if the probability that the verifier accepts is greater than $\epsilon + c2^{-m}$, for some constant c to be specified later, then $x \in L$. The analysis uses Lemmas 3.3, 3.4, 3.5 and 3.6, which we state now, but whose proofs we defer.

Suppose h maps $\mathcal{Z}_2^t$ to $\mathcal{Z}_2$. Denote by $\text{LINTEST}_m^t(h)$ the probability of the event

$$\forall\, x, x' \in \text{Span}(x_1, \ldots, x_m) \;:\; h(x) + h(x') = h(x + x')$$

when $x_1, \ldots, x_m$ are chosen uniformly and independently from $\mathcal{Z}_2^t$.

Lemma 3.3 There exists a constant c_1 such that the following is true. Suppose h: $\mathcal{Z}_2^t \to \mathcal{Z}_2$ satisfies

$$\text{LINTEST}_m^t(h) \;\geq\; c_1 2^{-m} .$$

Then there exists a linear function h_*: $\mathcal{Z}_2^t \to \mathcal{Z}_2$ such that $d(h, h_*) \leq 0.1$.

Suppose f maps $\mathcal{Z}_2^l$ to $\mathcal{Z}_2$ and g maps $\mathcal{Z}_2^{l^2}$ to $\mathcal{Z}_2$. Denote by $\text{QUADTEST}_m^l(f, g)$ the probability of the event

$$\forall\, x, x' \in \text{Span}(x_1, \ldots, x_m)\; \forall\, y \in \text{Span}(y_1, \ldots, y_m) \;:\; f(x)f(x') \;=\; g(y + x \circ x') - g(y)$$

when $x_1, \ldots, x_m$ are chosen uniformly and independently from $\mathcal{Z}_2^l$ and $y_1, \ldots, y_m$ are chosen uniformly and independently from $\mathcal{Z}_2^{l^2}$.

Lemma 3.4 There exists a constant c_2 such that the following is true. Suppose f: $\mathcal{Z}_2^l \to \mathcal{Z}_2$ and g: $\mathcal{Z}_2^{l^2} \to \mathcal{Z}_2$ are 0.1-close to linear functions f_*: $\mathcal{Z}_2^l \to \mathcal{Z}_2$ and g_*: $\mathcal{Z}_2^{l^2} \to \mathcal{Z}_2$, respectively, and further satisfy

$$\text{QUADTEST}_m^l(f, g) \;\geq\; c_2 2^{-m} .$$

Then $E^{-1}(f_*) \circ E^{-1}(f_*) = E^{-1}(g_*)$.

Suppose f maps $\mathcal{Z}_2^l$ to $\mathcal{Z}_2$ and h maps $\mathcal{Z}_2^t$ to $\mathcal{Z}_2$ and π: $\mathcal{Z}_2^l \to \mathcal{Z}_2^t$ is a projection function. Denote by $\text{PROJTEST}_m^{l,t}(f, h, \pi)$ the probability of the event

$$\forall\, x \in \text{Span}(x_1, \ldots, x_m)\; \forall\, z \in \text{Span}(z_1, \ldots, z_m) \;:\; h(z) \;=\; f(x + \pi^{-1}(z)) - f(x)$$

when $x_1, \ldots, x_m$ are chosen uniformly and independently from $\mathcal{Z}_2^l$ and $z_1, \ldots, z_m$ are chosen uniformly and independently from $\mathcal{Z}_2^t$.

Lemma 3.5 There exists a constant c_3 such that the following is true. Suppose f: $\mathcal{Z}_2^l \to \mathcal{Z}_2$ and h: $\mathcal{Z}_2^t \to \mathcal{Z}_2$ are 0.1-close to linear functions f_*: $\mathcal{Z}_2^l \to \mathcal{Z}_2$ and h_*: $\mathcal{Z}_2^t \to \mathcal{Z}_2$, respectively, and further satisfy

$$\text{ProjTest}_m^{l,t}(f,h,\pi) \;\geq\; c_3 2^{-m}$$

for some projection function π: $\mathcal{Z}_2^l \to \mathcal{Z}_2^t$. Then $\pi(E^{-1}(f_*)) = E^{-1}(h_*)$.

Suppose g maps $\mathcal{Z}_2^{l^2}$ to $\mathcal{Z}_2$ and $\alpha \in \mathcal{Z}_2^{l^2}$. Denote by $\text{CircTest}_m^l(g,\alpha)$ the probability of the event

$$\forall\, y \in \text{Span}(y_1,\ldots,y_m) \;:\; g(\alpha+y) - g(y) \;=\; 0$$

when $y_1,\ldots,y_m$ are chosen uniformly and independently from $\mathcal{Z}_2^{l^2}$.

Lemma 3.6 There exists a constant c_4 such that the following is true. Suppose g: $\mathcal{Z}_2^{l^2} \to \mathcal{Z}_2$ is 0.1-close to a linear function g_*: $\mathcal{Z}_2^{l^2} \to \mathcal{Z}_2$ and further satisfies

$$\text{CircTest}_m^l(g,\alpha) \;\geq\; c_4 2^{-m}\;.$$

Then $g_*(\alpha) = 0$.

To see that the Lemmas above suffice, let functions $f, g, f_1, \ldots, f_{p-1}$ be such that the probability that V accepts on each of the tests (of Section 3.2) is at least $c/2^m$ where $c = \max\{c_1, c_2, c_3, c_4\}$. Then we know that every test above must pass with probability at least $c/2^m$. Applying Lemma 3.3 to each function implies that all the functions are 0.1 close to linear functions (i.e., there exist strings $a, a', a_1, \ldots, a_{p-1}$ such that the functions given are the Hadamard encodings of these strings). Applying Lemma 3.4 we now infer that a' is actually equal to $a \circ a$. Applying Lemma 3.5 to each of the pairs of functions (f, f_k) implies that $\pi_k(a) = a_k$. Lastly Lemma 3.6 can be applied to show that a is of the form $C^{\text{aug}}(a_0)$ for some string a_0 such that $C(a_0) = 1$. Thus if the probability that the verifier V accepts is greater than $\epsilon + (c/2^m)$, then the probability that V_1 accepts is greater than ϵ, implying in turn that $x \in L$.

We feel that one of the important aspects of the above proof systems is that the number of free bits examined is related to the number of different "tables" that the verifier accesses rather than to the success probability of the corresponding "tests."

We now turn to the proofs of the lemmas. We start with the proof of Lemma 3.6.

Proof of Lemma 3.6: Suppose $g_*(\alpha) \neq 0$. Let $I[y]$ be the event that $g(y+\alpha) - g(y) = 0$. Our starting point is the self-corrector of [6, 10] which shows that in this case the probability of $I[y]$ when y is chosen uniformly at random from $\mathcal{Z}_2^{l^2}$ is at most 0.2. Observe that for $y_1, \ldots, y_m$ randomly chosen from the space $\mathcal{Z}_2^{l^2}$, every vector in their span (except for the all zeroes vector) is a random element from $\mathcal{Z}_2^{l^2}$. Further observe that for distinct non-zero strings $b_1 \ldots b_m \in \mathcal{Z}_2^m$ and $b'_1 \ldots b'_m \in \mathcal{Z}_2^m$ the vectors $y = \sum_{i=1}^m b_i y_i$ and $y' = \sum_{i=1}^m b'_i y_i$ are independently and uniformly distributed over $\mathcal{Z}_2^{l^2}$. Thus our analysis reduces to the following: we are tossing $N = 2^m - 1$ pairwise independent coins, each of which comes up "heads" with probability $p \leq 0.2$ and we wish to upper bound the probability that all N of them show "heads". An upper bound of $p/[(1-p)N]$ is obtained by a standard application of Chebychev's inequality. Thus the lemma is true for $c_4 = 1/2$. ▌

The proof of Lemma 3.5 is similar and is omitted from this version.

For the remaining lemmas we would like to proceed as above. The hitch is that the events that we would like to work with are not quite pairwise independent. However they satisfy a weaker form of independence which will suffice. We first introduce this notion and show how the second moment analysis can still be applied here. Our notion seems to be slightly weaker than some weak forms of independence that have been used in the literature [21, 1] and incomparable to some of the others [18, 23]. The definition that follows is given only for the special case of boolean random variables, but is easily generalized. We let $[N] = \{1, \ldots, N\}$. Refer to bottom of Section 2 for notation and conventions.

Definition 3.7 Let $X_1, \ldots, X_n$ be identically distributed, boolean valued random variables on a probability space Ω, and suppose $0 \le \delta \le 1$. We say that $X_1, \ldots, X_n$ are δ-weak pairwise independent if for every $b_1, b_2 \in \{0,1\}$

$$\left| \Pr_{\omega \overset{R}{\leftarrow} \Omega \,;\, i,j \overset{R}{\leftarrow} [N]} [\, X_i(\omega) = b_1 \text{ and } X_j(\omega) = b_2 \,] \;-\; \mu(b_1) \cdot \mu(b_2) \right| \;\le\; \delta \;,$$

where $\mu(b) = \Pr_{\omega \overset{R}{\leftarrow} \Omega}[X_1(\omega) = b]$ for all $b \in \{0,1\}$.

Notice that if $X_1, \ldots, X_n$ are pairwise independent then they are $1/N$-weak pairwise independent.

A bound similar to the second moment bound can be obtained for the weak pairwise independent variables.

Lemma 3.8 Let $X_1, \ldots, X_N$ be δ-weak pairwise independent random variables with $\Pr[X_1 = 1] = p$. Let $S_N = \sum_{i=1}^{N} X_i$. Then

$$\Pr[\, |S_N - Np| \ge k \,] \;\le\; \frac{\delta N^2}{k^2}$$

Proof: $\Pr[\cdot]$ and $\mathbf{E}[\cdot]$ stand for the probability and expectation, respectively, under the space underlying $X_1, \ldots, X_N$. Let $\mathbf{E}_{i,j}[\cdot]$ be shorthand for the expectation under the experiment of picking i, j at random from $[N]$. The δ-weak pairwise independence of $X_1, \ldots, X_n$ implies that $\mathbf{E}_{i,j}\mathbf{E}[X_i X_j] \le \delta + p^2$. Using this we have

$$\begin{aligned}
\mathbf{E}\left[(S_N - Np)^2 \right] &= \mathbf{E}\left[\textstyle\sum_{i,j=1}^{N} (X_i - p)(X_j - p) \right] \\
&= N^2 \cdot \mathbf{E}_{i,j}\, \mathbf{E}[\, (X_i - p)(X_j - p) \,] \\
&= N^2 \cdot \Big(\mathbf{E}_{ij}\, \mathbf{E}[\, X_i X_j \,] + p^2 - p\, \mathbf{E}_{i,j} \mathbf{E}[\, X_i + X_j \,] \Big) \\
&\le N^2 \cdot \Big(\delta + p^2 + p^2 - 2p^2 \Big) \\
&= \delta N^2 \;.
\end{aligned}$$

Conclude by applying Markov's inequality. ▌

We are now in a position to complete the proof of Lemma 3.3. Lemma 3.4 has a similar proof which is omitted from this version.

Proof of Lemma 3.3: Assume for contradiction that h is not 0.1-close to any linear function. For $x, x' \in \mathcal{Z}_2^t$, let $I[x, x']$ be the event that $h(x) + h(x') = h(x + x')$. The linearity test analysis of [10] implies that if x and x' are drawn randomly and independently from $\mathcal{Z}_2^t$ then $I[x, x']$ occurs with probability at most 7/9.

Let $x_1, \ldots, x_m$ be drawn randomly from $\mathcal{Z}_2^t$ and let $\mathcal{X}(b_1 \ldots b_m)$ denote the vector $\sum_{i=1}^m b_i x_i$. Then for distinct non-zero strings b and b' the vectors $\mathcal{X}(b)$ and $\mathcal{X}(b')$ are uniformly and independently distributed over $\mathcal{Z}_2^t$, and hence the event $I[\mathcal{X}(b), \mathcal{X}(b')]$ occurs with probability at most 7/9.

Now consider the probability that the events $I[\mathcal{X}(b), \mathcal{X}(b')]$ and $I[\mathcal{X}(c), \mathcal{X}(c')]$ turn out not to be independent when b, b', c, and c' are picked to be random non-zero vectors. The only cases when the events may be related are when one of the vectors in the set $\{b, b', b + b'\}$ turns out to be equal to one of the vectors in the set $\{c, c', c + c'\}$. The probability that any two of these are equal is at most $1/(2^m - 1)$. Thus the probability that any of these nine events occurs is at most $9/(2^m - 1)$. Thus the events represent a collection $X_1, \ldots, X_N$ of δ-weak pairwise independent random variables with N equal $2^m - 1$ choose 2, $\delta = 9/(2^m - 1)$ and $\mathbf{E}[X_1] \leq 7/9$. Thus Lemma 3.8 can be applied to infer that the probability that all the random variables are 1 is at most $729/[4(2^m - 1)]$. Thus the lemma is true for $c_1 = 729/2$. ▌

4 Chromatic Number

We present the proof of Theorem 1.5. It will be helpful, although not necessary, if the reader is familiar with the construction and proof of [17] which we extend.

Let $Q = 2^q$. Let $F = \mathrm{GF}(2^q)$ be the finite field of size $Q = 2^q$. Let $V = F^3$ and regard it as a 3 dimensional vector space over F. A set $W \subseteq V$ of vectors is *three wise independent* if for any $\vec{w}_1, \vec{w}_2, \vec{w}_3 \in W$ it is the case that the set of vectors $\{\vec{w}_1, \vec{w}_2, \vec{w}_3\}$ is independent. The number of nodes in a graph G is denoted $\|G\|$.

Lemma 4.1 There is a set $W \subseteq V$ of three-wise independent vectors of size $|W| = R$ which can be constructed in polynomial time.

Proof: Let $\alpha_1, \ldots, \alpha_n$ denote any fixed ordering of the $n = |F| - 1$ non-zero elements of F. The vectors are the columns of the matrix

$$\begin{bmatrix} 1 & 1 & 1 & \ldots & \ldots & 1 \\ \alpha_1 & \alpha_2 & \alpha_3 & \ldots & \ldots & \alpha_R \\ \alpha_1^2 & \alpha_2^2 & \alpha_3^2 & \ldots & \ldots & \alpha_R^2 \end{bmatrix}$$

For any $1 \leq i < j < k \leq R$ the determinant

$$\begin{vmatrix} 1 & 1 & 1 \\ \alpha_i & \alpha_j & \alpha_k \\ \alpha_i^2 & \alpha_j^2 & \alpha_k^2 \end{vmatrix}$$

is the Vandermonde determinant and is non-zero. So any three columns are linearly independent. ▌

Let W be as given by the Lemma 4.1, and fix some ordering $W = \{\vec{w}_1, \ldots, \vec{w}_R\}$ of it. View the given (R, Q) clique graph G as having vertex set $F \times [R]$ where $[R] = \{1, \ldots, R\}$. Picture the nodes as a $|F| = Q$ by R matrix, with the nodes in column i being $(\alpha_1, i), \ldots, (\alpha_Q, i)$. Each column is an independent set. We now specify a graph H^*. The vertex set of H^* is $V \times [R]$. Think of the nodes as arranged in a matrix of $|V| = Q^3$ rows and R columns. Call $\vec{w}_i$ the vector associated to column $i = 1, \ldots, R$. For each $\vec{v} \in V$ and node (a, i) of G, the $\vec{v}$-th *shift* of (a, i) is the node in H^* defined by $\pi_{\vec{v}}(a,i) = (\vec{v} + a\vec{w}_i, i)$. Note the shift preserves the column. The edge set of H^* is the set of all $\{\pi_{\vec{v}}(a,i)\,,\,\pi_{\vec{v}}(b,j)\,\}$ such that $\vec{v} \in V$ and $\{\,(a,i),(b,j)\,\}$ is an edge in G. Edge $\{(a_1,i_1),(a_2,i_2)\}$ in G is a *pre-image* of edge $\{(\vec{z}_1,i_1),(\vec{z}_2,i_2)\}$ in H^* if there exists $\vec{v} \in V$ such that $\pi_{\vec{v}}(a_1,i_1) = (\vec{z}_1,i_1)$ and $\pi_{\vec{v}}(a_2,i_2) = (\vec{z}_2,i_2)$.

Lemma 4.2 Every edge in H^* has a *unique* pre-image in G.

The proof is similar to that of Lemma 4.3 below hence is omitted.

The pre-image of a subgraph T of H^* is the subgraph of G induced by the pre-images of the edges in T. (By the above Lemma, it is indeed unique). A triangle is a complete graph on three nodes.

Lemma 4.3 The pre-image of a triangle in H^* is a triangle in G.

Proof: Let T be a triangle in H^* whose nodes are $(\vec{z}_1,i_1),(\vec{z}_2,i_2),(\vec{z}_3,i_3)$. By Lemma 4.2 each edge in T has a unique pre-image in G. So there exist $\vec{v}_1,\vec{v}_2,\vec{v}_3 \in V$ and edges $\{(a_{1,2},i_1),(b_{1,2},i_2)\}$, $\{(a_{2,3},i_2),(b_{2,3},i_3)\}$, $\{(a_{3,1},i_3),(b_{3,1},i_1)\}$ in G such that

$$\begin{aligned}
\vec{v}_1 + a_{1,2}\vec{w}_{i_1} &= \vec{v}_3 + b_{3,1}\vec{w}_{i_1} = \vec{z}_1\\
\vec{v}_2 + a_{2,3}\vec{w}_{i_2} &= \vec{v}_1 + b_{1,2}\vec{w}_{i_2} = \vec{z}_2\\
\vec{v}_3 + a_{3,1}\vec{w}_{i_3} &= \vec{v}_2 + b_{2,3}\vec{w}_{i_3} = \vec{z}_3
\end{aligned}$$

Equating the sum of the quantities in the first column with the sum of the quantities in the second column we get

$$\begin{aligned}
(\vec{v}_1+\vec{v}_2+\vec{v}_3) + a_{1,2}\vec{w}_{i_1} + a_{2,3}\vec{w}_{i_2} + a_{3,1}\vec{w}_{i_3} &= \\
\quad (\vec{v}_3+\vec{v}_1+\vec{v}_2) + b_{3,1}\vec{w}_{i_1} + b_{1,2}\vec{w}_{i_2} + b_{2,3}\vec{w}_{i_3}\ .
\end{aligned}$$

Simplifying we get

$$(a_{1,2}-b_{3,1})\vec{w}_{i_1} + (a_{2,3}-b_{1,2})\vec{w}_{i_2} + (a_{3,1}-b_{2,3})\vec{w}_{i_3} = \vec{0}\ .$$

The three wise independence of W implies that $a_{1,2} = b_{3,1}$ and $a_{2,3} = b_{1,2}$ and $a_{3,1} = b_{2,3}$. This means the edges $\{(a_{1,2},i_1),(b_{1,2},i_2)\}$, $\{(a_{2,3},i_2),(b_{2,3},i_3)\}$, $\{(a_{3,1},i_3),(b_{3,1},i_1)\}$ in G form a triangle. ▮

Let $\overline{\omega}(H^*)$ denote the size of a minimum clique cover of H^*. Given Lemmas 4.2 and 4.3, an argument analogous to that in [17] implies that

(1) $\omega(G) = R$ implies $\overline{\omega}(H^*) = |V| = Q^3$

(2) $\omega(G) \le R/g$ implies $\overline{\omega}(H^*) \ge |V|g = Q^3 g$.

Let H be the complement of H^*. It is well known that $\chi(H) = \overline{\omega}(H^*)$. Now note $\|H\| = \|H^*\| = |V \times R| = Q^3 R$. This completes the proof of Theorem 1.5.

The assumption $Q > R$ isn't really necessary; in general replace Q^3 by the cube of $\max(1+Q, 1+R)$.

5 Max 3SAT

For the proof of Theorem 1.3 we exploit the notion of a master prover and a bunch of other slave provers used by the canonical verifier (cf. Section 3). We also perform various optimizations following in the framework of [7]. In the latter category, we "weight" differently the basic tests; we modify their "placements;" we factor into the analysis a better analysis of the linearity tests, due to [10] which, although known before [7], seems to have been forgotten by them; we even reduce —to 13, from the 15 in [7]— the number of 3SAT clauses needed to write the quadratic test. A very skimpy description follows.

We first use the [19, 14] proof system to get a canonical two-prover proof system as discussed in Section 3.1. We then convert this into a Max 3SAT by expressing the task of the extended verifier by a 3cnf-formula: A linearity test requires 4 clauses, a quadraticity test 13, an output test 2 and an input test requires 4 clauses. We perform the above tests with probability 1, 16/27, 4/9 and 4/9 respectively. Thus the total number of clauses coming from this test is 388/27. The analysis is divided into the following cases. Here the function g is as in the protocol of Section 3.2.

Case 1. The function g is not 0.1 close to linear. In this case the linearity test fails with probability 2/9. Thus the fraction of clauses failing is at least 3/194.

Case 2. The function g is x-close to being linear, where $x \leq 0.1$. In this case, the linearity test fails with probability $3x - 6x^2$ and the at least one of the remaining tests fail with probability $\min\{2/9 - 3x, 2/9 - 2x, 2/9 - 2x\}$. Thus the test fails with probability at least 2/9, implying that the fraction of clauses failing is at least 3/194.

More interesting than the techniques, however, might be the realization that underlies them. Namely that, once again, minimizing the number of bits queried in a PCP is not the best way to go. Underlying our result is again a new measure of proof checking efficiency —different, however, from the free bit measure used above— but we don't have space to discuss it.

Acknowledgments

We thank Uri Feige and Joe Kilian for sending us a preliminary version of their work [13]; Oded Goldreich for pointing out the non-approximability factors under the P $\neq$ NP assumption that were mentioned in Section 1.1; Marcos Kiwi for comments on an early draft; Don Coppersmith for helpful discussions.

Work done while the first author was at the IBM T.J. Watson Research Center, New York.

References

[1] N. Alon, O. Goldreich, J. Håstad and R. Peralta. Simple constructions of almost k-wise independent random variables. *Proceedings of the Thirty First Annual Symposium on the Foundations of Computer Science,* IEEE, 1990.

[2] S. Arora, C. Lund, R. Motwani, M. Sudan and M. Szegedy. Proof verification and intractability of approximation problems. *Proceedings of the Thirty Third Annual Symposium on the Foundations of Computer Science,* IEEE, 1992.

[3] S. Arora and S. Safra. Probabilistic checking of proofs: a new characterization of NP. *Proceedings of the Thirty Third Annual Symposium on the Foundations of Computer Science,* IEEE, 1992.

[4] L. Babai, L. Fortnow and C. Lund. Non-deterministic Exponential time has two-prover interactive protocols. *Proceedings of the Thirty First Annual Symposium on the Foundations of Computer Science,* IEEE, 1990.

[5] L. Babai, L. Fortnow, L. Levin, and M. Szegedy. Checking computations in polylogarithmic time. *Proceedings of the Twenty Third Annual Symposium on the Theory of Computing,* ACM, 1991.

[6] D. Beaver and J. Feigenbaum. Hiding instances in multioracle queries. *Proceedings of the Seventh Annual Symposium on Theoretical Aspects of Computer Science,* Lecture Notes in Computer Science Vol. 415, Springer Verlag, 1990.

[7] M. Bellare, S. Goldwasser, C. Lund and A. Russell. Efficient probabilistically checkable proofs and applications to approximation. *Proceedings of the Twenty Fifth Annual Symposium on the Theory of Computing,* ACM, 1993. See also Errata sheet in *Proceedings of the Twenty Sixth Annual Symposium on the Theory of Computing,* ACM, 1994.

[8] M. Ben-Or, S. Goldwasser, J. Kilian and A. Wigderson. Multi-Prover interactive proofs: How to remove intractability assumptions. *Proceedings of the Twentieth Annual Symposium on the Theory of Computing,* ACM, 1988.

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

[10] M. Blum, M. Luby and R. Rubinfeld. Self-testing/correcting with applications to numerical problems. *Proceedings of the Twenty Second Annual Symposium on the Theory of Computing,* ACM, 1990.

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

[12] U. Feige, S. Goldwasser, L. Lovász, S. Safra, and M. Szegedy. Approximating clique is almost NP-complete. *Proceedings of the Thirty Second Annual Symposium on the Foundations of Computer Science,* IEEE, 1991.

[13] U. Feige and J. Kilian. Two prover protocols – Low error at affordable rates. *Proceedings of the Twenty Sixth Annual Symposium on the Theory of Computing,* ACM, 1994.

[14] U. Feige and L. Lovász. Two-prover one round proof systems: Their power and their problems. *Proceedings of the Twenty Fourth Annual Symposium on the Theory of Computing,* ACM, 1992.

[15] N. Kahale. On the second eigenvalue and linear expansion of regular graphs. *Proceedings of the Thirty Third Annual Symposium on the Foundations of Computer Science,* IEEE, 1992.

[16] R. Karp. Reducibility among combinatorial problems. *Complexity of Computer Computations,* Miller and Thatcher (eds.), Plenum Press, New York (1972).

[17] S. Khanna, N. Linial and S. Safra. On the hardness of approximating the chromatic number. *Proceedings of the Second Israel Symposium on Theory and Computing Systems*, 1993.

[18] D. Koller and N. Megiddo. Constructing small sample spaces satisfying given constraints. *Proceedings of the Twenty Fifth Annual Symposium on the Theory of Computing,* ACM, 1993.

[19] D. Lapidot and A. Shamir. Fully Parallelized Multi-prover protocols for NEXP-time. *Proceedings of the Thirty Second Annual Symposium on the Foundations of Computer Science,* IEEE, 1991.

[20] C. Lund and M. Yannakakis. On the hardness of approximating minimization problems. *Proceedings of the Twenty Fifth Annual Symposium on the Theory of Computing,* ACM, 1993.

[21] J. Naor and M. Naor. Small bias probability spaces: efficient constructions and applications. *Proceedings of the Twenty Second Annual Symposium on the Theory of Computing,* ACM, 1990.

[22] C. Papadimitriou and M. Yannakakis. Optimization, approximation, and complexity classes. *Proceedings of the Twentieth Annual Symposium on the Theory of Computing,* ACM, 1988.

[23] L. Schulman. Sample spaces uniform on neighborhoods. *Proceedings of the Twenty Fourth Annual Symposium on the Theory of Computing,* ACM, 1992.

[24] M. Yannakakis. On the approximation of maximum satisfiability. This SODA not yet defined! .

[25] D. Zuckerman. NP-Complete Problems have a version that is hard to Approximate. *Proceedings of the Eighth Annual Conference on Structure in Complexity Theory*, IEEE, 1993.