Improved Inapproximability Results for MaxClique, Chromatic Number and Approximate Graph Coloring

Subhash Khot
Department of Computer Science
Princeton University
khot@cs.princeton.edu *

Abstract

In this paper, we present improved inapproximability results for three problems : the problem of finding the maximum clique size in a graph, the problem of finding the chromatic number of a graph, and the problem of coloring a graph with a small chromatic number with a small number of colors.

Håstad's celebrated result [13] shows that the maximum clique size in a graph with n vertices is inapproximable in polynomial time within a factor $n^{1-\epsilon}$ for arbitrarily small constant $\epsilon > 0$ unless NP=ZPP. In this paper, we aim at getting the best subconstant value of ϵ in Håstad's result. We prove that clique size is inapproximable within a factor $\frac{n}{2^{(\log n)^{1-\gamma}}}$ (corresponding to $\epsilon = \frac{1}{(\log n)^{\gamma}}$) for some constant $\gamma > 0$ unless *NP $\subseteq$ ZPTIME$(2^{(\log n)^{O(1)}})$. This improves the previous best inapproximability factor of $\frac{n}{2^{O(\log n/\sqrt{\log\log n})}}$ (corresponding to $\epsilon = O(\frac{1}{\sqrt{\log\log n}})$) due to Engebretsen and Holmerin [7].*

A similar result is obtained for the problem of approximating chromatic number of a graph. Feige and Kilian [10] prove that chromatic number is hard to approximate within factor $n^{1-\epsilon}$ for any constant $\epsilon > 0$ unless NP=ZPP. We use some of their techniques to give a much simpler proof of the same result and also improve the hardness factor to $\frac{n}{2^{(\log n)^{1-\gamma}}}$ for some constant $\gamma > 0$. The above two results are obtained by constructing a new Hadamard code based PCP inner verifier.

We also present a new hardness result for approximate graph coloring. We show that for all sufficiently large constants k, it is NP-hard to color a k-colorable graph with $k^{\frac{1}{25}(\log k)}$ colors. This improves a result of Fürer [11] that for arbitrarily small constant $\epsilon > 0$, for sufficiently large constants k, it is hard to color a k-colorable graph with $k^{3/2-\epsilon}$ colors.

*This work was partially supported by S. Arora's David and Lucile Packard Fellowship.

1. Introduction

In this paper, we obtain improved inapproximability results for three problems, viz. the problem of finding the size of the largest clique in a graph, finding the chromatic number of a graph, and approximate coloring of a graph, i.e. coloring a graph with a small number of colors when the graph is guaranteed to have a small constant chromatic number. The first two results are obtained via new PCP constructions based on Hadamard codes while the third result is derived from a simple new reduction. From a conceptual point of view, the result on approximate graph coloring is the most interesting.

The best known (polynomial time) approximation algorithm (see [4]) for MaxClique achieves an approximation ratio of $O(\frac{n}{(\log n)^2})$ which suggests that this problem might be very hard to approximate. The first step towards proving strong inapproximability result for MaxClique was taken in the seminal paper by Feige et al. [9] which showed a connection between Probabilistically Checkable Proof Systems and inapproximability of MaxClique. The discovery of the PCP Theorem ([1], [2]) implied that MaxClique is inapproximable within a factor n^c for some constant $c > 0$ unless P=NP.

Bellare and Sudan [6] defined an important parameter of PCPs called *amortized free bit complexity* and showed that

Theorem 1.1 *If NP has probabilistically checkable proofs where the verifier uses logarithmic randomness and $\overline{f}$ amortized free bits, then MaxClique is inapproximable in polynomial time within a factor $n^{\frac{1}{1+\overline{f}}-\epsilon}$ for any constant $\epsilon > 0$ unless NP=ZPP.*

They constructed PCPs with 3 amortized free bits and obtained a hardness factor of $n^{1/4-\epsilon}$ for clique. This was improved to $n^{1/3-\epsilon}$ by Bellare et al [3].

In his breakthrough result, Håstad [13] proved an $n^{1-\epsilon}$ inapproximability factor for MaxClique. He obtained a PCP verifier that achieves amortized free bit complexity δ for arbitrarily small constant $\delta > 0$. A simpler and alternate proof of this result was obtained by Samorodnitsky and Trevisan [21] which was further simplified recently by Håstad and Wigderson [16]. Their verifiers are based on linearity testing algorithms and make a clever use of *recycling of queries* whose study was initiated by Trevisan [24]. These verifiers simultaneously achieve δ amortized free bits and $1+\delta$ *amortized query bits* which is optimal.

Our Results. All the papers mentioned above use the Long code which encodes a u-bit string by a 2^{2^u}-bit string and requires too much randomness to check it probabilistically. In this paper, we present a PCP verifier based on Hadamard code which encodes a u-bit string by a 2^u-bit string and allows randomness efficient checking. Our verifier is simpler to analyze than the Long code based verifier of Samorodnitsky and Trevisan. In addition to giving simpler proofs of the known results, our new techniques enable us to strengthen the inapproximability results for MaxClique and chromatic number, further closing the gap between the algorithmic results and the hardness results for these problems. We hope that the Hadamard code based PCPs may lead to other new results in the future.

The previous best inapproximability result for MaxClique was due to Engebretsen and Holmerin who used results from [21] to prove that MaxClique is inapproximable within a factor $\frac{n}{2^{O(\log n/\sqrt{\log\log n})}}$ unless NP $\subseteq$ ZPTIME($2^{O(\log n(\log\log n)^{3/2})}$). A possibility of obtaining even stronger inapproximability ratio was suggested by Feige's [8] result that one natural (and deep) idea for approximating maximum clique-size, the Lovász's θ-function, has approximation ratio as bad as $\frac{n}{2^{O(\sqrt{\log n})}}$. Srinivasan [22] raised the question whether a hardness factor of $\frac{n}{2^{O(\sqrt{\log n})}}$ can be proved for MaxClique, since arguably, the Lovász's θ-function might provide the "best" approximation guarantee. As a step towards resolving this question, we show that

Theorem 1.2 *It is not possible to approximate MaxClique in polynomial time within a factor* $\frac{n}{2^{(\log n)^{1-\gamma}}}$ *for some constant* $\gamma > 0$ *unless* $NP \subseteq ZPTIME(2^{(\log n)^{O(1)}})$.

Feige and Kilian [10] introduced a notion of *randomized PCPs* and applied this notion to Håstad's clique PCP [13] to obtain the following result :

Theorem 1.3 *It is hard to approximate chromatic number of a graph in polynomial time within a factor* $n^{1-\epsilon}$ *for any constant* $\epsilon > 0$ *unless NP=ZPP.*

Due to its dependence on Håstad's PCP, the analysis of [10] is complicated. However the PCP we construct in this paper is very easy to randomize and also gives a stronger hardness factor. We prove that

Theorem 1.4 *It is hard to approximate chromatic number of a graph in polynomial time within a factor* $\frac{n}{2^{(\log n)^{1-\gamma}}}$ *for some constant* $\gamma > 0$ *unless NP* $\subseteq$ *ZPTIME*($2^{(\log n)^{O(1)}}$).

Our result on the hardness of approximate graph coloring is quite disjoint from the above two results. It can be derived either from the Samorodnitsky and Trevisan's result [21] or from the very recent results of Håstad and Khot [15]. The latter approach is easier due to the perfect completeness of Håstad and Khot's PCP and that is what we present in this paper.

We note that there is a huge gap between the known algorithmic results and hardness results for approximate graph coloring. On the algorithmic side, Blum and Karger [5] give an algorithm to color a 3-colorable graph with $\tilde{O}(n^{3/14})$ colors whereas Karger, Motwani and Sudan [17] give an algorithm to color a K-colorable graph with $\tilde{O}(n^{1-3/(K+1)})$ colors. On the hardness side, Garey and Johnson [12] show that if, for every K, there exists an algorithm to color a K-colorable graph with $2K-6$ colors, then P=NP. But they do not specify an integer K for which it is NP-hard to color a K-colorable graph with $2K-6$ colors. Lund and Yannakakis [19] show that for every constant C, there exists a constant $K(C)$ such that it is NP-hard to color a $K(C)$-colorable graph with $C \cdot K(C)$ colors. Khanna, Linial and Safra [18] show that for every $K \geq 3$, it is NP-hard to color a K-colorable graph with $K' = K+2\lfloor\frac{K}{3}\rfloor-1$ colors. Fürer [11] proves that

Theorem 1.5 *If NP has a PCP verifier with logarithmic randomness and amortized free bit complexity* $\overline{f}$, *then for every constant* $\epsilon > 0$, *for all sufficiently large constants* K, *it is hard to color a* K*-colorable graph with* $K' = K^{1+\frac{1}{max(1+2\overline{f},2)}-\epsilon}$ *colors assuming NP*$\neq$*ZPP.*

The best amortized free bit complexity at that time was 2 giving $K' = K^{6/5-\epsilon}$. Even with current PCPs with arbitrarily low amortized free bit complexity, this theorem gives only $K' = K^{3/2-\epsilon}$. It was an interesting open problem (also raised by Khanna et al [18]) if one can replace K' by an arbitrary polynomial function or even a superpolynomial function of K. We resolve this question by showing that

Theorem 1.6 *For all sufficiently large constants* K, *it is NP-hard to color a* K*-colorable graph with* $K^{\frac{1}{25}(\log K)}$ *colors. Moreover this hardness result holds for graphs with bounded degree, in fact graphs with degree at most* $2^{K^{O(\log K)}}$.

A similar result holds when K is a slowly growing function of the number of vertices in a graph. Specifically we can show that

Theorem 1.7 *There exists an absolute constant* $\gamma > 0$ *such that for all* $K \leq 2^{(\log n)^{\gamma}}$, *it is hard to color a* K*-*

colorable n-vertex graph with $K^{\Omega(\log K)}$ colors unless NP $\subseteq$ DTIME($2^{(\log n)^{O(1)}}$).

These results fit in nicely with the fact that for arbitrarily small constant $\epsilon > 0$, it is hard to color a n^ϵ-colorable n-vertex graph with $n^{1-\epsilon}$ colors (this is just a restatement of Theorem 1.3).

Overview of the paper : Section 2 gives necessary background. Section 3 gives our main PCP construction. The hardness result for MaxClique follows directly from our PCP construction and it is proved in Appendix B. Section 4 introduces randomized PCPs and Section 5 proves hardness result for chromatic number. Section 6 proves hardness result for approximate graph coloring.

2. Preliminaries

A PCP verifier with parameters (r, q, c, s), $0 < s < c \leq 1$, uses r random bits, queries q bits from a proof, accepts a correct proof of a correct theorem with probability c and accepts a proof of an incorrect theorem with probability at most s. We call c and s, the completeness and soundness of the PCP respectively.

We recall the PCP theorem ([1],[2]) that every language in NP has a PCP verifier that uses logarithmic randomness, constant number of query bits and achieves perfect completeness ($c = 1$) and soundness $1/2$.

We say that a PCP verifier uses f *free bits* if there is a subset of f queries such that for any possible answer to these queries, there is only one possible answer to the remaining queries that would make the verifier accept. We define *amortized query complexity* as $\frac{q}{\log(c/s)}$ and *amortized free bit complexity* as $\overline{f} = \frac{f}{\log(c/s)}$. It turns out that amortized free bit complexity is a very important parameter for hardness results for MaxClique (see Theorem 1.1).

2.1. Inapproximability of Max-3Lin

We will use Hadamard codes in our PCP constructions. Hadamard codes are defined using linear functions, so we need to use an underlying NP-hard problem that features linear constraints. We define such a problem called Max-3Lin(ϵ) and state a theorem of Håstad [14] giving the hardness of this problem.

Theorem 2.1 *There exists an absolute constant $\mu < 1$ such that, for arbitrarily small constant $\epsilon > 0$, there exists a polynomial time reduction from a 3SAT formula ϕ with n variables to a system $\mathcal{L}$ of linear equations modulo 2 with N variables such that :*

- *Every equation contains exactly 3 variables and every variable appears in exactly 7 equations.*
- *If ϕ is satisfiable, there exists an assignment to the variables in $\mathcal{L}$ that satisfies $1-\epsilon$ fraction of equations.*
- *If ϕ is unsatisfiable, no assignment can satisfy more than μ fraction of the equations.*

Moreover, the reduction can achieve $\epsilon = \frac{1}{(\log N)^\beta}$ for some constant $\beta > 0$ if we allow the running time of the reduction and N to be slightly superpolynomial, i.e. $n^{O(\log \log n)}$.

We call the instance $\mathcal{L}$ given by this reduction as an instance of Max-3Lin(ϵ).

2.2. The Raz Verifier

Like many other PCP constructions ([13], [14], [21]), our construction makes use of the *Raz Verifier* which we define next. However we use a Raz Verifier based on Max-3Lin(ϵ) rather than the standard use of a Gap-3SAT based Raz Verifier.

The Raz Verifier is given an instance $\mathcal{L}$ of Max-3Lin(ϵ). It expects to have two proofs P and Q, where proof P is supposed to contain for every set U of u variables, a u-bit string $P(U)$ giving the values of these variables in some (global) assignment. The proof Q is supposed to contain for every set W of u equations, a $3u$-bit string $Q(W)$ giving the values of the $3u$ variables appearing in these u equations. We will denote by W also the set of these $3u$ variables.

The Raz Verifier works as follows : It randomly picks variables $U = (x_i)_{i=1}^u$ and then picks equations $W = (C_i)_{i=1}^u$ where equation C_i is chosen randomly from the constantly many equations containing variable x_i. It reads the bit-strings $P(U)$ and $Q(W)$ respectively. Let π be the projection from $3u$-bit strings to u-bit strings which corresponds to restricting an assignment to the set W to an assignment to the set U. The verifier accepts iff $Q(W)$ satisfies all the equations $(C_i)_{i=1}^u$ (we call this the linear constraints test) and $P(U) = \pi(Q(W))$, i.e. the values of the variables $(x_i)_{i=1}^u$ in $P(U)$ and $Q(W)$ are the same (we call this the projection test).

Completeness of the Raz Verifier is $\geq (1-\epsilon)^u \geq 1 - \epsilon u$. This is because if there is an assignment that satisfies $1-\epsilon$ fraction of equations, both the proofs P and Q can be consistent with this assignment. With probability $(1-\epsilon)^u$, all the equations $(C_i)_{i=1}^u$ will be satisfied and the verifier will accept. When at most $\mu < 1$ fraction of equations in $\mathcal{L}$ are satisfiable, the soundness can be upper bounded by the Parallel Repetition Theorem due to Raz [20] :

Theorem 2.2 *There exists an absolute constant $C_{lin} < 1$ such that the soundness of the Raz Verifier for Max-3Lin(ϵ) is at most C_{lin}^u.*

2.3. Fourier Transforms

We will deal with boolean functions $A : \mathbb{F}_2^u \to \{1, -1\}$. A function is called linear if $A(x \oplus y) = A(x) \cdot A(y)$, where $\oplus$ denotes vector addition over $\mathbb{F}_2$. There are precisely 2^u linear functions : for every $\alpha \in \mathbb{F}_2^u$, there is a function χ_α defined by

$$\chi_\alpha(a) = (-1)^{a \cdot \alpha} \quad \forall\, a \in \mathbb{F}_2^u$$

The space of all functions $A : \mathbb{F}_2^u \to \mathbf{R}$ is a real vector space where addition of two functions is defined as point-wise addition. One can define an inner product on this space as

$$< A_1, A_2 >= \frac{1}{2^u} \sum_{a \in \mathbb{F}_2^u} A_1(a) \cdot A_2(a)$$

It is easy to verfify that the set of all linear functions forms an orthonormal basis for this vector space w.r.t. the above inner product. It follows that any function $A : \mathbb{F}_2^u \to \mathbf{R}$ can be uniquely expressed as $A = \sum_\alpha \widehat{A}_\alpha \chi_\alpha$ where $\widehat{A}_\alpha$ are its *Fourier coefficients*. If A is a boolean function taking values ± 1, its Fourier coefficients satisfy the Parseval's identity $\sum_\alpha \widehat{A}_\alpha^2 = 1$.

A *projection function* $\pi : \mathbb{F}_2^{3u} \to \mathbb{F}_2^u$ is a function that maps vectors in $\mathbb{F}_2^{3u}$ to some fixed u coordinates. For $a \in \mathbb{F}_2^u$, let $\pi^{-1}(a)$ denote the unique vector $c \in \mathbb{F}_2^{3u}$ such that $\pi(c) = a$ and coordinates of c other than those projected by π are 0.

2.4. Hadamard Codes, their Decoding and Folding

Using a standard paradigm, our PCP verifier will expect an encoding of the proof supplied to the Raz Verifier. Specifically, we will use Hadamard codes which we define next.

Definition 2.3 *Hadamard code of* $p \in \mathbb{F}_2^u$ *is the* 2^u*-bit string* $\{\chi_p(a)\}_{a \in \mathbb{F}_2^u}$. *We denote it by* $Hadamard(p)$.

Recall that the string $x = Q(W)$ read by the Raz Verifier is supposed to satisfy certain linear constraints modulo 2. Let these constraints be $h_1 \cdot x = \zeta_1, \ldots, h_u \cdot x = \zeta_u$ where $h_1, \ldots, h_u \in \mathbb{F}_2^{3u}$ and $\zeta_1, \ldots, \zeta_u \in \mathbb{F}_2$.

Folding : We use a (by now standard) technique called *folding* that enables the verifier to ignore the linear constraints test.

Suppose that B is Hadamard code of x and x satisfies the constraints mentioned above. Let H be the linear subspace spanned by the vectors $h_1, \ldots, h_u$. Then for any vector b and any vector $h \in H,\ h = \oplus \rho_i h_i$, we have

$$B(b \oplus h) = B(b) \cdot (-1)^{\sum_i \rho_i \zeta_i}$$

Motivated by this observation, for an arbitrary function $B : \mathbb{F}_2^{3u} \to \{1, -1\}$, we define another function B' as :

$$\text{For } b = v_b \oplus_i \rho_i h_i, \quad \rho_1, \ldots, \rho_u \in \mathbb{F}_2$$
$$B'(b) = B(v_b) \cdot (-1)^{\sum_i \rho_i \zeta_i} \tag{1}$$

where v_b denotes the lexicographically smallest vector in the set of vectors $b \oplus H$ (a group theoretic coset of H). We call B' a folding of B over the linear constraints. We prove the following crucial lemma in Appendix A.

Lemma 2.4 *If* $\widehat{B'}_\beta \neq 0$*, then* β *must satisfy the linear constraints, i.e.* $h_i \cdot \beta = \zeta_i\ \forall\, i$.

We will require that the supposed Hadamard codes be folded over the respective constraints. This requirement can be enforced using the following access mechanism. When the verifier wants to read $B(b)$, it reads $B(v_b)$ instead and "calculates" the value of $B(b)$ from the expression (1).

We will eventually show that if our PCP verifier accepts the encoded proofs with a good probability, then these proofs can be decoded to construct proofs (P, Q) which the Raz Verifier accepts with a good probability. Decoding of a table B gives β with probability $\widehat{B}_\beta^2$. Folding ensures that any β given by this decoding procedure satisfies the linear constraints on W. Thus we can ignore the linear constraints test.

We remark that the previous PCP constructions use Gap-3SAT as the underlying NP-hard problem where the constraints are non-linear and one has to use Long codes.

3. The Main PCP Construction and Analysis

We now define and analyze our PCP verifier which we call V_{lin}. It crucially depends on the verifiers of Sudan and Trevisan [23] and Samorodnitsky and Trevisan [21]. The verifier V_{lin} is given an instance $\mathcal{L}$ of Max-3Lin(ϵ). As mentioned before, it expects to have proofs (P', Q') which are encodings of proofs (P, Q) supplied to the Raz Verfier using Hadamard codes. So $P'(U)$ ($Q'(W)$) is now supposed to contain the Hadamard code of the string $P(U)$ ($Q(W)$). The verifier V_{lin} proceeds as follows :

1. Pick a set U of u variables at random and k sets $(W_j)_{j=1}^k$ independently, where each set is picked in a similar manner as the Raz Verifier. Let π_j be the projection function between W_j and U.

2. Let A be the supposed Hadamard code of $P(U)$ and B_j be the supposed Hadamard code of $Q(W_j)$ in the proof. Tables B_j are assumed to be folded over respective linear constraints.

3. Pick $a_1, \ldots, a_k \in \mathbb{F}_2^u$ and $b_1, \ldots, b_k \in \mathbb{F}_2^{3u}$ randomly.

4. Accept iff for $1 \leq i, j \leq k$
$$A(a_i)B_j(b_j) = B_j(\pi_j^{-1}(a_i) \oplus b_j)$$

Theorem 3.1 *The verifier V_{lin} for Max-3Lin(ϵ) instance $\mathcal{L}$ with N variables*

- *Uses $r = u \log N + O(ku)$ random bits.*
- *Queries $2k + k^2$ bits from the proof with $f = 2k$ free bits.*
- *Has completeness at least $1 - \epsilon ku$.*
- *Has soundness $2^{-k^2} + \delta$ provided $C_{lin}^u < \delta^2$.*

Proof: The claims about the number of random bits and query bits are clear. To prove the completeness, we can assume that there is an assignment that satisfies $1 - \epsilon$ fraction of equations. We take the proofs (P, Q) to be consistent with this assignment and encode them with correct Hadamard codes to construct proofs (P', Q'). With probability $1 - \epsilon ku$, all the ku equations in the sets $(W_j)_{j=1}^k$ will be satisfied. With correct Hadamard codes,
$$A(a_i) = (-1)^{a_i \cdot P(U)}, \qquad B_j(b_j) = (-1)^{b_j \cdot Q(W_j)}$$

$$\begin{aligned} B_j(\pi_j^{-1}(a_i) \oplus b_j) &= (-1)^{b_j \cdot Q(W_j) + \pi_j^{-1}(a_i) \cdot Q(W_j)} \\ &= (-1)^{b_j \cdot Q(W_j)} \cdot (-1)^{a_i \cdot \pi_j(Q(W_j))} \\ &= B_j(b_j) \cdot A(a_i) \end{aligned}$$

since $\pi_j(Q(W_j)) = P(U)$ in a correct proof.

To prove the soundness, by Theorem 2.2, it is sufficient to show that if the soundness is $2^{-k^2} + \delta$ then there exist proofs (P, Q) which the Raz Verifier accepts with probability δ^2.

As shown in [21], the acceptance probability of the verifier is given by
$$\frac{1}{2^{k^2}} \sum_{S \subseteq [k] \times [k]} T_S \quad \text{where} \quad T_S = E_{\substack{U, W_1, \ldots, W_k; \\ a_1, \ldots, a_k, b_1, \ldots, b_k}}$$
$$\left[\prod_{(i,j) \in S} A(a_i) B_j(b_j) B_j(\pi_j^{-1}(a_i) \oplus b_j) \right]$$

If this probability is $\geq 2^{-k^2} + \delta$, there exists a nonempty set $S \subseteq [k] \times [k]$ such that $|T_S| \geq \delta$. The following ingenious observation by Samorodnitsky and Trevisan [21] enables us to assume that S is of the form $[2] \times [d]$ for some $1 \leq d \leq k$.

Lemma 3.2 *If $|T_S| \geq \delta$ for some non-empty set S, then $|T_{[2] \times [d]}| \geq \delta^2$ for some $1 \leq d \leq k$.*

We first consider the case when $S = [2] \times [d]$ and d is even. In this case
$$T_S = E \left[\prod_{j \in [d]} B_j(\pi_j^{-1}(a_1) \oplus b_j) B_j(\pi_j^{-1}(a_2) \oplus b_j) \right]$$

Using the following Fourier expansions and the fact that $\chi_\beta(\pi^{-1}(a)) = \chi_{\pi(\beta)}(a)$,
$$\begin{aligned} B_j(\pi_j^{-1}(a_1) \oplus b_j) &= \sum_{\beta_j} \widehat{B}_{j\beta_j} \chi_{\beta_j}(\pi_j^{-1}(a_1) \oplus b_j) && (2) \\ &= \sum_{\beta_j} \widehat{B}_{j\beta_j} \chi_{\pi_j(\beta_j)}(a_1) \chi_{\beta_j}(b_j) && (3) \end{aligned}$$

$$B_j(\pi_j^{-1}(a_2) \oplus b_j) = \sum_{\gamma_j} \widehat{B}_{j\gamma_j} \chi_{\pi_j(\gamma_j)}(a_2) \chi_{\gamma_j}(b_j)$$

we get
$$T_S = \sum_{\beta_j, \gamma_j, j \in [d]} \prod_{j \in [d]} \widehat{B}_{j\beta_j} \widehat{B}_{j\gamma_j} \cdot$$
$$E \left[\chi_{\oplus_j \pi_j(\beta_j)}(a_1) \cdot \chi_{\oplus_j \pi_j(\gamma_j)}(a_2) \cdot \prod_{j \in [d]} \chi_{\beta_j \oplus \gamma_j}(b_j) \right]$$

Taking expectation over b_j, we see that the terms in this sum are non-zero only if $\beta_j = \gamma_j \ \forall \ j$. Taking expectation over a_1, we see that we have nonzero terms only if $\oplus_{j \in [d]} \pi_j(\beta_j) = 0$. We conclude that
$$\delta^2 \leq T_S = E_{U, W_1, \ldots, W_k} \left[\sum_{\beta_j : \oplus_{j \in [d]} \pi_j(\beta_j) = 0} \prod_{j=1}^d \widehat{B}_{j\beta_j}^2 \right] \quad (4)$$

The case when $S = [2] \times [d]$ and d is odd is similar. In this case we have $T_S =$
$$E \left[A(a_1) A(a_2) \prod_{j \in [d]} B_j(\pi_j^{-1}(a_1) \oplus b_j) B_j(\pi_j^{-1}(a_2) \oplus b_j) \right]$$

Using Fourier expansions of $A, B_1, \ldots, B_d$ and simplifying, we get
$$\delta^2 \leq T_S = E_{U, W_1, \ldots, W_k} \left[\sum_{\substack{\alpha, \ \beta_j : \\ \alpha = \oplus_{j \in [d]} \pi_j(\beta_j)}} \widehat{A}_\alpha^2 \prod_{j=1}^d \widehat{B}_{j\beta_j}^2 \right] \quad (5)$$

Now we define proofs (P, Q) for the Raz Verifier as follows. For a set W, pick β with probability $\widehat{B}_\beta^2$ and define $Q(W) = \beta$. For a set U, pick sets $(W_j)_{j=2}^d$ at random and pick $(\beta_j)_{j=2}^d$ with probability $\prod_{j=2}^d \widehat{B}_{j\beta_j}^2$. If d is even, define $P(U) = \oplus_{j=2}^d \pi_j(\beta_j)$. If d is odd, pick α with probability $\widehat{A}_\alpha^2$ and define $P(U) = \alpha \oplus_{j=2}^d \pi_j(\beta_j)$.

It is easy to see that the acceptance probability of the Raz Verifier on these proofs (expected over construction of (P, Q)) is precisely the expressions in (4) or (5). So there exists at least one choice of proofs (P, Q) which is accepted by the Raz Verifier with probability at least δ^2 concluding the proof of Theorem 3.1. ∎

4. Randomized PCPs and Chromatic Number

Now we *randomize* the PCP system constructed in Section 3 to obtain improved hardness result for approximating chromatic number $\chi(G)$ of a graph G. Randomizing this system is easier than randomizing earlier PCP systems based on Long codes. First we introduce the notion of *randomized PCPs* defined by Feige and Kilian [10].

Definition 4.1 *An accepting pattern τ for a PCP verifier is a pair $\tau = (S, \nu)$ such that for some choice of the random string, S is the set of query bits read by the verifier and ν is a setting of these bits for which the verifier would accept. The set of all accepting patterns is denoted by $\mathcal{T}$. A proof Π is said to be consistent with a pattern $\tau = (S, \nu)$ if the values of the bits in proof Π corresponding to the set S match their values given by the setting ν.*

Definition 4.2 *A language L has randomized PCP system with parameters (r, f, ρ, s), where we call ρ the covering parameter, if there is a probabilistic verifier V that can check membership proofs Π for language L using r random bits, f free query bits and satisfies :*

(Soundness condition) :

$$x \notin L \Longrightarrow \forall \text{ proofs } \Pi,\ \ Pr[\, V \text{ accepts } \Pi\,] \ \le s$$

(Covering condition) : If $x \in L$, there exists a collection of proofs $\{\Pi_1, \Pi_2, \ldots\}$ with a probability distribution on these proofs such that

$$\forall\ \tau\ \in \mathcal{T},\ \ Pr_i\,[\,\Pi_i \text{ is consistent with } \tau\,]\ \ge \rho$$

The following theorem of Feige and Kilian [10] gives a connection between RPCPs and the hardness of approximating chromatic number of a graph.

Theorem 4.3 *If there is a RPCP system for 3SAT with parameters (r, f, ρ, s), then for any integer h and $m = 2^f/s$, there is a randomized reduction from a 3SAT formula ϕ to a graph G with $N' = m^h$ vertices such that*

- *If ϕ is satisfiable, $\chi(G) \le \frac{1}{\rho^h} \ln N'$.*
- *if ϕ is unsatisfiable, with probability $1/2$, $\chi(G) \ge \frac{N'}{h 2^{r+f}}$.*

This reduction runs in time polynomial in N'.

We need the following theorem which we state without proof.

Theorem 4.4 *There exists an absolute constant $\mu < 1$ such that for any constant $\epsilon > 0$, there is a polynomial time reduction from a 3SAT formula ϕ with n variables to a Max-3Lin instance $\mathcal{L}$ with N variables such that*

- *If ϕ is unsatisfiable, at most μ fraction of equations in $\mathcal{L}$ can be satisfied.*
- *If ϕ is satisfiable, there exists a set of assignments $\mathcal{A} = \{\sigma_1, \sigma_2, \ldots\}$ for $\mathcal{L}$ such that every equation in $\mathcal{L}$ is satisfied by at least $(1-\epsilon)$ fraction of assignments in $\mathcal{A}$. The number of assignments in $\mathcal{A}$ can be bounded by $O(\frac{1}{\epsilon^2})$.*

Moreover this theorem holds with $\epsilon = \frac{1}{(\log N)^\beta}$ for some constant $\beta > 0$ if we allow the running time of the reduction and N to be slightly superpolynomial, i.e. $n^{O(\log\log n)}$.

We call $\mathcal{L}$ an instance of "coverable" Max-3Lin(ϵ).

5. Randomized PCP for Coverable Max-3Lin(ϵ)

A randomized PCP for coverable Max-3Lin(ϵ) is obtained by a simple modification of the verifier V_{lin} in Section 3 and employing a randomization technique due to Feige and Kilian [10].

Let (P, Q) be the proofs provided to the Raz Verifier. We construct a new verifier V_{rand} as follows. The verifier V_{rand} has access to proofs $(\widehat{P}, \widehat{Q})$ where for some fixed l-bit string x, $\widehat{P}(U)$ is supposed to contain Hadamard code of the $(u+l)$-bit string $P(U) \circ x$ and $\widehat{Q}(W)$ is supposed to contain Hadamard code of the $(3u+l)$-bit string $Q(W) \circ x$ ($\circ$ denotes concatenation of strings).

If $\pi : \mathbb{F}_2^{3u} \to \mathbb{F}_2^{u}$ is the projection between W and U, we define a new projection function π' in the following way : $\pi' : \mathbb{F}_2^{3u+l} \to \mathbb{F}_2^{u+l}$ is defined by setting for every $\beta \in \mathbb{F}_2^{3u}$, $\eta \in \mathbb{F}_2^{l}$, $\pi'(\beta \circ \eta) = \pi(\beta) \circ \eta$.

The new verifier V_{rand} has access to proofs $(\widehat{P}, \widehat{Q})$ and it works as follows.

1. Pick sets $U, W_1, \ldots, W_k$ and corresponding projections $\pi_1, \ldots, \pi_k$. Construct new projection functions $\pi'_1, \ldots, \pi'_k$.
2. Let $A = \widehat{P}(U)$ and $B_j = \widehat{Q}(W_j)$ for $1 \le j \le k$.
3. Pick vectors $a_1, \ldots, a_k \in \mathbb{F}_2^{u+l}$ and $b_1, \ldots, b_k \in \mathbb{F}_2^{3u+l}$. Write them as $a_i = a'_i \circ a''_i$ and $b_i = b'_i \circ b''_i$ where a''_i and b''_i are l-bit vectors.

4. If $\{a_i'', b_i'' : 1 \leq i \leq k\}$ are linearly dependent, then reject. Otherwise accept iff

$$A(a_i) \cdot B_j(b_j) = B_j(\pi_j'^{-1}(a_i) \oplus b_j) \; \forall \, i, j$$

Theorem 5.1 *The RPCP system with verifier V_{rand} for Max-3Lin(ϵ) instance $\mathcal{L}$ with N variables*

- *Uses $r = u \log N + O(ku)$ random bits and $2k$ free query bits.*
- *Covering parameter $\rho \geq 2^{-(2k+1)}$ provided $\epsilon k u \leq 1/2$.*
- *Soundness $s \leq 2^{-k^2+1}$ provided $u = \Omega(k^2)$.*

Proof: We note that $\{a_i'', b_i''\}$ is a collection of $2k$ vectors randomly chosen from a space of dimension l. We will have $l = u \gg k$ and the probability that they are linearly dependent is negligible.

Since V_{rand} works in a similar manner as V_{lin}, the soundness analysis for V_{lin} applies to V_{rand} as well and the soundness of V_{rand} is bounded by $2^{-k^2} + \delta$ provided $C_{lin}^u < \delta^2$.

Now we will prove that this PCP system has a good covering parameter. For a (global) assignment σ to the instance $\mathcal{L}$, let $\sigma(U)$ denote the assignment to the variables in set U under σ. We define proofs $(\widehat{P}_{\sigma,x}, \widehat{Q}_{\sigma,x})$ corresponding to the assignment σ and a fixed l-bit string x where $\widehat{P}_{\sigma,x}(U)$ contains the Hadamard code of the $(u+l)$-bit string $\sigma(U) \circ x$ and $\widehat{Q}_{\sigma,x}(W)$ contains the Hadamard code of the $(3u+l)$-bit string $\sigma(W) \circ x$. Consider the collection of proofs

$$\{ \, (\widehat{P}_{\sigma,x}, \; \widehat{Q}_{\sigma,x}) \mid \; x \in \{0,1\}^l, \; \sigma \in \mathcal{A}\} \tag{6}$$

where $\mathcal{A}$ is a set of assignments such that every equation in $\mathcal{L}$ is satisfied by at least $1 - \epsilon$ fraction of assignments in $\mathcal{A}$ (see Theorem 4.4). We consider the uniform probability distribution on these proofs.

Patterns τ for the verifier V_{rand} correspond to any fixed setting of the bits $(A(a_1), \ldots, A(a_k), \; B_1(b_1), \ldots, B_k(b_k))$ where $A = Hadamard(\sigma(U) \circ x), B_j = Hadamard(\sigma(W_j) \circ x)$. The values of these bits in the proof $(\widehat{P}_{\sigma,x}, \widehat{Q}_{\sigma,x})$ are (in {0,1} notation)

$$\begin{aligned} &(a_1' \cdot \sigma(U) \oplus a_1'' \cdot x, \ldots, a_k' \cdot \sigma(U) \oplus a_k'' \cdot x, \\ &b_1' \cdot \sigma(W_1) \oplus b_1'' \cdot x, \ldots, b_k' \cdot \sigma(W_k) \oplus b_k'' \cdot x) \end{aligned} \tag{7}$$

Since $\{a_i'', b_i'' : 1 \leq i \leq k\}$ are linearly independent, if x is a random string from $\{0,1\}^l$, the bit pattern (7) matches every $2k$ bit string with probability $1/2^{2k}$. On the other hand, if $\sigma \in \mathcal{A}$ is chosen randomly, with probability $1-\epsilon k u$, all the equations in $(W_j)_{j=1}^k$ will be satisfied. It follows that a randomly chosen proof from (6) is consistent with the pattern τ with probability $\geq (1 - \epsilon k u)/2^{2k} \geq 1/2^{2k+1}$, giving a good covering parameter. ∎

The parameters of Theorem 5.1 along with Theorem 4.3 are sufficient to prove Theorem 1.3. Theorem 1.4 can be proved in a similar manner, but plugging in superconstant values of u, k and a subconstant value $\epsilon = \frac{1}{(\log N)^\beta}$ given by Theorem 4.4.

6. Hardness of Approximate Graph Coloring

In this section we prove Theorem 1.6. We use a verifier from a recent paper by Håstad and Khot [15]. Actually this result can be derived from Samorodnitsky and Trevisan's verifier or also from the RPCP system of Section 5. However using Håstad and Khot's verifier is simpler due to its perfect completeness and that is what we present here.

We briefly describe the verifier of Håstad and Khot. It requries the proof to be the so called *Standard Written Proof* (see [13], [14]). This means that it is based on the Raz Verifier constructed from a Gap-3SAT instance ϕ. The proof for the Raz Verifier contains for every set U of u variables, a u-bit string $\sigma(U)$, giving the values of these variables in a supposedly satisfying assignment σ to the formula ϕ. It also contains for every set W of u clauses, a $3u$-bit string $\sigma(W)$, giving the values of all the variables appearing in these clauses according to the assignment σ. The *Standard Written Proof* is obtained by replacing the strings $\sigma(U)$ and $\sigma(W)$ by their respective *Long codes.* [1]

Definition 6.1 *Long code of a string $a \in \{0,1\}^t$ is obtained by writing down for every function $f : \{0,1\}^t \mapsto \{-1,1\}$ the value $f(a)$. The length of this code is 2^{2^t}.*

Håstad and Khot's verifier picks a set U of u variables at random and sets $(W_j)_{j=1}^k$ where each W_j is a set of u clauses, the i^{th} clause picked randomly from the clauses containing the i^{th} variable in the set U. The strings $\{\sigma(W_j)\}_{j=1}^k$ are all supposed to project down to the string $\sigma(U)$.

For sets U, W, a projection $\pi : \{0,1\}^{3u} \mapsto \{0,1\}^u$ between them, and a function $f : \{0,1\}^u \mapsto \{-1,1\}$, we can think of f also as a function on $\{0,1\}^{3u}$ by defining $f(y) = f(\pi(y)) \; \forall y \in \{0,1\}^{3u}$.

Håstad and Khot [15] give the following test :

1. Pick a set U of u variables at random and $2k$ random functions $(f_i, f_i')_{i=1}^k$ on it. Pick sets $(W_j)_{j=1}^k$ and random functions (g_j, g_j') on W_j.
2. Let A be a supposed long code of $\sigma(U)$ and B_j be a supposed long code of $\sigma(W_j)$ in the proof where σ is a supposed satisfying assignment to ϕ.

[1] Actually one requires the long codes to be "folded", but this is unimportant for our purpose.

3. Accept iff for $1 \leq i, j \leq k$

$$B(g_j \cdot f_i \cdot (f'_i \wedge g'_j)) = B_j(g_j) A(f_i)(A(f'_i) \wedge B_j(g'_j))$$

They prove that

Theorem 6.2 *This test uses $4k + k^2$ queries of which $4k$ are free. It has perfect completeness and the soundness is at most 2^{-k^2+1} provided $u \geq \Omega(k^2)$.*

We again use the randomization technique used in Section 5. For a global assignment σ, instead of expecting long codes of $\sigma(U)$ and $\sigma(W)$, we expect long codes of $\sigma(U) \circ x$ and $\sigma(W) \circ x$ for some fixed l-bit string x. The projection function is modified accordingly. We call the new verifier V_{color}. If σ is a satisfying assignment to ϕ, different choices of x give different correct proofs $\Pi(x)$ for V_{color}. Note that Theorem 6.2 applies to the verifier V_{color} as well.

6.1. The FGLSS Graph

Given the PCP verifier V_{color} that uses r random bits and f free bits, we define the corresponding FGLSS graph (see [9]) as follows : The vertices of this graph are all accepting patterns $\tau = (S, \nu)$. There is an edge between two patterns (S, ν) and (S', ν') if the sets S, S' have a bit in common and ν, ν' assign different values to this bit, i.e. if these patterns are conflicting.

Note that there is one set S of queries for every choice of the random string used by the verifier and there are $2^f = 2^{4k}$ settings ν such that (S, ν) is an accepting pattern. So the FGLSS graph has 2^{r+4k} vertices. Feige et al [9] observed the following fundamental connection between acceptance probability of a proof and the size of maximum independent set in the corresponding FGLSS graph : For a proof Π, the set of patterns that are consistent with this proof form an independent set in the FGLSS graph and the size of this independent set is

$$2^r \cdot (\text{acceptance probability for proof } \Pi)$$

Similarly an independent set in the FGLSS graph gives a proof whose acceptance probability is proportional to the size of this independent set. Thus there is a one-to-one correspondence between proofs for the verifier and independent sets in the FGLSS graph.

Since the soundness of the verifier V_{color} is at most 2^{-k^2+1}, the size of a maximum independent set in the corresponding FGLSS graph is at most 2^{r-k^2+1} and hence the number of colors needed to color this graph is at least

$$\frac{\text{size of the graph}}{\text{size of maximum independent set}} \geq \frac{2^{r+4k}}{2^{r-k^2+1}} = 2^{k^2-1+4k}$$

We investigate the completeness case next.

6.2. Proof of Theorem 1.6

In the completeness case, we know that there are several correct proofs (hence several independent sets of size 2^r), one for every choice of string x. We may expect that these independent sets cover the FGLSS graph. This essentially turns out to be the case : these independent sets cover all but a tiny portion of the FGLSS graph. The crucial observation is that we can easily identify this tiny portion and throw it away so that the remaining FGLSS graph can be colored with small number (2^{5k}) of colors.

The vertices of the FGLSS graph are the accepting patterns (S, ν) of the verifier. For every choice of $(U, (W_j)_{j=1}^k, (f_i, f'_i)_{i=1}^k, (g_j, g'_j)_{j=1}^k)$ made by the verifier, let A be the supposed long code of $\sigma(U)$, and let B_j be the supposed long code of $\sigma(W_j)$ where σ is the supposed satisfying assignment for the Gap-3SAT formula ϕ. In the accepting pattern (S, ν), S is the set of query bits

$$S = (\ \{A(f_i), A(f'_i)\}_{i=1}^k, \{B_j(g_j), B_j(g'_j)\}_{j=1}^k, \\ \{\ B_j(g_j \cdot f_i \cdot (f'_i \wedge g'_j))\ \}_{i,j=1}^k\ )$$

The patterns (S, ν) correspond to the settings ν of the $4k$ bits ($\{A(f_i), A(f'_i)\}_{i=1}^k, \{B_j(g_j), B_j(g'_j)\}_{j=1}^k$). Note that the values of the remaining k^2 bits in the set S are determined by the values of these $4k$ bits.

Consider a vertex corresponding to the pattern (S, ν). We want that this vertex should be covered by an independent set corresponding to some proof $\Pi(x)$ for some $x \in \{0,1\}^l$. This requirement translates into a condition on the functions $(f_i, f'_i, g_j, g'_j)_{i,j=1}^k$ which we state below :

Definition 6.3 *A choice of functions $(f_i, f'_i, g_j, g'_j)_{i,j=1}^k$ is called good if*
$\forall\, a \in \{0,1\}^u$, $\forall\, b_1, b_2, \ldots, b_k \in \{0,1\}^{3u}$,
$\forall\, z \in \{-1,1\}^{4k}$, $\exists\, x \in \{0,1\}^l$ *such that*

$$z = (f_1(a \circ x), f'_1(a \circ x), \ldots, f_k(a \circ x), f'_k(a \circ x), \\ g_1(b_1 \circ x), g'_1(b_1 \circ x), \ldots, g_k(b_k \circ x), g'_k(b_k \circ x))$$

Lemma 6.4 *If $l = 5k$, then the probability that a choice of functions (f_i, f'_i, g_j, g'_j) is not good is $\leq 2^{-2^{k-1}}$ provided k is large enough.*

Proof: Fix $z, a, b_1, \ldots, b_k$. Note that (f_i, f'_i, g_j, g'_j) are defined by setting value ± 1 with equal probability at every point independently. So for every $x \in \{0,1\}^l$, the probability that

$$z \neq (f_1(a \circ x), f'_1(a \circ x), \ldots, f_k(a \circ x), f'_k(a \circ x), \\ g_1(b_1 \circ x), g'_1(b_1 \circ x), \ldots, g_k(b_k \circ x), g'_k(b_k \circ x))$$

is $1 - 2^{-4k}$. The probability that this holds for every $x \in \{0,1\}^l$ is $(1 - 2^{-4k})^{2^l} \leq 2^{-2^k}$. Taking a union bound over all the choices of $z, a, (b_i)_{i=1}^k$ (there are $2^{O(ku)}$ such choices and $u = O(k^2)$), we see that a choice of functions is not good with probability at most $2^{-2^k+O(k^3)} \leq 2^{-2^{k-1}}$. ∎

We stress again that it is feasible to determine the "bad" choices of functions. We remove the vertices in the FGLSS graph which correspond to a bad choice of functions. By Lemma 6.4, the fraction of vertices removed is very small. So in the soundness case, we still need at least 2^{k^2} colors to color the modified FGLSS graph.

In the compltenesss case, consider any vertex (S, ν) of the modified FGLSS graph where S corresponds to a set of query locations ($\{A(f_i), A(f'_i)\}_{i=1}^k$, $\{B_j(g_j), B_j(g'_j)\}_{j=1}^k$) and ν is some setting of these bits. In a correct proof $\Pi(x)$ for verifier V_{color}, the tables A, B_j are long codes of some strings $a \circ x, b_j \circ x$. Hence

$$A(f_i) = f_i(a \circ x),\ A(f'_i) = f'_i(a \circ x),$$
$$B_j(g_j) = g_j(b_j \circ x),\ B_j(g'_j) = g'_j(b_j \circ x) \qquad (8)$$

In the modified FGLSS graph, we are guaranteed that for some choice of x, the bits in (8) match the bit-pattern ν (see Definition 6.3). Thus the independent sets corresponding to proofs $\{\Pi(x)\}_{x \in \{0,1\}^l}$ cover every vertex of the modified FGLSS graph. Hence the modified FGLSS graph can be colored with $2^l = 2^{5k}$ colors.

We have given a reduction from 3SAT to a graph which can either be colored with 2^{5k} colors or requires 2^{k^2} colors to color it. Taking $K = 2^{5k}$ proves Theorem 1.6. It is easy to show that the FGLSS graph has degree at most $2^{K^{O(\log K)}}$, we omit the details.

6.3. On the Proof of Theorem 1.7

For the lack of space, we omit the proof of Theorem 1.7. However we would like to mention that it *does not* follow from the proof of Theorem 1.6. Just choosing u, k to be slowly growing functions in the proof of Theorem 1.6 gives the result only for $K \leq 2^{O(\sqrt{\log \log n})}$. Instead, as mentioned before, one can prove Theorem 1.6 from the RPCP system in Section 5, though it is a bit complicated. The use of Hadamard codes helps again and one can plug in parameters as slowly growing functions of n giving Theorem 1.7. We do not discuss this further.

7. Acknowledgments

I sincerely thank Prof. Sanjeev Arora and Prof. Johan Håstad for suggesting that I apply my ideas to improving hardness results for MaxClique and chromatic number respectively. The hardness result for approximate graph coloring really came out of my discussions with Johan and I thank him for kindly allowing me to include it in this paper. I thank Sanjeev, Prof. Amit Sahai and anonymous refrees for their valuable comments.

References

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

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

[3] M. Bellare, O. Goldreich, M. Sudan. Free bits, PCPs and non-approximability. *Technical Report TR95-024, Electronic Colloquium on Computational Complexity*, 1995.

[4] R. Boppana, M. Haldórsson. Approximating maximum independent sets by excluding subgraphs. *Bit*, 32(2) : 180-196, Jan 1992.

[5] A. Blum, D. Karger. An $\tilde{O}(n^{3/14})$ coloring algorithm for 3-colorable graphs. *Information Processing Letters*, 61, 49-53, 1997.

[6] M. Bellare, M. Sudan. Improved non-approximability results. *In Proceedings of the 26th Annual ACM Symposium on Theory of Computing*, 184-193, 1994.

[7] L. Engebretsen, J. Holmerin. Clique is hard to approximate within $n^{1-o(1)}$. *In Proceedings of 27th International Colloquium on Automata, Languages and Programming*, volume 1853 of Lecture Notes in Computer Science, 2-12, Springer-Verlag, Berlin, July 2000.

[8] U. Feige. Randomized graph products, chromatic numbers, and the Lovasz θ-function. *In Proceedings of the 27th Annual ACM Symposium on Theory of Computing*, 635-640, 1995.

[9] U. Feige, S. Goldwasser, L. Lovász, S. Safra, M. Szegedy. Approximating clique is almost NP-complete. *In Proceedings of the 32nd Annual IEEE Symposium on Foundations of Computer Science*, 2-12, 1991.

[10] U. Feige, J. Kilian. Zero-knowledge and chromatic number. *In Proceedings of the 11th Annual Conference on Computational Complexity*, 1996.

[11] M. Fürer. Improved hardness results for approximating the chromatic number. *In Proceedings of the 36th Annual IEEE Sumposium on Foundations of Computer Science*, 414-421, 1995.

[12] M. R. Garey and D. S. Johnson. The complexity of near-optimal graph coloring. *Journal of the ACM*, vol 23, 43-49, 1976.

[13] J. Håstad. Clique is hard to approximate within $n^{1-\epsilon}$. *In Proceedings of the 37th Annual IEEE Symposium on Foundations of Computer Science*, 627-636, 1996.

[14] J. Håstad. Some optimal inapproximability results. *In Proceedings of the 29th Annual ACM Symposium on Theory of Computing*, 1-10, 1997.

[15] J. Håstad, S. Khot. Query efficient PCPs with perfect completeness. Accepted for presentation at the *42nd Annual IEEE Symposium on Foundations of Computer Science*, 2001.

[16] J. Håstad, A. Wigderson. Simple Analysis of graph tests for linearity and PCP. Accepted for presentation at 2001 Computational Complexity conference.

[17] D. Karger, R. Motwani and M. Sudan. Approximate graph coloring by semidefinite programming. *Journal of the ACM*, 45, 246-265.

[18] S. Khanna, N. Linial and S. Safra. On the hardness of approximating the chromatic number. *In Proceedings of the 2nd Israel Symposium on Theory and Computing Systems*, 250-260, IEEE Computer Society Press, 1993.

[19] C. Lund, M. Yannakakis. On the hardness of approximating minimization problems. *Journal of the ACM*, 41, 960-981, 1999.

[20] R. Raz. A parallel repetition theorem. *SIAM J. of Computing*, vol 27(3), 763-803, 1998.

[21] A. Samorodnitsky, L. Trevisan. A PCP characterization of NP with optimal amortized query complexity. *In Proceedings of the 32nd Annual ACM Symposium on Theory of Computing*, 191-199, 2000.

[22] A. Srinivasan. The value of strong inapproximability results for clique. *In Proceedings of the 32nd Annual ACM Symposium on Theory of Computing*, 144-152, 2000.

[23] M. Sudan, L. Trevisan. Probabilistically checkable proofs with low amortized query complexity. *In Proceedings of the 39th Annual IEEE Symposium on Foundations of Computer Science*, 1998.

[24] L. Trevisan. Recycling queries in PCPs and in linearity tests. *In Proceedings of the 30th ACM Symposium on Theory of Computing*, 1998.

A. Proof of Lemma 2.4

Proof: By definition,

$$\begin{aligned}
\widehat{B'}_\beta &= \frac{1}{2^{3u}} \sum_b B'(b) \cdot \chi_\beta(b) \\
&= \frac{1}{2^{3u+1}} \sum_b (B'(b) \cdot \chi_\beta(b) + B'(b \oplus h_i) \cdot \chi_\beta(b \oplus h_i)) \\
&= \frac{1}{2^{3u+1}} \sum_b B'(b) \cdot \chi_\beta(b)(1 + (-1)^{\zeta_i} \chi_\beta(h_i))
\end{aligned}$$

This sum is zero unless $h_i \cdot \beta = \zeta_i$. Thus β satisfies all constraints if $\widehat{B'}_\beta \neq 0$. ∎

B. Improved Inapproximability Result for MaxClique

In this section, we prove Theorem 1.2. We use the verifier V_{lin} from Theorem 3.1 with superconstant values of parameters u, k and subconstant value of ϵ as given by Theorem 2.1 and apply the following theorem from [7].

Theorem B.1 *If there is a PCP Verifier for 3SAT using r random bits, f free query bits, completeness c and soundness s, then for any $R > r$ and $D = (R+2)/\log(1/s)$, there is a randomized reduction from 3SAT to a graph G with $N' = 2^{R+Df}$ vertices such that :*

- *If the 3SAT formula is satisfiable, with probability $2/3$, G has a clique of size at least $c^D 2^R/2$.*
- *If the formula is unsatisfiable, with probability $2/3$, maximum clique size in G is at most 2^r.*

The reduction runs in time polynomial in N' and the time taken by the verifier.

We construct a PCP verifier for 3SAT as follows : Using Theorem 2.1, we transform given 3SAT formula to an instance $\mathcal{L}$ of Max-3Lin(ϵ) with $\epsilon = \frac{1}{(\log N)^\beta}$ and $N = n^{O(\log\log n)}$. Then we use Theorem 3.1 to construct a PCP verifier for $\mathcal{L}$ with parameters : $\epsilon = \frac{1}{(\log n)^\beta}$, $u = \frac{1}{2}(\log N)^{3\beta/4}$, $k = (\log N)^{\beta/4}$, $\delta = 2^{-k^2}$, $r \leq (\log N)^{1+3\beta/4}$, $f = 2k$, $c \geq 1/2$, $s \leq 2 \cdot 2^{-k^2}$. Finally we apply Theorem B.1 with $R = r(\log N)^{\beta/4}$.

With these parameters, the gap between maximum clique sizes in Theorem B.1 is $\frac{N'}{2^{(\log N')^{1-\gamma}}}$ for some constant $\gamma > 0$ where N' is the size of the graph produced by the reduction in Theorem B.1. We omit the straightforward calculation. This proves Theorem 1.2 under the assumption NP $\not\subseteq$ BPTIME($2^{(\log n)^{O(1)}}$). One can use techniques from [7] to get the same result under the assumption NP $\not\subseteq$ ZPTIME($2^{(\log n)^{O(1)}}$). We omit the details.