Hardness of Approximate Hypergraph Coloring*

Venkatesan Guruswami† Johan Håstad‡ Madhu Sudan¶

September 23, 2002

Abstract

We introduce the notion of covering complexity of a verifier for probabilistically checkable proofs (PCP). Such a verifier is given an input, a claimed theorem, and an oracle, representing a purported proof of the theorem. The verifier is also given a random string and decides whether to accept the proof or not, based on the given random string. We define the covering complexity of such a verifier, on a given input, to be the minimum number of proofs needed to "satisfy" the verifier on every random string, i.e., on every random string, at least one of the given proofs must be accepted by the verifier. The covering complexity of PCP verifiers offers a promising route to getting stronger inapproximability results for some minimization problems, and in particular, (hyper-)graph coloring problems. We present a PCP verifier for NP statements that queries only four bits and yet has a covering complexity of one for true statements and a super-constant covering complexity for statements not in the language. Moreover, the acceptance predicate of this verifier is a simple Not-all-Equal check on the four bits it reads. This enables us to prove that for *any* constant c, it is NP-hard to color a 2-colorable 4-uniform hypergraph using just c colors, and also yields a super-constant inapproximability result under a stronger hardness assumption.

Key words. Graph coloring, Hypergraph coloring, Hardness of approximations, PCP, covering PCP, Set splitting.

AMS subject classification. 68Q17, 68Q15.

*A preliminary version of this paper [15] appears in the *Proceedings of the 41st Annual IEEE Symposium on Foundations of Computer Science*, IEEE Computer Society Press, Los Alamitos, CA, 2000, pp. 149-158.

†University of California at Berkeley, Computer Science Division, Berkeley, CA 94720. Email: `venkat@lcs.mit.edu`. This work was done while the author was at MIT and was supported in part by NSF grants CCR-9875511 and CCR-9912342, and in part by an IBM Graduate Fellowship.

‡Department of Numerical Analysis and Computer Science, Royal Institute of Technology, SE-100 44 Stockholm, Sweden. Email: `johanh@nada.kth.se`, supported in part by the Göran Gustafsson foundation and in part by NSF grant CCR-9987077.

¶Laboratory for Computer Science, MIT, 200 Technology Square, Cambridge, MA 02139. Email: `madhu@mit.edu`. Supported in part by an MIT-NEC Research Initiation Award, a Sloan Foundation Fellowship, NSF Career Grant CCR-9875511, NSF Grant CCR-9912342, and MIT-NTT Grant MIT 2001-04.

1 Introduction

The study of probabilistically checkable proof (PCP) systems has led to major breakthroughs in theoretical computer science in the past decade. In particular this study has led to a surprisingly clear understanding of the complexity of finding approximate solutions to optimization problems. A recurring theme in this study is the association of new complexity measures to verifiers of PCP systems, and construction of efficient verifiers under the new measure. The new measures are then related to some special subclass of optimization problems to gain new insight about the approximability of problems in this subclass of optimization problems. This paper presents yet another such complexity measure, the *covering complexity* of a verifier, and relates it to a subclass of optimization problems, namely *hypergraph coloring* problems. Below we elaborate on some of the notions above, such as PCP, approximability, hypergraph coloring, and introduce our new complexity measure.

Probabilistically checkable proofs. The centerpiece of a PCP system is the probabilistic verifier. This verifier is a randomized polynomial time algorithm whose input is a "theorem", and who is also given oracle access to a "proof". Using the traditional equivalence associated with randomized algorithms, it is convenient to think of the verifier as having two inputs, the "theorem" and a "random string". Based on these two inputs the verifier settles on a strategy to verify the proof — namely, it decides on a sequence of queries to ask the oracle, and prepares a predicate P. It then queries the oracle and if it receives as response bits $a_1, \ldots, a_q$, it applies the predicate $P(a_1, \ldots, a_q)$ and accepts iff the predicate is satisfied.[1] The quality of the PCP system is roughly related to its ability to distinguish valid proofs (true "theorems" with correct "proofs") from invalid theorems (incorrect "theorems" from any purported "proof") — hopefully the verifier accepts valid proofs with much higher probability than it does invalid theorems.

To study the power of PCP systems in a complexity-theoretic setting, we quantify some of the significant resources of the verifier above, and then study the resources needed to verify proofs of membership for some hard language. Fix such a language L and consider a verifier V whose goal is to verify proofs of membership in L. The above paragraph already hints at four measures we may associate with such a verifier and we define them in two steps. For functions $r, q : \mathbb{Z}^+ \to \mathbb{Z}^+$ we say that a V is (r, q)-restricted if, on input x (implying the theorem $x \in L$) of length n, V requires a random string of length $r(n)$ and makes $q(n)$ queries to the proof oracle. We say that V verifies L with completeness c and soundness s, if (1) For every $x \in L$, there exists an oracle Π such that V, on input x and oracle access to Π, outputs accept with probability at least c, and (2) For every $x \notin L$ and every oracle Π, V outputs accept with probability at most s. The class of all languages L that have an (r, q)-restricted verifier verifying it with completeness c and soundness s is denoted $\mathrm{PCP}_{c,s}[r, q]$.

Covering Complexity. In the variant of PCPs that we consider here, we stick with (r, q)-restricted verifiers, but alter the notion of completeness and soundness. Instead of focusing on the one proof that maximizes the probability with which the verifier accepts a given input, here we allow multiple proofs to be provided to the verifier. We say that a set of proofs $\{\Pi_1, \ldots .\Pi_k\}$ *covers* a verifier V on input x if for every random string, there exists one proof Π_i such that V accepts Π_i on this random string. We are interested in the smallest set of proofs that satisfy this property and the cardinality of this set is said to be the *covering complexity* of the verifier on this input.

[1]This description of a verifier is somewhat restrictive. More general definitions allow the verifier to be adaptive, deciding on later queries based on response to previous ones. For this paper, the restricted version suffices.

Analogous to the class PCP, we may define the class $\mathrm{cPCP}_{c,s}[r,q]$ (for covering PCP) to be the class of all languages for which there exist (r,q)-restricted verifiers that satisfy the following conditions: (Completeness) If $x \in L$, the covering complexity of V on x is at most $1/c$. (Soundness) If $x \notin L$ then the covering complexity of V on x is at least $1/s$.

Notions somewhat related to covering complexity have been considered in the literature implicitly and explicitly in the past. Typically these notions have been motivated by the approximability of minimization problems, such as graph coloring, set cover, and the closest vector problem. Our specific notion is motivated by graph and hypergraph coloring problems. We describe our motivation next. We defer the comparison with related notions to later in this section.

Hypergraph coloring, approximability, and inapproximability. An l-uniform hypergraph H is given by a set of vertices V and a set of edges E where an edge $e \in E$ is itself a subset of V of cardinality l. A k-coloring of H is a map from V to the set $\{1, \ldots, k\}$ such that no edge is *monochromatic*. The hypergraph coloring problem is that of finding, given H, the smallest k for which a k-coloring of H exists. When $l = 2$, then the hypergraph is just a graph, and the hypergraph coloring problem is the usual graph coloring problem.

Graph and hypergraph coloring problems have been studied extensively in the literature from both the combinatorial and algorithmic perspective. The task of determining if a l-uniform graph is k-colorable is trivial if $l = 1$ or $k = 1$, and almost so if $l = k = 2$. Every other case turns out to be NP-hard. The case of $l = 2$, $k \geq 3$ is a classical NP-hard problem, while the case of $k = 2$, $l \geq 3$ was shown NP-hard by Lovasz [23]. Thus, even the property of a hypergraph being 2-colorable is non-trivial. This property, also called *Property B*, has been studied in the extremal combinatorics literature for long. Much work has been done on determining sufficient conditions under which a hypergraph family is 2-colorable and on solving the corresponding algorithmic questions [11, 6, 7, 25, 26, 29, 27].

The hardness of the coloring problem motivates the study of the *approximability* of the graph and hypergraph coloring problems. In context of these problems, an (l, k, k')-approximation algorithm is one that produces (in polynomial time) a k'-coloring of every k-colorable l-uniform hypergraph for some $k' > k$, with the "approximation" being better as k' gets closer to k. Even this problem turns out to be non-trivial, with the best known algorithms for coloring even 3-colorable graphs requiring $n^{\Omega(1)}$ colors [9, 19], where n is the number of vertices. Similarly, inspired in part by the approximate graph coloring algorithms, several works [1, 10, 22] have provided approximation algorithms for coloring 2-colorable hypergraphs. The best known result for 2-colorable 4-uniform hypergraphs is a polynomial time coloring algorithm that uses $\tilde{O}(n^{3/4})$ colors [1, 10].

To justify the intractability of the approximation versions of the hypergraph coloring problem, one looks for *inapproximability* results. Inapproximability results show that it is NP-hard to achieve the goals of an (l, k, k')-approximation algorithm by producing a polynomial time computable reduction from, say, SAT to a "gap" problem related to hypergraph coloring. Here we assume a conservative definition of such a reduction, namely, the many-one reduction. The many-one version of such a reduction would reduce a formula φ to an l-uniform hypergraph H such that H is k-colorable if φ is satisfiable, and H is not k'-colorable if φ is not satisfiable. Since the existence of an (l, k, k')-approximation algorithm now gives the power to decide if φ is satisfiable or not, this shows that the approximation problem is NP-hard. In the sequel, when we say that an (l, k, k')-approximation problem is NP-hard, we always implicitly mean that the "gap version" of the problem is NP-hard.

This methodology combined with the PCP technique has been employed heavily to get hardness results of graph coloring problems. This approach started with the results of [24], and culminates with, essentially tight, the results of [12] who show that the $(2, n^{\varepsilon}, n^{1-\varepsilon})$-approximation problem is NP-hard under randomized reductions. However, for graphs whose chromatic number is a small constant, the known hardness results are much weaker. For example, for 3-colorable graphs the best known hardness result only rules out coloring using 4 colors [20, 16]. This paper is motivated by the quest for strong (super-constant) inapproximability for coloring graphs whose chromatic number is a small constant. We do not get such results for graph coloring, but do get such inapproximability results for hypergraph coloring and in particular for coloring 4-uniform hypergraphs.

Graph coloring and covering PCPs. In examining the reasons why the current techniques have been unable to show strong hardness results for inapproximability of coloring 3-colorable graphs, a natural question arises: Are PCPs really necessary to show such hardness results, or would something weaker suffice? To date there are no reasons showing PCPs are necessary. And while the first result showing the intractability of coloring 3-colorable graphs with 4 colors [20] did use the PCP technique, [16] show that PCPs are not needed in this result. The starting point of our work is the observation that *covering* PCPs are indeed necessary for showing strong hardness results for graph coloring. Specifically, in Proposition 2.1, we show that if the $(2, c, \omega(1))$-approximation problem for coloring is NP-hard for $c < \infty$, then $\mathrm{NP} \subseteq \mathrm{cPCP}_{\gamma,o(1)}[\log, 2]$ for some $\gamma > 0$. (Similar results can also be derived from hardness results for coloring hypergraphs, though we don't do so here.)

Previous approaches have realized this need implicitly, but relied on deriving the required results via PCPs. In particular, they use the trivial containment $\mathrm{PCP}_{1,s}[r, q] \subseteq \mathrm{cPCP}_{1,s}[r, q]$ and build upon the latter result to derive hardness for coloring. (Notice, that we do not have such a simple containment when the completeness parameter is not equal to 1. This special case of $c = 1$ is important in general and is referred to as *perfect completeness.*) For our purposes, however, this trivial containment is too weak. In particular it is known that $\mathrm{PCP}_{c,s}[\log, q] \subseteq \mathrm{P}$ for every c, s, q such that $c > s2^q$ (cf. [8, Lemma 10.6]). Thus it is not possible to show $\mathrm{NP} \subseteq \mathrm{cPCP}_{\gamma,o(1)}[\log, O(1)]$ for any constant $\gamma > 0$, using the trivial containment mentioned above (and such a covering PCP is essential for super-constant hardness results for coloring hypergraphs). Thus it becomes evident that a direct construction of covering PCPs may be more fruitful and we undertake such constructions in this paper.

Related Notions. Typically, every approach that applies PCP to minimization problems has resulted, at least implicitly, in some new complexity measures. Two of those that are close to, and likely to be confused with, the notion of covering complexity are the notions of "multiple assignments" [2], and the "covering parameter" of [12]. Here we clarify the distinctions.

In the former case, the multiple assignments of [2], the proof oracle is expected to respond to each query with an element of a large alphabet (rather than just a bit). When quantifying the "quality" of a proof, however, the oracle is allowed to respond with a subset of the alphabet, rather than just a single element, and the goal of the prover is to pick response sets of small sizes so that on every random string, the verifier can pick one elements from each response set to the different queries so that it leads to acceptance. Once again we have a notion of covering all random strings with valid proofs, but this time the order of quantifiers is different. The notion of multiple assignments is interesting only when the alphabet of the oracles responses are large, while our notion remains interesting even when the oracle responds with an element of a binary alphabet.

The second related notion is the covering parameter of Feige and Kilian [12]. Since the names are confusingly similar (we apologize for not detecting this at an early stage), we refer to their notion as the FK-covering parameter. In a rather simplified sense, their parameter also allows multiple proofs to be presented to the verifier. But their notion of coverage requires that on every random string and every possible accepting pattern of query responses for the verifier, there should exist a proof which gives this accepting pattern (and is hence accepted). For any fixed verifier and input, the FK-covering number is always larger than ours, since we don't need every accepting pattern to be seen among the proofs. Though the notions appear close, the motivation, the application, and the technical challenges posed by the FK-covering parameter and ours are completely different. Both notions arise from an attempt to study graph coloring, but their focus is on general graphs (with high chromatic number), while ours is on graphs of small chromatic number. In their case, separation of the FK-covering parameter is sufficient, but not necessary, to give inapproximability of coloring. For our parameter, separation is necessary, but not sufficient to get the same. Finally, in their constructions the challenge is to take a traditional PCP and enhance it to have small FK-covering completeness and they use the PCP directly to argue that the soundness is not large. In our case, the completeness is immediate and the soundness needs further analysis.

Gadgets and covering complexity. Returning to our notion of covering complexity, while it seems essential to study this to get good hardness results on coloring, the reader should also be warned that this notion is somewhat less robust that usual notions that one deals with in PCPs. Specifically, prior notions were not very sensitive to the predicate applied by the verifier in deciding its final output. They could quantify the power of the verifier by simple parameters such as number of bits read, or number of accepting configurations. Here we are forced to pay attention to the verifier's computations and restrict these to get interesting results. It is reasonable to ask why this happens and we attempt to give some justification below.

In standard PCPs, it is often possible to use "gadgets" (whenever they are available) to convert the acceptance predicate of the verifier from one form to another, for only a small loss in the performance. For example suppose one has a PCP verifier V_1 that reads three bits of a proof and accepts if they are not all equal (NAE). Such a verifier would directly prove the hardness of the "Max NAE-3SAT" problem. But by application of a gadget the same verifier can be transformed into one that proves the hardness of the "Max 3SAT" problem. The gadget notices that the function NAE(a, b, c) for three Boolean variables a, b, c is simply $(a \vee b \vee c) \wedge (\neg a \vee \neg b \vee \neg c)$, which is a conjunction of two 3SAT clauses. Thus a transformed verifier V_2 which picks three bits of the proof as V_1 does, and then picks one of the two clauses implied by the check performed by V_1 and verifies just this one clause, is now a verifier whose acceptance predicate is a 3SAT condition. Furthermore, if the acceptance probability of V_1 on the same proof is $1 - \alpha$, then the acceptance probability of V_2 on some given proof is exactly $1 - \alpha/2$. Thus if V_1 proves inapproximability of Max NAE-3SAT then V_2 proves inapproximability of Max 3SAT.

Unfortunately, a similar transformation does not apply in the case of covering complexity. Notice that two proofs, the oracle that always responds with 0 and the one that responds with 1, always suffices to cover any verifier whose acceptance predicate is 3SAT. Yet there exist NAE 3-SAT verifiers that can not be covered by any constant number of proofs. (For example, the verifier that picks 3 of the n bits of the proof uniformly and independently at random and applies the NAE 3-SAT predicate to them, needs $\Omega(\log n)$ proofs to be covered.) Thus even though a gadget transforming NAE 3SAT to 3SAT does exist, it is of no use in preserving covering complexity of verifiers. This non-robust behavior of cPCP verifiers forces us to be careful in designing our verifiers and our two results differ in mainly the predicate applied by the verifier.

Our results. Our first result is a containment of NP in the class $\mathrm{cPCP}_{1,\varepsilon}[O(\log n), 4]$, for every $\varepsilon > 0$. If the randomness is allowed to be slightly super-logarithmic, then the soundness can be reduced to some explicit $o(1)$ function. Technically, this result is of interest in that it overcomes the qualitative limitation described above of passing through standard PCPs. Furthermore, the proof of this result is also of interest in that it shows how to apply the (by now) standard Fourier-analysis based techniques to the studying of covering complexity as well. Thus it lays out the hope for applying such analysis to other cPCP's as well.

Unfortunately, the resulting cPCP fails to improve inapproximability of graph coloring or even hypergraph coloring. As noted earlier covering PCPs are only necessary, but not sufficient to get hardness results for hypergraph coloring. In order to get hardness results for hypergraph coloring from covering PCPs, one needs verifiers whose acceptance condition is a NAE SAT (not-all-equal) predicate (though, in this case, it is also reasonable to allow the responses of the queries to be elements of a non-binary alphabet, and a result over q-ary alphabet will give a result for q-colorable hypergraphs).

Keeping this objective in mind, we design a second verifier (whose query complexity is also 4 bits), but whose acceptance predicate simply checks if the four queried bits are not all equal. The verifier has perfect completeness and its covering soundness can be made an arbitrarily large constant (Theorem 4.2). This result immediately yields a super-constant lower bound on coloring 2-colorable 4-uniform hypergraphs: we prove that c-coloring such hypergraphs is NP-hard for any constant c (Theorem 4.4), and moreover there exists a constant $c_0 > 0$ such that, unless $\mathrm{NP} \subseteq \mathrm{DTIME}(n^{O(\log\log n)})$, there is no polynomial time algorithm to color a 2-colorable 4-uniform hypergraph using $c_0 \frac{\log\log n}{\log\log\log n}$ colors (Theorem 4.6). A similar hardness result also holds for coloring 2-colorable k-uniform hypergraphs for any $k \geq 5$ by reduction from the case of 4-uniform hypergraphs (Theorem 4.7). Prior to our work, no non-trivial inapproximability results seem to be known for coloring 2-colorable hypergraphs, and in fact it was not known if 3-coloring a 2-colorable 4-uniform hypergraph is NP-hard.

We note that we do not have analogous results for the hardness of coloring 2-colorable 3-uniform hypergraphs. The difficulty in capturing the problem stems from the difficulty of analyzing the underlying maximization problem. The natural maximization version of hypergraph 2-coloring is the following: color the vertices with two colors so that a maximum number of hyperedges are non-monochromatic. For l-uniform hypergraphs, this is problem is known as Max l-Set Splitting. For $l = 4$ (the case we study here), a tight hardness result of $7/8 + \varepsilon$ is known [17] and this fact works its way into our analysis. For $k = 3$, a tight hardness result is not known for the maximization version (see [14]) and our inability to show hardness results for 3-uniform hypergraphs seems to stem from this fact.

Organization. In Section 2, we go over some of the definitions more formally and relate covering complexity to approximability of hypergraph coloring. In Section 3, we analyze a simple cPCP verifier that makes 4 queries and has perfect completeness and $o(1)$ soundness. In Section 4, we analyze a more complicated cPCP verifier with similar parameters whose acceptance condition is not-all-equal-sat. This yields the hardness result for coloring 2-colorable, 4-uniform hypergraphs.

This is the complete version of the conference paper [15].

2 Preliminaries

In this section we introduce covering PCPs formally, and establish a connection (in the wrong direction) between covering PCPs and inapproximability of hypergraph coloring.

2.1 Probabilistically checkable proofs (PCPs)

We first give a formal definition of a PCP. Below verifiers are probabilistic oracle Turing machines whose output, on input x and random string r with oracle O, is denoted $V^O(x,r)$. The output is a bit with 1 denoting acceptance and 0 denoting rejection.

Definition 1 *Let c and s be real numbers such that $1 \geq c > s \geq 0$. A probabilistic polynomial time oracle Turing machine V is a PCP verifier with soundness s and completeness c for a language L iff*

- *For $x \in L$ there exists oracle Π such that $\mathrm{Prob}_r[V^{\Pi}(x,r) = 1] \geq c$.*
- *For $x \notin L$, for all Π, $\mathrm{Prob}_r[V^{\Pi}(x,r) = 1] \leq s$.*

Two parameters of interest in a PCP are the number of random bits used by the verifier and the number of queries it makes to the proof oracle. Most of the time the symbols of Π are bits and whenever this is not the case, this is stated explicitly.

Definition 2 *For functions $r, q : \mathbb{Z}^+ \to \mathbb{Z}^+$, a verifier V is (r,q) restricted if, on any input of length n, it uses at most $r(n)$ random bits and makes at most $q(n)$ queries to Π.*

We can now define classes of languages based on PCPs.

Definition 3 (PCP) *A language L belongs to the class $\mathrm{PCP}_{c,s}[r,q]$ if there is an (r,q)-restricted verifier V for L with completeness c and soundness s.*

Next we have the definition of covering PCP.

Definition 4 (Covering PCP) *A language L belongs to the class $\mathrm{cPCP}_{c,s}[r,q]$ if there is an (r,q)-restricted verifier V such that on input x:*

(i) if $x \in L$ then there is a set of proofs $\{\Pi_1, \ldots, \Pi_k\}$ for $k \leq 1/c$ such that for every random string r there exists a proof Π_i such $V^{\Pi_i}(x,r) = 1$; and

(ii) if $x \notin L$, then for every set of k proofs $\{\Pi_1, \Pi_2, \ldots, \Pi_k\}$ with $k < 1/s$, there is a random string r for which V rejects every *Π_i, $1 \leq i \leq k$.*

One usually requires "perfect completeness" ($c = 1$) when seeking PCP characterizations. It is clear from the above definitions that $\mathrm{PCP}_{1,s}[r,q] \subseteq \mathrm{cPCP}_{1,s}[r,q]$ and thus obtaining a PCP characterization for a language class is at least as hard as obtaining a covering PCP characterization with similar parameters.

2.2 Covering PCPs and Graph Coloring

We now verify our intuition that "good" covering PCPs (i.e., those which have a large gap in covering complexity between the completeness and soundness cases) are necessary for strong lower bounds on the approximating the chromatic number. As usual, for a graph G, we denote by $\chi(G)$ its chromatic number, i.e., the minimum number of colors required in a proper coloring of G.

Below, we use the phrase "it is NP-hard to distinguish $f(n)$-colorable graphs from $g(n)$-colorable graphs" to mean that "the $(2, f, g)$-approximation problem is NP-hard". As mentioned in Section 1, note that we are using a conservative definition of NP-hardness and hence this statement implies that there is a many-one reduction from SAT that maps satisfiable instances of SAT to $f(n)$ colorable graphs and maps unsatisfiable instances to graphs that are not $g(n)$-colorable. Under this assumption, we show how to get nice covering PCPs. Below and throughout this paper, the function log denotes logarithms to base two.

Proposition 2.1 *Suppose for functions $f, g : \mathbb{Z}^+ \to \mathbb{Z}^+$, given a graph G on n vertices, it is NP-hard to distinguish between the cases $\chi(G) \leq f(n)$ and $\chi(G) \geq g(n)$. Then*

$$\mathrm{NP} \subseteq \mathrm{cPCP}_{\lceil \log f(n) \rceil^{-1}, \lceil \log g(n) \rceil^{-1}}[O(\log n), 2] \ .$$

Proof: Let the vertex set of G be $V = \{v_1, v_2, \ldots, v_n\}$. The covering PCP consists of proofs $\Pi_1, \Pi_2, \ldots, \Pi_k$ that correspond to "cuts" $\Gamma_1, \ldots, \Gamma_k$ of G, i.e., each Π_i is n-bits long, with the j^{th} bit being 1 or 0 depending on which side of the cut Γ_i contains v_j. The verifier simply picks two vertices v_{j_1} and v_{j_2} at random such that they are adjacent in G, and then check if the j_1^{th} and j_2^{th} bits differ in *any* of the k proofs. The minimum number k of proofs required to satisfy the verifier for all its random choices is clearly the *cut cover number* $\kappa(G)$ of G, i.e., the minimum number of cuts that cover all edges of G. It is easy to see that $\kappa(G) = \lceil \log \chi(G) \rceil$, and therefore the claimed result follows. $\square$

One can get a similar result for any base q, by letting the proofs be q-ary strings and the verifier read two q-ary symbols from the proof. In light of this, we get the following.

Corollary 2.2 *Suppose that there exists an $\varepsilon > 0$ such that it is NP-hard, given an input graph G, to distinguish between the cases when G is 3-colorable and when $\chi(G) \geq n^{\varepsilon}$. Then* NP $\subseteq$ $\mathrm{cPCP}_{1,(\varepsilon \log_3 n)^{-1}}[O(\log n), 2]$ *where the covering PCP is over a ternary alphabet, and the verifier's action is to simply read two ternary symbols from the proof and check that they are not equal.*

In light of the above Corollary, very powerful covering PCP characterizations of NP are necessary in order to get strong hardness results for coloring graphs with small chromatic number. A result similar to Proposition 2.1, with an identical proof, also holds for hypergraph coloring, and thus motivates us to look for good covering PCP characterizations of NP in order to prove hardness results for coloring 2-colorable hypergraphs.

Proposition 2.3 *Suppose that there exists a function $f : \mathbb{Z}^+ \to \mathbb{Z}^+$ such that, given an input r-uniform hypergraph on n vertices, it is NP-hard to distinguish between the cases when it is 2-colorable and when it is not $f(n)$-colorable. Then,* NP $\subseteq$ $\mathrm{cPCP}_{1,\frac{1}{\log f(n)}}[O(\log n), r]$. *In particular, if c-coloring 2-colorable r-uniform hypergraphs is NP-hard for every constant c, then* NP $\subseteq$ $\mathrm{cPCP}_{1,\frac{1}{k}}[O(\log n), r]$ *for every constant $k \geq 1$.*

3 PCP Construction I

We now move on to the constructions of our proof systems. To a reader familiar with PCPs we first give a preview of our constructions. Both our PCPs (of this section and the next) go through the standard path. We start with strong 2-prover 1-round proof systems of Raz [28], apply the composition paradigm [5], and then use the long code of [8] at the bottom level. One warning: in the literature it is common to use a variant of the long code — called the "folded long code" — we do not use the folded version. (Readers unfamiliar with the terms above may find elaborations in Section 3.1.)

As usual, the interesting aspects in the constructions are choice of the inner verifiers and the analyses of their soundness. The inner verifiers that we use are essentially from [17]: The inner verifier in Section 4 is exactly the same as the one used by [17, Section 7] to show hardness of Max 4-Set Splitting, while the one in this section is a small variant. The goals of the analyses are different, since we are interested in the number of proofs required to cover all random strings. Despite the difference, we borrow large parts of our analysis from that of [17]. In the current section our analysis essentially shows that if our verifier, on some fixed input, rejects every proof oracle with probability at least γ, then on any set of k proofs nearly γ^k fraction of random strings end up rejecting all the proofs. Thus the standard soundness of the verifier we construct *is* of interest and we analyze this using lemmas from [17]. The analysis of the verifier in Section 4 does involve some new components and we will comment upon these in the next section.

3.1 Preliminaries: Label cover, Long codes, Proof composition

Our PCP constructions (also) follow the paradigm of *proof composition*, by composing an "outer verifier" with an "inner verifier". In its most modern and easy to apply form, one starts with an *outer proof system* which is a *2-Prover 1-Round proof system* (2P1R) construction for NP. We abstract the 2P1R by a graph-theoretic optimization problem called LABEL COVER. The specific version of LABEL COVER we refer to is the maximization version $\text{LabelCover}_{\max}$ discussed in [3] (see [3] for related versions and the history of this problem).

Label Cover. A $\text{LabelCover}_{\max}$ instance $\mathcal{LC}$ consists of a bipartite graph $H = (U, W, F)$ with vertex set $U \cup W$ and edge set F, "label sets" L_U, L_W which represent the possible *labels* that can be given to vertices in U, W respectively, and *projection functions* $\pi_{u,w} : L_W \to L_U$ for each $u \in U$ and $w \in W$ such that $(u, w) \in F$. The optimization problem we consider is to assign a label $\ell(u) \in L_U$ (resp. $\ell(w) \in L_W$) to each $u \in U$ (resp. $w \in W$) such that the fraction of edges $e = (u', w')$ with $\ell(u') = \pi_{u',w'}(\ell(w'))$ (call such an edge "satisfied") is maximized. The optimum value of a $\text{LabelCover}_{\max}$ instance $\mathcal{LC}$, denoted $\mathsf{OPT}(\mathcal{LC})$, is the maximum fraction of "satisfied" edges in any label assignment. In the language of $\text{LabelCover}_{\max}$, the PCP theorem [5, 4] together with the parallel repetition theorem of Raz [28] yields parts (i)-(iii) of the theorem below. Here we need an additional property that is also used in [17, Sections 6, 7]. First we need a definition: For $u \in U, w \in W$, a set $\beta \subseteq L_W$ and $0 < \varepsilon \le 1$ define

$$\mu_\varepsilon(\beta, u, w) = \sum_{x \in L_U} \min\left\{1, \varepsilon \cdot \left|\pi_{u,w}^{-1}(x) \cap \beta\right|\right\}. \tag{1}$$

The definition above is quite technical (and borrowed directly from [17]) but the intuition is that β projects mostly onto different elements of L_U iff the "measure" μ is large.

Theorem 3.1 ([3, 17]) *There exist $d_0, e_0 < \infty$ and $c > 0$ and a transformation that, given a parameter $\delta > 0$, maps instances φ of* SAT *to instances $\mathcal{LC} = (U, W, F, L_U, L_W, \{\pi_{u,w} | (u, w) \in F\})$ of* $\text{LabelCover}_{\max}$*, in time $n^{O(\log \delta^{-1})}$, such that*

(i) $|U|, |W| \le n^{d_0 \log \delta^{-1}}$ where n is the size of the SAT *instance φ.*

(ii) $|L_U|, |L_W| \le \delta^{-e_0}$.

(iii) If φ is satisfiable then $\mathsf{OPT}(\mathcal{LC}) = 1$, while if φ is not satisfiable then $\mathsf{OPT}(\mathcal{LC}) \le \delta$.

(iv) For every $0 < \varepsilon \leq 1$, $w \in W$, *and every* $\beta \subseteq L_W$, $|\beta| \geq \varepsilon^{-1}$,

$$\mathop{\mathbf{E}}_{u \in_R N(w)} \left[\frac{1}{\mu_\varepsilon(\beta, u, w)} \right] \leq (\varepsilon|\beta|)^{-c} \tag{2}$$

where $N(w) = \{u \in U | (u, w) \in F\}$.

Remark: As mentioned earlier, conditions (i)-(iii) are standard for $\text{LabelCover}_{\max}$. The need for Condition (iv) is inherited from some lemmas of [17] that we use (specifically, Lemmas 3.3 and 3.4). This condition is shown in Lemma 6.9 of [17].

To use the hardness of label cover, we use the standard paradigm of proof composition. The use of this paradigm requires an error-correcting code, which in our case is again the long code. We define this next.

The Long Code. We first remark on some conventions and notation we follow through the rest of this paper: We represent Boolean values by the set $\{1, -1\}$ with 1 standing for FALSE and -1 for TRUE. This representation has the nice feature that XOR just becomes multiplication. For any domain D, denote by $\mathcal{F}_D$ the space of all Boolean functions $f : D \to \{1, -1\}$. For any set D, $|D|$ denotes its cardinality.

We now describe a very redundant error-correcting code, called the *long code*. The long code was first used by [8], and has been very useful in most PCP constructions since.

The long code of an element x in a domain D, denoted $\mathsf{LONG}(x)$, is simply the evaluations of all the $2^{|D|}$ Boolean functions in $\mathcal{F}_D$ at x. If A is the long code of a, then we denote by $A(f)$ the coordinate of A corresponding to function f, so that $A(f) = f(a)$.

We note that most of the proofs used in the literature use the "folded long code" which is a code of half the length of the long code, involving evaluations of the elements x at exactly one of the functions f or $-f$ (but not both). For reasons that will become clearer later, we *cannot* use the folded long code here and work with the actual long code.

Constructing a "Composed" PCP. Note that Theorem 3.1 implies a PCP where the proof is simply the labels of all vertices in U, W of the $\text{LabelCover}_{\max}$ instance and the verifier picks an edge $e = (u, w) \in F$ at random and checks if the labels of u and w are "consistent", i.e., $\pi_{u,w}(\ell(w)) = \ell(u)$. An alternative is to choose a random neighbor w' of u and instead checking $\pi_{u,w}(\ell(w)) = \pi_{u,w'}(\ell(w'))$. By defining $\ell(u)$ to be the most common value of $\pi_{u,w'}(\ell(w'))$ it is easy to see that the probability of acceptance in the latter PCP (that uses w, w' for the check) is at most the probability of acceptance in the former PCP (that uses u, w for the check).

By the properties guaranteed in Theorem 3.1, either PCP uses $O(\log n \log \delta^{-1})$ randomness, has perfect completeness and soundness at most δ. While the soundness is excellent, the number of bits it reads from the proof in total (from the two "locations" it queries) is large (namely, $O(\log \delta^{-1})$). In order to improve the query complexity, one "composes" this "outer" verification with an "inner" verification procedure. The inner verifier is given as input a projection function $\pi : L_W \to L_U$, and has oracle access to purported encodings, via the encoding function Enc of some error-correcting code, of two labels $a \in L_U$ and $b \in L_W$, and its aim is to check that $\pi(b) = a$ (with "good" accuracy) by making very few queries to $\mathsf{Enc}(a)$ and $\mathsf{Enc}(b)$. The inner verifiers we use have a slightly different character: they are given input two projections π_1 and π_2 (specifically $\pi_{u,w}$ and $\pi_{u,w'}$) and have oracle access to purported encodings $\mathsf{Enc}(b)$ and $\mathsf{Enc}(c)$ of two labels $b, c \in L_W$, and the aim is to test whether $\pi_1(b) = \pi_2(c)$. This interesting feature was part of and necessary for Håstad's construction for set splitting [17], and our PCPs also inherit this feature.

In our final PCP system, the proof is expected to be the encodings of the labels $\ell(w)$ of all vertices $w \in W$ using the encoding Enc. For efficient constructions the code used is the *long code* of [8], i.e., $\mathsf{Enc} \stackrel{\text{def}}{=} \mathsf{LONG}$. We denote the portion of the (overall) proof that corresponds to w by $\mathsf{LP}(w)$, and in a "correct" proof $\mathsf{LP}(w)$ would just be $\mathsf{LONG}(\ell(w))$ (the notation LP stands for "long proof").

The construction of a PCP now reduces to the construction of a good *inner verifier* that given a pair of strings B, C which are purportedly long codes, and projection functions π_1 and π_2, checks if these strings are the long codes of two "consistent" strings b and c whose respective projections agree (i.e., satisfy $\pi_1(b) = \pi_2(c)$). Given such an inner verifier IV, one can get a "composed verifier" V_{comp} using standard techniques as follows (given formula φ the verifier first computes the $\text{LabelCover}_{\max}$ instance $\mathcal{LC}$ in polynomial time and then proceeds with the verification):

1. Pick $u \in U$ at random and $w, w' \in N(u)$ at random
2. Run the inner verifier with input $\pi_{u,w}$ and $\pi_{u,w'}$ and oracle access to $\mathsf{LP}(w)$ and $\mathsf{LP}(w')$.
3. Accept iff the inner verifier IV accepts

We denote by $V_{\text{comp}}(\mathsf{IV})$ the composed verifier obtained using inner verifier IV. The (usual) soundness analysis of the composed PCP proceeds by saying that if there is a proof that causes the verifier V_{comp} to accept with large, say $(s + \varepsilon)$, probability, where s is the soundness we are aiming for, then this proof can be "decoded" into labels for $U \cup W$ that "satisfy" more than a fraction δ of the edges in the $\text{LabelCover}_{\max}$ instance, and by Theorem 3.1 therefore the original formula φ was satisfiable. In our case, we would like to make a similar argument and say that if at most k proofs together satisfy all tests of V_{comp}, then these proofs can be "decoded" into labels for $U \cup W$ that satisfy more than δ fraction of edges of $\mathcal{LC}$.

3.2 The Inner Verifier

We now delve into the specification of our first "inner verifier", which we call Basic-IV4. This inner verifier is essentially the same as the one for 4-set splitting in [17], but has a different acceptance predicate. Recall the inner verifier is given input two projections functions $\pi_1, \pi_2 : L_W \to L_U$ and has oracle access to two tables $B, C : \mathcal{F}_{L_W} \to \{1, -1\}$, and aims to check that B (resp. C) is the long code of b (resp. c) which satisfy $\pi_1(b) = \pi_2(c)$.

> `Inner Verifier` Basic-IV4$_p^{B,C}$ (π_1, π_2)
> Choose uniformly at random $f \in \mathcal{F}_{L_U}$, $g_1, h_1 \in \mathcal{F}_{L_W}$
> Choose at random $g', h' \in \mathcal{F}_{L_W}$ such that $\forall b \in L_W$,
> $\mathbf{Pr}[g'(b) = 1] = p$ and $\mathbf{Pr}[h'(b) = 1] = p$
> Set $g_2 = -g_1(f \circ \pi_1 \wedge g')$; $h_2 = -h_1(-f \circ \pi_2 \wedge h')$.
> **Accept** iff $(B(g_1) \neq B(g_2)) \vee (C(h_1) \neq C(h_2))$

For a technical reason, as in [17], the final inner verifier needs to run the above inner verifier for the bias parameter p chosen at random from an appropriate set of values. The specific distribution we use is the one used by Håstad [17] (the constant c used in its specification is the constant from Equation (2) in the statement of Theorem 3.1).

Inner Verifier $\mathrm{IV4}_\gamma^{B,C}$ (π_1, π_2)
Set $t = \lceil 1/\gamma \rceil$, $\varepsilon_1 = \gamma$ and $\varepsilon_i = \gamma^{1+2/c}\varepsilon_{i-1}$ for $1 < i \le t$.
Choose $p \in \{\varepsilon_1, \ldots, \varepsilon_t\}$ uniformly at random.
Run Basic-$\mathrm{IV4}_p^{B,C}(\pi_1, \pi_2)$.

Note that the inner verifier above has perfect completeness. Indeed when B, C are long codes of b, c where $\pi_1(b) = \pi_2(c) = a$ (say), then for each $f \in \mathcal{F}_{L_U}$, if $f(a) = 1$ then $B(g_1) = g_1(b)$ while $B(g_2) = B(-g_1(f \circ \pi_1 \wedge g')) = -g_1(b)$ and so these are not equal, and similarly for the case when $f(a) = -1$.

3.3 Covering Soundness analysis

Let $X(\gamma)$ be the indicator random variable for the rejection of a particular proof $\Pi = \{\mathsf{LP}(w) : w \in W\}$ by the composed verifier $V_{\mathrm{comp}}(\mathrm{IV4}_\gamma)$ (henceforth $V_1(\gamma)$). The probability that $V_1(\gamma)$ rejects Π taken over its random choices is clearly the expectation

$$\mathop{\mathbf{E}}_{u,w,w',p,f,g_1,h_1,g_2,h_2}[X(\gamma)] = \mathbf{E}\left[\left(\frac{1 + B(g_1)B(g_2)}{2}\right)\left(\frac{1 + C(h_1)C(h_2)}{2}\right)\right] . \qquad (3)$$

Here B, C are shorthand for $\mathsf{LP}(w)$ and $\mathsf{LP}(w')$ respectively and equal $\mathsf{LONG}(\ell(w))$ and $\mathsf{LONG}(\ell(w'))$ respectively in a "correct" proof. We wish to say that no k proofs can together satisfy all the tests which $V_1(\gamma)$ performs. Now, if $X_k(\gamma)$ is the indicator random variable for the rejection of a set of k proofs $\{\mathsf{LP}_i(w) : w \in W\}$, $1 \le i \le k$, by the verifier $V_1(\gamma)$, then the overall probability that $V_1(\gamma)$ rejects all these k proofs, taken over its random choices, is exactly

$$\mathop{\mathbf{E}}_{u,w,w',p,f,g_1,h_1,g_2}[X_k(\gamma)] = \frac{1}{4^k}\left(\mathbf{E}\Big[\prod_{i=1}^{k}(1 + B_i(g_1)B_i(g_2))(1 + C_i(h_1)C_i(h_2))\Big]\right) , \qquad (4)$$

where we use the shorthand B_i, C_i for $\mathsf{LP}_i(w), \mathsf{LP}_i(w')$ respectively.

We now argue (see Lemma 3.2 below) that if this rejection probability is much smaller than 4^{-k}, then there is a way to obtain labels $\ell(u)$ for $u \in V \cup W$ by "decoding" Π_1 such that more than δ fraction of the edges (u, w) are satisfied by this labeling, i.e., $\ell(u) = \pi_{u,w}(\ell(w))$. Together with Theorem 3.1, this implies that the rejection probability (from Equation (4)) for any set of k proofs for a false claim of satisfiability (of φ), can be made arbitrarily close to $\frac{1}{4^k}$, and in particular is non-zero, and thus the covering soundness of the composed verifier is at most $1/k$.

Lemma 3.2 *There exists $a' < \infty$ such that for every integer $k \ge 1$, every ε, $0 < \varepsilon < 4^{-k}$, and all $\gamma \le \varepsilon/8$, if $\mathbf{E}[X_k(\gamma)] < \frac{1}{4^k} - \varepsilon$, then $\mathsf{OPT}(\mathcal{LC}) > \gamma^{-a'\gamma^{-1}}$.*

Before presenting the formal proof of Lemma 3.2, we first highlight the basic approach. The power of arithmetizing the rejection probability for a set of k proofs as in Equation (4) is that one can expand out the product and analyze the expectation of

$$\frac{1}{4^k}\sum_{S,T \subseteq [k]} B_S(g_1)B_S(g_2)C_T(h_1)C_T(h_2), \qquad (5)$$

where $B_S = \prod_{i \in S} B_i$ and $C_T = \prod_{i \in T} C_i$, where empty products are defined to be 1. A special term is $S = T = \emptyset$ which is the constant 1. We analyze the rest of the terms individually. We can now imagine two new proofs $\tilde{B} = B_S$ and $\tilde{C} = C_T$ which are exclusive-ors of subsets of the k given proofs. Now one can apply existing techniques from [17] to analyze terms involving the tables $\tilde{B}$ and $\tilde{C}$ and show that $\tilde{B}(g_1)\tilde{B}(g_2)$ and $\tilde{C}(h_1)\tilde{C}(h_2)$ cannot be too negative, and similarly if the expectation of $\tilde{B}(g_1)\tilde{B}(g_2)\tilde{C}(h_1)\tilde{C}(h_2)$ is too much below zero, then in fact $\mathsf{OPT}(\mathcal{LC})$ is quite large. In short, at a high level, we are saying that if there exist k proofs such that the verifier accepts at least one of them with good probability, then some exclusive-or of these proofs is also accepted by the verifier with good probability, and we know this cannot happen by the soundness analysis of [17] for the case of a single proof. This intuition is formalized via the next two lemmas from [17].

Before stating the lemmas, we make a slight digression to point out the relevance of not employing folding here. Folded long codes are typically used as follows: Given a table $A = \mathsf{LP}(w)$ supposedly giving the long code of the encoding of the label assigned to w, conceptually we assume we have a long proof $A' = \mathsf{FoldedLP}(w)$ which respects the constraints $A'(f) = -A'(-f)$. We generate such an A' for ourselves from A' by setting $A'(f) = A(f)$ if $f(x_0) = 1$ and $A'(-f) = -A(f)$ if $f(x_0) = -1$, where x_0 is some fixed element of the concerned domain (i.e., L_U or L_W as the case might be). Such a table A', which satisfies $A'(-f) = -A'(f)$ for every function f, is said to be *folded*. We then pretend the verifier works with the long code, but carry out the soundness analysis only for folded tables. In our case also we could do the same to analyze the acceptance of a single proof. However when faced with multiple proofs, the intermediate tables we consider, such as B_S above, need not be folded even if the original proofs we were given were folded — in particular, this will be the case when S has even cardinality. Thus our analysis needs to work with non "folded tables" as well. This is why we work with the long code directly.

Now we go back to the technical lemmas.

Lemma 3.3 ([17]) *For every $\gamma > 0$ and for all $B : \mathcal{F}_{L_W} \to \{1, -1\}$, and all $w \in W$*

$$\mathop{\mathbf{E}}_{p,v\in N(w),f,g,g'} [B(g_1)B(g_2)] \geq -4\gamma \ ,$$

where the distribution of p, f, g_1, g_2 is the same as the one in $IV4_\gamma$.

This lemma is Lemma 7.9 in [17] combined with calculation in the first half of Lemma 7.14 in the same paper. Similarly the next lemma follows from Lemma 7.12 of the same paper and a similar calculation.

Lemma 3.4 ([17]) *There exists $a < \infty$ such that for every $\gamma > 0$ and all proof tables $\{B_w\}$ and $\{C_w\}$, indexed by $w \in W$ with $B_w, C_w : \mathcal{F}_{L_W} \to \{1, -1\}$, we have $\mathbf{E}\left[B_w(g_1)B_w(g_2)C_{w'}(h_1)C_{w'}(h_2)\right]$ is at least*

$$-7\gamma - \mathsf{OPT}(\mathcal{LC})\gamma^{-a\gamma^{-1}} \ ,$$

where the expectation is taken over $p, u, w, w', g_1, h_1, g_2, h_2$, and where the distribution of p, g_1, g_2, h_1, h_2 is the same as the one in $IV4_\gamma$.

We are now ready to prove Lemma 3.2.

Proof of Lemma 3.2: The proof is actually simple given Lemmas 3.3 and 3.4. We pick a $\gamma > 0$ that satisfies $\gamma < \frac{\varepsilon}{8}$. By Equation (4), if $\mathbf{E}[X_k(\gamma)] < 4^{-k} - \varepsilon$, then there exist subsets S_1, S_2 of $\{1, 2, \ldots, k\}$, $S_1 \cup S_2 \neq \emptyset$, such that

$$\mathbf{E}[B_{S_1}(g_1)B_{S_1}(g_2)C_{S_2}(h_1)C_{S_2}(h_2)] < -\varepsilon \tag{6}$$

where B_{S_1} (resp. C_{S_2}) denotes $\Pi_{j\in S_1} B_j$ (resp. $\Pi_{j\in S_2} C_j$).

Suppose one of S_1, S_2 is empty, say $S_2 = \emptyset$. Lemma 3.3 applied to B_{S_1} (which is a function mapping $\mathcal{F}_{LW} \to \{1,-1\}$), gives $\mathbf{E}[B_{S_1}(g_1)B_{S_1}(g_2)] \geq -4\gamma$ which together with Equation (6) above yields $\gamma > \frac{\varepsilon}{4}$, a contradiction since $\gamma \leq \varepsilon/8$.

Now suppose both S_1 and S_2 are non-empty. Now we apply Lemma 3.4 to B_{S_1} and C_{S_2} to get that the expectation in Equation (6) is at least $-7\gamma - \mathsf{OPT}(\mathcal{LC})\gamma^{-a\gamma^{-1}}$. Together with Equation (6) this yields (using $\varepsilon \geq 8\gamma$)

$$\mathsf{OPT}(\mathcal{LC}) > \gamma\gamma^{a\gamma^{-1}} > \gamma^{a'\gamma^{-1}}$$

for some *absolute* constant a'. $\square$

We are now ready to state and prove the main Theorem of this section.

Theorem 3.5 *For every constant k,* $\mathrm{NP} \subseteq \mathrm{cPCP}_{1,\frac{1}{k}}[\log, 4]$.

Proof: The theorem follows from Lemma 3.2 and Theorem 3.1. Let $\varepsilon = \frac{1}{2} \cdot 4^{-k}$ and $\gamma = \varepsilon/8$, and pick $\delta > 0$ small enough so that $\gamma^{-a'\gamma^{-1}} > \delta$. By Lemma 3.2 we have $\mathbf{E}[X_k(\gamma)] < \frac{1}{4^k} - \varepsilon = \frac{1}{2\cdot 4^k}$ implies $\mathsf{OPT}(\mathcal{LC}) > \delta$. Consider the PCP with verifier $V_{\mathrm{comp}}(\mathrm{IV4}_\gamma)$. Using Theorem 3.1, we get that if the input formula φ is not satisfiable, the verifier $V_{\mathrm{comp}}(\mathrm{IV4}_\gamma)$ rejects any k proofs with probability at least $\frac{1}{2\cdot 4^k}$. Since it clearly has perfect completeness and makes only 4 queries, the claimed result follows. $\square$

Remark on tightness of the analysis: In fact, Lemma 3.2 can be used to show that for any $\varepsilon > 0$, there exists a (covering) PCP verifier that makes 4 queries, has perfect completeness and which rejects any set of k proofs with probability at least $\frac{1}{4^k} - \varepsilon$. Note that this analysis is in fact *tight* for the verifier $V_{\mathrm{comp}}(\mathrm{IV4})$ since a random set of k proofs is accepted with probability $1 - 4^{-k}$. It would have been sufficient to prove that for any k proofs the set of verifier coins causing the verifier to reject all k proofs is nonempty. We do not know a simpler proof of this weaker statement.

4 PCP Construction II and Hardness of Hypergraph Coloring

In the previous section we gave a PCP construction which made only 4 queries into the proof and had covering soundness smaller than any desired constant. This is already interesting in that it highlights the power of taking the covering soundness approach (since as remarked in the introduction one cannot achieve arbitrarily low soundness using classical PCPs with perfect completeness that make some fixed constant number of queries). We next turn to applying this to get a strong inapproximability result for hypergraph coloring.

The predicate tested by the inner verifier $\mathrm{IV4}_\gamma$ is $F(x,y,z,w) = (x \neq y) \vee (z \neq w)$, and to get a hardness result for hypergraph coloring, we require the predicate to be $\mathrm{NAE}(x,y,z,w)$ which is true unless all of x,y,z,w are equal. Note that $\mathrm{NAE}(x,y,z,w)$ is true whenever $F(x,y,z,w)$ is true, so one natural approach is to simply replace the predicate F tested by $\mathrm{IV4}_\gamma$ by NAE without losing perfect completeness. The challenge of course is to prove that the covering soundness does not suffer in this process, and this is exactly what we accomplish. For completeness we describe the inner verifier below.

`Inner Verifier` IV-NAE$4_\gamma^{B,C}$ (π_1, π_2)
Pick p as in $\mathrm{IV4}_\gamma$.
Pick $f, g_1, h_1, g', h', g_2, h_2$ as in Basic-IV4_p.
Accept iff not all of $B(g_1), B(g_2), C(h_1), C(h_2)$ are equal.

To analyze the soundness of the resulting composed verifier, we need to understand the "not-all-equal" predicate NAE. Note that NAE(x, y, z, w) rejects iff

$$\frac{1}{8}(1 + xy + xz + xw + yz + yw + zw + xyzw) = 1$$

and this sum equals zero otherwise. With similar notation as in the previous section this implies that for a given choice of g_1, g_2, h_1, h_2, the verifier rejects all k proofs iff

$$8^{-k} \sum_{S_1, S_2, S_3, S_4, S_1 \oplus S_2 \oplus S_3 \oplus S_4 = \emptyset} B_{S_1}(g_1) B_{S_2}(g_2) C_{S_3}(h_1) C_{S_4}(h_2) = 1, \tag{7}$$

where $\oplus$ denotes the exclusive-or of characteristic vectors, or worded differently, symmetric difference of sets. If the verifier accepts one the proofs then the right hand side of (7) must equal zero. Hence we study the expected value of this quantity.

Before proceeding with the analysis we shed some insight into the analysis and explain what is new this time. Let $T = S_1 \oplus S_2 = S_3 \oplus S_4$. The terms corresponding to T being the empty set are exactly the terms that appeared in the analysis of the verifier of Section 3. Let us turn our attention to terms where $T \neq \emptyset$. Typically, when a sum as the above appears, we would just go ahead and analyze the individual terms. Unfortunately, it turns out that we are unable to do this in our case. To see why, consider a typical summand above, namely $B_{S_1}(g_1) B_{S_1 \oplus T}(g_2) C_{S_3}(h_1) C_{S_3 \oplus T}(h_2)$. These are more general than the terms analyzed in Section 3, which were of the form $B_S(g_1) B_S(g_2) C_{S'}(h_1) C_{S'}(h_2)$. The first two elements of such a product come from an identical distribution, and similarly for the last two elements of the product. This in turn enabled a certain "pairing" up of terms from which a good solution to the label cover instance could be extracted (see the analysis in Lemma 7.12 of [17] for more details). But now, since $T \neq \emptyset$, the first two tables, B_{S_1} and $B_{S_1 \oplus T}$, are different, and so are the last two. Therefore, we now have to deal with individual terms which are the product of four elements each of which comes from a different distribution. It does not seem possible to analyze such a term by itself and extract meaningful solutions to the label cover instance.

To cope with this problem, we now bunch together terms $B_{S_1}(g_1) B_{S_1 \oplus T}(g_2) C_{S_3}(h_1) C_{S_3 \oplus T}(h_2)$ that involve the same T but different S_1 and S_3. (Alternatively, one could think of this as fixing T, and then picking S_1 and S_3 as random subsets of $[k]$ and considering the expectation of the terms over S_1, S_3 as well.) This makes the distribution of the first pair as a whole identical to that of the second pair, and allows us to analyze the terms above. More formally, for each non-empty $T \subseteq [k]$, we define

$$B^T(g_1, g_2) = \sum_S B_S(g_1) B_{S \oplus T}(g_2), \tag{8}$$

and similarly for C. Using this notation the sum in Equation (7) equals

$$8^{-k} \Bigg(1 + \sum_S B_S(g_1) B_S(g_2) + \sum_S C_S(h_1) C_S(h_2) + \sum_S B_S(g_1) B_S(g_2) C_S(h_1) C_S(h_2) + \sum_{T \neq \emptyset} B^T(g_1, g_2) C^T(h_1, h_2)\Bigg) \tag{9}$$

where the first four terms correspond to the case where $T = \emptyset$. Lemma 3.3 can be used to lower bound the expectation of the first two sums over g_1, g_2, h_1, h_2 and Lemma 3.4 can be used to lower bound the expectation of the third sum as a function of the optimum of the label cover instance. Thus we only need to study the last sum.

We show that if the last term is too negative, then one can extract an assignment of labels to the provers. The intuition behind the proof is as follows. B^T and C^T are two functions chosen independently from the same distribution. Further, the queried pairs (g_1, g_2) and (h_1, h_2) are also chosen from the same distribution, but are not independent of each other (and are related via f). If we ignore this dependence for a moment, then we get:

$$\mathbf{E}[B^T(g_1, g_2)C^T(h_1, h_2)] = \mathbf{E}[B^T(g_1, g_2)]\,\mathbf{E}[C^T(h_1, h_2)] = \mathbf{E}[B^T(g_1, g_2)]\,\mathbf{E}[B^T(g_1, g_2)] \geq 0,$$

and this would be good enough for us. Unfortunately, (g_1, g_2) and (h_1, h_2) are not independent. The intuition behind the proof of the next inequality is that if this correlation affects the expectation of $\mathbf{E}[B^T(g_1, g_2)C^T(h_1, h_2)]$, then there is some correlation between the tables for B^T and C^T and so a reasonable strategy for assigning labels to w and w' can be extracted. Specifically, we get the following lemma:

Lemma 4.1 *There exists $a' < \infty$ such that the following holds: Let $T \neq \emptyset$, $0 < \varepsilon < 2^{-k}$, and $\gamma \leq \varepsilon/8$ be such that*

$$\mathbf{E}\left[B^T(g_1, g_2)C^T(h_1, h_2)\right] \leq -\varepsilon,$$

where the expectation is taken over the distribution of $u, w, w', p, g_1, h_1, f, g', h', g_2, h_2$ as in IV-NAE4$_\gamma$. Then $\mathsf{OPT}(\mathcal{LC}) \geq \gamma^{a'\gamma^{-1}}$.

As usual we postpone the proof of the lemma, and instead prove the resulting theorem.

Theorem 4.2 *For every constant k, $\mathrm{NP} \subseteq \mathrm{cPCP}_{1,\frac{1}{k}}[\log, 4]$, where moreover the predicate verified by the PCP upon reading bits x, y, z, w is $\mathrm{NAE}(x, y, z, w)$.*

Proof: We only have to analyze the soundness of the verifier. Let a be the constant from Lemma 3.4 and a' be the constant from Lemma 4.1. Let $b = \max\{a, a'\}$. Let $\gamma = 2^{-(k+5)}$, and let $\delta < \gamma^{b\gamma^{-1}}$. To create a verifier for an instance SAT, reduce the instance of SAT to an instance of Label Cover using Theorem 3.1 with parameter δ and then use the verifier based on using IV-NAE4$_\gamma$ as the inner verifier. To show soundness, we need to show that if this verifier is covered by k proofs, then the instance of Label Cover has an optimum greater than δ.

Suppose we have k proofs such that the verifier always accept one of the proofs. This implies that the expectation, over $u, w, w', p, f, g_1, g_2, h_1, h_2$, of (9) is 0. This implies that at least one of summands in (9) is less than or equal to $-2^{-(k+2)}$ in expectation (since there are at most $4 \cdot 2^k$ summands in the expression). If it is a summand in one of the first two sums then this contradicts Lemma 3.3. If it is a summand in the third sums then by Lemma 3.4, we get that $\mathsf{OPT}(\mathcal{LC}) \geq \gamma^{a\gamma^{-1}} > \delta$. If it is a summand in the last sum, then by Lemma 4.1 we get that $\mathsf{OPT}(\mathcal{LC}) \geq \gamma^{a'\gamma^{-1}} > \delta$. Thus in the last two cases we get that the optimum is more than δ as desired. □

Before going on to the proof of Lemma 4.1, we discuss the consequences of Theorem 4.2 to hypergraph coloring. Before doing so, we just note that in fact one can prove a stronger claim in Theorem 4.2 that given any k proofs, the probability that the verifier rejects all of them is at least $\frac{1}{8^k} - \varepsilon$, for $\varepsilon > 0$ as small as we seek. The proof is really the same as that of Theorem 4.2, since we have argued that all terms in the expansion (9) are arbitrarily small in the case when optimum value of the label cover instance is very small. Once again this soundness analysis is tight, since a random set of k proofs will, in expectation, satisfy a fraction $1 - \frac{1}{8^k}$ of the verifier's checks.

4.1 Hardness results for hypergraph coloring

Since the predicate used by the PCP of Theorem 4.2 is that of 4-set splitting, we get the following Corollary.

Corollary 4.3 *For every constant $k \geq 2$, given an instance of 4-set splitting, it is NP-hard to distinguish between the case when there is a partition of the universe that splits all the 4-sets, and when for every set of k partitions there is at least one 4-set which is is not split by any of the k partitions.*

The above hardness can be naturally translated into a hardness result for coloring 4-uniform hypergraphs, and this gives us our main result:

Theorem 4.4 (Main Theorem) *For any constant $c \geq 2$, it is NP-hard to color a 2-colorable 4-uniform hypergraph using c colors.*

Proof: Follows from the above Corollary since a 4-set splitting instance can be naturally identified with a 4-uniform hypergraph whose hyperedges are the 4-sets, and it is easy to see that the minimum number of partitions k needed to split all 4-sets equals $\lceil \log c \rceil$ where c is the minimum number of colors to color the hypergraph such that no hyperedge is monochromatic. $\square$

In light of the discussion after the proof of Theorem 4.2, we in fact have the following stronger result.

Theorem 4.5 *For any constant $c \geq 2$ and every $\varepsilon > 0$, it is NP-hard to color a 2-colorable 4-uniform hypergraph using c colors such that at least a fraction $(1 - \frac{1}{8^{\lceil \log c \rceil}} + \varepsilon)$ of the hyperedges are properly colored (i.e., are not monochromatic).*

Theorem 4.6 *Assume* NP $\not\subseteq$ DTIME$(n^{O(\log \log n)})$. *Then there exists an absolute constant $c_0 > 0$ such that there is no polynomial time algorithm that can color a 2-colorable 4-uniform hypergraph using $c_0 \frac{\log \log n}{\log \log \log n}$ colors, where n is the number of vertices in the hypergraph.*

Proof: This follows since the covering soundness of the PCP in Theorem 4.2 can be made an explicit $o(1)$ function. Indeed, nothing prevents from having a k that is a function of n. We need to have $\gamma = 2^{-(k+5)}$ and to reach a contradiction $\delta < \gamma^{O(\gamma^{-1})}$. The proof size we need is $n^{O(\log \delta^{-1})} 2^{\delta^{-O(1)}}$. We can thus have $n^{O(\log \log n)}$ size proofs by letting $\delta^{-1} = (\log n)^{O(1)}$ and $2^k = \Omega(\gamma^{-1}) = \Omega(\frac{\log \log n}{\log \log \log n})$. Similarly to Theorem 4.4, this implies 2^k-coloring a 2-colorable 4-uniform hypergraph is hard unless NP $\subseteq$ DTIME$(n^{O(\log \log n)})$. $\square$

We now show that a hardness result similar to Theorem 4.4 also holds for 2-colorable k-uniform hypergraphs for any $k \geq 5$.

Theorem 4.7 *Let $k \geq 5$ be an integer. For any constant $\ell \geq 2$, it is NP-hard to color a 2-colorable k-uniform hypergraph using ℓ colors.*

Proof: The proof works by reducing from the case of 4-uniform hypergraphs, and the claimed hardness then follows using Theorem 4.4.

Let $\mathcal{H}$ be a 4-uniform hypergraph with vertex set V. Suppose that $k = 4s + t$ where $1 \leq t \leq 4$. Construct a k-uniform hypergraph $\mathcal{H}'$ as follows. The vertex set of $\mathcal{H}'$ is $V^{(1)} \cup V^{(2)} \cup \cdots \cup V^{(s\ell+1)}$ where the sets $V^{(j)}$ are independent copies of V. On each $V^{(j)}$, take a collection $\mathcal{F}^{(j)}$ of 4-element subsets of $V^{(j)}$ that correspond to the hyperedges in $\mathcal{H}$. A hyperedge of $\mathcal{H}'$ (which is a $(4s+t)$-element subset of $\bigcup_j V^{(j)}$) is now given by the union of s 4-sets belonging to s different $\mathcal{F}^{(j)}$'s, together with t vertices picked from a 4-set belonging to yet another $\mathcal{F}^{(j)}$. More formally, for every set of $(s+1)$ distinct indices $j_1, j_2, \ldots, j_{s+1}$, every choice of elements $e_{j_i} \in \mathcal{F}^{(j_i)}$ for $i = 1, \ldots, s+1$, and every t-element subset $f_{j_{s+1}}$ of $e_{j_{s+1}}$, there is a hyperedge $(e_{j_1} \cup \cdots \cup e_{j_s} \cup f_{j_{s+1}})$ in $\mathcal{H}'$.

If $\mathcal{H}$ is 2-colorable then clearly any 2-coloring of it induces a 2-coloring of $\mathcal{H}'$, and hence $\mathcal{H}'$ is 2-colorable as well.

Suppose $\mathcal{H}$ is not ℓ-colorable and that we are given an ℓ-coloring of $\mathcal{H}'$. Since $\mathcal{H}$ is not ℓ-colorable, each $\mathcal{F}^{(j)}$, for $1 \leq j \leq s\ell + 1$, must contain a monochromatic set g_j. By the pigeonhole principle, there must be a color c such that $(s+1)$ different g_j's have color c. The hyperedge of $\mathcal{H}'$ constructed from those $(s+1)$ sets is then clearly monochromatic (all its vertices have color c) and we conclude that $\mathcal{H}'$ is not ℓ-colorable.

Since the reduction runs in polynomial time when k and ℓ are constants the proof is complete. $\square$

4.2 Discrete Fourier transforms

Before going on to the proof of Lemma 4.1 we now introduce a tool that has been crucial in the analysis on inner verifiers. This was hidden so far from the reader but already used in the proofs of Lemmas 3.3 and 3.4 in [17]. Now we need to introduce them explicitly.

In general we consider functions mapping D to $\{1, -1\}$. For $\alpha \subseteq D$ and $f \in \mathcal{F}_D$, let $\chi_\alpha(f) = \prod_{x \in \alpha} f(x)$. Notice that $\chi_{\{x\}}$ is the long code of x. For any function A mapping $\mathcal{F}_D$ to the reals, we have the corresponding Fourier coefficients

$$\hat{A}_\alpha = 2^{-2^{|D|}} \sum_f A(f)\chi_\alpha(f)$$

where $\alpha \subseteq D$. We have the Fourier inversion formula given by

$$A(f) = \sum_\alpha \hat{A}_\alpha \chi_\alpha(f)$$

and Plancherel's equality that states that

$$\sum_\alpha \hat{A}_\alpha^2 = 2^{-2^{|D|}} \sum_f A(f)^2.$$

In the case when A is a Boolean function the latter sum is clearly 1.

We think of an arbitrary table A as being somewhat close to (or coherent with) the long code of x if there exists a small set α containing x such that $\hat{A}_\alpha$ is non-negligibly large. Thus, when viewing the long proofs of w and w', our goal is to show that the $\mathsf{LP}(w)$ and $\mathsf{LP}(w')$ have coherence with the long codes of strings x and y such that $\pi_{u,w}(x)$ and $\pi_{u,w'}(y)$ are equal.

4.3 Proof of Lemma 4.1

Fix $T \subseteq [k]$. Throughout this section the quantities that we define depend on T, but we don't include it as a parameter explicitly.

Recall we need to show that if the expectation, over $u, w, w', p, g_1, g_2, h_1, h_2$ of $B^T(g_1, g_2)C^T(h_1, h_2)$ is too negative (less than $-\varepsilon$), then we can assign labels to the label cover problem with acceptance probability more that δ. Recall that $g_2 = g_1(f \circ \pi_1 \wedge g')$ is defined in terms of other random variables f and g' and similarly h_2 in terms of f and h'. For brevity, we let X denote the quantity $B^T(g_1, g_2)C^T(h_1, h_2)$. Notice that X is a random variable depending on all the variables above. We first analyze the expectation of X over g_1 and h_1 (for fixed choice of u, w, w', p, f, g' and h'). Next we calculate the expectation over f, g' and h'. In both stages we get exact expressions. Finally we make some approximations for the expectation over u, w, w'. (The careful reader may observe that we don't take expectations over p — in fact the lemma holds for every choice of p of the inner verifier IV-NAE4$_\gamma$.)

The crux of this proof are the functions $B^*, C^* : \mathcal{F}_{L_W} \to \{1, -1\}$, defined as follows: We let

$$B^*(g) = \mathop{\mathbf{E}}_h\Big[\sum_S B_S(h)B_{S\oplus T}(gh)\Big]$$

and

$$C^*(g) = \mathop{\mathbf{E}}_h\Big[\sum_S C_S(h)C_{S\oplus T}(gh)\Big] .$$

Note that for a fixed choice of f and g' we have $\mathop{\mathbf{E}}_{g_1}[B^T(g_1, g_2)] = B^*(-((f \circ \pi_1) \wedge g'))$. We get a similar expression for C^T and thus we get:

$$\mathop{\mathbf{E}}_{g_1,h_1}[X] = B^*(-((f \circ \pi_1) \wedge g'))C^*(-((-f \circ \pi_2) \wedge h')) .$$

Let us call the above quantity Y.

In what follows, we rely crucially on the properties of the Fourier coefficients of B^* and C^*. Let $\hat{F}_\beta$ and $\hat{G}_\beta$ denote the Fourier coefficients of B^* and C^* respectively. From the definitions and some standard manipulation, we get

$$\hat{F}_\beta = \sum_S \hat{B}_{\beta,S}\hat{B}_{\beta,S\oplus T} \quad \text{and} \quad \hat{G}_\beta = \sum_S \hat{C}_{\beta,S}\hat{C}_{\beta,S\oplus T}.$$

Using simple Fourier expansion, we can rewrite the quantity we are analyzing as:

$$\begin{aligned} Y &= B^*(-((f \circ \pi_1) \wedge g'))C^*(-((-f \circ \pi_2) \wedge h')) \\ &= \sum_{\beta,\beta'} \hat{F}_\beta \hat{G}_{\beta'} \chi_\beta(-((f \circ \pi_1) \wedge g'))\chi_{\beta'}(-((-f \circ \pi_2) \wedge h')) \\ &= \sum_{\beta,\beta'} \hat{F}_\beta \hat{G}_{\beta'} \prod_{y\in\beta}(-(f(\pi_1(y)) \wedge g'(y))) \prod_{z\in\beta'}(-(-f(\pi_2(z)) \wedge h'(z))) \end{aligned}$$

The main property about the Fourier coefficients of B^* and C^* is that their L_1 norm is bounded. Specifically, we have:

$$\sum_\beta |\hat{F}_\beta| \le \sum_{\beta,S} |\hat{B}_{\beta,S}\hat{B}_{\beta,S\oplus T}| \le \sum_S \left(\sum_\beta \hat{B}^2_{\beta,S}\right)^{1/2} \left(\sum_\beta \hat{B}^2_{\beta,S\oplus T}\right)^{1/2} \le 2^k. \tag{10}$$

We start by defining the strategy we use to assign labels and prove that if the expectation (of Y) is large, then the labels give an assignment to the label cover instance with objective of at least $\delta = \gamma^{a'\gamma^{-1}}$.

Strategy. Given $w \in W$, and tables $B_1, \ldots, B_k$ corresponding to $\mathsf{LP}(w)$ in k different proofs, compute B_S for every $S \subseteq [k]$, B^* and its Fourier coefficients. Pick a non-empty set $\beta \subseteq L_W$ with probability $2^{-k}|\hat{F}_\beta|$ and assign as label to w, an element $x \in \beta$ chosen uniformly at random. With remaining probability, since $\sum_{\beta \neq \emptyset} |\hat{F}_\beta|$ may be less than 2^k, assign no label to w.

Preliminary analysis. We now give a preliminary expression for the success probability of the strategy. Consider picking u, w, and w' (and the associated π_1 and π_2) at random and checking for the event $\pi_1(\ell(w)) = \pi_2(\ell(w'))$. The probability of this event is lower bounded by the probability that $\pi_1(\beta)$ and $\pi_2(\beta')$ intersect and we assign the elements corresponding to this intersection to w and w'. The probability of these events is at least:

$$\mathop{\mathbf{E}}_{u,w,w'}\left[\sum_{\beta,\beta':\pi_1(\beta)\cap\pi_2(\beta')\neq\emptyset} \frac{2^{-2k}}{|\beta|\cdot|\beta'|}|\hat{F}_\beta||\hat{G}_{\beta'}|\right]. \tag{11}$$

Below we show that this quantity is large if the expectation of Y is too small. We now return to the expectation of Y.

An exact expression for the expectation of Y. We start with some notation. Fix u, w, w', p and π_1 and π_2. For $x \in L_U$ and $\beta, \beta' \subseteq L_W$, let $s_x(\beta) = |\beta \cap \pi_1^{-1}(x)|$ and let $t_x(\beta') = |\beta' \cap \pi_2^{-1}(x)|$. Since the argument of s_x is always β and the argument of t_x is always β', we use the shorthand s_x for $s_x(\beta)$ and t_x for $t_x(\beta')$. Further for real p and non-negative integers s, t, let $\alpha(p, s, t) = \frac{1}{2}\left((-1)^s(1-2p)^t + (1-2p)^s(-1)^t\right)$, and let $\eta(p, s) = \frac{1}{2}\left((-1)^s + (1-2p)^s\right)$. Next we show that

$$\mathop{\mathbf{E}}_{f,g',h'}[Y] = \sum_{\beta,\beta'} \hat{F}_\beta \hat{G}_{\beta'} \prod_{x \in L_U} \alpha(p, s_x, t_x). \tag{12}$$

To prove the above it suffices to show that

$$\mathop{\mathbf{E}}_{f,g',h'}\left[\prod_{y\in\beta}[-(f(\pi_1(y)) \wedge g'(y))] \prod_{z\in\beta'}[-(-f(\pi_2(z)) \wedge h'(z))]\right] = \prod_{x\in L_U} \alpha(p, s_x, t_x).$$

Factors corresponding to y and z with different projections on L_U are independent, and thus the expectation can be broken down into a product of expectations, one for each $x \in L_U$. Fix $x \in L_U$ and consider the term

$$\begin{aligned}
&\mathop{\mathbf{E}}_{f,g',h'}\left[\prod_{y\in\beta\cap\pi_1^{-1}(x)} (-(f(\pi_1(y)) \wedge g'(y))) \prod_{z\in\beta'\cap\pi_2^{-1}(x)} (-(-f(\pi_2(z)) \wedge h'(z)))\right] \\
= &\mathop{\mathbf{E}}_{f,g',h'}\left[\prod_{y\in\beta\cap\pi_1^{-1}(x)} (-(f(x)) \wedge g'(y))) \prod_{z\in\beta'\cap\pi_2^{-1}(x)} (-(-f(x)) \wedge h'(z)))\right].
\end{aligned}$$

If $f(x) = 1$ (or "false") the first product equals $(-1)^{s_x}$ and the second equals $(1-2p)^{t_x}$. Similarly, If $f(x) = -1$ the first product equals $(1-2p)^{s_x}$ and the second equals $(-1)^{t_x}$. The events happen with probability $1/2$ each and thus giving that the expectation above (for fixed x) equals

$$\frac{1}{2}\left((-1)^{s_x}(1-2p)^{t_x} + (1-2p)^{s_x}(-1)^{t_x}\right) = \alpha(p, s_x, t_x).$$

Taking the product over all x's gives (12).

Inequalities on $\mathbf{E}[Y]$. For every u, we now show how to lower bound the expectation of Y, over w, w', f, g', h' in terms of a sum involving only β's and β' that *intersect* in their projections. This brings us much closer to the expression derived in our preliminary analysis of the success probability of our strategy for assigning labels and we lower bound a closely related quantity. Specifically we now use the inequality $\mathop{\mathbf{E}}_{w,w',f,g',h'}[Y] \leq -\varepsilon$ (as guaranteed by the Lemma statement) to show:

$$\mathop{\mathbf{E}}_{u,w,w'}\left[\sum_{\beta,\beta' | \pi_1(\beta)\cap\pi_2(\beta')\neq\emptyset} 2|\hat{F}_\beta \hat{G}_{\beta'}| e^{-\mu_p(\beta)/2}\right] \geq \varepsilon, \tag{13}$$

where $\mu_p(\beta) = \mu_p(\beta, u, w) = \sum_x \min\{1, ps_x\}$ is the quantity defined in Equation (1). (A similar inequality with $\mu_p(\beta')$ in the exponent follows by symmetry.)

To prove the above, consider the following expression, which is closely related to the expectation of Y as given by (12).

$$Y_1 = \left(\sum_\beta \hat{F}_\beta \prod_x \eta(p, s_x)\right)\left(\sum_{\beta'} \hat{G}_{\beta'} \prod_x \eta(p, t_x)\right).$$

First we note that $\mathop{\mathbf{E}}_{w,w'}[Y_1] = \left(\mathop{\mathbf{E}}_{w}\left[\left(\sum_\beta \hat{F}_\beta \prod_x \eta(p, s_x)\right)\right]\right)^2 \geq 0$. (Here we are using the fact that the tables B^T and C^T are chosen from the same distribution.) Next we note that the difference between Y and Y_1 arises only from terms involving β, β' such that $\pi_1(\beta)\cap\pi_2(\beta') \neq \emptyset$. To verify this, note that if $s = 0$ or $t = 0$ then $\alpha(p, s, t) = \eta(p, s)\cdot\eta(p, t)$. Since either s_x or t_x equals 0 for every x, if $\pi_1(\beta)\cap\pi_2(\beta') = \emptyset$, we get that terms corresponding to such pairs of β, β' vanish in $Y - Y_1$. We conclude:

$$\mathop{\mathbf{E}}_{w,w',f,g',h'}[Y] = \mathop{\mathbf{E}}_{w,w'}[Y_1] + \sum_{\beta,\beta':\pi_1(\beta)\cap\pi_2(\beta')\neq\emptyset} \hat{F}_\beta \hat{G}_{\beta'} \Big(\prod_x \alpha(p, s_x, t_x) - \prod_x (\eta(p, s_x)\cdot\eta(p, t_x))\Big).$$

Using $\mathop{\mathbf{E}}_{w,w',f,g',h'}[Y] \leq -\varepsilon$, $\mathbf{E}[Y_1] \geq 0$ and taking absolute values we get,

$$\begin{aligned}
\varepsilon &\leq \left|\sum_{\beta,\beta':\pi_1(\beta)\cap\pi_2(\beta')\neq\emptyset} \hat{F}_\beta \hat{G}_{\beta'} \Big(\prod_x \alpha(p, s_x, t_x) - \prod_x (\eta(p, s_x)\cdot\eta(p, t_x))\Big)\right| \\
&\leq \sum_{\beta,\beta':\pi_1(\beta)\cap\pi_2(\beta')\neq\emptyset} |\hat{F}_\beta \hat{G}_{\beta'}| \left(\prod_x |\alpha(p, s_x, t_x)| + \prod_x |\eta(p, s_x)|\right) \qquad (14)
\end{aligned}$$

where the last inequality uses $|\eta(p, t)| \leq 1$ for every $t \geq 0$.

Next we simplify the terms of the LHS above. First we show that for every $t \geq 0$,

$$|\alpha(p, s, t)|, |\eta(p, s)| \leq e^{-\min\{1, ps\}/2}. \tag{15}$$

First we note both $|\alpha(p, s, t)|$ and $|\eta(p, s)|$ are upper bounded by $\frac{1}{2}(1 + (1-2p)^s) \leq \frac{1}{2}(1 + e^{-2ps})$. If $p \geq s^{-1}$, then we have $\frac{1}{2}(1 + e^{-2ps}) \leq \frac{1}{2}(1 + e^{-2}) \leq e^{-1/2}$ as required. If $p \leq \frac{1}{s}$, let us set $\lambda(z) = 2e^{-z/2} - (1 + e^{-2z})$. To show (15), we need to prove that $\lambda(z) \geq 0$ for $z \in [0, 1]$. We have $\lambda''(z) = \frac{1}{2}e^{-z/2} - 4e^{-2z}$ and hence $\lambda''(z) \leq 0$ in the interval in question and we only have to check the inequality at the end points. We have $\lambda(0) = 0$ and $\lambda(1) = 2e^{-1/2} - (1 + e^{-2}) > 0$.

Using (15) we conclude that

$$\prod_x |\alpha(p, s_x, t_x)| + \prod_x |\eta(p, s_x)| \le 2\prod_x e^{-\min\{1, ps_x\}} = 2e^{-\mu_p(s_x)}.$$

Substituting the above into (14) gives (13).

Based on (13). we want to prove that the strategy for assigning labels is a good one. First we prove that large sets β do not contribute much to the LHS of the sum in (13). Define

$$K = p^{-1}(\varepsilon^{-1}2^{2k+4}(4k+8+2\log\varepsilon^{-1}))^{1/c}.$$

We have

Lemma 4.8 *We have*

$$\mathbf{E}\Big[\sum_{\pi(\beta)\cap\pi(\beta')\neq\emptyset, |\beta|\ge K} 2|\hat{F}_\beta \hat{G}_{\beta'}| e^{-\mu_p(\beta)/2}\Big] \le \varepsilon/4.$$

Proof: By Property (iv) of Theorem 3.1 we have that the probability that $\mu_p(\beta) \le (4k+8+2\log\varepsilon^{-1})$ is at most $\varepsilon 2^{-(2k+4)}$. A similar chain of inequalities as (10) shows that $\sum |\hat{G}_{\beta'}| \le 2^k$. The sum in the lemma can hence be estimated as

$$\begin{aligned} 2^{k+1}\sum_{|\beta|\ge K} |\hat{F}_\beta| \mathop{\mathbf{E}}_u[e^{-\mu_p(\beta)/2}] &\le 2^{k+1}\sum_{|\beta|\ge K} |\hat{F}_\beta|(2^{-(2k+4)}\varepsilon + e^{-(2k+4+\log\varepsilon^{-1})}) \\ &\le 2^{2k+1}(2^{-(2k+4)}\varepsilon + 2^{-(2k+4)}\varepsilon) \\ &\le \varepsilon/4, \end{aligned}$$

and the lemma follows. □

By the same argument applied to β' of size at least K, together with Equation (13), we get

$$\mathbf{E}\Big[\sum_{\pi(\beta)\cap\pi(\beta')\neq\emptyset, |\beta|,|\beta'|\le K} 2|\hat{F}_\beta \hat{G}_{\beta'}|\Big] \ge \varepsilon/2. \tag{16}$$

We now relate to the probability of success of our strategy for assigning labels. From (11) we know this quantity is at least

$$\begin{aligned} &\mathop{\mathbf{E}}_{u,w,w'}\Big[\sum_{\beta,\beta':\pi_1(\beta)\cap\pi_2(\beta')\neq\emptyset} \frac{2^{-2k}}{|\beta|\cdot|\beta'|}|\hat{F}_\beta||\hat{G}_{\beta'}|\Big] \\ \ge\; &\mathop{\mathbf{E}}_{u,w,w'}\Big[\sum_{\pi_1(\beta)\cap\pi_2(\beta')\neq\emptyset, |\beta|,|\beta'|\le K} \frac{2^{-2k}}{K^2}|\hat{F}_\beta||\hat{G}_{\beta'}|\Big] \\ \ge\; &\frac{\varepsilon 2^{-(2k+2)}}{K^2} \end{aligned}$$

where the last inequality uses (16). (It is easy to convert this randomized strategy for assigning labels to a deterministic one that does equally well.) The dominating factor in the expression is the term $p^{-O(1)}$ (from the definition of K) which can be calculated to be $\gamma^{O(\gamma^{-1})}$ and the proof of Lemma 4.1 is complete. □

4.3.1 Comparison to previous proof of Theorem 4.4

We point out that the conference version of this paper [15] contained a different proof of Theorem 4.4. The current proof is significantly simpler, and furthermore it is only a minor adjustment of similar proofs in [17]. The key observation to make the current proof possible is the insight that we should treat the terms of (7) in the collections given by $B^T(g_1, g_2)C^T(h_1, h_2)$ as it does not seem possible to handle them one by one in an efficient manner. The previous proof did not make this observation explicitly and ended up being significantly more complicated. This "simplicity" in turn has already enabled some further progress on the hypergraph coloring problem — in particular, using this style of analysis, Khot [21] shows a better super-constant hardness for a-colorable 4-uniform hypergraphs for $a \geq 7$.

4.3.2 Subsequent related work

In a very recent work, Holmerin [18] showed that the vertex cover problem considered on 4-uniform hypergraphs is NP-hard to approximate within a factor of $(2 - \varepsilon)$ for arbitrary $\varepsilon > 0$. (A subset S of vertices of a hypergraph H is said to be a vertex cover if every hyperedge of H intersects S.) He proves this by modifying the soundness analysis of Håstad's 4-set splitting verifier (which is also the verifier we use in Section 4) to show that any proof which sets only a fraction ε of bits to -1 will cause some 4-tuple tested by the verifier to consist of only 1's. This in turn shows that for every constant $\varepsilon > 0$, given a 2-colorable 4-uniform hypergraph, it is NP-hard to find an independent set that consists of a fraction ε of vertices. Note that this result is stronger as a small independent set implies a large chromatic number and it thus immediately implies the hardness of coloring such a 2-colorable 4-uniform hypergraph with $1/\varepsilon$ colors, and hence our main result (Theorem 4.4). We stress that the verifier in Holmerin's paper is the same as the one in this paper; however, the analysis in [18] obtains our result without directly referring to covering complexity.

Acknowledgments

We would like to thank the anonymous referees and Oded Goldreich for useful comments on the presentation of the paper.

References

[1] N. Alon, P. Kelsen, S. Mahajan and H. Ramesh. Coloring 2-colorable hypergraphs with a sublinear number of colors. *Nordic Journal of Computing*, 3 (1996), pp. 425-439.

[2] S. Arora, L. Babai, J. Stern, and Z. Sweedyk. The hardness of approximate optima in lattices, codes, and systems of linear equations. *Journal of Computer and System Sciences*, 54(2):317–331, April 1997.

[3] S. Arora and C. Lund. Hardness of Approximations. In *Approximation Algorithms for NP-hard Problems*, (Dorit Hochbaum, ed.), PWS, 1996.

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

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

[6] J. Beck. On 3-chromatic hypergraphs. *Discrete Mathematics*, 24 (1978), pp. 127-137.

[7] J. Beck. An algorithmic approach to the Lovász Local Lemma. *Random Structures and Algorithms*, 2 (1991), pp. 343-365.

[8] M. Bellare, O. Goldreich and M. Sudan. Free bits, PCP's and non-approximability – towards tight results. *SIAM Journal on Computing*, 27(3):804-915, 1998.

[9] A. Blum and D. R. Karger. Improved approximation for graph coloring. *Information Processing Letters*, 61(1):49–53, January 1997.

[10] H. Chen and A. Frieze. Coloring bipartite hypergraphs. *Proc. of 5th IPCO*, pp. 345-358, Lecture Notes in Computer Science, vol. 1084, Springer, 1996.

[11] P. Erdös. On a combinatorial problem I. *Nordisk Mat. Tidskrift*, 11 (1963), pp. 5-10.

[12] U. Feige and J. Kilian. Zero-knowledge and the chromatic number. *Journal of Computer and System Sciences*, 57:187–199, 1998.

[13] M. Goemans and D. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM*, 42:1115-1145, 1995.

[14] V. Guruswami. Inapproximability results for set splitting and satisfiability problems with no mixed clauses. *Proc. of the 3rd Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX'00)*, pp. 155-166, 2000. Journal version accepted to *Algorithmica.*

[15] V. Guruswami, J. Håstad, and M. Sudan. Hardness of Approximate Hypergraph Coloring. *Proceedings of 41st Annual IEEE Symposium on Foundations of Computer Science*, 2000, pp. 149-158.

[16] V. Guruswami and S. Khanna. On the hardness of 4-coloring a 3-colorable graph. *Proc. of Complexity 2000*, pp. 188-197.

[17] J. Håstad. Some optimal inapproximability results. Accepted to appear in *Journal of the ACM*, available from http://www.nada.kth.se/ johanh/papers.html. Preliminary version in *Proceedings of STOC'97.*

[18] J. Holmerin. Vertex cover on 4-regular hyper-graphs is hard to approximate within $(2 - \varepsilon)$. *Proceedings of the 34th ACM Symposium on Theory of Computing (STOC)*, May 2002, to appear.

[19] D. R. Karger, R. Motwani and M. Sudan. Approximate graph coloring using semidefinite programming. *Journal of the ACM*, 45 (1998), pp. 246-265.

[20] S. Khanna, N. Linial and S. Safra. On the hardness of approximating the chromatic number. *Combinatorica*, 20(3) (2000), pp. 393-415.

[21] S. Khot. Hardness results for approximate hypergraph coloring. *Proceedings of the 34th ACM Symposium on Theory of Computing (STOC)*, May 2002, to appear.

[22] M. Krivelevich and B. Sudakov. Approximate coloring of uniform hypergraphs. *Proc. of the 6th European Symposium on Algorithms (ESA'98)*, LNCS 1461, pp. 477-489, 1998.

[23] L. Lovász. Coverings and colorings of hypergraphs. *Proc. 4th Southeastern Conf. on Combinatorics, Graph Theory, and Computing*, pp. 3-12, Utilitas Mathematica Publishing, Winnipeg, 1973.

[24] C. Lund and M. Yannakakis. On the hardness of approximating minimization problems. *Journal of the ACM*, 41:960-981, 1994.

[25] C. McDiarmid. A random recoloring method for graphs and hypergraphs. *Combinatorics, Probability and Computing*, 2 (1993), pp. 363-365.

[26] C. McDiarmid. Hypergraph coloring and the Lovász Local Lemma. *Discrete Mathematics*, 167/168 (1997), pp. 481-486.

[27] J. Radhakrishnan and A. Srinivasan. Improved bounds and algorithms for hypergraph 2-coloring. *Random Structures and Algorithms*, 16 (2000), pp. 4-32. Preliminary version in *Proc. of* 39^{th} *FOCS*, (1998), pp. 684-693.

[28] R. Raz. A parallel repetition theorem. *SIAM Journal on Computing*, 27(3):763–803, 1998.

[29] J. H. Spencer. Coloring n-sets red and blue. *J. Combinatorial Theory, Series A*, 30 (1981), pp. 112-113.