Revisiting Alphabet Reduction in Dinur's PCP

Venkatesan Guruswami
Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, USA
venkatg@cs.cmu.edu

Jakub Opršal
Computer Science Department, Durham University, UK
jakub.oprsal@durham.ac.uk

Sai Sandeep
Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, USA
spallerl@andrew.cmu.edu

Abstract

Dinur's celebrated proof of the PCP theorem alternates two main steps in several iterations: gap amplification to increase the soundness gap by a large constant factor (at the expense of much larger alphabet size), and a composition step that brings back the alphabet size to an absolute constant (at the expense of a fixed constant factor loss in the soundness gap). We note that the gap amplification can produce a Label Cover CSP. This allows us to reduce the alphabet size via a direct long-code based reduction from Label Cover to a Boolean CSP. Our composition step thus bypasses the concept of Assignment Testers from Dinur's proof, and we believe it is more intuitive – *it is just a gadget reduction.* The analysis also uses only elementary facts (Parseval's identity) about Fourier Transforms over the hypercube.

2012 ACM Subject Classification Theory of computation → Problems, reductions and completeness; Theory of computation → Approximation algorithms analysis

Keywords and phrases PCP theorem, CSP, discrete Fourier analysis, label cover, long code

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2020.34

Category APPROX

Funding *Venkatesan Guruswami*: Supported in part by NSF grants CCF-1526092, CCF-1814603, and CCF-1908125.
Jakub Opršal: Received funding from the UK EPSRC grant EP/R034516/1. Most of the work has been done during a visit to Carnegie Mellon University.
Sai Sandeep: Supported in part by NSF grant CCF-1908125.

Acknowledgements We would like to thank Irit Dinur, Margalit Glasgow, Oded Goldreich, Prahladh Harsha, Johan Håstad, Jaikumar Radhakrishnan, and Madhu Sudan for useful comments and feedback. We also thank the anonymous reviewers for valuable comments on the presentation.

1 Introduction

Constraint Satisfaction Problem (CSP) is a canonical NP-complete problem. Assuming P $\neq$ NP, no polynomial time algorithm can find a satisfying assignment to a satisfiable CSP instance. If we are happy with the easier goal of satisfying a $1 - o(1)$ fraction of constraints, does there exist an efficient algorithm to do so? Answering this in the negative, the fundamental PCP theorem [1, 2] implies that for some fixed integers $k, q \geq 2$ and $c < 1$, it is NP-hard to find an assignment satisfying a fraction c of constraints in a satisfiable CSP of arity k over alphabet $\{0, 1, \ldots, q-1\}$. Further this result holds for the combinations $(q, k) = (2, 3)$ and $(3, 2)$. The PCP theorem lies at the center of a rich body of work that has yielded numerous inapproximability results, including many optimal ones.

The PCP theorem was originally proved using algebraic techniques such as the low-degree test and the sum-check protocol. In a striking work, Dinur [7] gave an alternative combinatorial proof of the PCP theorem. Her proof works by amplifying the "Unsat value" of a CSP instance – the fraction of constraints any assignment should violate. The goal is to show that it is NP-hard to distinguish if the Unsat value of a CSP instance is equal to 0 or at least a constant $c > 0$. Starting with a NP-hard problem such as 3-coloring with m constraints, we can already deduce that it is NP-hard to identify if Unsat value is equal to 0 or at least $1/m$. The Unsat value is increased slowly and iteratively via two steps – *gap amplification* and *alphabet reduction*. In gap amplification, we incur a constant factor blow up in the size of the instance, and get a constant factor improvement in the Unsat value. However, this step also blows up the alphabet size. To alleviate this, alphabet reduction brings back the alphabet size to an absolute constant while losing a constant factor in the Unsat value (and blows up the instance size by a constant factor). Combining both the steps, we can increase the Unsat value by a constant factor (say 2) incurring a constant factor blow up in the size of the instance. Repeating this $\log m$ times proves the PCP theorem.

In this paper we revisit the alphabet reduction step. Dinur implemented this step by an "inner" PCP construction, which is in effect a gadget reducing a specific predicate ψ to be tested to a collection Ψ of constraints over a fixed (say Boolean) alphabet, such that if ψ is unsatisfiable, then a constant fraction of constraints of Ψ must be violated by any assignment.[1] This inner PCP is then applied to all constraints in the CSP instance (say $\mathcal{G}$) produced by the gap amplification step. The collection of inner PCPs as such only ensure that each constraint of $\mathcal{G}$ is individually satisfiable, which is not very meaningful. To ensure that the inner PCPs together ensure that the constraints of $\mathcal{G}$ are all satisfiable by a *single consistent* assignment, error-correcting codes are used to encode the purported assignments to the variables of $\mathcal{G}$. The inner PCP is also replaced by an *Assignment Tester* that ascertains whether the specific assignment given by these encodings satisfies the predicate ψ being checked.

The key observation driving this work is that instead of designing the inner PCP for *arbitrary* constraints (as in Dinur's paper), we can first reduce the CSP instance $\mathcal{G}$ produced by gap amplification to a *Label Cover* instance. Label Cover is a special kind of CSP which has arity 2, and whose underlying relations are functions (so the value of one of the variables in each constraint is determined by the value taken by the other variable in that constraint). Conveniently for us, we also observe that Dinur's gap amplification step in fact already produces a CSP with this Label Cover structure, allowing us to skip the reduction step.[2] We can thus focus on alphabet reduction when the CSP we are reducing from has the Label Cover structure, and is over a fixed, albeit large, alphabet. We then follow the influential Label Cover and Long Code framework, originally proposed in [5] and strengthened in [11] and since then applied in numerous works on inapproximability, to reduce the CSP obtained from gap amplification, now viewed as Label Cover, to a Boolean CSP. Finally, we reduce the Boolean CSP back to a Label Cover instance (see Section 4) that can be plugged in as input to the gap amplification step.

[1] While this might seem circular, as this is what the PCP reduction is trying to accomplish in the first place, the key is that this inner reduction can be highly inefficient (even triply exponential blow up is okay!), as it is applied to a constraint of constant size.

[2] Technically, the gap amplification step produces a version of Label Cover whose constraints are *rectangular* rather than functions, but this is a minor difference that can be easily accommodated in reductions from Label Cover.

Our main result is the following, which can be viewed as reproving a case of alphabet reduction from [8, 7].

▶ **Theorem 1.** *There is a polynomial time reduction from Label Cover with soundness* $1-\delta$ *to a fixed template CSP with soundness* $1-C\delta$ *for an absolute constant* $C>0$.

We analyze our reduction using Fourier analysis as pioneered by Håstad [11]. Usually, in this framework, we reduce from low soundness Label Cover to strong (and at times optimal) soundness of CSP. But here we start with a high soundness Label Cover, and we reduce to high soundness CSP.

We highlight a couple of differences from previous works that make our proof easier:

- We have the freedom to choose any CSP rather than trying to prove inapproximability of a CSP. We choose the following 4-ary predicate R in our reduction: $(u,v,w,x)\in R$ if and only if $u\neq v\vee w\neq x$. This predicate appears in [11] in the context of proving optimal hardness for NAE-4SAT.
- In [11] and [5], the objective is to prove optimal inapproximability, or at least to get good soundness. However, our present goal is to prove "just" a nontrivial soundness. (On the other hand, we also start with high soundness Label Cover.) This allows us to use a very convenient test distribution leading to a simple analysis.
- We remark that a similar statement as Theorem 1 can be also deduced using [5, Section 4.1.1]. We believe that the test presented in this paper is more direct since we benefit from ideas in [11].
- It is possible to perform alphabet reduction using the Hadamard code instead of the long code as described in [10, 12]; the latter [12], similarly to our proof, avoids explicit use of Assignment Testers.
- Long code tests correspond exactly to testing whether a function is a *polymorphism* of the corresponding CSP, and as such corresponds to gadget reductions in the algebraic approach to CSP (see e.g. [4]). The PCP theorem surpasses these algebraic (gadget) reductions; this is even more evident when extending the scope from CSPs to *promise constraint satisfaction problems* (PCSPs) as there are PCSPs that can be shown to be NP-hard by using PCP theorem via a natural reduction through Label Cover, but cannot be shown to be NP-hard using only algebraic reductions [3, 6]. In this sense, the present paper shows that this strength of the PCP theorem comes from the Gap Amplification step.

Alphabet Reduction is an essential step in both the original proof of the PCP theorem as well as Dinur's proof and deserves further attention. Our proof of alphabet reduction bypasses the concept of Assignment Testers and is more intuitive in our opinion as it is *nothing but a gadget reduction*. Our proof is elementary using only Parseval's identity from Fourier Analysis over the hypercube. Dinur's analysis used the Friedgut-Naor-Kalai theorem [9] about Boolean functions with most of the Fourier mass at level 1. We believe that this makes our proof more accessible to readers that are new to PCPs. We also hope that this material might be useful in teaching the proof of PCP theorem as it relies only on techniques that any such basic course would cover anyway.

Outline

We start by formally defining CSP, Label Cover, and other preliminaries in Section 2. Then, in Section 3, we prove the main reduction from Label Cover to CSP. In Section 4, we show how the reduction can be used in the alphabet reduction step of Dinur's proof.

2 Preliminaries

2.1 CSPs and Label Cover

Loosely speaking, a CSP over some domain Σ is a decision problem which gets as input a set of variables and a set of constraints on the values of these variables. The goal is to decide whether there is an assignment of values from Σ to the variables such that all the constraints are satisfied. Usually, the shape of the constraints is somehow restricted. We first give a formal definition of the general CSP, and then two restrictions that we will use further.

▶ **Definition 2** (CSP). *The constraint satisfaction problem over an alphabet Σ takes as input a set of variables $V = \{x_1, \ldots, x_n\}$ and a finite set of* constraints *where each constraint is a pair $((x_{i_1}, \ldots, x_{i_k}), R)$ where k is a number (the arity of the constraint), $i_1, \ldots, i_k \in \{1, \ldots, n\}$, and $R \subseteq \Sigma^k$. The goal is to decide whether there exists an assignment $s\colon V \to \Sigma$ such that for each constraint $((x_{i_1}, \ldots, x_{i_k}), R)$ we have $(s(x_{i_1}), \ldots, s(x_{i_k})) \in R$.*

A fixed template CSP is a restriction of the general CSP that requires that the constraints only involve relations from a fixed list of relations over the given alphabet (a template). In the case the domain is Boolean, often negation of variables is allowed. Below, we formally define a Boolean fixed template CSP with a template consisting of a single relation allowing for negation of variables.

▶ **Definition 3** (Boolean fixed template CSP). *Let $\Sigma = \{0, 1\}$ be a Boolean domain, and fix a relation $R \subseteq \Sigma^k$. The constraint satisfaction problem associated with R, denoted by $\mathrm{CSP}(R)$, takes input as a set of variables V and a set of constraints of the form $((x_{i_1}, \ldots, x_{i_k}), R)$ where each x_i is either a variable x, or a negation of a variable (i.e., an expression $\neg x$). An assignment $s\colon V \to \Sigma$ is said to satisfy the constraint $((x_{i_1}, \ldots, x_{i_k}), R)$ if $(s(x_{i_1}), \ldots, s(x_{i_k})) \in R$ where $s(\neg x)$ is defined as $\neg s(x)$ for each $x \in V$.*

The restriction of CSP to binary constraints is traditionally referred to as Label Cover.

▶ **Definition 4** (Label Cover). *In an instance of Label Cover problem, we are given a tuple $(G = (V, E), \Sigma, \Pi)$ where*

1. *G is a graph on vertex set V.*
2. *Each vertex in G has to be assigned a label from Σ.*
3. *For each edge $e = (u, v) \in E$, there is a relation $\Pi_e \subseteq \Sigma \times \Sigma$. This relation corresponds to a constraint between u and v.*

A labeling of graph is a function $s\colon V \to \Sigma$ that assigns a label to each vertex of G. Such labeling is said to satisfy a constraint e if and only if $(s(u), s(v)) \in \Pi_e$.

For a Label Cover instance or in general for a CSP instance I, we use $\mathrm{size}(I)$ to denote $m + n$, where m is the number of constraints and n is the number of variables. We remark that Label Cover usually refers to the case when G is bipartite, and the constraint relations are functions. However, in this work, we find it convenient to consider a (closely related) version which has *rectangular relations*.

▶ **Definition 5** (Rectangular relation). *A relation $R \subseteq A \times B$ is said to be rectangular if there is a set C and functions $\pi\colon A \to C$ and $\sigma\colon B \to C$ such that $(a, b) \in R$ if and only if $\pi(a) = \sigma(b)$. Equivalently, R is rectangular if for all $a, a' \in A$ and $b, b' \in B$ such that $(a, b) \in R$, $(a, b') \in R$, and $(a', b) \in R$, we have $(a', b') \in R$.*

2.2 The long code

Loosely speaking, the *long code* is the longest (error-correcting) code over the Boolean alphabet that does not repeat bits. It is constructed as follows: the long code is a Boolean code of length 2^n which encodes a value $i \in [n]$ into a tuple p_i whose k-th coordinate, $k < 2^n$, is the i-th least significant digit of k in binary.

The long code be also described in another way: we view a Boolean tuple of length 2^n as an n-ary function $p\colon \{0,1\}^n \to \{0,1\}$ (each coordinate of the tuple encodes one value of p). In this perspective, the code words of the long code are functions p_i defined as $p_i(x_1,\dots,x_n) = x_i$. These functions are often called *dictators*.

We also remark that in the conjunction with the long-code, a rectangular constraint can be expressed as an identity. More precisely, given a rectangular relation $R \subseteq [n] \times [m]$, say $R = \{(i,j) : \pi(i) = \sigma(j)\}$ for some $\pi\colon [n] \to [k]$ and $\sigma\colon [m] \to [k]$, then the long codes p_i and p_j of values i, j satisfy

$$p_i(x_{\pi(1)},\dots,x_{\pi(n)}) = p_j(x_{\sigma(1)},\dots,x_{\sigma(m)})$$

for all $x_1,\dots,x_k \in \{0,1\}$ if and only if $(i,j) \in R$. This is a key property of rectangular relations that is used implicitly in our proof.

2.3 Boolean Fourier analysis

As usual in Boolean Fourier analysis, we treat TRUE as -1 and FALSE as $+1$. In particular, in this notation, "negation" is expressed as $\neg x = -x$, "xor" $x \oplus y$ is expressed as $x \oplus y = xy$, and "or" is the expressed by the following function:

$$x \vee y = \begin{cases} -1 & \text{if } x = -1 \text{ or } y = -1, \text{ and} \\ 1 & \text{otherwise.} \end{cases}$$

Throughout the paper, we will use all the same symbols to denote the coordinatewise (or bitwise) application of these functions to tuples, e.g. $(x_1,x_2) \oplus (y_1,y_2) = (x_1y_1, x_2y_2)$.

We define an inner product space on functions from $\{-1,+1\}^n \to \mathbb{R}$ as $\langle f,g\rangle = \mathbb{E}_x[f(x)g(x)]$. For a set $\alpha \subseteq [n]$, let

$$\chi_\alpha(x_1,\dots,x_n) = \prod_{i\in\alpha} x_i.$$

The set of such functions form an orthonormal basis for all functions from $\{-1,+1\}^n$ to $\mathbb{R}$ in the above defined inner product space. Moreover, if $\alpha \neq \emptyset$, then $\mathbb{E}_x[\chi_\alpha(x)] = 0$.

▶ **Definition 6** (Fourier expansion). *Given a function $f\colon \{-1,+1\}^n \to \mathbb{R}$, we can thus write it uniquely as a linear combination of this basis –*

$$f = \sum_{\alpha\subseteq[n]} \hat{f}(\alpha)\chi_\alpha$$

The real quantities $\hat{f}(\alpha)$ are called the Fourier coefficients *of f. We abuse the notation $\hat{f}(i)$ to denote $\hat{f}(\{i\})$.*

The following simple but crucial identity follows from the definitions and is all that we will need in our analysis.

▶ **Theorem 7** (Parseval's Identity). *For each Boolean valued function f, i.e., $f\colon \{-1,+1\}^n \to \{-1,+1\}$,*

$$\sum_{\alpha\subseteq[n]} \hat{f}(\alpha)^2 = 1.$$

2.3.1 Connection to the long code

We remark, that the function $\chi_{\{i\}}$ corresponds to a valid long code: the function p_i encoding the value i. Also observe that there is a connection between the natural distance defined by the inner product $\langle f, g\rangle$ on Boolean functions and *relative Hamming distance* of f and g: This is thanks to the fact that if $x, y \in \{-1, 1\}$ then $x = y$ if and only if $xy = 1$, and consequently, the relative Hamming distance of f to the long code word $p_i = \chi_{\{i\}}$ can be expressed as $\big(1 - \hat{f}(i)\big)/2$. This means that the closest valid long code to a function f is the p_i for which $\hat{f}(i)$ is maximal.

These ideas are manifested in the common strategy in rounding a Boolean function f to a long code: First make sure that coefficients $\hat{f}(\alpha)$ for large sets α are small enough, then decode to a value i that belongs to a small-enough (ideally 1-element) set α with a large-enough $\hat{f}(\alpha)$.

3 Label Cover to CSP

This section describes our gadget reduction from Label Cover to CSP(R) where R is the 4-ary relation over Boolean domain defined as

$$R = \{(x, x', z, z') \mid x \neq x' \vee z \neq z'\}.$$

▶ **Theorem 8.** *There exists absolute constant C such that given a Label Cover instance (not necessarily bipartite) $G = (V, E, \Sigma, \Pi)$ with rectangular constraints, there is a reduction from G that outputs an instance I of* CSP(R) *such that* size(I) $= O($size($G)$) *and*

- *If G is satisfiable, then I is satisfiable as well.*
- *If no labeling can satisfy $1 - \delta$ fraction of constraints of G, then no assignment can satisfy $1 - C\delta$ fraction of constraints in I for all δ.*

Since the domain of CSP(R) is Boolean, the above reduces from an alphabet Σ of arbitrary size to the alphabet of size 2. We note that the constant in $O($size$(G))$ above hides exponential dependency on $|\Sigma|$.

We describe the reduction as a probabilistic checker of a solution to G encoded using a long code, i.e., the proof contains for each $u \in V$ a word $f_u \colon \{-1, 1\}^{|\Sigma|} \to \{-1, 1\}$. In other words, we design the test in such a way that if $s \colon V \to \Sigma$ is a solution to G, then the assignment $f_u \mapsto p_{s(u)}$ passes the test. This will then immediately give the completeness of the reduction. The test is as follows: Sample an edge $e = (u, v)$ from E uniformly at random, and then with equal probability do one of the following

1. run a long code test inside f_u;
2. run a long code test inside f_v;
3. run a constraint test between f_u and f_v.

We describe the long code test and the constraint test below. Both query the respective tables of f_u and f_v at some 4 bits that are generated by a certain randomized algorithm, and then check whether these 4 Boolean values satisfy the predicate R defined above.

This checker can be viewed as a gadget reduction in the following sense: We replace each vertex $u \in V$ with $2^{|\Sigma|}$ Boolean variables labeled by $f_u(x)$ for $x \in \{-1, 1\}^{|\Sigma|}$ (we see an assignment to such variables as a function $f_u \colon \{-1, 1\}^{|\Sigma|} \to \{-1, 1\}$), and each edge $e = (u, v)$ with a set of weighted 4-ary constraints on f_u and f_v, each involving the relation R and some 4 values of f_u and f_v (the result is therefore an instance of CSP(R)). These constraints depend only on the relation Π_e.

To simplify some notation, we assume $\Sigma = [n]$. We also assume that the tables for f_u's are *folded* so f_u is forced to satisfy $f(-x) = -f(x)$. This is a standard technique. Such a folding is ensured by including only one variable of each pair $x, -x$, and if the test queries f_u at the bit corresponding to some x that is not included, the bit $f(-x)$ is queried instead, and the value is negated. As a consequence of this folding, we have to allow for negation of variables in CSP(R). An important and useful consequence of this is that all "even" Fourier coefficients of f vanish, i.e., $\hat{f}(\beta) = 0$ for all β such that $|\beta|$ is even. We remark that folding can be avoided in the construction of the gadget. Nevertheless, it considerably simplifies the calculations below. Further, for calculations, it is useful to view R as a predicate $\rho\colon \{\pm 1\}^4 \to \{0,1\}$ defined as

$$\rho(x,x',z,z') = 1 - \frac{(xx'+1)(zz'+1)}{4}.$$

It is easy to check that $\rho(x,x',z,z') = 1$ if and only if $(x,x',z,z') \in R$.

Let us now describe the two probabilistic checkers.

3.1 Long code test

The long code test has on input a table of a function f ($= f_u$ or f_v), and it is supposed to check whether this function is a code word of the long code, i.e., there is i such that $f = p_i$. We design the test so that all these words pass with probability 1. Since we are only using the predicate R, this further limits possible checks. In fact, we include all checks of the form $f(x_1,x_2,x_3,x_4) \in R$ that are passed by all dictators.[3]

Long code test. Given $f\colon \{-1,1\}^n \to \{-1,1\}$ to test against being a long code word. Choose $x,y,z,\mu \in \{-1,1\}^n$ uniformly at random. Test whether

$$(f(x), f(x \oplus (\mu \vee y)), f(z), f(z \oplus (\mu \vee \neg y))) \in R. \tag{1}$$

Note that for all $x,y,z,\mu \in \{-1,1\}$, $(x, x \oplus (\mu \vee y), z, z \oplus (\mu \vee \neg y)) \in R$. This implies that any dictator function passes the test with probability 1, and therefore provides the completeness of the test. We also note that this test can give some false positives, e.g. the function $-p_i\colon x \mapsto -p_i(x)$ passes the test with probability 1, but is not a long code word. It is in fact a negation of the word p_i. It can be checked that all functions that pass are either long code words, or their negations. In the decoding, we simply decode the above function f to i.

The following lemma bounds the probability that the test accepts in the means of the Fourier coefficients. We remark, that since we want to ensure that f is as close to a valid long code as possible, the probability should decrease as the coefficients $\hat{f}(\alpha)$ for $\alpha \neq \{i\}$ increase. Indeed, the lemma states that this is the case.

▶ **Lemma 9.** *Assuming that f is folded, the probability the long code test accepts is at most*

$$1 - \frac{3}{16} \sum_{|\alpha|>1} \hat{f}(\alpha)^2.$$

[3] Any function that passes any such test with probability 1 is called a *polymorphism of R*, see also [4].

Proof. Assume $f(x) = \sum_\alpha \hat{f}(\alpha)\chi_\alpha(x)$. The probability that the test accepts is

$$\begin{aligned}
&\mathbb{E}_{x,y,z,\mu}\, \rho(f(x),f(x \oplus (\mu \vee y)), f(z), f(z \oplus (\mu \vee \neg y))) \qquad (2)\\
&\quad = \mathbb{E}_{x,y,z,\mu}\left[1 - \frac{\big(f(x)f(x \oplus (\mu \vee y) + 1\big)\big(f(z)f(z \oplus (\mu \vee \neg y)) + 1\big)}{4}\right]\\
&\quad = \frac{3}{4} - \frac{1}{4}\,\mathbb{E}_{x,y,\mu}\, f(x)f(x \oplus (\mu \vee y)) - \frac{1}{4}\,\mathbb{E}_{y,z,\mu}\, f(z)f(z \oplus (\mu \vee \neg y))\\
&\qquad - \frac{1}{4}\,\mathbb{E}_{x,y,z,\mu}\, f(x)f(x \oplus (\mu \vee y))f(z)f(z \oplus (\mu \vee \neg y))
\end{aligned}$$

We further simplify this expression one term at a time.

$$\begin{aligned}
\mathbb{E}_{x,y,\mu}\, f(x)f(x \oplus (\mu \vee y)) &= \mathbb{E}_{x,y,\mu} \sum_{\alpha,\beta} \hat{f}(\alpha)\hat{f}(\beta)\chi_\alpha(x)\chi_\beta(x \oplus (\mu \vee y)) \qquad (3)\\
&= \sum_{\alpha,\beta} \hat{f}(\alpha)\hat{f}(\beta)\,\mathbb{E}_x[\chi_\alpha(x)\chi_\beta(x)]\,\mathbb{E}_{y,\mu}[\chi_\beta(\mu \vee y)]\\
&= \sum_{\alpha} \hat{f}(\alpha)^2\,\mathbb{E}_{y,\mu}[\chi_\alpha(y \vee \mu)] = \sum_{\alpha} \hat{f}(\alpha)^2(-1/2)^{|\alpha|}.
\end{aligned}$$

The third equality follows since χ_α and χ_β are orthogonal if $\alpha \neq \beta$. The last equality follows from the fact that $\mathbb{E}_{y,\mu}[y \vee \mu] = (-1) \cdot 3/4 + 1 \cdot 1/4 = -1/2$ and coordinates are chosen independently. Similarly, we get that

$$\mathbb{E}_{y,z,\mu}\, f(z)f(z \oplus (\mu \vee \neg y)) = \sum_{\alpha} \hat{f}(\alpha)^2(-1/2)^{|\alpha|}. \qquad (4)$$

Moving to the next term, we get

$$\begin{aligned}
&\mathbb{E}_{x,y,z,\mu}\, f(x)f(x \oplus (\mu \vee y))f(z)f(z \oplus (\mu \vee \neg y))\\
&\quad = \sum_{\alpha,\beta} \hat{f}(\alpha)^2\hat{f}(\beta)^2\,\mathbb{E}_{y,\mu}\,\chi_\alpha(\mu \vee y)\chi_\beta(\mu \vee \neg y) = \sum_{\alpha \cap \beta = \emptyset} \hat{f}(\alpha)^2\hat{f}(\beta)^2(-1/2)^{|\alpha \cup \beta|}. \qquad (5)
\end{aligned}$$

The last equality follows since $\mathbb{E}_{y,\mu}[(\mu \vee y) \oplus (\mu \vee \neg y)] = \mathbb{E}_{y,\mu}[\neg\mu] = 0$ and $\mathbb{E}_{y,\mu}[\mu \vee y] = \mathbb{E}_{y,\mu}[\mu \vee \neg y] = -1/2$. The overall acceptance probability is then

$$\begin{aligned}
&\frac{3}{4} - \frac{1}{2}\sum_{\alpha} \hat{f}(\alpha)^2(-1/2)^{|\alpha|} - \frac{1}{4}\sum_{\alpha \cap \beta = \emptyset} \hat{f}(\alpha)^2\hat{f}(\beta)^2(-1/2)^{|\alpha \cup \beta|}\\
&\qquad = 1 - \frac{1}{2}\sum_{\alpha} \hat{f}(\alpha)^2\big((-1/2)^{|\alpha|} + 1/2\big) - \frac{1}{4}\sum_{\alpha \cap \beta = \emptyset} \hat{f}(\alpha)^2\hat{f}(\beta)^2(-1/2)^{|\alpha \cup \beta|} \qquad (6)
\end{aligned}$$

where for the last equality, we used Parseval's identity. Further, we assumed that f is folded, and therefore $\hat{f}(\alpha) = 0$ for all α such that $|\alpha|$ is even. Restricting the sums to α and β with odd cardinality, and using that for such disjoint α and β, $|\alpha \cup \beta|$ is even, the last expression of (6) can be bounded from above by

$$1 - \frac{1}{2}\sum_{|\alpha|>1} \hat{f}(\alpha)^2\,(3/8) - \frac{1}{4}\sum_{\alpha \cap \beta = \emptyset} \hat{f}(\alpha)^2\hat{f}(\beta)^2(1/2)^{|\alpha \cup \beta|} \leq 1 - \frac{3}{16}\sum_{|\alpha|>1} \hat{f}(\alpha)^2 \qquad (7)$$

which concludes the proof. ◀

3.2 Constraint test

The constraint test has on input tables for functions f and g corresponding to some u and v such that $(u, v) \in E$, and it is supposed to test (assuming f and g are correct long code words) whether the values these functions encode satisfy the constraint given by a rectangular relation Π_e. We construct the test in a similar way as the long code test: We test functions f and g in 4 bits in such a way that long code words encoding satisfying values pass. In contrast with the long code test, we do not include all such tests, but only a selection; in particular, we include only tests that query two values from each function.

We assume that the constraint relation Π_e is given by $\pi, \sigma \colon [n] \to [m]$ such that $(i, j) \in \Pi_e$ if and only if $\pi(i) = \sigma(j)$, and we denote by y^π the vector in $\{-1, 1\}^n$ such that $y^\pi(i) = y(\pi(i))$.

Constraint test. Given $f, g \colon \{-1, 1\}^n \to \{-1, 1\}$ to test against satisfying a constraint Π_e given by $(i, j) \in \Pi_e$ if and only if $\pi(i) = \sigma(j)$ for fixed $\pi, \sigma \colon [n] \to [m]$. Choose $x, z \in \{-1, 1\}^n$ and $y \in \{-1, 1\}^m$ uniformly at random, and test whether

$$(f(x), f(x \oplus y^\pi), g(z), g(z \oplus (\neg y)^\sigma)) \in R. \tag{8}$$

Note that if both f and g are dictators, say $f = p_i$ and $g = p_j$, such that $\pi(i) = \sigma(j) = k$ then the above test accepts with probability 1. Indeed, the tuple gets evaluated to

$$(x(i), (x \oplus y^\pi)(i), z(j), (z \oplus (\neg y)^\sigma)(j)) = (x(i), x(i) \oplus y(k), z(j), z(j) \oplus \neg y(k))$$

which is in R for all x, y and z. This provides the completeness of the test.

In the analysis below, we will use the following notation.

▶ **Definition 10.** *Let $\alpha \subseteq [n]$ and $\pi \colon [n] \to [m]$, we denote by $\pi[\alpha]$ the subset of $[m]$ defined by $\pi[\alpha] = \{k : |\pi^{-1}(k) \cap \alpha| \text{ is odd}\}$.*

The goal of the constraint check is to ensure that functions f, g which are far from valid long codes that encode values satisfying the constraint pass with low probability. Unfortunately, the test gives a lot of false positives: it accepts any pair of functions χ_α and χ_β such that $\pi[\alpha] = \sigma[\beta]$ with probability 1.[4] This is nevertheless good-enough since the long code test provides that relevant α and β contain only one element, and $\pi[\{i\}] = \sigma[\{j\}]$ if and only if $\pi(i) = \sigma(j)$.

Naturally, the pairs of α and β with $\pi[\alpha] = \sigma[\beta]$ will appear in the analysis below. A useful fact that will simplify the computation below is that $\prod_{i \in \alpha} x_{\pi(i)} = \prod_{i \in \pi[\alpha]} x_i$, for all $x_1, \ldots, x_m \in \{-1, 1\}$, which implies that

$$\chi_\alpha(x^\pi) = \chi_{\pi[\alpha]}(x).$$

▶ **Lemma 11.** *Given that both f and g are folded, the probability that the consistency test accepts is at most*

$$1 - \frac{1}{4} \sum_{i,j : \pi(i) \neq \sigma(j)} \hat{f}(i)^2 \hat{g}(j)^2.$$

[4] We note that $\pi[\alpha] = \sigma[\beta]$ is equivalent to $\chi_\alpha(x_{\pi(1)}, \ldots, x_{\pi(n)}) = \chi_\beta(x_{\sigma(1)}, \ldots, x_{\sigma(n)})$ for all $x \in \{-1, 1\}^m$.

Proof. We can compute the acceptance probability in the same way as before, i.e., as

$$\frac{3}{4} - \frac{1}{4}\,\mathbb{E}_{x,y}[f(x)f(x \oplus y^\pi)] - \frac{1}{4}\,\mathbb{E}_{z,y}[g(z)g(z \oplus (\neg y)^\sigma)] \\ - \frac{1}{4}\,\mathbb{E}_{x,y,z}[f(x)f(x \oplus y^\pi)g(z)g(z \oplus (\neg y)^\sigma)] \quad (9)$$

We have

$$\mathbb{E}_{x,y}[f(x)f(x \oplus y^\pi)] = \sum_\alpha \hat{f}(\alpha)^2\,\mathbb{E}_y[\chi_\alpha(y^\pi)] = \sum_\alpha \hat{f}(\alpha)^2\,\mathbb{E}_y[\chi_{\pi[\alpha]}(y)] = 0 \quad (10)$$

where the last equality holds since $|\alpha|$ is odd, and consequently $\pi[\alpha] \neq \emptyset$. Similarly, $\mathbb{E}_{x,z}[g(z)g(z \oplus (\neg y)^\sigma)]$ vanishes. Thus the probability that the test accepts is

$$\begin{aligned} &\frac{3}{4} - \frac{1}{4}\,\mathbb{E}_{x,y,z}\, f(x)f(x \oplus y^\pi)g(z)g(z \oplus (\neg y)^\sigma) \quad (11)\\ &\qquad = \frac{3}{4} - \frac{1}{4}\sum_{\alpha,\beta} \hat{f}(\alpha)^2\hat{g}(\beta)^2\,\mathbb{E}_y[\chi_\alpha(y^\pi)\chi_\beta(-y^\sigma)] \\ &\qquad = \frac{3}{4} + \frac{1}{4}\sum_{\alpha,\beta} \hat{f}(\alpha)^2\hat{g}(\beta)^2\,\mathbb{E}_y[\chi_\alpha(y^\pi)\chi_\beta(y^\sigma)] \\ &\qquad = \frac{3}{4} + \frac{1}{4}\sum_{\alpha,\beta} \hat{f}(\alpha)^2\hat{g}(\beta)^2\,\mathbb{E}_y[\chi_{\pi[\alpha]}(y)\chi_{\sigma[\beta]}(y)] \\ &\qquad = \frac{3}{4} + \frac{1}{4}\sum_{\alpha,\beta:\pi[\alpha]=\sigma[\beta]} \hat{f}(\alpha)^2\hat{g}(\beta)^2 \\ &\qquad = 1 - \frac{1}{4}\sum_{\alpha,\beta:\pi[\alpha]\neq\sigma[\beta]} \hat{f}(\alpha)^2\hat{g}(\beta)^2 \end{aligned}$$

where the second equality follows from $|\beta|$ being odd, and the last equality follows from the Parseval's identity. Since $\pi(i) = \sigma(j)$ implies that $\pi[\{i\}] = \sigma[\{j\}]$, the claim follows. ◀

3.3 The full test

Putting the analysis of the two tests together we get the following.

▶ **Lemma 12.** *Given that both f and g are folded, the probability that the joint test accepts is at most*

$$1 - \frac{1}{16}\Big(\sum_{|\alpha|>1} f(\alpha)^2 + \sum_{|\beta|>1} g(\beta)^2 + \sum_{i,j:\pi(i)\neq\sigma(j)} \hat{f}(i)^2\hat{g}(j)^2\Big)$$

Proof. Follows directly from Lemmas 9 and 11. ◀

Finally, we are ready to prove the main theorem of this section.

Proof of Theorem 8. The completeness follows in a straightforward way from the two comments after the description of the tests. We prove the soundness. Suppose that the test passes with probability $1 - \delta$. We will show that this implies that there is an assignment to the Label Cover instance that satisfies $(1 - 16\delta)$-fraction of constraints.

Our decoding is as follows: for a node $v \in V$, decode v to $i \in \Sigma$ with probability proportional to $\hat{f}_v(i)^2$. Intuitively, we decode to the value i with higher probability if f is closer to the code word $p_i = \chi_{\{i\}}$ or its negative $-p_i$ (see also Section 2.3.1). We will show that in expectation, this decoding satisfies at least $1 - 16\delta$ fraction of constraints, which proves that there exists a labeling that satisfies at least $1 - 16\delta$ fraction of constraints.

Let $1-\delta_e$ denote the probability that the test passes when we pick edge e. As test passes with probability $1-\delta$, we know that $\mathbb{E}_e[\delta_e] = \delta$. Suppose that we pick $e = (u, v)$ with f and g being the functions corresponding to u and v respectively. From the above lemma, we have that

$$1 - \delta_e \leq 1 - \frac{1}{16}\Big(\sum_{|\alpha|>1} \hat{f}(\alpha)^2 + \sum_{|\beta|>1} \hat{g}(\beta)^2 + \sum_{i,j:\pi(i)\neq\sigma(j)} \hat{f}(i)^2\hat{g}(j)^2\Big), \tag{12}$$

and therefore,

$$16\delta_e \geq \sum_{|\alpha|>1} \hat{f}(\alpha)^2 + \sum_{|\beta|>1} \hat{g}(\beta)^2 + \sum_{i,j:\pi(i)\neq\sigma(j)} \hat{f}(i)^2\hat{g}(j)^2. \tag{13}$$

The probability that our decoding satisfies edge e of Label Cover is at least

$$\begin{aligned}
\sum_{i,j:\pi(i)=\sigma(j)} \hat{f}(i)^2\hat{g}(j)^2 &= 1 - \sum_{\substack{\alpha,\beta \\ |\alpha|>1, \text{ or } |\beta|>1, \text{ or } \alpha=\{i\} \text{ and } \beta=\{j\} \text{ and } \pi(i)\neq\sigma(j)}} \hat{f}(\alpha)^2\hat{g}(\beta)^2 \qquad (14)\\
&\geq 1 - \sum_{\substack{\alpha,\beta \\ |\alpha|>1}} \hat{f}(\alpha)^2\hat{g}(\beta)^2 - \sum_{\substack{\alpha,\beta \\ |\beta|>1}} \hat{f}(\alpha)^2\hat{g}(\beta)^2 - \sum_{\substack{i,j \\ \pi(i)\neq\sigma(j)}} \hat{f}(i)^2\hat{g}(j)^2\\
&= 1 - \sum_{|\alpha|>1} \hat{f}(\alpha)^2 - \sum_{|\alpha|>1} \hat{g}(\alpha)^2 - \sum_{i,j:\pi(i)\neq\sigma(j)} \hat{f}(i)^2\hat{g}(j)^2\\
&\geq 1 - 16\delta_\epsilon
\end{aligned}$$

where the first equality follows from Perseval's identity. Thus, the expected number of constraints satisfied by the labeling is at least $\mathbb{E}_e[1 - 16\delta_e] = 1 - 16\delta$ which proves the required claim with $C = 1/16$. ◀

Theorem 1 is stated without the assumption that the constraints are rectangular. This slightly more general version follows from Theorem 8 by a standard reduction which we describe below, in the proof of Theorem 13.

4 CSP to Label Cover

In this section, we recall the basic structure of Dinur's proof of PCP Theorem, and show how the previous reduction can be used in the alphabet reduction step of Dinur's proof. The resulting proof requires a gap amplification step for which we refer to Dinur's paper [7].

We first prove that the previous reduction can be combined with standard reductions to get back Label Cover from the CSP.

▶ **Theorem 13** (Alphabet reduction). *Given a Label Cover instance $G = (V, E, \Pi, \Sigma_0)$ with rectangular constraints, there is a polynomial time reduction that outputs another Label Cover instance with rectangular constraints $G' = (V', E', \Pi', \Sigma)$ with alphabet size Σ such that $|\Sigma|$ is an absolute constant, $size(G') = O(size(G))$ and*

- *If G is satisfiable, then G' is satisfiable as well.*
- *If every labeling violates δ fraction of constraints of G, then every labeling violates $C\delta$ fraction of constraints in G' for an absolute constant C.*

Proof. We first convert the Label Cover instance G to a CSP instance I as in Theorem 8. The CSP instance can be converted to bipartite Label Cover using standard clause-variable Label-coverization technique. We include the proof here for the sake of completeness. We

have n vertices $x_1, x_2, \ldots, x_n$ corresponding to the variables of I on the left L, and m vertices corresponding to constraints $C_1, C_2, \ldots, C_m$ of I on the right R. The label set is binary on the left, and satisfying assignments (at most 16) on the right corresponding to the possible assignments to four variables in the constraint. We add an edge between $u \in L$ and $v \in R$ if $x_u \in C_v$. The constraint on this edge enforces that the assignment to x_u is consistent with the assignment C_v assigns to x_u.

If there is a satisfying labeling to G, there is a satisfying assignment to I. Using this, we can assign the variables on the left the satisfying assignment, and the corresponding assignment to tuples for the vertices of constraints on the right, and thus get a satisfying assignment to G'. Suppose that every labeling violates at least δ fraction of constraints of G. From Theorem 8, every assignment violates at least $C\delta$ fraction of constraints in I. Suppose there is a labeling to G' that satisfies δ' fraction of constraints. Consider the assignment obtained by this labeling on the left. This assignment violates at least $C\delta m$ number of constraints in I. Note that this should violate at least $C\delta m$ constraints in G' and thus $\delta' \geq C'\delta$ for an absolute constant C'. The constraints are in fact projections, and thus are rectangular too. ◀

In order for us to use this as Composition step in the PCP of Dinur, we need the final observation that the output of Gap Amplification applied to a CSP with rectangular constraints results in a Label Cover with rectangular constraints. Dinur achieves gap amplification by "graph powering" which is described more formally below.

▶ **Definition 14** (Constraint Graph Powering). *Given a d-regular Label Cover (a.k.a. Constraint graph) $G = (V, E, \Sigma, \Pi)$, we obtain t-th power of it $G^t = (V, E', \Sigma', \Pi')$ as follows:*

- *Vertices. The vertices in G^t are the same as vertices in G.*
- *Edges. u and v are connected by k parallel edges in E' if there are exactly k paths of length t between u and v in G.*
- *Alphabet. The alphabet of G^t is $\Sigma^{d^{\lceil t/2 \rceil}}$. A value $a \in \Sigma^{d^{\lceil t/2 \rceil}}$ is interpreted as assigning values $a \colon \Gamma(u) \to \Sigma$ to $d^{\lceil t/2 \rceil}$ elements from Σ. This value is treated as u's opinion on $\Gamma(u)$, the set of all vertices within $\lceil t/2 \rceil$ distance from u.*
- *Constraints. An edge $(u, v) \in E'$ is satisfied by $a, b \in \Sigma^{d^{\lceil t/2 \rceil}}$ if and only if the following holds: there is an assignment $\sigma \colon \Gamma(u) \cup \Gamma(v) \to \Sigma$ that satisfies every constraint $c(e)$ where $e \in E \cap (\Gamma(u) \times \Gamma(v))$, and such that*

$$\forall u' \in \Gamma(u), \sigma(u) = a_{u'}; \forall v' \in \Gamma(v), \sigma(v) = b_{v'}$$

where $a_{u'}$ (and respectively $b_{v'}$) is the value a (and resp. b) assigned to u' (and resp. v').

The output G^t is also a binary CSP, and hence can be viewed as a Label Cover. We claim that if every constraint of G is rectangular, then every constraint of G^t is rectangular as well. Let $e = (u, v)$ be an edge in E' with constraint relation as R_e. Suppose $(a, b), (a', b), (a, b') \in R_e$. This implies that for all $(u', v') \in E \cap (\Gamma(u) \times \Gamma(v))$ with constraint relation c_e,

$$(a_{u'}, b_{v'}), (a'_{u'}, b_{v'}), (a_{u'}, b'_{v'}) \in R_{c_e}.$$

Since R_{c_e} is rectangular, $(a'_{u'}, b'_{v'}) \in R_{c_e}$ as well. As this holds for all such u' and v', $(a', b') \in R_e$, thus proving that R_e is a rectangular relation.

Combined with the preprocessing step, the gap amplification theorem of Dinur can be rewritten as follows.

▶ **Theorem 15** (Gap amplification). *Fix a parameter t. Given a Label Cover $G = (V, E, \Pi, \Sigma)$ where Σ is an absolute constant, there is a polynomial time reduction to output a rectangular Label Cover instance $G' = (V', E', \Pi', \Sigma')$ with the alphabet size $|\Sigma'| = c(|\Sigma|, t)$ such that*

- *If G is satisfiable, G' is satisfiable as well.*
- *If every labeling violates at least δ fraction of the constraints of G, then every labeling violates at least $\Omega(\delta\sqrt{t})$ fraction of the constraints of G'.*

Choosing t large enough constant and iterating Theorem 13 and Theorem 15 $\log(m)$ times proves the PCP theorem.

References

1. Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and the hardness of approximation problems. *Journal of the ACM*, 45(3):501–555, 1998. doi:10.1145/278298.278306.
2. Sanjeev Arora and Shmuel Safra. Probabilistic checking of proofs: A new characterization of NP. *Journal of the ACM*, 45(1):70–122, 1998. doi:10.1145/273865.273901.
3. Per Austrin, Venkatesan Guruswami, and Johan Håstad. $(2 + \epsilon)$-Sat is NP-hard. *SIAM J. Comput.*, 46(5):1554–1573, 2017. doi:10.1137/15M1006507.
4. Libor Barto, Andrei Krokhin, and Ross Willard. Polymorphisms, and how to use them. In Andrei Krokhin and Stanislav Živný, editors, *The Constraint Satisfaction Problem: Complexity and Approximability*, volume 7 of *Dagstuhl Follow-Ups*, pages 1–44. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 2017. doi:10.4230/DFU.Vol7.15301.1.
5. Mihir Bellare, Oded Goldreich, and Madhu Sudan. Free bits, PCPs, and nonapproximability-towards tight results. *SIAM J. Comput.*, 27(3):804–915, 1998. doi:10.1137/S0097539796302531.
6. Jakub Bulín, Andrei Krokhin, and Jakub Opršal. Algebraic approach to promise constraint satisfaction. In *Proceedings of the 51st Annual ACM SIGACT Symposium on the Theory of Computing (STOC '19)*, New York, NY, USA, 2019. ACM. doi:10.1145/3313276.3316300.
7. Irit Dinur. The PCP theorem by gap amplification. *Journal of the ACM*, 54(3):12, 2007. doi:10.1145/1236457.1236459.
8. Irit Dinur and Omer Reingold. Assignment testers: Towards a combinatorial proof of the PCP theorem. *SIAM J. Comput.*, 36(4):975–1024, 2006.
9. Ehud Friedgut, Gil Kalai, and Assaf Naor. Boolean functions whose Fourier transform is concentrated on the first two levels. *Adv. in Applied Math.*, 29:427–437, 2002.
10. Venkatesan Guruswami and Ryan O'Donnell. The PCP theorem and hardness of approximation: Notes on lectures 7–9. https://courses.cs.washington.edu/courses/cse533/05au/, 2005.
11. Johan Håstad. Some optimal inapproximability results. *Journal of the ACM*, 48(4):798–859, 2001. doi:10.1145/502090.502098.
12. Jaikumar Radhakrishnan and Madhu Sudan. On Dinur's proof of the PCP theorem. *Bull. Amer. Math. Soc.*, 44:19–61, 2007.

A Derandomization of the gadget decoding

In this appendix, we provide a derandomization of the decoding used in Theorem 8. This requires only a little additional argument. The idea is, instead of decoding to i with probability $\hat{f}(i)^2$, to decode to the i with the largest $\hat{f}(i)^2$. We set i_f to be such i. In this light, the reduction can be analyzed by analyzing completeness and soundness of the gadget separately without considering the rest of the instance. The following lemma then shows that the gadget has perfect completeness and soundness 99% not depending on the parameters n and m (the alphabet sizes), π and σ.

▶ **Lemma 16.** *There is a gadget with inputs n, m, k, $\pi\colon [n] \to [k]$, and $\sigma\colon [m] \to [k]$ that produces an instance of* $\mathrm{CSP}(R)$ *with variables $f(a_1, \ldots, a_n)$ and $g(a_1, \ldots, a_m)$ such that*
1. *if $\pi(i) = \sigma(j)$ then p_i and p_j satisfy all the constraints, and*
2. *if at least 99% of the constraints are satisfied, then $\pi(i_f) = \sigma(i_g)$.*

Proof. First, we bound the probability that the test accepts. For the long code test, starting with the first expression in (7), we obtain the following bound.

$$1 - \frac{1}{2}\sum_{|\alpha|>1} \hat{f}(\alpha)^2\,(3/8) - \frac{1}{4}\sum_{\alpha\cap\beta=\emptyset} \hat{f}(\alpha)^2\hat{f}(\beta)^2(1/2)^{|\alpha\cup\beta|} \le 1 - \frac{3}{16}\sum_{|\alpha|>1}\hat{f}(\alpha)^2 - \frac{1}{16}\sum_{i\neq j}\hat{f}(i)^2\hat{f}(j)^2$$

For the consistency test, we use the bound from Lemma 11. Thus the overall probability that the whole test accepts is at most

$$1 - \frac{1}{16}\sum_{|\alpha|>1}\hat{f}(\alpha)^2 - \frac{1}{48}\sum_{i\neq j}\hat{f}(i)^2\hat{f}(j)^2 - \frac{1}{16}\sum_{|\alpha|>1}\hat{g}(\alpha)^2 - \frac{1}{48}\sum_{i\neq j}\hat{g}(i)^2\hat{g}(j)^2 - \frac{1}{12}\sum_{i,j:\pi(i)\neq\sigma(j)}\hat{f}(i)^2\hat{g}(j)^2.$$

Given that the acceptance probability is at least $99\% > 1 - 1/96$, we get that

$$\sum_{|\alpha|>1}\hat{f}(\alpha)^2 \le 1/6 \tag{15}$$

$$\sum_{i\neq j}\hat{f}(i)^2\hat{f}(j)^2 \le 1/2 \tag{16}$$

$$\sum_{|\alpha|>1}\hat{g}(\alpha)^2 \le 1/6 \tag{17}$$

$$\sum_{i\neq j}\hat{g}(i)^2\hat{g}(j)^2 \le 1/2 \tag{18}$$

$$\sum_{i,j:\pi(i)\neq\sigma(j)}\hat{f}(i)^2\hat{g}(j)^2 \le 1/8 \tag{19}$$

From Parseval's identity and (15), we get that $1 \ge \sum_i \hat{f}(i)^2 \ge 5/6$. Recall that i_f is such i that $\hat{f}(i)^2$ is maximal. Then using the above and (16), we obtain that

$$\hat{f}(i_f)^2 \ge \hat{f}(i_f)^2\sum_i \hat{f}(i)^2 \ge \sum_i \hat{f}(i)^4 = \sum_{i,j}\hat{f}(i)^2\hat{f}(j)^2 - \sum_{i\neq j}\hat{f}(i)^2\hat{f}(j)^2 \ge (5/6)^2 - 1/2 = 4/9. \tag{20}$$

Similarly, from (17) and (18), we get $\hat{g}(i_g)^2 \ge 4/9$. Finally, since $\hat{f}(i_f)^2\hat{g}(i_g)^2 \ge (4/9)^2 > 1/8$, we have that $\pi(i_f) = \sigma(i_g)$ otherwise (19) cannot be true. ◀

Theorem 8 can be also directly obtained from this lemma albeit with a worse constant than in the above proof: Let $C = 1\%$ and assume that $\delta < 1$. Given that the resulting CSP instance has an assignment fails no more than $C\delta$-fraction of the constraints, we derive that in at least $(1-\delta)$-fraction of the gadgets, no more than C-fraction of constraints are unsatisfied. Lemma 16 then shows that the assignment $s\colon u \mapsto i_{f_u}$ is an assignment of the Label Cover instance that satisfies all the constraints corresponding to these gadgets. This completes the proof.