NOTES ON PCPS AND UNIQUE GAMES

EDWARD KIM

Abstract. We expound on the basic theory of approximating NP-hard problems. Starting with the PCP Theorem, we introduce background for the Unique Games Conjecture (UGC) and the optimality of the Goemans-Williamson Algorithm for MAXCUT assuming the conjecture is true.

Contents

These notes borrow heavily from Arora and Barak's textbook [1], O'Donnell's textbook [13], and Prahladh Harsha's lecture notes on PCPs and Unique Games given at DIMACS and TIFR [6], [7]. Supplementary reading can be found in Luca Trevisan's exposition on the UGC [15] and Subhash Khot's survey [11].

1. Approximation Algorithms and CSPs

1.1. **Definitions and Examples.** The theory of approximating NP-hard problems roots itself in the following question: "Is it possible to efficiently approximate NP-complete problem to some arbitrary degree of accuracy?" Since Cook-Levin result demonstrated that the SAT decision problem is NP-complete, the question could be rephrased as "Can we find an efficient algorithhm for SAT which outputs an assignment which satisfies a $1-\delta$ fraction of the clauses for any constant $\delta > 0$?" To this end, let us first introduce a few motivating examples which will find utility in our forthcoming analysis.

Consider $\mathsf{Max3CNF}$, the problem of, given a 3CNF formula φ as input, outputting an assignment which maximizes the number of clauses satisfied in φ. Given any assignment x to the variables, say as an n-bit string while viewing $\varphi : \{0,1\}^n \to \{0,1\}$, there is some $0 \leq \rho_{\varphi,x} \leq 1$ representing the fraction of clauses satisfied in φ. Take $\mathsf{Opt}(\varphi)$ to be the maximum such value over all such assignments:

$$\mathsf{Opt}(\varphi) = \max_{x\in\{0,1\}^n} \rho_{\varphi,x}$$

An polynomial-time algorithm which solves $\mathsf{Max3CNF}$ is one which takes a 3CNF φ as input and outputs $\mathsf{Opt}(\varphi)$. Naturally, $\mathsf{Max3CNF}$ is NP-hard since its corresponding decision problem $\mathsf{3CNF}$ is NP-complete i.e φ is satisfiable iff $\mathsf{Opt}(\varphi) = 1$. However, we can ask if there exists an algorithm A which outputs an assignment which satisfies at least some $\beta \cdot \mathsf{Opt}(\varphi)$ fraction of the clauses of φ for some $\beta \leq 1$. To formalize this:

Definition 1.1. For $\beta \leq 1$, a polynomial-time algorithm A is deemed as a β-approximation algorithm for $\mathsf{Max3CNF}$ if $A(\varphi)$ outputs an assignment which satisifies at least $\beta \cdot \mathsf{Opt}(\varphi)$ fraction of φ's clauses for every 3CNF instance φ. More specifically, the algorithm A outputs some value such that:

$$\beta \cdot \mathsf{Opt}(\varphi) \leq A(\varphi) \leq \mathsf{Opt}(\varphi) \tag{1.1}$$

If $\mathsf{Opt}(\varphi) = 1$, then φ is said to be *satisfiable.*

A canonical example of a simple approximation algorithm for $\mathsf{Max3CNF}$ is the following scheme: for every variable in φ choose with uniform probability an assignment from $\{0,1\}$. The probability of any given clause being satisfied by such a random assignment is $\frac{7}{8}$. Thus

$$\mathbb{E}_{x\in\{0,1\}^n}[\#\text{ satisfied clauses of }\varphi(x)] = \frac{7m}{8}$$

where m is the number of satisfiable clauses of φ. This gives us a $\frac{7}{8}$-approximation algorithm for $\mathsf{Max3CNF}$. Here's another example:

Example 1.2. Let $\mathsf{MaxE3Lin}$ define the problem of the following form: let ξ be defined as a system of linear equations over the field $\mathbb{F}_2$ where every equation contains *exactly* 3 variables from n variables $x_1, \cdots, x_n$.

Find an assignment to the variables which maximizes the number of satisified linear equations in ξ. An example is the following system over variables x_1, x_2, x_3, x_4:

$$\begin{aligned} x_1 + x_2 + x_3 &= 0 \\ x_1 + x_2 + x_4 &= 1 \\ x_1 + x_3 + x_4 &= 1 \end{aligned}$$

The $\mathsf{E3Lin}$ represents a linear system of Exactly $\mathsf{3}$ variables. We can extend the notation treated above for $\mathsf{Max3CNF}$ to this situation. In other words, if $\rho_{\xi,x}$ is the fraction of satisfied linear constraints of ξ under assignment $x \in \mathbb{F}_2^n$.

$$\mathsf{Opt}(\xi) = \max_{x \in \{0,1\}^n} \rho_{\xi,x}$$

A comment regarding this problem. Firstly, if the given E3Lin instance has a guaranteed solution i.e $\mathsf{Opt}(\varphi) = 1$, then Gaussian Elimination will always output an assignment which satisfies all linear constaints in polynomial-time. It's more interesting to consider instances where no such solution satisfing all of the constraints exists over $\mathbb{F}_2^n$. In these cases, $\mathsf{Opt}(\xi) = 1 - \epsilon$ for some $\epsilon \leq 1$. This problem also has a fairly simple $\frac{1}{2}$-approximation algorithm: Set all $x_1 = \cdots = x_n = 0$ or $x_1 = \cdots = x_n = 1$ depending on which satisfies the most constraints. This scheme must satisfy at least $\frac{1}{2}$ of the constraints for any instance ξ.

Example 1.3. The problem MaxCut will be casted as follows: given an undirected graph $G = (V, E)$, find the largest cut of G. There is a straightforward LP formulation for this problem:

$$\max \sum_{(u,v) \in E} e_{u,v} \tag{1.2}$$

$$e_{u,v} \leq \min \begin{cases} x_1 + x_2 \\ 2 - (x_1 + x_2) \end{cases} \tag{1.3}$$

$$e_{u,v} \in [0,1] \tag{1.4}$$

$$x_v \in [0,1] \tag{1.5}$$

where the $e_{u,v}$ variables represent the edges $e \in E$ of the input graph and the x_v represent the vertex variables $v \in V$. This will actually be a relaxation from its respective integer program in which the variables can take the integer values:

$$e_{u,v} = \begin{cases} 1 \text{ if } e_{u,v} \in \mathcal{C} \\ 0 \text{ otherwise} \end{cases} \quad x_v = \begin{cases} 0 \text{ if } v \text{ is in partition } S \\ 1 \text{ otherwise} \end{cases} \tag{1.6}$$

where $\mathcal{C} \subseteq E$ is the edges constituting the cut and $(S, \bar{S})$ is the partition defining the cut. Naturally, a solution to the Integer program will also be a solution to the LP above, which implies that

$$\mathsf{Opt}_{LP} \geq \mathsf{MaxCut}(G)$$

There actually exists an Semi-definite Programming (SDP) relaxation for MaxCut called the *Goemans-Williamson* algorithm which will be treated in Section 6.2.

1.2. **Constraint Satisfaction Problems.** Generally speaking, it's useful to concretize these problems into a unified framework. The examples presented in the last section have some common interpretation shared between them: they could all be seen as manifestations of Constaint-Satisfaction Problem (CSP):

Definition 1.4. Let Ω be a finite set deemed as the *domain*. A constraint satisfaction problem (CSP) Ψ over domain Ω is a finite set of predicates $\psi : \Omega^r \to \{0,1\}$ where r would be the *arity* of predicate ψ. The predicates can be of different arities. The arity of the CSP Ψ would be the maximum of all the arities of the predicates in Ψ.

An *instance* $\mathcal{I}$ over CSP Ψ is a set of tuples (S, ψ) where if r is the arity of predicate ψ, $S = (v_1, \cdots, v_r)$ is some ordered tuple of variables taken from finite set V consisting of variables contained in CSP Ψ. These tuples are called the *constaints* of $\mathcal{I}$. In addition, we add the conditon that every variable show up in at least one constraint. Variables which do not appear in any of the constraints can simply be removed.

Now given some instance $\mathcal{I}$, an *assignment* $F : V \to \Omega$ is simply some map between the variables and the domain. We say that F satisfies a constaint (S, ψ) if $\psi(F(S)) = 1$. For tuple $S = (v_1, \cdots, v_r)$, we define $F(S) = (F(v_1), \cdots, F(v_r))$. Consider the fraction of constraints of satisfied by F in $\mathcal{I}$:

$$\mathsf{Val}_{\mathcal{I}}(F) = \mathbb{E}_{(S,\psi)\sim\mathcal{I}}[\psi(F(S))] \tag{1.7}$$

By taking the maximum fraction over all such assignments:

$$\mathsf{Opt}(\mathcal{I}) = \max_{F:V\to\Omega} \mathsf{Val}_{\mathcal{I}}(F) \tag{1.8}$$

Example 1.5.

- For $\mathsf{Max3CNF}$, the domain would be $\Omega = \{0,1\}$ and the CSP Ψ would be composed of the predicate $\vee_3 : \{0,1\}^3 \to \{0,1\}$ just taking the logical ORs of the three input variables. Any instance φ would be composed of $(S, \vee_3)$ where S would be the variables inputted into $\vee_3$. Hence, an assignment F would be an assignment into the variables found in 3CNF φ, showing that $\mathsf{Opt}(\varphi)$ aligns with the definition given in the last section.

- For $\mathsf{MaxE3Lin}$, the domain would be $\Omega = \mathbb{F}_2$ and Ψ consist of predicates of the form $(x_1, \cdots, x_3) = x_1 + x_2 + x_3$ and $(x_1, \cdots, x_3) = x_1 + x_2 + x_3 + 1$ representing both types of linear constraints found in an system of three-variable equations over $\mathbb{F}_2$. An instance ξ would consist of constraints (S, ψ) where S would be a three variable tuple containing the variables showing up in the linear constraint ψ.

- For MaxCut, the domain can be defined as $\Omega = \{-1,1\}$ with Ψ set to the simple predicate $\neq$: $\{-1,1\}^2 \to \{0,1\}$. This simply tests if the inputted values are not equal. The variable set V of an instance $\mathcal{I}$ would be indexed by the vertices contained in a graph $G = (V, E)$. There is one constraint tuple, $((v_i, v_j), \neq)$ for every edge $(v_i, v_j) \in E$. Thus, an assignment $F : V \to \{-1,1\}$ would encode a partition of V into a cut C with a constraint becoming satisified if the corresponding edge is contained in C.

- The CSP for $\mathsf{Max3Color}$, the maximization counterpart for the NP-complete decision problem, $\mathsf{3Color}$, is similarly defined to that of MaxCut except our domain would be $\Omega = \{0,1,2\}$. This signifies the three possible colors to color any vertex $v \in V$ in an input graph $G = (V, E)$.

As implied in the examples above, there is a generic method to formulate a maximization problem in respect to a given CSP Ψ:

Definition 1.6. For a given CSP Ψ, formulate $\mathsf{MaxCSP}(\Psi)$ as the problem: given an instance $\mathcal{I}$, output an assignment F which satisfies the largest number of constraints in $\mathcal{I}$.

1.3. **Gap Problems.** The NP-hardness theory frequently relies on Karp reductions from decision problems to decision problems. In light of this, we can tailor optimization problems to related promise problems in the form of so-called *gap problems*.

Definition 1.7. A *promise problem* is defined as a tuple $(\mathsf{YES}, \mathsf{NO})$ where $\mathsf{YES}, \mathsf{NO} \subseteq \Sigma^*$ in respect to some alphabet Σ. Furthermore, we require that $\mathsf{YES} \cap \mathsf{NO} = \emptyset$ but not necessarily that $\mathsf{YES} \cup \mathsf{NO} = \Sigma^*$.

Definition 1.8. Given a $\mathsf{MaxCSP}(\Psi)$ problem, we define $\mathsf{Gap}_{\alpha,\beta}\mathsf{MaxCSP}(\Psi)$ for $\alpha < \beta$ as the promise problem: given an instance $\mathcal{I}$:

$$\mathcal{I} \in \mathsf{YES} \iff \mathsf{Opt}(\mathcal{I}) \geq \beta \tag{1.9}$$

$$\mathcal{I} \in \mathsf{NO} \iff \mathsf{Opt}(\mathcal{I}) < \alpha \tag{1.10}$$

Furthermore, an algorithm A decides $\mathsf{MaxCSP}(\Psi)$ if for input instance $\mathcal{I}$ it accepts iff $\mathcal{I} \in \mathsf{Yes}$ and rejects iff $\mathcal{I} \in \mathsf{No}$. If $\mathcal{I}$ is in neither $\mathsf{Yes}, \mathsf{No}$, we do not care what the algorithm outputs.

In particular, we deem a $\mathsf{Gap}_{\alpha,\beta}\mathsf{MaxCSP}(\Psi)$ problem as NP-hard if for every language $L \in \mathsf{NP}$, there exists a polynomial-time reduction f taking x to CSP Ψ instances such that:

$$x \in L \implies \mathsf{Opt}(f(x)) \geq \beta$$
$$x \notin L \implies \mathsf{Opt}(f(x)) < \alpha$$

The NP-hardness of approximation algorithms reduces to that of gap problems as shown in the below observation:

Theorem 1.9. *Suppose* $\mathsf{Gap}_{\alpha,\beta}\mathsf{MaxCSP}(\Psi)$ *is* NP*-hard for CSP* Ψ*, then approximating* $\mathsf{MaxCSP}(\Psi)$ *to at least an* $\frac{\alpha}{\beta}$ *factor is* NP*-hard.*

Proof. Suppose there exists an algorithm A which can $\frac{\alpha}{\beta}$-approximate $\mathsf{MaxCSP}(\Psi)$. For an instance $\mathcal{I}$ such that $\mathsf{Opt}(\mathcal{I}) \geq \beta$:

$$A(\mathcal{I}) \geq \frac{\alpha}{\beta} \cdot \mathsf{Opt}(\mathcal{I}) = \frac{\alpha}{\beta} \cdot \beta = \alpha$$

Else if $\mathsf{Opt}(\mathcal{I}) < \alpha$

$$A(\mathcal{I}) \leq \mathsf{Opt}(\mathcal{I}) < \alpha$$

by Definition 1.1 adapted to $\mathsf{MaxCSP}(\Psi)$ instances. Hence, the algorithm can decide $\mathsf{Gap}_{\alpha,\beta}\mathsf{MaxCSP}(\Psi)$ by checking it's outputted value in respect to α. $\square$

This in particular demonstrates that showing the hardness of approximating a particular problem is equivalent to showing the hardness of its corresponding gap problem.

2. The PCP Theorem

2.1. **Intuitions.** This section will be centered around the seminal PCP Theorem [2], [3], which characterized NP in a framework considered unconventional at the time. PCPs, or Probabilistically Checkable Proofs, represent a twist on the idea of NP. Recall that NP roughly represents the languages which have verifiers which can check proofs of membership in polynomial time. PCPs represent an extension of this definition where the verifier can be *probabilistic* and is granted *random access* to the proof string π. If we allow the verfier to simply query π by outputting a index i, it has access to $\pi[i]$. Since we can express an index in $\log n$ bits, this in theory gives the verifier access to proof strings of exponential length. To formalize these notions, we begin with definitions:

Definition 2.1. Given a language L and $r, q : \mathbb{N} \to \mathbb{N}$, a *(r(n),q(n))*-$\mathsf{PCP}$*-verifier* for L consists of a polynomial-time algorithm V with the following properties:

- For input strings $x \in \{0,1\}^n$, $\pi \in \{0,1\}^{\leq N}$ for $N = q(n)2^{r(n)}$, V makes $r(n)$ coin flips and decides $q(n)$ queries addresses $i_1, \cdots, i_{q(n)}$ of the proof π. Based on these queries, it outputs 1 for "accept" or 0 for "reject".
- (Completeness) For $x \in L$, there exists some proof π such that $V(x, \pi, r) = 1$ for all random coin tosses r. In other words:

$$\mathbb{P}_r[V(x, \pi, r) = 1] = 1 \tag{2.1}$$

- (Soundness) For $x \notin L$, for all proofs π:

$$\mathbb{P}_r[V(x, \pi, r) = 1] \leq \frac{1}{2} \tag{2.2}$$

Define the class $\mathsf{PCP}(r(n), q(n))$ as the set of languages L which has a $(c \cdot r(n), d \cdot q(n))$-$\mathsf{PCP}$-verifier for some $c, d > 0$.

Remark. Sometimes the completeness criterion is too strong for our purposes (see the comments on $\mathsf{MaxE3Lin}$ problem). In these cases, we like to denote the class $\mathsf{PCP}_{\beta,\alpha}(r(n), q(n))$ as the languages L which have a $(r(n), q(n))$-NP-verifier such that the completeness and soundness criteria are amended as below:

- (Completeness) For $x \in L$, there exists some proof π such that $V(x, \pi, r) = 1$ for all random coin tosses r. In other words:

$$\mathbb{P}_r[V(x, \pi, r) = 1] \geq \beta \tag{2.3}$$

- (Soundness) For $x \notin L$, for all proofs π:

$$\mathbb{P}_r[V(x, \pi, r) = 1] \leq \alpha \tag{2.4}$$

Here, β is the *completeness parameter* while α is the *soundness parameter*. The class introduced in the original definition would thus be denoted as $\mathsf{PCP}_{1,\frac{1}{2}}(r(n), q(n))$. PCP verifiers whose completeness parameter is one ($\beta = 1$) is deemed as *perfectly complete*.

The PCP Theorem says that NP is *exactly* the class of PCPs which uses a *logarithmic* number of random bits and a *constant* number of queries.

Theorem 2.2. *(The* PCP *Theorem* [2], [3]*)*

$$\mathsf{NP} = \mathsf{PCP}_{1,\frac{1}{2}}(O(\log n), O(1)) \tag{2.5}$$

Actually, one direction of this theorem is not too difficult to see:

Proposition 2.3. For every constants $Q \in \mathbb{N}, c > 0$, $\mathsf{PCP}_{1,\frac{1}{2}}(c \cdot \log n, Q) \subseteq \mathsf{NP}$

Proof. Begin with the observation that $\mathsf{PCP}_{1,\frac{1}{2}}(r(n), q(n)) \subseteq \mathsf{NTIME}(q(n)2^{r(n)})$. This is justified by the view of an NTIME machine simulating the verifier by trying all possible coin tosses and queries to the input string x and proof string π. It can then count all of the accepting paths to determine the probability of acceptance. If $q = O(1)$ and $r = O(\log n)$, then the right side of the inclusion will be $\mathsf{NTIME}(2^{O(\log n)}) = \mathsf{NP}$. $\square$

Remark. The queries a PCP verifier makes could be *adaptive* or *non-adaptive*. Adaptive queries can be dependent on the outcome of previous queries while non-adaptive queries are independent of one another. The verifiers in these notes will all be non-adaptive for the sake of presentation. The PCP Theorem still holds when the verifier makes adaptive queries. The only change would be that the proof length would be at most $2^{r(n)+q(n)}$ rather than at most $q(n)2^{r(n)}$.

2.2. **Equivalence of PCP Theorems.** It may be difficult to understand the importance of the PCP Theorem in its form presented in Theorem 2.2. It turns out there are other equivalent forms of the PCP Theorem more palatable in the context of our goal to prove hardness of approximation results.

Theorem 2.4. *(*PCP *Theorem:* $\mathsf{Gap3SAT}$*-hardness) The problem* $\mathsf{Gap}_{\alpha,1}\mathsf{Max3SAT}$ *is* NP*-hard. In other words there exists a constant* $\alpha \leq 1$ *such that, for every* NP *language* L*, there exists a polynomial-time reduction* f *mapping* L *to 3CNF formulas such that:*

$$x \in L \implies \mathsf{Opt}(f(x)) = 1$$
$$x \notin L \implies \mathsf{Opt}(f(x)) \leq \alpha$$

An immediate consequence of Theorem 2.4 and Theorem 1.9 is that if there exists an α-approximation algorithm for $\mathsf{Max3SAT}$, then $\mathsf{P} = \mathsf{NP}$. With this, we have the first steps towards an inapproximability result: if $\mathsf{P} \neq \mathsf{NP}$, there exists no efficient $\mathsf{Max3SAT}$ algorithm which can approximate better than an α factor. Note that we haven't actually found a concrete value for α yet. This will be addressed once we prove Håstad's 3-bit PCP for NP in a future section.

Theorem 2.5. *(*PCP*-Theorem:* $\mathsf{GapMaxCSP}$*- hardness) There exists constants* $q \in \mathbb{N}$ *and* $\alpha \leq 1$ *where the problem* $\mathsf{Gap}_{\alpha,1}\mathsf{Max\text{-}qCSP}$ *is* NP*-hard. To elaborate, for every* NP *language* L*, there exists a polynomial time reduction mapping an* L *to a instance* $f(x)$ *of some CSP* Ψ *over domain* $\Omega = \{0, 1\}$ *where* Ψ *consists of* q*-ary predicates, such that*

$$x \in L \implies \mathsf{Opt}(f(x)) = 1$$
$$x \notin L \implies \mathsf{Opt}(f(x)) \leq \alpha$$

Theorem 2.6. *All the PCP Theorems above are equivalent to each other.*

Before we embark on the proof, let us establish an equivalence between PCPs and CSPs:

Lemma 2.7. *(Equivalence between PCPs and CSPs) Theorem 2.2 and Theorem 2.5 are equivalent.*

Proof. First, assume $\mathsf{NP} = \mathsf{PCP}_{1,\frac{1}{2}}(O(\log n), O(1))$. We will outline a procedure to convert the verifier V into an instance $\mathcal{I}$ for a q-ary CSP Ψ for some constant q. For some input string $x \in \{0, 1\}^n$ and proof string π, let $r \in \{0, 1\}^{c \cdot \log n}$ be the coin flips made by V and $V_{x,r}$ be the deterministic procedure which is executed on input x and coin flip r such that $V_{x,r} = 1$ iff V accepts proof π on input x and coin flip r. We can define the domain of our constructed CSP Ψ to be $\Omega = \{0, 1\}$ and the predicates to be $\{V_{x,r}\}_r$. Now our instance $\mathcal{I}$ of Ψ is casted as the tuples $(S, V_{x,r})$ where S will be at most a q-sized tuple indicating which indices of the proof π are queried when conditioned on r. This yields a polynomially-sized qCSP instance $\mathcal{I}$. Furthermore, since the verifier V runs in polynomial time, it's execution can be simulated on all r to output the instance $\mathcal{I}$ in polynomial-time. Thus, we have given a polynomial-time reduction from an input x to its corresponding CSP instance $\mathcal{I}$, so Theorem 2.5.

Conversely, suppose we had a reduction from NP to $\mathsf{Gap}_{\alpha,1}\mathsf{Max\text{-}qCSP}$ as stated in Theorem 2.5. We devise polynomial-time reduction taking an instance $\mathsf{Max\text{-}qCSP}$ to a polynomial-time PCP verifier V using logarithmic number of random bits and a constant number of queries to the supplied proof π. For an input $x \in \{0,1\}^n$, the proof will be expected to be an assignment to its respective instance $f(x)$ in the notation utilized in Theorem 2.5. Verifier V makes coin flips $r \in \{0,1\}^{c \log n}$ to choose one constraint tuple (S, ψ) where ψ is some q-ary predicate. Only a logarithmic number of random bits are required to query any constraint in instance $f(x)$ as the polynomial-time NP reduction can only generate a polynomial number of such constraints. The PCP only has to make q-queries to the proof π to find the assignments to the variables listed in $S = (v_1, \cdots, v_q)$. By the properties listed in Theorem 2.5, the PCP verifier V must have completeness 1 and soundness $\leq \frac{1}{2}$ as claimed. $\square$

Lemma 2.8. *(Equivalnce between* $\mathsf{GapMaxCSP}$*and* $\mathsf{GapMax3SAT}$ *Theorem 2.4 and Theorem 2.5 are equivalent.*

Proof. Since any $\mathsf{3Sat}$ instance can be seen as a particular type of $\mathsf{3CSP}$ instance, one direction is immediate. Conversely, if we assume Theorem 2.5, we aim to find some constant α' such that there exist a reduction from an instance of $\mathsf{Gap}_{1-\alpha,1}\mathsf{Max\text{-}qCSP}$ for $q \in \mathbb{N}$ claimed in Theorem 2.5 to an instance of $\mathsf{Gap}_{1-\alpha',1}\mathsf{Max3Sat}$. Let the CSP Ψ be comprised q-ary predicates ψ. For an instance $\mathcal{I}$ of Ψ, an constraint tuple (S, ψ) can be expressed as a logical AND of 2^q clauses where each clause is a logical OR of q variables or their negations. In other words, (S, ψ) is essentially a CNF of width q and of size at most 2^q. We can then construct an "equivalent" 3CNF as follows: add extra symbols $\Pi_1, \cdots, \Pi_{(q-3)2^q}$. It can be shown that there exists a 3CNF ψ'_S of size at most $(q-2)2^q$ such that:

(1) For every $x \in \{0,1\}^q$ which causes $\psi(x) = 1$, there exists an assignment Π such that $\psi'(x, \Pi) = 1$.
(2) Else if $\psi(x) = 0$, then for all assignments Π, $\psi'(x, \Pi) = 0$

Now take the total 3CNF defined by a conjection of all such ψ'_S:

$$\psi_{\mathcal{I}} = \bigwedge_{(S,\psi)\in\mathcal{I}} \psi'_S$$

The total formula $\psi_{\mathcal{I}}$ is determined by at most $mq2^q$ number of clauses and at most $n + m(q-3)2^q$ number of variables. If an instance $\mathcal{I}$ has all of its constraints satisfiable by some assignment, then by proprety (1) listed above, there must exist a assignment to the variables of the $\psi_{\mathcal{I}}$ such that it is satisifed. On the other hand, if for all assignments to instance $\mathcal{I}$ only satisfy at most an $1-\alpha$ fraction of constraints, then fraction of clauses satisfied can be at most $1 - \alpha + \alpha(1 - \frac{1}{(q-2)2^q a}) = 1 - \frac{\alpha}{(q-2)2^q}$. This is due to the fact that for each unsatisfied ψ'_S, the most number of its clauses which can be satisfied by any assignment will be $(q-2)2^q - 1$. Taking $\alpha' = \frac{\alpha}{(q-2)2^q}$ yields the required constant. $\square$

3. Label-Cover and Projection Games

We now introduce a problem which manages to provide a natural paradigm for capturing the essence of CSPs and proving inapproximability results. These "projection games" were introduced by Bellare, Goldreich, and Sudan [4]. The NP-hardness of the gap problem version of Label Cover was used by Håstad to show tight inapproximability results for $\mathsf{Max3SAT}$ and $\mathsf{MaxE3Lin}$ [8].

Definition 3.1. A *Label Cover (LC) Problem* instance $\mathcal{G}$ is defined by a bipartite graph $(A \sqcup B, E)$, finite alphabets Σ_A, Σ_B, and a set of projections $\pi_e : \Sigma_A \to \Sigma_B$ for every edge $e \in E$. Define an *assignment* as consisting of two maps $\mathfrak{A} : A \to \Sigma_A$, $\mathfrak{B} : B \to \Sigma_B$. An edge $e = (a, b) \in E$ is said to be satisfied by this assignment if the assignment is compatible with projection π_e:

$$\pi_e(\mathfrak{A}(a)) = \mathfrak{B}(b) \tag{3.1}$$

The value of this game will be

$$\mathsf{Opt}(\mathcal{G}) = \max_{(\mathfrak{A},\mathfrak{B})} \mathbb{E}_{e\sim E}[e \text{ satisfied}] \tag{3.2}$$

In other words, the value will be the largest fraction of edges satisfied by any assignment to the vertices. The corresponding gap problem for Label Cover, $\mathsf{Gap}_{\alpha,\beta}\mathsf{LC}$, is defined as the promise problem:

$$\mathsf{YES} = \{\mathcal{G} \mid \mathsf{Opt}(\mathcal{G}) \geq \beta\}$$
$$\mathsf{NO} = \{\mathcal{G} \mid \mathsf{Opt}(\mathcal{G}) < \alpha\}$$

In the case of perfect completeness, we abbreviate $\mathsf{Gap}_{\alpha,1}\mathsf{LC}$ as simply $\mathsf{Gap}_{\alpha}\mathsf{LC}$.

There are a few observations worthy of mentioning here. The first regards a type of equivalence between CSP instances and Label Cover instances. Specifically, let $\mathcal{I}$ be an instance of a given CSP Ψ over domain Ω. We can translate this CSP instance into a Label Cover instance as follows: Let the left-hand partition A of our bipartite graph be indexed by the set of constraint tuples (S, ψ) and the right-hand partition B be indexed by the variables of the CSP V. Draw an edge from a constraint tuple (S, ψ) to a variable v if that variable appears in S. Set $\Sigma_A = \Omega^r, \Sigma_B = \Omega$ where r is the arity of the CSP, and for every edge $e = ((v_1, \cdots, v_r), \psi), v)$ define the projection $\pi_e : \Sigma_A \to \Sigma_B$ to be

$$\pi_e(\omega_1, \cdots, \omega_r) = \omega_i \text{ if } v_i = v$$

On the other hand, every Label Cover instance can be seen as a 2CSP over a sufficiently large domain: the predicates of the CSP would be all 2-ary predicates $\pi : \Sigma_A \times \Sigma_B \to \{0, 1\}$ representing every possible map from $\Sigma_A \to \Sigma_B$. Thus, the domain of our CSP can be defined as $\Omega = \Sigma_A \cup \Sigma_B$. The corresponding instance of this CSP would be (S, π_e) where π_e represents the predicate corresponding to the edge e's projection map π_e and $S = (a, b)$ would be the vertices of e between A and B respectively.

Theorem 3.2. *(Weak Projection Games Theorem) Label Cover is* NP*-hard to approximate within some constant.*

3.1. **Some Structural Results of PCPs.**

3.2. **Raz's Parallel Repetition Theorem.** By combining the PCP Theorem (Theorem 2.2) along with Raz's Parallel Repetition Theorem [14], the NP-hardness of Label Cover comes to fruition:

Theorem 3.3. *(Projection Games Theorem) For every* $\epsilon > 0$*, there exist alphabets* Σ_A, Σ_B *where* $|\Sigma_A|, |\Sigma_B| \leq \mathsf{poly}(\frac{1}{\epsilon})$ *such that* $\mathsf{Gap}_{\epsilon}\mathsf{LC}$ *is* NP*-hard.*

4. Håstad's 3-bit PCP

4.1. **Inapproximability Results for $\mathsf{MaxE3Lin}$ and $\mathsf{Max3Sat}$.** The significance of Håstad's PCP for NP is its use in showing tight inapproximability results for $\mathsf{MaxE3Lin}$ and hence $\mathsf{Max3Sat}$. The theorem is first stated below:

Theorem 4.1. *(Håstad's 3-bit PCP, [8]) For every* $\delta > 0$ *and* $L \in \mathsf{NP}$*, there exists a PCP verifier for* L *over the boolean alphabet such that for every*

- *The verifier V queries 3 bits of the proof $x_{q_1}, x_{q_2}, x_{q_3} \in \pi$ such that verification predicate is a three variable linear equation over $\mathbb{F}_2$ depending on the queried bits $x_{q_1}, x_{q_2}, x_{q_3}$.*
- *If $x \in L$, then there exists a proof π:*

$$\mathbb{P}[V(x,\pi) = 1] \geq 1 - \delta \tag{4.1}$$

- *If $x \notin L$, then for all proofs π:*

$$\mathbb{P}[V(x,\pi) = 1] \leq \frac{1}{2} + \delta \tag{4.2}$$

On closer inspection, Theorem 4.1 reveals that a gap problem based on $\mathsf{MaxE3Lin}$ is actually NP-hard to decide:

Corollary 4.1.1. *For every $\delta > 0$, the promise problem $\mathsf{Gap}_{\frac{1}{2}+\delta,1-\delta}\mathsf{MaxE3Lin}$ is NP-hard.*

This gives our first threshold result: Corollary 4.1.1 states that the simple $\frac{1}{2}$-approximation algorithm introduced in Example 1.2 is optimal in the sense that the existence of an efficient algorithm improving on the $\frac{1}{2}$ constant factor would show that $\mathsf{P} = \mathsf{NP}$. From this result, by reducing $\mathsf{MaxE3Lin}$ to $\mathsf{Max3Sat}$, another threshold result appears:

Corollary 4.1.2. *For every $\delta > 0$, the promise problem $\mathsf{Gap}_{\frac{7}{8}+\delta,1-\delta}\mathsf{Max3Sat}$ is NP-hard.*

Proof. It suffices to transform $\mathsf{MaxE3Lin}$ CSP instances to MaxSat CSP instances. For predicates of the form $x + y + z = 1$, consider the conjunction of the clauses:

$$\begin{gathered} x \vee y \vee z \\ \bar{x} \vee y \vee \bar{z} \\ \bar{x} \vee y \vee \bar{z} \\ x \vee \bar{y} \vee \bar{z} \end{gathered}$$

The case for predicates of the form $x + y + z = 0$ are handled with a similar idea. For an assignment of a $\mathsf{MaxE3Lin}$ instance which satisfies a linear constraint, it must satisfy all four of the associated MaxSat constraints. In the case where an assignment does not satisfy a linear constraint, at most three of the four clauses will be satisified. Hence if an assignment satifies $\frac{1}{2} + \delta$ of the constraints, then it can satisify at most:

$$(\frac{1}{2} + \delta)4 + (\frac{1}{2} - \delta)\frac{3}{4} = \frac{7}{8} + \frac{3\delta}{4}$$

and if at least $1 - \delta$ of the linear constaints are satisfied then at least $1 - \delta$ of the clauses must also be satisfied. ☐

The proof of Theorem 4.1 leverages the NP-hardness of $\mathsf{Gap}_\epsilon\mathsf{LC}$ mentioned in Theorem 3.3. H åstad devised a dictatorship test based on $\mathsf{E3Lin}$ predicates then composed this test with checking the validity of a $\mathsf{Gap}_\epsilon\mathsf{LC}$ proof. In fact, the three queries to the proof arise from a modified Blum-Luby-Rubinfeld Linearity Test. The details will be treated in the upcoming subsections.

4.2. **The Long Code.** The workhorse for Håstad's proof will be the use of the *long code*. A long code is esstentially the truth table of a dictator function. The idea roughly revolve around the idea that dictator functions are locally-testable as linear functions (à la BLR Test) and have an useful error-correction property. Correct proofs will be expected to encode labels to vertices in a Label Cover instance as long codes of dictators corresponding to the indices of the labels. In other words, for a label $\ell \in \Sigma$, a corresponding index $i(\ell)$ can be computed with $\log|\Sigma|$ bits. The total number of such bitstrings will be, of course, length $2^{\log|\Sigma|} = |\Sigma|$. If we additionally encode this bitstring into the dictator function $\chi_{i(\ell)}(x_1, \cdots, x_\Sigma) = x_{i(\ell)}$, the truth table for this function will be of total length $2^{|\Sigma|}$. This reveals that the long code requires a *double-exponential* proof length. The truth table will serve as query access to the modified BLR test mentioned above.

4.3. **Aside on Dictatorship Testing.** The first step towards testing if an input boolean function $f : \{-1,1\}^n \to \{-1.1\}$ is a dictator arises from the Blum-Luby-Rubinfeld Test (BLR) which we restate below:

(1) Sample $x, y \sim_R \{-1,1\}^n$
(2) Accept iff $f(x)f(y) = f(xy)$

Theorem 4.2. *Suppose the BLR test accepts $f : \{-1,1\}^n \to \{-1,1\}$ with probability $1-\epsilon$, then f is ϵ-close to a linear function χ_{S*} for some $S^* \subseteq [n]$.*

Certainly the dictators $\chi_i,\ i \in [n]$ are linear functions. Hence, if $f = \chi_i$ for some i, then the BLR test accepts with probability one. However, we have the rest of the parity functions $\chi_S, |S| \geq 2$ which the BLR Test cannot distinguish. We need to amend the test to ensure that parity functions of higher weight are rejected with high probability. Håstad proposed modifiying the vanilla BLR test to add noise to the sampled product xy. Although this sacrifices perfect completeness, it penalizes large parity functions:

(1) Sample $x, y \sim \{-1,1\}^n$
(2) Sample $z \sim N_{1-2\epsilon}(xy)$
(3) Accept iff $f(x)f(y) = f(z)$

Lemma 4.3. *(Completeness of the Noisy BLR Test) If $f = \chi_i$ is a dictator for some i, then*

$$\mathbb{P}[\text{ \textit{Noisy BLR test accepts} }] \geq 1 - \epsilon$$

Proof. Note that if $z \sim N_{1-2\epsilon}(xy)$ for some $x \in \{-1,1\}^n$, then y can be expressed as below:

$$z_i = \begin{cases} x_i y_i \text{ with probability } 1-\epsilon \\ -x_i y_i \text{ with probability } \epsilon \end{cases} \tag{4.3}$$

By the acceptance criterion,

$$f(z) = f(x)f(y) \implies z_i = x_i y_i$$

This occurs with probability $1-\epsilon$ by the observation above, yielding completeness as claimed. □

Lemma 4.4. *Suppose that for some constant $\nu > 0$:*

$$\mathbb{P}[\text{ \textit{Noisy BLR test accepts} }] \geq \frac{1}{2} + \nu$$

then

$$2\nu \leq \sum_{S \subseteq [n]} \hat{f}(S)^3 (1-2\epsilon)^{|S|} \tag{4.4}$$

Proof. The proof begins by noticing the accepting probability can be expressed as:

$$\mathbb{P}[\text{ Noisy BLR test accepts }] = \frac{1}{2} + \frac{1}{2}\mathbb{E}_{x,y,z}\left[f(x)f(y)f(z)\right]$$

By our assumption, we prove that:

$$\begin{aligned}
\frac{1}{2}+\nu \leq \frac{1}{2}+\frac{1}{2}\mathbb{E}_{x,y,z}\left[f(x)f(y)f(z)\right] &\implies \\
2\nu \leq \mathbb{E}_{x,y,z}\left[f(x)f(y)f(z)\right] &\implies \\
2\nu \leq \mathbb{E}_{x,y,z}\left[f(x)f(y)\mathbb{E}_{z\sim N_{1-2\epsilon}(xy)}\left[f(z)\right]\right] &\implies \\
2\nu \leq \mathbb{E}_{x}\left[f(x)\mathbb{E}_{y}\left[f(y)\mathcal{T}_{1-2\epsilon}f(xy)\right]\right]& \\
2\nu \leq \mathbb{E}_{x}\left[f(x)(f*\mathcal{T}_{1-2\epsilon}f)(x)\right] &\implies \\
2\nu \leq \sum_{S\subseteq[n]}\widehat{f}(S)\widehat{f}(S)\widehat{\mathcal{T}_{1-2\epsilon}f}(S) &\implies \\
2\nu \leq \sum_{S\subseteq[n]}\widehat{f}(S)^3(1-2\epsilon)^{|S|}&
\end{aligned}$$

as claimed. □

The operator $\mathcal{T}_\rho$ is the *noise operator* which, given boolean function $f:\{-1,1\}^n\to\{-1,1\}$, maps f to another boolean function $\mathcal{T}_\rho(f)$ with the following Fourier decomposition:

$$\mathcal{T}_\rho(f)=\sum_{S\subseteq[n]}\rho^{|S|}\widehat{f}(S)\chi_S \tag{4.5}$$

Corollary 4.4.1. *(Soundness of the Noisy BLR Test) There exists some $S^*\subseteq[n]$ such that*

$$|\widehat{f}(S^*)|\geq 2\nu \quad |S^*|\leq O\left(\frac{1}{\epsilon}\log\frac{1}{\nu}\right) \tag{4.6}$$

Proof. From Theorem 4.4, we deduce that:

$$\begin{aligned}
2\nu \leq \sum_{S\subseteq[n]}\widehat{f}(S)^3(1-2\epsilon)^{|S|} &\leq \max_S \widehat{f}(S)(1-2\epsilon)^{|S|}\sum_{S\subseteq[n]}\widehat{f}(S)^2\\
&=\max_S \widehat{f}(S)(1-2\epsilon)^{|S|}
\end{aligned}$$

By taking the maximum attained as S^*,

$$2\nu\leq\widehat{f}(S^*)(1-2\epsilon)^{|S^*|}$$

Since both $\widehat{f}(S^*)\geq 2\nu$ and $(1-2\epsilon)^{|S^*|}\geq 2\nu$ (as f is boolean):

$$e^{-2\epsilon|S^*|}\geq 2\nu$$

which shows that $|S^*|\leq O\left(\frac{1}{\epsilon}\log\frac{1}{\nu}\right)$. □

4.4. **The Code Consistency Test.** The consistency test will compose the dictatorship test formulated above with checking the validity of an alleged long code as well as satisfiablity of the labels encoded. More formally, the input will be a $\mathsf{Gap}_\epsilon\mathsf{LC}$ instance $((A\sqcup B,E),\Sigma_A,\Sigma_B,\{\pi_e\}_{e\in E})$ and the proof will be the long codes encoding the assignments to the vertices, $\mathfrak{A}:A\to\Sigma_A$, $\mathfrak{B}:B\to\Sigma_B$. The consistency check will aim to perform two tasks simultaneously given an edge (a,b) and its associated projection constraint π: Let $f_a:\{-1,1\}^{|\Sigma_A|}\to\{-1,1\}$, $f_b:\{-1,1\}^{|\Sigma_B|}\to\{-1,1\}$ be the functions claimed to be long codes for the labels assigned to vertices a,b: $\mathfrak{A}(a),\mathfrak{B}(b)$. Recall these will be encoded as truth table strings on the proof string π.

Validity Check: Check that the functions f_a, f_b are indeed valid long codes for some labels $\mathfrak{A}(a), \mathfrak{B}(b)$.
Consistency Check: Verify that the assigned labels satisfy π

$$\pi(\mathfrak{A}(a)) = \mathfrak{B}(b)$$

Before presenting the test, some notation is in order. Let $(x \circ \pi)_i = x_{\pi(i)},\ \forall i \in \Sigma_A$ for $x \in \{-1,1\}^{|\Sigma_B|}$. It worth keeping in mind that $x \circ \pi \in \{-1,1\}^{|\Sigma_A|}$.

(1) Sample an edge $e = (a,b) \in E$. Let π be the associated projection constraint.
(2) Sample $x \sim \{-1,1\}^{|\Sigma_B|},\ y \sim \{-1,1\}^{|\Sigma_A|}$
(3) Sample $z \sim N_{1-2\delta}((x \circ \pi)y)$
(4) Accept iff $f_a(z) = f_b(x) f_a(y)$

Note that if the labels assigned a, b satisfied π and f_a, f_b are valid long codes: $f_a(x \circ \pi) = f_b(x)$ for all $x \in \{-1,1\}^{|\Sigma_B|}$.

Proposition 4.5. (Completeness) If $\mathsf{Gap}_\delta\mathsf{LC}$ instance $((A \sqcup B, E), \Sigma_A, \Sigma_B, \{\pi_e\}_{e\in E})$ satisfies all constraints, then there exists code words f_a, f_b such that

$$\mathbb{P}[\text{ Consistency Test Accepts }] \geq 1 - \delta \tag{4.7}$$

Proof. Suppose the assignment atisfies all constaints of the $\mathsf{Gap}_\epsilon\mathsf{LC}$ instance above. Set $f_a(x_1, \cdots, x_{\Sigma_A}) = x_{\mathfrak{A}(a)},\ f_b(x_1, \cdots, x_{\Sigma_B}) = x_{\mathfrak{B}(b)}$. The following then hold:

$$\begin{aligned}
\mathbb{P}[\text{ Consistency Test Accepts }] &= \mathbb{P}_{x,y,z}\left[f_a(z) = f_b(x) f_a(y)\right] \\
&= \mathbb{P}_{x,y,\mu}\left[(x \circ \pi)_{\mathfrak{A}(a)} y_{\mathfrak{A}(a)} \mu_{\mathfrak{A}(a)} = x_{\mathfrak{B}(b)} y_{\mathfrak{A}(a)}\right] \quad (\mu \sim N_{1-2\delta}(1^{|\Sigma_A|})) \\
&= \mathbb{P}_{x,\mu}\left[(x \circ \pi)_{\mathfrak{A}(a)} \mu_{\mathfrak{A}(a)} = x_{\mathfrak{B}(b)}\right] \\
&= \mathbb{P}_{x,\mu}\left[x_{\pi(\mathfrak{A}(a))} \mu_{\mathfrak{A}(a)} = x_{\mathfrak{B}(b)}\right] \\
&= \mathbb{P}_{x,\mu}\left[x_{\mathfrak{B}(b)} \mu_{\mathfrak{A}(a)} = x_{\mathfrak{B}(b)}\right] \\
&= 1 - \delta
\end{aligned}$$

□

4.5. **Folded Functions.**

4.6. **Soundness and Decoding a Labeling.**

Proposition 4.6. Suppose that for a $\mathsf{Gap}_\epsilon\mathsf{LC}$ instance, the consistency test above accepts with at least $\frac{1}{2} + \delta$, then there exists a labeling which satisfies at least δ of the projection constraints.

Note that the soundness criterion intuitively implies the use of a method to extract out a labeling from a high enough acceptance probability. This is the "local-correctability" property of linear functions coming into play. We first prove the following lemma:

Definition 4.7. For $S \subseteq \Sigma_A$ and $\pi : \Sigma_A \to \Sigma_B$, define $\pi_2(S) = \{j \in \Sigma_B \mid |\pi^{-1}(j) \cap S| \text{ is odd}\}$

Lemma 4.8. *Assume the test accepts with at least $\frac{1}{2} + \delta$ probability, then for at least δ fraction of the edges (a,b), the associated codes $f_a : \{-1,1\}^{|\Sigma_A|} \to \{-1,1\},\ f_b : \{-1,1\}^{|\Sigma_B|} \to \{-1,1\}$ satisfy the following inequality:*

$$\sum_{S \subseteq [n]} \widehat{f}_a(S)^2 \widehat{f}_b(\pi_2(S))(1-2\delta)^{|S|} \geq \delta \tag{4.8}$$

Proof. The proof is remarkably similar to that of Lemma 4.4. From our assumption, it is immediate that:

$$\frac{1}{2}+\frac{1}{2}\mathbb{E}_{(u,v),x,y,\mu}\left[f_b(x)f_a(y)f_a((x\circ\pi)y\mu)\right]\geq\frac{1}{2}+\delta\quad(\mu\sim N_{1-2\delta}(1^{|\Sigma_A|})) \tag{4.9}$$

which implies that:

$$\mathbb{E}_{(u,v),x,y,\mu}\left[f_b(x)f_a(y)f_a((x\circ\pi)y\mu)\right]\geq 2\delta \tag{4.10}$$

By an averaging argument, there must exist at least δ fraction of the edges which satisfy:

$$\mathbb{E}_{x,y,\mu}\left[f_b(x)f_a(y)f_a((x\circ\pi)y\mu)\right]\geq\delta \tag{4.11}$$

Hence, through a Fourier-analytic argument:

$$\begin{aligned}
\mathbb{E}_{x,y,\mu}\left[f_b(x)f_a(y)f_a((x\circ\pi)y\mu)\right]\geq\delta &\implies\\
\mathbb{E}_{x,y}\left[f_b(x)f_a(y)\mathbb{E}_{\mu}\left[f_a((x\circ\pi)y\mu)\right]\right]\geq\delta &\implies\\
\mathbb{E}_{x,y}\left[f_b(x)f_a(y)\mathcal{T}_{1-2\delta}(f_a)(x\circ\pi)y)\right]\geq\delta &\implies\\
\mathbb{E}_{x}\left[f_b(x)(f*\mathcal{T}_{1-2\delta}(f_a))(x\circ\pi)\right]\geq\delta &\implies\\
\sum_S\widehat{f}_b(S)\widehat{f}_a(\pi^{-1}(S))^2(1-2\delta)^{|S|}\geq\delta &\implies\\
\sum_S\widehat{f}_b(\pi(S))\widehat{f}_a(S)^2(1-2\delta)^{|S|}\geq\delta &\implies\\
\sum_S\widehat{f}_b(\pi_2(S))\widehat{f}_a(S)^2(1-2\delta)^{|S|}\geq\delta &
\end{aligned}$$

The last implication follows from the observation that:

$$\chi_S(x\circ\pi)=\prod_{i\in S}x_{\pi(i)}=\prod_{i\in\pi(S)}x_i=\prod_{i\in\pi_2(S)}x_i=\chi_{\pi_2(S)}(x)$$

□

We can now finish the soundness proof by decoding the labeling from our supplied proof string π. The actual procedure is not complicated:

(1) For each $a\in A$, do the following steps:
(2) Sample $S\sim(\widehat{f}_a(S))^2$
(3) Set $\mathfrak{A}(a)\xleftarrow{R}S\quad$ (uniformly sample from S)

Do the same for each $b\in B$ except sample the subset $T\sim\widehat{f}_b^2(T)$ from the Fourier spectrum of f_b:

(1) For each $b\in A$, do the following steps:
(2) Sample $T\sim(\widehat{f}_b(T))^2$
(3) Set $\mathfrak{B}(b)\xleftarrow{R}T\quad$ (uniformly sample from T)

Since f_a,f_b are assumed to be folded (odd) functions, both $\widehat{f}_a(\emptyset)=\widehat{f}_b(\emptyset)=0$, so we will never sample the empty set in any of the procedures above. We now aim to show the inequality:

Lemma 4.9.

$$\mathbb{P}_{(a,b)\in E}[(a,b)\textit{ is satisfied}]\geq\sum_S\sum_{T\subseteq\pi(S)}\widehat{f}_a(S)^2\widehat{f}_b(T)^2\cdot\frac{1}{|S|} \tag{4.12}$$

Proof. Here is one way to get a label which satisfies a chosen edge (a,b). The first procedure samples a subset from the Fourier spectrum of f_a. The second procedure then samples from the Fourier spectrum of f_b and receives a subset $T \subseteq \pi(S)$. The second procedure then uniformly samples from T to receive some label $t \in T$. Since T is contained in the image of S under π, there must exist at least one element in $s \in S$ such that $\pi(s) = t$, showing that this event occurs with probability at least $\frac{1}{|S|}$ □

Proposition 4.10. (Soundness) For the labeling scheme constructed above, the probability:

$$\mathbb{P}_{(a,b)\in E}[(a,b) \text{ is satisfied}] \geq 4\delta^4 \tag{4.13}$$

Proof. By Lemma 4.9:

$$\begin{aligned}
\mathbb{P}_{(a,b)\in E}[\text{(a,b) is satisfied}] &\geq \sum_S \sum_{T\subseteq\pi(S)} \widehat{f}_a(S)^2 \widehat{f}_b(T)^2 \cdot \frac{1}{|S|} \\
&\geq \sum_S \widehat{f}_a(S)^2 \widehat{f}_b(\pi_2(S))^2 \frac{1}{|S|} \\
&= \sum_S \left(\widehat{f}_a(S)\widehat{f}_b(\pi_2(S))\frac{1}{\sqrt{|S|}}\right)^2 \cdot \left(\sum_S \widehat{f}(S)^2\right) \\
&\geq \left(\sum_S \widehat{f}_a(S)^2 \widehat{f}_b(\pi_2(S))\frac{1}{\sqrt{|S|}}\right)^2 \quad \text{(Cauchy-Schwarz)} \\
&\geq 4\delta\left(\sum_S \widehat{f}_a(S)^2 \widehat{f}_b(\pi_2(S))(1-2\delta)^{|S|}\right)^2
\end{aligned}$$

The last inequality holds since $\frac{1}{\sqrt{x}} \geq 2\sqrt{\delta}(1-2\delta)^x$ for all $x, \delta > 0$. By virtue our bound derived in Lemma 4.8, we find that for at least an δ of the edges, $\sum_S \widehat{f}_a(S)^2 \widehat{f}_b(\pi_2(S))(1-2\delta)^{|S|} \geq \delta$. Hence:

$$\mathbb{P}_{(a,b)\in E}[(a,b) \text{ is satisfied}] \geq 4\delta \cdot \delta^3 = 4\delta^4 \tag{4.14}$$

□

By Theorem 3.3, we know that $\mathsf{Gap}_{4\delta^4}\mathsf{LC}$ is NP-hard for every $\delta \in [0, \frac{1}{2}]$. Since $\delta \geq 4\delta^4$ for $\delta \in [0, \frac{1}{2}]$. The argument for the completeness of this PCP verifier still holds if we set our δ to be $\delta' = 4\delta^4$. This demonstrates that we gave a PCP system for the NP-hard $\mathsf{Gap}_{4\delta^4}\mathsf{LC}$:

Proposition 4.11. Suppose for an input $\mathsf{Gap}_\delta\mathsf{LC}$ instance say $\mathcal{I}$ and associated proof string π causes the test to pass with at least $\frac{1}{2} + \delta$:

$$\mathbb{P}[\text{Consistency Test Passes}] \geq \frac{1}{2} + \delta \tag{4.15}$$

then there exists a assignment for $\mathcal{I}$ such that it satisfies at least $\delta' = 4\delta^4$ fraction of edges in $\mathcal{I}$.

This gives a PCP system for NP as desired.

5. Unique Games

5.1. **Definitions.** The PCP Theorem culminated in a proof of the NP-hardness of Label Cover by Håstad. Although these results gave proofs of the NP-hardness of $\mathsf{Gap}_{\frac{7}{8}+\epsilon,1-\epsilon}$-$\mathsf{Max3SAT}$ and $\mathsf{Gap}_{\frac{1}{2}+\epsilon,1-\epsilon}$-$\mathsf{MaxE3Lin}$, similar hardness proofs for other canonical problems such as MaxCut didn't seem to follow from these ideas. In his seminal paper, Khot proposed a relaxation of the Label Cover Problem [9]. The instances of this relaxed version are called *Unique Games*:

Definition 5.1. A *Unique Label Cover Problem* with m labels (UniqueLC(m)) instance $\mathcal{U}$ is defined by a bipartite graph $(A \sqcup B, E)$ where $|A| = |B| = n$ for some $n \in \mathbb{N}$, finite alphabet $\Sigma_A = \Sigma_B = \Sigma$ such that $|\Sigma| = m$, and a set of *permutations* $\pi_e : [m] \to [m]$ for every edge $e \in E$. Define an *assignment* as consisting of a map $\sigma : A \sqcup B \to [m]$. An edge $e = (a, b) \in E$ is said to be satisfied by this assignment if the assignment is compatible with projection π_e:

$$\pi_e(\sigma(a)) = \sigma(b) \tag{5.1}$$

The value of this game will be

$$\mathsf{Opt}(\mathcal{G}) = \max_{\sigma} \mathbb{E}_{e\sim E}[e \text{ satisfied}] \tag{5.2}$$

In other words, the value will be the largest fraction of edges satisfied by any assignment to the vertices. The corresponding gap problem for Label Cover, $\mathsf{Gap}_{\alpha,\beta}\mathsf{UniqueLC}(m)$, is defined as the promise problem:

$$\begin{aligned} \mathsf{YES} &= \{\mathcal{U} \mid \mathsf{Opt}(\mathcal{U}) \geq \beta\} \\ \mathsf{NO} &= \{\mathcal{U} \mid \mathsf{Opt}(\mathcal{U}) < \alpha\} \end{aligned}$$

In the case of perfect completeness, we abbreviate $\mathsf{Gap}_{\alpha,1}\mathsf{UniqueLC}(m)$ as simply $\mathsf{Gap}_{\alpha}\mathsf{UniqueLC}(m)$.

In addition, Khot formulated the *Unique Games Conjecture* and utilized it to prove several inapproxability results assuming the conjecture is true.

Conjecture 5.2. (Unique Games Conjecture [9]) For any constant $\delta > 0$, there exists sufficiently large $m \in \mathbb{N}$ such that $\mathsf{Gap}_{\delta,1-\delta}\mathsf{UniqueLC}(m)$ is NP-hard.

Remark. The $1 - \delta$ constant is crucial to the validity of the conjecture. For an instance of $\mathsf{UniqueLC}(m)$ for any m with a guaranteed solution, there exists a polynomial-time algorithm finding an assignment which satisfies all projection constraints: Start with a vertex and set it to a label. If it is an endpoint of an edge e, follow e to the other side and find a label which satisfies as many neighbors as possible. Repeat in a breadth-first search fashion. In virtue of the projection maps being permutations, this amounts to searching for the guaranteed solution in time $O(mn^2)$.

Example 5.3. The MaxCut problem for an input graph $G = (V, E)$ can be cast as a $\mathsf{UniqueLC}(|V|)$ instance. The two partitions of the bipartite graph A, B will be indexed by the vertices V. Draw an edge between two vertices v_1, v_2 if $(v_1, v_2) \in E$. Set the alphabet to be $\Sigma = \{-1, 1\}$ and the projection maps simply be the "swap" map $-1 \mapsto 1, 1 \mapsto -1$.

Example 5.4. $\mathsf{MaxE2LinModp}$ for prime p denotes the problem of finding an assignment which maximizes the number of satisfied linear constraints consisting of exactly two variables over field $\mathbb{F}_p$. An example of an instance of this problem over variables x_1, x_2, x_3, x_4 is shown below:

$$x_1 + x_3 = 3$$
$$x_2 + x_4 = 2$$
$$x_1 + x_4 = 1$$

An instance of this problem can also be translated as an instance of $\mathsf{UniqueLC}(m)$.

Definition 5.5. A promise problem P is said to be UG-*hard* if there is some constant $\delta > 0$ such that for all $m \in \mathbb{N}$, there exists a polynomial-time reduction f from $\mathsf{Gap}_{\delta,1-\delta}\mathsf{UniqueLC}(m)$ to P, in the sense that for a $\mathsf{Gap}_{\delta,1-\delta}\mathsf{UniqueLC}(m)$ instance $\mathcal{I}$:

$$\mathsf{Opt}(\mathcal{I}) \geq 1 - \delta \implies f(x) \in \mathsf{Yes}$$
$$\mathsf{Opt}(\mathcal{I}) < \delta \implies f(x) \in \mathsf{No}$$

6. UG-hardness of MAXCUT

6.1. **Intuitions.** Recall that the keen insight behind Håstad's 3-bit PCP for NP is the embedding of a dictatorship test within a Label Cover instance. In a similar spirit, the proof for the optimality of the Goemans-Williamson algorithm will hinge on crafting a clever dictatorship test embedded within a MaxCut instance. By composing this test with

6.2. **Goemans-Williamson Algorithm for MaxCut.** Example 1.3 presented an LP-based approximation algorithm for MaxCut. Goemans and Williamson [5] designed an SDP to give an α_{GW}-approximation algorithm for MaxCut where:

$$\alpha_{GW} = \min_{-1 \leq \rho \leq 1} \frac{2}{\pi} \frac{\cos^{-1}(\rho)}{1-\rho} \approx 0.87856 \tag{6.1}$$

To begin, a semi-definite program is a generalization of a linear program where instead of optimizing over a vector of variables $\vec{x}$, the program considers a positive semi-definite matrix of variables i.e a matrix whose eigenvalues are non-negative. An equivalent formulation considers inner products between pairs of vectors:

$$\max \sum_{i,j} c_{ij} \langle v_i, v_j \rangle$$
$$\text{under constraints } a_{ij}^k \langle v_i, v_j \rangle \leq b_{ij}^k$$
$$v_1, \cdots v_n \in \mathbb{R}, \quad k \in [C] \text{ constraints}$$

Note that this form also subsumes quadratic programming by setting $c_{ij} = a_{ij}^k = b_{ij}^k = 0$ for all $i \neq j$ and $k \in [C]$. The Goemans-Williamson algorithm concerns the solution to the semi-definite program given some graph $G = (V, E)$:

$$\max \sum_{i,j} \frac{1 - \langle v_i, v_j \rangle}{2} \tag{6.2}$$

$$\langle v_i, v_i \rangle = 1 \text{ for all } i \in V \tag{6.3}$$

Let us first show that indeed the program is a relaxation of the integer program crafted in Example 1.3. Indeed, if we set any unit vector $\vec{u}$ such that for a cut defined by $(S, \bar{S})$:

$$\vec{v_i} = \begin{cases} u \text{ if } i \in S \\ -u \text{ if } i \notin S \end{cases}$$

A direct calculation yields that the term $\frac{1-\langle v_i, v_j \rangle}{2} = 0$ if $v_i = v_j = u$ else it is equal to 1. Thus, by applying this observation to the maximum cut,

$$\mathsf{Opt}_{GW}(G) \geq \mathsf{MaxCut}(G)$$

So far the semi-definite program appears to be rather simple. The insight made by Goemans and Williamson lies in the rounding procedure of the solution outputted by the program. The procedure proceeds by first drawing a random hyperplane passing through the origin and taking the two partitions of the cut S_+, S_- to be the solution v_i which lie on positive and negative sides of the hyperplane respectively. We can calculate the expected size of the cut:

$$\mathbb{E}[|E(S_+, S_-)|] = \sum_{(i,j) \in E} \mathbb{P}[v_i, v_j \text{ lie on different sides of the hyperplane}]$$

Now if θ_{ij} denotes the angle between v_i, v_j, a simple geometric argument shows that:

$$\mathbb{P}[v_i, v_j \text{ lie on different sides of the hyperplane}] = \frac{\theta_{ij}}{\pi}$$

By definition of the dot product:

$$\frac{1 - \langle v_i, v_j \rangle}{2} = \frac{1 - \cos(\theta_{ij})}{2}$$

and the magical inequality:

$$\frac{\theta_{ij}}{\pi} \geq \alpha_{GW} \cdot \frac{1 - \cos(\theta_{ij})}{2}$$

the two can be combined to yield that:

$$\frac{\mathbb{E}[|E(S_+, S_-)|]}{\mathsf{Opt}_{GW}(f)} \geq \alpha_{GW} \tag{6.4}$$

Furthermore, the first inequality above implies that the Goemans-Williamson algorithm is indeed an α_{GW} approximation algorithm:

$$\frac{\mathbb{E}[|E(S_+, S_-)|]}{\mathsf{MaxCut}(G)} \geq \alpha_{GW} \tag{6.5}$$

6.3. **MaxCut is UG-hard.** After introducing the basic background, we are finally ready to show a non-trivial inapproximability result assuming the UGC:

Theorem 6.1. *(MaxCut is UG-hard) The problem* $\mathsf{Gap}_{\frac{\cos^{-1}(\rho)}{\pi}+\epsilon, \frac{1-\rho}{2}-\epsilon}\mathsf{MaxCut}$ *is* UG*-hard*

Note that an immediate corollary of Theorem and the UGC is that the Goemans-Williamson algorithm is tight:

Corollary 6.1.1. *Assuming the UGC, if an α_{GW}-approximation algorithm for* MaxCut *exists, then* $\mathsf{P} = \mathsf{NP}$.

Proof. Using Theorem 1.9, we see that for small ϵ:

$$\frac{\frac{\cos^{-1}(\rho)}{\pi}+\epsilon}{\frac{1-\rho}{2}-\epsilon} \approx \frac{2}{\pi}\frac{\cos^{-1}(\rho)}{1-\rho}$$

Setting $\rho \approx -0.6934$ yields the desired result. □

Onwards to the proof of Theorem 6.1. We first reason about the relationship between $\mathsf{UniqueLC}(m)$ instances and MaxCut instances. In particular, we wish to demonstrate that proving UG-hardness for MaxCut is equivalent to constructing a 2-query PCP for $\mathsf{GapUniqueLC}(m)$ for all m.

Theorem 6.2. $\mathsf{Gap}_{\alpha,\beta}\mathsf{MaxCut}$ *is* UG*-hard iff there exists a constant* $\delta > 0$ *such that for all* m *there exists a 2-query PCP for* $\mathsf{Gap}_{\delta,1-\delta}\mathsf{UniqueLC}(m)$ *with completeness* β *and soundness* α*.*

Proof. First assume that for some $\delta > 0$ and all m, there exists a polynomial-time reduction from $\mathsf{Gap}_{\delta,1-\delta}\mathsf{UniqueLC}(m)$ to $\mathsf{Gap}_{\alpha,\beta}\mathsf{MaxCut}$. Let $G = (V, E), \mathcal{C}$ be the graph outputted by reduction and the cut outputted by the cut approximation algorithm. Construct a PCP samples two vertices from $v_1, v_2 \sim V$ by querying the proof tape encoding $G, \mathcal{C}$ and outputs "accept" if $(v_1, v_2) \in \mathcal{C}$. The completeness and soundness parameters immediately follow from definition of $\mathsf{Gap}_{\alpha,\beta}\mathsf{MaxCut}$. Conversely, assume the existance of a 2-query PCP for $\mathsf{Gap}_{\delta,1-\delta}\mathsf{UniqueLC}(m)$ with the above properties. We will calculate the acceptance probabilities of this PCP given a proof string and an input instance by finding the max cut. Let the vertices be the proof locations of the input proof string π. It suffcies to draw an edge between two vertices weighted by the probability their corresponding proof locations are queried by the verifier. □

Thus, we can proceed by constructing a 2-query PCP for $\mathsf{Gap}_{\delta,1-\delta}\mathsf{UniqueLC}(m)$ with the completeness $\frac{1-\rho}{2} - \epsilon$ and soundness $\frac{\cos^{-1}(\rho)}{\pi} + \epsilon$. As with Håstad's 3-bit PCP, we aim to construct a dictatorship test which generates the parameters needed. Once again, a proof string π containing an assignment of labels to the vertices of a $\mathsf{UniqueLC}(m)$ instance would be encoded into a truth table of a dictator i.e into a *long code.*

6.4. **Dictatorship Testing Redux.** A key difference between dictatorship test crafted in Håstad's 3-bit PCP for NP and the one will aim to utilize here is the number of queries the PCP can make. Namely, the reduction here can only use *two queries* to the proof.

6.5. **Majority is the Stablest (MIS).** Before we craft our dictator test, we require a tool bounding the noise sensitivity of a boolean function with small low-degree influence. This is captured by the "Majority is the Stablest" theorem:

Theorem 6.3. *("Majority is the Stablest" (MIS) [12]) For every* $\epsilon > 0$*,* $\rho \in (-1, 0)$*, there exists* $\tau > 0$ *such that if for all* $i \in [n]$*,* $\mathsf{Inf}_i(f) < \tau$ *for function* $f : \{-1, 1\}^n \to [-1, 1]$*, then*

$$\mathsf{NS}_\rho(f) < \frac{\cos^{-1}(\rho)}{\pi} + \epsilon \tag{6.6}$$

Recall that the noise sensitivity of a boolean function $f : \{-1, 1\}^n \to \{-1, 1\}$ is defined as:

$$\mathsf{NS}_\rho(f) = \mathbb{P}_{y\sim N_\rho(x),x}[f(x) \neq f(y)] \tag{6.7}$$

where $y \sim N_\rho(x)$ refers to sampling a string y under the procedure:

$$y_i = \begin{cases} x_i & \text{with probabilty} \frac{1+\rho}{2} \\ -x_i & \text{with probabilty} \frac{1-\rho}{2} \end{cases}$$

Now equation 6.7 can be re-expressed in terms of the noise stability of f:

$$\mathsf{NS}_\rho(f) = \frac{1}{2} - \frac{1}{2}\mathsf{Stab}_\rho(f) \tag{6.8}$$

where

$$\mathsf{Stab}_\rho(f) = \mathbb{E}_{y\sim N_\rho(x),x}[f(x)f(y)]$$

Through Fourier-analytic techniques, we derive that:

$$\mathsf{NS}_\rho(f) = \frac{1}{2} - \frac{1}{2}\sum_{S\subseteq[n]} \hat{f}(S)^2\rho^{|S|} \tag{6.9}$$

A generalization of the MIS theorem will serve our purposes:

Theorem 6.4. *(Generalized MIS* [10],[12]*) For all $\epsilon > 0$, $\rho \in (0,1)$, there exists some $\tau > 0$ and finite d such that if $f : \{-1,1\}^n \to [-1,1]$ and for all $i \in [n]$, $\mathsf{Inf}_i^{\le d}(f) \le \tau$:*

$$\frac{1}{2} - \frac{1}{2}\sum_{S\subseteq[n]} \hat{f}(S)^2\rho^{|S|} < \frac{\cos^{-1}(\rho)}{\pi} + \epsilon \tag{6.10}$$

6.6. **The 2-query PCP.** With the MIS theorem, we can begin our analysis of the promised 2-query PCP. Before the procedure, define $(x\circ\pi)_i = x_{\pi(i)}$ for $x \in \{-1,1\}^n,\ i \in [n],\ \pi \in S_n$.

(1) Sample a vertex $a \in A$ uniformly.

(2) Sample two of its neighbors $b, b' \in B$ uniformly. Let $\pi_{b,a}$ and $\pi_{b',a}$ denote the *inverses* of the constraints associated to $(a,b),(a,b')$ respectively.

(3) Sample $x \sim \{-1,1\}^m$ and $y \sim N_\rho(x)$

(4) Accept if $f_b(x\circ\pi_{b,a}) \neq f'_b(y\circ\pi_{b',a})$

For the proof of Theorem 6.1, invoke Theorem 6.4 for $\frac{\epsilon}{2}$ and ρ assumed. This will yield a τ and a degree upper bound d. Set parameter

$$\delta = \frac{\epsilon\tau^2}{8d}$$

6.6.1. *Completeness.* Suppose we have a $\mathsf{UniqueLC}(m)$ instance $\mathcal{U}$ such that $\mathsf{Opt}(\mathcal{U}) \ge 1-\delta$. Through a simple union bound argument, the probability that both $\pi_{b,a}, \pi_{b',a}$ are satisfied by the assignment is at least $1-2\delta$. Now if both are indeed satisfied and the test accepts, it must be true that:

$$\begin{aligned} f_b(x\circ\pi_{b,a}) \neq f'_b(y\circ\pi_{b',a}) &\iff (x\circ\pi_{b,a})_{\sigma(b)} \neq (x\circ\pi_{b',a})_{\sigma(b')} \\ &\iff x_{\pi_{b,a}(\sigma(b))} \neq x_{\pi_{b',a}(\sigma(b'))} \\ &\iff x_{\sigma(a)} \neq y_{\sigma(a)} \end{aligned}$$

where for the last equivalence, we invoked the assumption that σ satisfies both $\pi_{b,a}, \pi_{b',a}$. Recall that σ is the assignment of labels to the vertices of the biparitite graph. The last expression occurs with probabilty $\frac{1-\rho}{2}$ by step three of the verification algorithm. Hence,

$$\mathbb{P}[\text{Test accepts}] \geq \frac{(1-2\delta)(1-\rho)}{2}$$

By observing that $\delta < \frac{\epsilon}{2}$, the inequality above reduces to:

$$\mathbb{P}[\text{Test accepts}] \geq \frac{(1-\epsilon)(1-\rho)}{2} \geq \frac{1-\rho}{2} - \epsilon \tag{6.11}$$

as desired.

6.6.2. *Soundness.* To show soundness, we prove the contrapositive, namely if

$$\mathbb{P}[\text{Test accepts}] \geq \frac{\cos^{-1}(\rho)}{\pi} + \epsilon$$

then there exists a labeling which satisfies more than a δ fraction of constraints.

Proof. First:

$$\mathbb{P}[\text{Test accepts}] = \mathbb{E}_{a,b,b'}\left[\mathbb{E}_{x,y\sim N_\rho(x)}\left[\frac{1}{2} - \frac{1}{2} f_b(x \circ \pi_{b,a}) f_{b'}(y \circ \pi_{b',a})\right]\right] \geq \frac{\cos^{-1}(\rho)}{\pi} + \epsilon \tag{6.12}$$

An averaging argument on the vertex $a \in A$ tells us that, since the test passes with at least $\frac{\cos^{-1}(\rho)}{\pi} + \epsilon$ probability, there must exist at least $\epsilon/2$ fraction of the vertices in A such that the test conditioned on picking a from this fraction passes with probability at least $\frac{\cos^{-1}(\rho)}{\pi} + \epsilon/2$. Otherwise, if there existed fewer than a $\epsilon/2$ fraction such that the test passed with at least this probability, then the total probability of the test passing is *at most* $(\epsilon/2)\cdot 1 + (1-\epsilon/2)(\frac{\cos^{-1}(\rho)}{\pi} + \epsilon/2) < \frac{\cos^{-1}(\rho)}{\pi} + \epsilon$, which is a contradiction. Let us label the vertices picked from this $\epsilon/2$ fraction as *good* vertices.

So say we picked one of these good vertices say a. Let us define the below function:

Definition 6.5. Define $g_a : \{-1,1\}^m \to [-1,1]$ as

$$g_a(x) = \mathbb{E}_b\left[f_b(x \circ \pi_{b,a})\right] \tag{6.13}$$

where the expectation is drawn uniformly over the neighbors b of a.

This allows us to re-express the inequality 6.12:

$$\mathbb{E}_{b,b'}\left[\mathbb{E}_{x,y\sim N_\rho(x)}\left[\frac{1}{2} - \frac{1}{2} f_b(x \circ \pi_{b,a}) f_{b'}(y \circ \pi_{b',a})\right]\right] \tag{6.14}$$

$$= \mathbb{E}_{x,y\sim N_\rho(x)}\left[\frac{1}{2} - \frac{1}{2}\mathbb{E}_b\left[f_b(x \circ \pi_{b,a})\right]\mathbb{E}_{b'}\left[f_{b'}(y \circ \pi_{b',a})\right]\right] \tag{6.15}$$

$$= \frac{1}{2} - \frac{1}{2}\mathbb{E}_{x,y\sim N_\rho(x)}\left[g_a(x) g_a(y)\right] \tag{6.16}$$

$$= \frac{1}{2} - \frac{1}{2}\mathsf{Stab}_\rho(g_a) \tag{6.17}$$

$$\geq \frac{\cos^{-1}(\rho)}{\pi} + \epsilon/2 \tag{6.18}$$

By invoking the contrapositive of the generalized MIS theorem (Theorem 6.4) on g_a, we deduce that there must exist some index $i_a \in [m]$ such that:

$$\mathsf{Inf}_{i_a}^{\leq d}(g_a) \geq \tau \tag{6.19}$$

Fortunately, there is a method to equate the Fourier coefficients g_a with those of f_b:

$$\begin{aligned}
g_a(x) &= \mathbb{E}_b\left[f_b(x \circ \pi_{b,a})\right] \\
&= \mathbb{E}_b\left[\sum_{S\subseteq[n]} \hat{f}_b(S)\chi_S(x \circ \pi_{b,a})\right] \\
&= \mathbb{E}_b\left[\sum_{S\subseteq[n]} \hat{f}_b(S)\chi_{\pi_{b,a}(S)}(x)\right] \\
&= \mathbb{E}_b\left[\sum_{S\subseteq[n]} \hat{f}_b(S)\chi_{\pi_{b,a}(S)}(x)\right] \\
&= \sum_{S\subseteq[n]} \mathbb{E}_b\left[\hat{f}_b(\pi_{b,a}^{-1}(S))\right]\chi_S(x)
\end{aligned}$$

where the last equality follows from reparameterizing the sum. By expanding g_a on the left-hand side of the equality into its Fourier expansion, from the orthogonality of characters:

$$\hat{g_a}(S) = \mathbb{E}_b\left[\hat{f}_b(\pi_{b,a}^{-1}(S))\right] \tag{6.20}$$

Starting from inequality 6.19,

$$\begin{aligned}
\tau \leq \mathsf{Inf}_{i_a}^{\leq d}(g_a) &= \sum_{\substack{S\subseteq[n], i_a\in S \\ |S|\leq d}} \hat{g_a}^2(S) \\
&= \sum_{\substack{S\subseteq[n], i_a\in S \\ |S|\leq d}} \left(\mathbb{E}_b\left[\hat{f}_b(\pi_{b,a}^{-1}(S))\right]\right)^2 \\
&\leq \sum_{\substack{S\subseteq[n], i_a\in S \\ |S|\leq d}} \mathbb{E}_b\left[\hat{f}_b^2(\pi_{b,a}^{-1}(S))\right] \\
&\mathbb{E}_b\left[\sum_{\substack{S\subseteq[n], i_a\in S \\ |S|\leq d}} \hat{f}_b^2(\pi_{b,a}^{-1}(S))\right] \\
&= \mathbb{E}_b\left[\mathsf{Inf}_{\pi_{b,a}^{-1}(i_a)}^{\leq d}(f_b)\right]
\end{aligned}$$

The inequality uses Cauchy-Schwarz. We use another averaging argument to see that there must exist at least a $\tau/2$ fraction of a's neighbors b such that

$$\mathsf{Inf}_{\pi_{b,a}^{-1}(i_a)}^{\leq d}(f_b) \geq \tau/2$$

otherwise the total influence term would be at most $\tau/2 \cdot 1 + (1-\tau/2)(\tau/2) < \tau$. For each b in that $\tau/2$ fraction, we pick a label uniformly at random from the set:

$$S_b = \{\ell \mid \mathsf{Inf}_\ell^{\leq d}(f_b) \geq \tau/2\}$$

which must be *non-empty* by the averaging argument made above. Notice that the label picked will satisfy $\pi_{b,a}$ by construction. We can thus lower bound the probability of constraint $\pi_{b,a}$ being satisfied:

$$\mathbb{P}[\pi_{(b,a)} \text{ is satisfied}] \geq \frac{\epsilon}{2}\frac{\tau}{2}\frac{1}{|S_b|} \tag{6.21}$$

Through a Fourier-analytic argument, we can upper-bound $|S_b|$:

$$\frac{|S_b|\tau}{2} \leq \sum_{i=1}^{|S_b|} \mathsf{Inf}_i^{\leq d}(f_b) \leq \sum_{i=1}^{m} \mathsf{Inf}_i^{\leq d}(f_b) = \sum_{\substack{S\subseteq[m]\\|S|\leq d}} |S|\hat{f}^2(S) \leq d \sum_{S\subseteq[m]} \hat{f}^2(S) = d$$

This yields that $|S_b| \leq \frac{2d}{\tau}$. So the probability bound of 6.21 would become:

$$\mathbb{P}[\pi_{(b,a)} \text{ is satisfied}] \geq \frac{\epsilon}{2}\frac{\tau}{2}\frac{\tau}{2d} = \frac{\epsilon\tau^2}{8d} = \delta \tag{6.22}$$

as desired. This completes the proof of Theorem 6.1. □

7. Semi-definite Programming and Integrality Gaps

References

1. Sanjeev Arora and Boaz Barak, *Computational complexity: a modern approach*, Cambridge University Press, 2009.
2. Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy, *Proof verification and the hardness of approximation problems*, Journal of the ACM (JACM) **45** (1998), no. 3, 501–555.
3. Sanjeev Arora and Shmuel Safra, *Probabilistic checking of proofs: A new characterization of np*, Journal of the ACM (JACM) **45** (1998), no. 1, 70–122.
4. Mihir Bellare, Oded Goldreich, and Madhu Sudan, *Free bits, pcps, and nonapproximability—towards tight results*, SIAM Journal on Computing **27** (1998), no. 3, 804–915.
5. Michel X Goemans and David P Williamson, *Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming*, Journal of the ACM (JACM) **42** (1995), no. 6, 1115–1145.
6. Prahladh Harsha, *Limits of approximation algorithms: Pcps and unique games (tifr/imsc lecture notes, spring 2010)*, (2010).
7. Prahladh Harsha, Moses Charikar, Matthew Andrews, Sanjeev Arora, Subhash Khot, Dana Moshkovitz, Lisa Zhang, Ashkan Aazami, Dev Desai, Igor Gorodezky, et al., *Limits of approximation algorithms: Pcps and unique games (dimacs tutorial lecture notes)*, arXiv preprint arXiv:1002.3864 (2010).
8. Johan Håstad, *Some optimal inapproximability results*, Journal of the ACM (JACM) **48** (2001), no. 4, 798–859.
9. Subhash Khot, *On the power of unique 2-prover 1-round games*, Proceedings of the thiry-fourth annual ACM symposium on Theory of computing, 2002, pp. 767–775.
10. Subhash Khot, Guy Kindler, Elchanan Mossel, and Ryan O'Donnell, *Optimal inapproximability results for max-cut and other 2-variable csps?*, SIAM Journal on Computing **37** (2007), no. 1, 319–357.
11. Subhash Khot and Nisheeth K Vishnoi, *On the unique games conjecture*, FOCS, vol. 5, Citeseer, 2005, p. 3.
12. Elchanan Mossel, Ryan O'Donnell, and Krzysztof Oleszkiewicz, *Noise stability of functions with low influences: invariance and optimality*, 46th Annual IEEE Symposium on Foundations of Computer Science (FOCS'05), IEEE, 2005, pp. 21–30.
13. Ryan O'Donnell, *Analysis of boolean functions*, Cambridge University Press, 2014.
14. Ran Raz, *A parallel repetition theorem*, SIAM Journal on Computing **27** (1998), no. 3, 763–803.
15. Luca Trevisan, *On khot's unique games conjecture.*, Bulletin (New Series) of the American Mathematical Society **49** (2012), no. 1.

Email address: `ehkim@cs.unc.edu`