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-Williams Algorithm for MAXCUT assuming the conjecture is true.

These notes borrow heavily from Arora and Barak's textbook, O'Donnell's textbook, and Prahladh Harsha's lecture notes on PCPs and Unique Games given at DIMACS and TIFR [1], [5], [6], [9].

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 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 $0 \le \rho_{\varphi,x} \le 1$ representing the fraction of clauses satisfied in φ. Take $\mathsf{Opt}(\varphi)$ to be the maximum such value over all such assignments:

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

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 $\beta \le 1$. To formalize this:

Definition 1.1. For $\beta \le 1$, 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:

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

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:

$$x_1 + x_2 + x_3 = 0$$
$$x_1 + x_2 + x_4 = 1$$
$$x_1 + x_3 + x_4 = 1$$

The $\mathsf{E3Lin}$ represents a linear system of $\mathsf{Exactly\ 3}$ 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. There exists an Semi-definite Programming (SDP) relaxation for MaxCut.

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 $\mathcal{I}$:

$$\mathsf{Val}_{\mathcal{I}}(F) = \mathbb{E}_{(S,\psi)\sim\mathcal{I}}[\psi(F(S))] \tag{1.2}$$

By taking the maximum fraction over all such assignments:

$$\mathsf{Opt}(\mathcal{I}) = \max_{F: V \to \Omega} \mathsf{Val}_{\mathcal{I}}(F) \tag{1.3}$$

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 $\mathcal{I}$:

$$\mathcal{I} \in \mathsf{YES} \iff \mathsf{Opt}(\mathcal{I}) \geq \beta \tag{1.4}$$

$$\mathcal{I} \in \mathsf{NO} \iff \mathsf{Opt}(\mathcal{I}) < \alpha \tag{1.5}$$

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. □

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:

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

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 constant 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 $\mathsf{NTIME}(2^{O(\log n)}) = \mathsf{NP}$. □

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, for every* NP *language L, there exists a polynomial-time reduction f mapping L to 3CNF formulas such that:*

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

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) For some constants $q \in \mathbb{N}, \alpha \in (0,1)$, 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 where Ψ consists of q-ary predicates, such that*

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

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)*

Proof. To be written... □

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 $\mathsf{MaxE3Lin}$ [7].

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:

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

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. **Raz's Parallel Repetition Theorem.** To be written...

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. Unique Games

4.1. **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 [8]. The instances of this relaxed version are called *Unique Games*:

Definition 4.1. A *Unique Label Cover Problem* with m labels (UniqueLC(m)) instance $\mathcal{U}$ is defined by a bipartite graph $(A \sqcup B, E)$, 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:

$$\pi_e(\sigma(a)) = \sigma(b) \tag{4.1}$$

The value of this game will be

$$\mathsf{Opt}(\mathcal{G}) = \max_\sigma \mathbb{E}_{e\sim E}[e \text{ satisfied}] \tag{4.2}$$

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:

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

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

Theorem 4.2. *(Unique Games Conjecture* [8]*) 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.*

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. Prahladh Harsha, *Limits of approximation algorithms: Pcps and unique games (tifr/imsc lecture notes, spring 2010)*, (2010).
6. 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).
7. Johan Håstad, *Some optimal inapproximability results*, Journal of the ACM (JACM) **48** (2001), no. 4, 798–859.
8. 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.

9. Ryan O'Donnell, *Analysis of boolean functions*, Cambridge University Press, 2014.

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