PROBABILISTIC CHECKING OF PROOFS AND HARDNESS OF APPROXIMATION

YOUNG KUN KO

Abstract. This paper introduces basic notions of classic computational complexity theory and probabilistically checkable proof. It sketches the proof of the PCP theorem, and discusses its applications to the hardness of approximation of various problems.

Contents

1. Introduction 1
1.1. Defining Classical Complexity Classes P and NP 1
1.2. NP-complete problems 2
1.3. **NP**and proof verifying 2
2. PCP Theorem 3
2.1. Definitions 3
2.2. Main Theorem 5
3. Hardness Results 6
3.1. Hardness of MAX3SAT 6
3.2. Hardness of SET-COVER 6
3.3. Unique Games Conjecture 10
4. Conclusion 10
Acknowledgments 11
References 12

1. Introduction

1.1. **Defining Classical Complexity Classes P and NP.** In Classical Complexity theory, determinism and non-determinism of Turing machines defines the following two different classes of problems.

Definition 1.1. The Class P is a set of languages whose membership problem can be deterministically decided in polynomial time

Definition 1.2. The Class NP is a set of languages whose membership problem can be non-deterministically decided in polynomial time

An equivalent definition would be that P is a set of problems whose solutions *can be found* in polynomial time, while NP is a set of problems whose solutions *can be checked* in polynomial time, since if the path that non-deterministic machine takes is given with the input, a deterministic machine can verify (in polynomial time) that the input is indeed a valid solution.

Example 1.3 (Shortest Path Problem)**.** Consider an undirected graph $G = (V, E)$ and $s, t \in V$. Then using Dijkstra's Algorithm, the shortest path between s and t can be found in $O(|E| + |V| \log |V|)$ time. Therefore it is in P.

Example 1.4 (Satisfiability Problem or SAT Problem)**.** Consider a boolean formula ϕ with varibles $x_1, x_2, \ldots x_n$. Then given an assignment to these varibles (i.e. just plugging in the values), one can check in polynomial time whether ϕ is satisfied or not satisfied. Therefore $\mathsf{SAT} \in NP$

For more explicit and concrete definitions for classical complexity theory such as the definition of Turing machine and equivalence of different computing models, refer to [6], which is an introductory textbook to this field. Since a non-deterministic machine can simulate deterministic machine, it is trivial that $P \subseteq NP$. However, the following still remains as a big conjecture in computer science.

Conjecture 1.5 (P vs. NP)**.** $P \neq NP$

1.2. **NP-complete problems.** Note that one can modify (or reduce) a problem to some other problem. If such reduction can be done in polynomial time, the reduction relation is denoted by $\leq_p$. With the notion of reduction, one can find the following set of languages, or problems.

Definition 1.6. A language L is NP-complete if

- $L \in NP$
- $L' \leq_p L$ $(\forall L' \in NP)$

A quite remarkable result is that such problem actually exists.

Theorem 1.7 (Cook-Levin Theorem)**.** SAT *is NP-complete*

For the full proof of the theorem, refer to [3]. One noteworthy characteristic of this proof is that it uses the locality of Turing machine computation, and you can write a boolean formula that accepts if and only if each step of the turing machine is valid. With the above result and the property of reduction, if a problem can reduced to an instance of SAT, that problem is also **NP**-complete. Following problems are known to be **NP**-complete

- 3-SAT
- VERTEX COVER
- INDEPENDENT SET
- CLIQUE
- SET COVER
- INTEGER PROGRAMMING

For full description of these problems, refer to [6]

1.3. **NPand proof verifying.** One noteworthy observation to make is that mathematical proof is **NP**. Consider a mathematical theorem and a proof. Verifier reads the written proof and accepts if the proof follows logical rules and rejects otherwise. If proof were written in n bits, the verifier reads the whole n bits and accept it if the proof is valid, and reject otherwise. In otherwords, **NP** is a set of problems such that if proof is written say in n bits, it can be determined in polynomial time with resepct to n whether the solution is valid. Probabilistic Checking of Proof starts from this view of **NP**.

2. PCP Theorem

2.1. Definitions.

Definition 2.1 (Algebraic Circuit)**.** An algebraic circuit C is a directed acyclic graph where each node computes $+, -$ or $\times$ of incoming edges. The output is carried over to the outgoing edges.

Definition 2.2 (CKT-SAT)**.** Given an algebraic circuit C, if C has a satisfying assignment then $C \in$ CKT-SAT The membership problem is denoted as CKT-SAT.

Definition 2.3. For functions $r, q : \mathbb{N} \to \mathbb{N}$, *a probabilistic polynomial-time verifier* V is $(r(n), q(n))$-restricted if, for every input of size n, it uses at most $r(n)$ random bits and examines at most $q(n)$ bits in the membership proof. A Language $L \in \mathbf{PCP}(r(n), q(n))$ if there exists a $(r(n), q(n))$-restricted polynomial-time verifier that, for every input x, behaves as follows:

Completeness: if $x \in L$, then there exists a membership proof π such that V accepts (x, π) with probability 1

Soundness: if $x \notin L$, then for any membership proof π, V accepts π with probability at most $1/2$

Under this definition, is not hard to see that $\mathbf{NP} = \mathbf{PCP}(0, n)$. Now we define outer verifier.

Definition 2.4 (Outer verifier)**.** For $r, p, c, a : \mathbb{N} \to \mathbb{N}$, an $(r(n), p(n), c(n), a(n))$ *outer verifier* is a randomized Turing machine V which expects the membership proof for an input of size n to be an oracle of answer size $a(n)$. Given an input $x \in \{0,1\}^n$ and an oracle π of answer size $a(n)$, V runs in poly(n) time and behaves as follows:

(1) Uses $r(n)$ random bits
(2) Constructs a circuit C of size $c(n)$
(3) Computes $p(n)$ locations in π, denoted $q_1, \ldots, q_{p(n)}$
(4) Queries the oracle: V queries the oracle π in locations $q_1, \ldots, q_{p(n)}$. Let a_i denote the string in q_i position.
(5) Makes a decision: V outputs **accept** if $a_1 \circ \ldots \circ a_{p(n)}$ is a satisfying assignment to circuit C. Otherwise **reject**. Denote the decision by $V^\pi(x; R)$

Definition 2.5 (**RPCP**)**.** For $r, p, c, a : \mathbb{N} \to \mathbb{N}$ and $e < 1$, a language $L \in \mathbf{RPCP}(r(n), p(n), c(n), a(n), e)$, if there exists a $(r(n), p(n), c(n), a(n))$ outer verifier that satisfies the following properties for every input x.

Completeness: If $x \in L$, then there exists an oracle π such that

$$\mathbf{Pr}_R[V^\pi(x; R) = \mathbf{accept}] = 1.$$

Soundness: IF $x \notin L$, then for all oracles π,

$$\mathbf{Pr}_R[V^\pi(x; R) = \mathbf{accept}] \leq e.$$

Similarly, we define inner verifier.

Definition 2.6. For $l, a \in \mathbb{N}$, let $\mathcal{F}_{l,a}$ denote the family of functions $\{f | f : [l] \to \{0,1\}^a\}$. i.e. $\mathcal{F}_{l,a}$ is a family of l-letter strings over the alphabet $\{0,1\}^a$.

Definition 2.7 (Valid Encoding/Decoding Scheme)**.** For functions $l, a : \mathbb{N} \to \mathbb{N}$, an $(l(n), a(n))$-encoding scheme is an ensemble of functions $E = \{E_n\}_{n\in\mathbb{N}}$, such that $E_n : \{0,1\}^n \to \mathcal{F}_{l(n),a(n)}$. An (l, a)-decoding scheme is an ensemble of functions $E^{-1} = \{E_n^{-1}\}_{n\in\mathbb{N}}$, such that $E_n^{-1} : \mathcal{F}_{l(n),a(n)} \to \{0,1\}^n$. An encoding/decoding scheme pair (E, E^{-1}) is valid if for all $x \in \{0,1\}^*$, $E_{|x|}^{-1}(E_{|x|}(x)) = x$.

Definition 2.8 (Inner verifier)**.** For functions, $r, p, c, a : \mathbb{N} \to \mathbb{N}$, positive integer k and fraction e, a $(k, r(n), p(n), c(n), a(n), e)$ *inner verifier system* is a triple (V, E, E^{-1}), where V is a probabilistic Turing machine and (E, E^{-1}) is a valid $(l(n), a(n))$-encoding/decoding scheme for some function $l : \mathbb{N} \to \mathbb{N}$. The input to the verifier is a circuit. The number of input nodes of the circuit is a multiple of k. Let C be this circuit, n be its size and km be the number of inputs to C. Then, V expects the membership proof to consist of $k + 1$ oracles $X_1, \ldots, X_{k+1}$, each of answer size $a(n)$. Here X_{k+1} is denoted π. Verifier runs in $poly(n)$ time and behaves as follows:

(1) Uses $r(n)$ random bits: V reads C and picks a random string $R \in \{0,1\}^{r(n)}$.
(2) Constructs a circuit of size $c(n)$: Based on C and R, V computes a circuit C' of size $c(n)$
(3) Computes $p(n)$ locations in the oracle : V computes $p(n)$ locations denoted $q_1, \ldots, q_{p(n)}$ in the oracles. Each q_i is a pair (q_i^1, q_i^2) where $q_i^1 \in [k + 1]$ denotes the oracle in which it is contained and q_i^2 denotes the position in oracle.
(4) Queries the oracle: V uses these $p(n)$ locations computed above to query the oracles. Let $a_1, \ldots, a_{p(n)}$ denote the strings received as answers, where each $a_i \in \{0,1\}^{a(n)}$
(5) Makes a decision: V outputs **accept** if $a_1 \circ \ldots \circ a_{p(n)}$ is a satisfying assignment to circuit C'. Otherwise **reject**. Denote the decision by $V^\pi(C; R)$s

Completeness: Let $x_1, \ldots, x_k$ be m-bit strings such that $x_1 \circ \ldots \circ x_k$ is a satisfying assignment for C. Then there exists an oracle π such that

$$\mathbf{Pr}_R[V^{X_1,\ldots,X_k,\pi}(C; R) = \mathbf{accept}] = 1,$$

where $X_j = E_m(x_j)$ for $1 \geq j \geq k$.

Soundness: If for some oracles $X_1, \ldots, X_k, \pi$, the probability

$$\mathbf{Pr}_R[V^{X_1,\ldots,X_k,\pi}(C; R) = \mathbf{accept}] \geq e,$$

then $E_m^{-1}(X_1) \circ \ldots \circ E_m^{-1}(X_k)$ is a satisfying assignment for C.

Following relationship holds between outer verifier and inner verifier

Proposition 2.9. *If there exists a $(k, r(n), p(n), c(n), a(n), e)$-inner verifier system, then CKT-SAT $\in$ $\mathbf{RPCP}(r(n), p(n), c(n), a(n), e)$.*

Proof. Recall that inner verifier system uses $r(n)$ random bits, and expects the proof to be an oracle of answer size $a(n)$. If we ignore the special structure of $X_1, \ldots, X_k$ and only consider π as the proof and all queries are on π, circuit size is preserved, thus it is also an $(r(n), p(n), c(n), a(n))$-outer verifier. By definition, it can check membership proofs for CKT-SAT, as it satisfies both completeness and soundness condition. Thus CKT-SAT $\in$ $\mathbf{RPCP}(r(n), p(n), c(n), a(n), e)$. □

2.2. **Main Theorem.**

Theorem 2.10 (PCP theorem)**.** [1]. *There exists a positive integer q such that*

$$\mathbf{NP} = \bigcup_{c>0} \mathbf{PCP}(c \log n, q)$$

Following three theorems will be stated and used without the full proof, since it requires technical details.

Theorem 2.11 (Composition lemma)**.** [2] *Suppose $(p, r_1(n), p_1(n), c_1(n), a_1(n), e_1)$-inner verifier system exists with $0 < e_1 < 1$ and valid $p \in \mathbb{N}$. Then, for all functions $r, c, a : \mathbb{N} \to \mathbb{N}$ and every positive fraction e,*

$$\mathbf{RPCP}(r(n), p, c(n), a(n), e) \subseteq \mathbf{RPCP}(r(n)+r_1(\tau), p_1(\tau), a_1(\tau), c_1(\tau), e+e_1-ee_1),$$

where $\tau = c(n)$

Theorem 2.12. *For every constant $k \in \mathbb{N}$, there exist constants $c_1, c_2, c_3, p \in \mathbb{N}$ and real number $e < 1$, such that for some functions $c(n) = O(\log^{c_2} n)$ and $a(n) = O(\log^{c_3} n)$, there exists a $(k, c_1 \log n, p, c(n), a(n), e)$ inner verifier system.*

Theorem 2.13. *For every constant $k \in \mathbb{N}$, there exist constants $c_1, p \in \mathbb{N}$ and a positive real number $e < 1$, such that there exists a $(k, c_1 n^2, p, 2^p, 1, e)$ inner verifier system.*

Proposition 2.14. *If a $(k, r(n), p(n), c(n), a(n), e)$ inner verifier system exists, then for every positive integer l, a $(k, lr(n), lp(n), lc(n), la(n), e^l)$ inner verifier system exists.*

Proof. Repeat inner verifier l times and accept the input only if it is accepted every time. □

Theorem 2.15. *There exists a constant C such that for every language $L \in \mathbf{NP}$, there exists a constant c_L such that*

$$L \in \mathbf{RPCP}\left(c_L \log n, C, 2^C, 1, \frac{1}{2}\right).$$

Proof. It suffices to show that CKT-SAT $\in \mathbf{RPCP}\left(c_L \log n, C, 2^C, 1, 1/2\right)$ for some constants c_L, C. First, use theorem 2.12 with $k = 1$, and reduce the error to $1/16$ using proposition 2.14. This gives $(1, c_1 \log n, p, c(n), a(n), 1/16)$ inner verifier system where $a(n) = \log^{d_1} n$ and $c(n) = \log^{d_2} n$ for some constants d_1, d_2, c_1, p. Due to proposition 2.9,

$$\text{CKT-SAT} \in \mathbf{RPCP}\left(c_1 \log n, p, \log^{d_2} n, \log^{d_1} n, \frac{1}{16}\right). \tag{1}$$

Apply the same argument with constant $k = 1$ to the inner verifier in theorem 2.12, gives $(p, c' \log n, c'', \log^{d'} n, \log^{d''} n, 1/16)$ inner verifier system. Now apply 1 and to the new inner verifier. Then we can obtain

(2)

$$\text{CKT-SAT} \in \mathbf{RPCP}\left(c_1 \log n + c' d_1 \log\log n, c'', (d_1 \log\log n)^{d'}, (d_1 \log\log n)^{d'}, \frac{2}{16}\right).$$

Now use c'' as k in inner verifier constructed in theorem 2.13 and amplify using proposition 2.14, giving $(c'', gn^2, h, 2^h, 1, 1/16)$ inner verifier system.

Apply $(c'', gn^2, h, 2^h, 1, 1/16)$ to 2 using theorem 2.11. This gives

$$\text{(3)}\quad \text{CKT-SAT} \in \mathbf{RPCP}\left(c_1 \log n + c' d \log\log n + g(d\log\log n)^{2d'}, h, 2^h, 1, \frac{3}{16}\right).$$

since $c'd\log\log n + g(d\log\log n)^{2d'} = o(\log n)$ and h, c_1 are constants, this proves the theorem. □

3. Hardness Results

3.1. **Hardness of MAX3SAT.** It is not immediate that the above result implies hardness of approximation results. But the following theorem provides a link between those two.

Definition 3.1 (MAX3SAT)**.** Given a $3SAT$ formula ϕ, find the maximum fraction of clauses that can be satisfied by assignments, denoted by $\mathsf{OPT}(\phi)$

Theorem 3.2. *If* $\mathbf{NP} = \mathbf{PCP}(O(\log n), O(1))$*, it is* $\mathbf{NP}$*-hard to approximate MAX-3SAT within a factor* $1-\epsilon$ *for some* $\epsilon > 0$

Proof. We reduce a language in $\mathbf{NP}$with V a $(c_L \log n, q)$ restricted verifier to a 3SATformula ϕ_x with at most $q2^q 2^{c_L \log n}$ clauses with the following property.

YES: If $x \in L$, then $\mathsf{OPT}(\phi_x) = 1$
NO: If $x \notin L$, then $\mathsf{OPT}(\phi_x) < 1 - \frac{1}{q2^{q+1}}$

For every query Q_i to the oracle π, associate a boolean variable v_i. Now fix a random string $R \in \{0,1\}^{c_L \log n}$ and thus fix queries $Q_{i_1}, \ldots, Q_{i_q}$ using R. Construct function $\psi_{x,R} : \{0,1\}^q \to \{0,1\}$

$$\psi_{x,R}(a_{i_1}, \ldots, a_{i_q}) = \begin{cases} 1 & \text{iff } V \text{ accepts on } a_{i_1}, \ldots, a_{i_q} \\ 0 & \text{otherwise} \end{cases}$$

Express $psi_{x,R}$ as a CNF formula with at most 2^q clauses, each of length at most q. Then convert this to 3-CNF formula using upto q auxiliary variables. Let $\phi_{x,R}$ denote this 3-CNF formula, then

$$\text{(4)}\qquad \phi_x = \bigwedge_R \phi_{x,R}$$

since there are $2^{c_L \log n}$ choice of R, then the number of clauses in ϕ_x is at most $m = q2^q 2^{c_L \log n}$
Now if $x \in L$, $\psi_{x,R}$ is satisfied for every R. Thus $\mathsf{OPT}(\phi_x) = 1$.
Now suppose $x \notin L$. Suppose that $\mathsf{OPT}(\phi_x) \geq 1 - \frac{1}{2^{q+1}}$, thus having at most ϵm clauses unsatisfied, where $\epsilon = 1/q2^{(q+1)}$. But then the number of R's such that $\phi_{x,R}$ is not satisfied is at most ϵm. Thus the probability that V accepts π is at least $1/2$, implying $x \in L$. Contradiction. Thus $\mathsf{OPT}(\phi_x) < (1-\epsilon)$.
If there exists an algorithm that distinguishes between YES and NO instance, this distinguishes between $x \in L$ and $x \notin L$. Thus no algorithm can approximate MAX3SAT within $1-\epsilon$ factor in polynomial time, unless $\mathbf{P} = \mathbf{NP}$ □

3.2. **Hardness of SET-COVER.** Using the result, we can show hardness result for various problems, such as SET-COVER. First, we introduce the gap version of problems.

Definition 3.3 (GAP-Problems)**.** Given Problem $\mathcal{P}$, its gap version $Gap - \mathcal{P}_{g(n)}$ is following problem such that for some function $h(n)$,

- YES instance: instances I of $\mathcal{P}$ such that $\mathsf{OPT}(I) \leq h(n)$
- NO instance: instances I of $\mathcal{P}$ such that $\mathsf{OPT}(I) \leq g(n)h(n)$. The function $g(n) \leq 1$ is called the *gap*.

Definition 3.4 (LABEL-COVER)**.** An instance of LABEL COVER is denoted by $\mathcal{L}(G(V, W, E), [M], [N], \{\pi_{vw}\}_{(v,w)\in E})$ where

- $G(V, W, E)$ is a regular bipartite graph
- $[M], [N]$ are sets of labels, and V, W get labels from $[N], [M]$ respectively.
- $\{\pi_{vw}\}_{(v,w)\in E}$ denote the constraints on each edge. For every edge (v, w) in E, we have a map $\pi_{vw} : [M] \to [N]$

A labeling $l : V \to [N], W \to [M]$ satisfies (the constraint on) an edge (v, w) if $\pi_{vw}(l(w)) = l(v)$. Given an instance $\mathcal{L}$, find a labeling that satisfies the maximum fraction, denoted by $\mathsf{OPT}(\mathcal{L})$ of edges.

Definition 3.5 (SET-COVER)**.** Given elements $[n]$ and sets $S_1, S_2, \ldots S_m \subseteq [n]$ such that $\bigcup_i S_i = [n]$. Find minimum t and sets $S_{i_1}, \ldots, S_{i_t}$ such that $\bigcup_{k=1}^{t} S_{i_k} = [n]$

Definition 3.6 (MAX3SAT(5))**.** Given a $3SAT$ formula with the restriction that each variable is in exactly 5 clauses, find the maximum fraction of clauses that can be satisfied by assignments.

Note by the PCP theorem, GAP-MAX3SAT(5) is **NP**-hard. Equivalently, it is **NP**-hard to tell whether formula ϕ is

YES: $\mathsf{OPT}(\phi) = 1$
NO: $\mathsf{OPT}(\phi) \leq c < 1$ for some constant c

Now, we reduce GAP-MAX3SAT(5) to an instance of LABEL-COVER.

Theorem 3.7. *It is **NP**-hard to tell whether a label cover instance $\mathcal{L}$ is*

YES: $\mathsf{OPT}(\mathcal{L}) = 1$
NO: $\mathsf{OPT}(\mathcal{L}) \leq c' < 1$ *for some constant* c'

where $M = 7, N = 2$

Proof. Given an instance ϕ of MAX3SAT(5), construct an instance $\mathcal{L}_\phi$ of LABEL COVER with the following property

V: For each variable x_i, make corresponding vertex in V
W: For each clause C_j, make corresponding vertex in W
E: $(x_i, C_j) \in E$ if and only if $x_i \in C_j$

Note that by the property of MAX3SAT(5), $deg(v) = 5$ $(\forall v \in V)$ and $deg(w) = 3$ $(\forall w \in W)$. Since V gets label from [2], let the labels correspond to boolean value $\{0, 1\}$. Since W gets label from [7], let the labels correspond to 7 satisfying assignment of the clause. Then set $\pi_{x_i,C_j} : [7] \to [2]$ be the assignment of variable x_i by the satisfying assignment.

Claim: If $\mathsf{OPT}(\phi) = 1$ then $\mathsf{OPT}(\mathcal{L}_\phi) = 1$. If $\mathsf{OPT}(\phi) \leq c$ then $\mathsf{OPT}(\mathcal{L}_\phi) \leq 1 - \frac{1-c}{3}$

If $\mathsf{OPT}(\phi) = 1$ then $\mathsf{OPT}(\mathcal{L}_\phi) = 1$, all labels on edges will be satisfied.
Now, suppose $\mathsf{OPT}(\phi) \leq c$. Consider labeling l corresponding to some assignment. Then at least $1 - c$ fraction of clauses remain unsatisfied under this assignment. Let $C_j = x_a \vee x_b \vee x_c$ be such clause. Since it is an unsatisfied clause, one of three

edges, $(x_a, C_j), (x_b, C_j), (x_c, C_j)$ will have invalid label. Thus at least $\frac{1-c}{3}$ of the edges are violated. Hence $\mathsf{OPT}(\mathcal{L}_\phi) \leq 1 - \frac{1-c}{3}$. □

Definition 3.8 (Graph Product)**.** Given a bipartite graph $G = (V, W, E)$, define $G' = (V', W', E')$ as k-product of G where

- $V' := V^k$
- $W' := W^k$
- $(v', w') \in E'$ iff $(v_i, w_i) \in E \ \forall 1 \leq i \leq k$ where $v' = (v_1, \ldots, v_k), w' = (w_1, \ldots, w_k)$

Using the definition of Graph Product, we can define

$$\mathcal{L}' = \mathcal{L}^{\otimes k}(G'(V', W', E'), [M'], [N'], \{\pi'_{v'w'}\})$$

where

- $[M'] := [M]^k$, $[N'] := [N]^k$
- $\pi'_{v'w'}(b_1, \ldots, b_k) := (\pi_{v_1 w_1}(b_1), \ldots, \pi_{v_k w_k}(b_k))$

Now following theorem can be used to amplify the gap.

Theorem 3.9 (Raz's Parallel Repetition Theorem)**.** *If* $\mathsf{OPT}(\mathcal{L}) \leq c < 1$*, then there exists a* $c' < 1$ *depending only on* c, M, N *such that* $\mathsf{OPT}(\mathcal{L}^{\otimes k}) \leq c'^k$ [5]

The proof of this theorem is beyond the scope of this paper.

Corollary 3.10. *There is a reduction from SAT to a gap version of LABEL COVER* $\mathcal{L}$ *such that*

YES: $\mathsf{OPT}(\mathcal{L}) = 1$
NO: $\mathsf{OPT}(\mathcal{L}) \leq 2^{-\gamma k}$ *for some constant* $\gamma > 0$

where $M = 7^k, N = 2^k$ *and* $|V|, |W| = n^{O(k)}$

Proof. This is the direct application of Theorem 3.7 and Theorem 3.9 □

Now it remains to reduce the above instance to a SET-COVER instance. But we first define a partition system and pseudolabeling.

Definition 3.11 (Partition System)**.** A partition system $P(\mathcal{U}, m, h, t)$ is the universe set $\mathcal{U} = [m]$ and t pairs of sets $(A_1, \bar{A}_1), \ldots, (A_t, \bar{A}_t)$ where $A_i \subseteq [m]$ such that for any indices $i_1 < \ldots < i_h$, $A^*_{i_1} \cup \ldots \cup A^*_{i_h} \neq [m]$ where A^*_i is either A_i or $\bar{A}_i$

Definition 3.12 (Pseudolabeling)**.** Given a label cover instance $\mathcal{L}$, an assignment $v \to T_v, w \to T_w$ for all $v \in V, w \in W$ where $T_v \subseteq [N], T_w \subseteq [M]$ is called (δ, h')-Pseudolabeling if for at least δ fraction of edges of $\mathcal{L}$, $|T_v|, |T_w| \leq h'$ and $\pi_{vw}(T_w) \cup T_v \neq \emptyset$

Lemma 3.13. *Given* h, t*, there exists a partition system with* $m = O(2^h h \log t)$*. This system can be constructed with high probability*

Proof. Pick random collection of sets $A_1, \ldots, A_t$ from $2^{\mathcal{U}}$. Fix h of the sets, say $A_{i_1}, \ldots, A_{i_h}$ Then

$$\mathbf{Pr}[\bigcup_{k=1}^{h} A_{i_k} = \mathcal{U}] = \mathbf{Pr}[x \in \bigcup_{k=1}^{h} A_{i_k}]^m = \left(1 - \frac{1}{2^h}\right)^m \tag{5}$$

Note that there are $\binom{t}{h}2^h$ ways of fixing h sets. By union bound,

$$\mathbf{Pr}[(A_1,\ldots,A_t)\text{does not form a partitioning system}] \leq \binom{t}{h}2^h\left(1-\frac{1}{2^h}\right)^m \tag{6}$$

w.h.p, we have a partition system. □

Lemma 3.14. *If a label cover instance $\mathcal{L}$ has a (δ,h')-Pseudolabeling, then there is a labeling which satisfies more than $\frac{\delta}{h'^2}$ fraction of the edges and thus $\mathsf{OPT}(\mathcal{L}) \geq \frac{\delta}{h'^2}$*

Proof. Given any pseudolabeling, for a vertex v, pick an element from T_v and let it be the label. By the definition of pseudolabeling, for at least δ fraction of edges (v,w), at least one pair from $T_v \times T_w$ is consistent. An edge is satisfied with probability $\frac{\delta}{h'^2}$. Thus the expected fraction of satisfied edges is $\frac{\delta}{h'^2}$. Thus $\mathsf{OPT}(\mathcal{L}) \geq \frac{\delta}{h'^2}$. □

Now we define a SET-COVER instance $SC_{\mathcal{L}}$.

Definition 3.15. $SC_{\mathcal{L}}$ is

- For each $(v,w) \in E$ define $P_{vw}(\mathcal{U}_{vw}, m, h, t = N = 2^k)$ as constructed in lemma 3.13. Define $\mathcal{U} = \bigcup \mathcal{U}_{vw}$
- Note that for each partition system, we constructed sets. Construct following set of sets.

$$S_{v,i} := \bigcup_{w:(v,w)\in E} A_i^{vw}$$

$$S_{w,j} := \bigcup_{w:(v,w)\in E} \bar{A}^{vw}_{\pi_{vw}(j)}$$

  where A_i^{vw} is the set created in P_{vw}

Proposition 3.16 (Completeness)**.** *If $\mathsf{OPT}(\mathcal{L}) = 1$ then $SC_{\mathcal{L}}$ has a cover of size $|V| + |W|$*

Proof. Let $l : V \to [N], W \to [M]$ be the correct labeling. Thus for all $(v,w) \in E$, $\pi_{vw}(l(w)) = l(v)$. Then $\{S_{v,l(v)}, S_{w,l(w)} : v \in V, w \in W\}$ cover the universe. □

Proposition 3.17 (Soundness)**.** *If $SC_{\mathcal{L}}$ has a set cover of size $\rho h|W|$, then there is a $(1-4\rho,h)$-Pseudolabeling of $\mathcal{L}$*

Proof. Suppose there is a set cover P of size $\rho h|W|$. For every v, set $T_v := \{i : S_v, i \in P\}$. Similarly, set $T_w := \{j : S_w, j \in P\}$. Disregard the vertices for which $|T_v| \geq h/2$. Since there are $\rho h|W|$ sets in all, we would be disregarding at most 2ρ fraction of the vertices. And more than $1-4\rho$ fraction of pairs (v,w) has $|T_v|, |T_w| \leq h/2$
Fix one such pair. Consider the sets $\{S_{v,i} : i \in T_v\}$ and $\{S_{w,j} : j \in T_w\}$. They cover $\mathcal{U}_v w$. Thus, $\{A_i^{vw} : i \in T_v\}$ and $\{\bar{A}_j^{vw} : j \in T_w\}$ Note that there are at most h sets. By the property of the partition system,there must be a complementary pair, which implies that there is $i \in T_v$ and $j \in T_w$ such that $\pi(j) = i$. Thus the assignment was a $(1-4\rho,h)$-pseudolabeling. □

Now we prove the main theorem.

Theorem 3.18. *It is hard to approximate SET-COVER within factor $\Omega(\log n)$*

Proof. By lemma 3.14, we know that $\mathsf{OPT}(SC_{\mathcal{L}}) \geq (1-4\rho)/h^2$. Choose $\rho = 1/8$. If $h = O(2^{\gamma k/2})$, $\mathsf{OPT}(\mathcal{L}) \leq 2^{-\gamma k}$ implies that size of set cover for $SC_{\mathcal{L}}$ is at least $h|W|/8$, an $\Omega(h)$ hardness result.
Set k so that $N' = n^{O(k)} \approx 2^h O(hk)$. (Set $k = O(\log\log n)$) The size of the set cover instance is $n^{O(k)}2^h O(hk) \approx N'^2$ Thus $\Omega(h) = \Omega(\log N')$. □

3.3. **Unique Games Conjecture.** Here we conclude the section by stating Unique Games Conjecture, which was proposed to obtain better hardness results.

Definition 3.19 (Unique Game)**.** A Unique Game $\mathcal{U}(G(V,E),[n],\{\pi_e|e\in E\})$ is a constraint satisfaction problem defined as follows: $G(V,E)$ is a directed graph whose vertices represent variables and edges represent constraints. The goal is to assign to each vertex a label from the set $[n]$. The constraint on an edge $e = (v,w) \in E$ is described by a bijection $\pi_e : [n] \to [n]$. A labeling $L : V \to [n]$ satisfies the constraint on edge $e = (v,w)$ iff $\pi_e(L(v)) = L(w)$. Let $\mathsf{OPT}(\mathcal{U})$ denote the maximum fraction of constraints that can be satisfied by any labeling:

$$\mathsf{OPT}(\mathcal{U}) := \max_{L:V\to[n]} \frac{1}{|E|}|\{e \in E | L \text{ satisfies } e\}|.$$

Note the similarity between label cover problem. In fact all unique games can be reduced to label cover problem.

Conjecture 3.20 (Unique Games Conjecture)**.** *For every $\epsilon, \delta > 0$, there exists a constant $n = n(\epsilon,\delta)$, such that given a Unique Game instance is NP-hard to distinguish between two cases:*

- YES *Case:* $\mathsf{OPT}(\mathcal{U}) \geq 1-\epsilon$.
- NO *Case:* $\mathsf{OPT}(\mathcal{U}) \leq \delta$.

Due to the reducibility of label cover problem, Conjecture 3.20 is equivalent to numerous conjectures. A survey [4] summarizes the inapproximability results known under Unique Games Conjecture. One notable result is a reduction from Unique Game to Min-2SAT-Deletion, which will only be stated.

Definition 3.21 (Min-2SAT-Deletion)**.** Given a 2SATformula ϕ, find the minimum fraction of clauses that need to be deleted from ϕ to make it satisfiable. Such fraction is denoted as $\mathsf{OPT}(\phi)$

Theorem 3.22. *Let $1/2 < c < 1$ be a fixed constant. Then, for every sufficiently small constant $\epsilon > 0$, there exists a constant $\delta > 0$ and a polynomial time reduction that maps an instance $\mathcal{U}(G(V,E),[n],\{\pi_e|e\in E\})$ to a 2SATformula ϕ such that*

- YES *Case: If* $\mathsf{OPT}(\mathcal{U}) \geq 1-\epsilon$ *then* $\mathsf{OPT}(\phi) \leq O(\epsilon)$
- NO *Case: If* $\mathsf{OPT}(\mathcal{U}) \leq \delta$ *then* $\mathsf{OPT}(\phi) \geq \Omega(\epsilon^c)$

If such reduction exists, it is **NP**-hard to distinguish between those two instances. The proof requires advanced techniques from boolean function analysis thus will not be stated.

4. Conclusion

The PCP theorem states that we only need to check constant bits to check whether the proof is valid or invalid. This led to some interesting results and brought a huge insight to the field of hardness of approximation. Unique Games

Conjecture, a big step in the field of hardness of approximation, still remains unsolved but led to some interesting implications. Various open problems remain in this field and the results leading up to these problems are promising.

Acknowledgments. It is a pleasure to thank my mentor, Matthew Wright, and Professor Subhash Khot who introduced me to the field of Probabilistic Checking of Proofs and related fields.

References

[1] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and the hardness of approximation problems. *J. ACM*, 45(3):501–555, May 1998.

[2] Sanjeev Arora and Shmuel Safra. Probabilistic checking of proofs: a new characterization of np. *J. ACM*, 45(1):70–122, January 1998.

[3] Stephen A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the third annual ACM symposium on Theory of computing*, STOC '71, pages 151–158, New York, NY, USA, 1971. ACM.

[4] Subhash Khot. On the unique games conjecture (invited survey). In *IEEE Conference on Computational Complexity*, pages 99–121, 2010.

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

[6] M. Sipser. *Introduction To The Theory Of Computation*. Computer Science Series. Thomson Course Technology, 2006.