PCP Theorem And Hardness Of Approximation
For MAX-SATISFY Over Finite Fields

Shangwei Zhao
Key Laboratory of Mathematics Mechanization
Institute of Systems Science, AMSS, Academia Sinica, Beijing, 100080, China

Abstract. This paper will survey the PCP theorem and its application in solving the MAX-SATISFY. For the problem MAX-SATISFY over $F_2$, a random assignment satisfies half the equations and thus yields a 2-approximation algorithm. It is the best possible approximation algorithm in polynomial time based on $P \neq NP$, even if each equation only contains exactly three variables, which can be proved by the 3-bit PCP theorem due to Johan Håstad. Similar results can be extended to MAX-SATISFY over finite fields.

1. Introduction

Consider the problem MAX-SATISFY over $F_2$, that is, given a system of $m$ linear equations with $n$ variables over $F_2$, find a solution satisfying the maximal number of equations. This is an optimization problem and is NP complete, which shows that it is hard to find its exact solutions unless $P = NP$. Hence, it is meaningful to consider the approximation algorithms of this problem.

Since for each equation, the probability of a random assignment satisfying the equation is $1/2$, a random assignment for the system of equations will satisfies half of the equations. Therefore, we can get an approximation algorithm with approximation ratio 2.

The PCP theorem is one of the most important results in complexity in recent years and it provides a tool to prove the inapproximability result of optimization problems [2, 1]. In this paper, we will introduce the basic theory of PCP and its application to the inapproximability results.

Based on the basic PCP theorem, Johan Håstad gives a 3-bit PCP theorem, which can be used to prove that the 2-approximation algorithm is the best possible polynomial time approximation algorithm for MAX-SATISFY over $F_2$ unless $P = NP$ [3]. Precisely, he proves that it is NP-hard to approximate MAX-SATISFY over $F_2$. He also shows that it is NP-hard to approximate MAX-SATISFY over $F_p$ with approximation ratio $p - \epsilon$ for arbitrary $\epsilon > 0$.

We will give a brief introduction to these results.

2. Basic definitions

First, we will give some definitions about approximation algorithms of optimization problems.

Definition 2.1 Let $O$ be a maximization problem and let $r \geq 1$ be a real number. For an instance $x$ of $O$ let $OPT(x)$ be the optimal value. An $r$-approximation algorithm is an
algorithm that on each input $x$ outputs a number $\text{cost}(x,s)$ such that $\text{OPT}(x)/r \leq \text{cost}(x,s) \leq \text{OPT}(x)$, where $s$ is a feasible solution.

**Definition 2.2** Let $O$ be a minimization problem and let $r \geq 1$ be a real number. For an instance $x$ of $O$ let $\text{OPT}(x)$ be the optimal value. An $r$-approximation algorithm is an algorithm that on each input $x$ outputs a number $\text{cost}(x,s)$ such that $\text{OPT}(x) \leq \text{cost}(x,s) \leq r \cdot \text{OPT}(x)$, where $s$ is a feasible solution.

**Definition 2.3** An efficient $r$-approximation algorithm is an $r$-approximation algorithm that runs in worst-case polynomial time.

We can also use the formulation “having performance ratio $r$” or “approximation ratio $r$” instead of saying “being an $r$-approximation algorithm”.

**Definition 2.4** An optimization problem $O$ is said to have a polynomial-time approximation scheme, if there exists a variety of approximation algorithms $\{A_k\}$ such that for every fixed $k \geq 1$, $A_k$ is a $(1 + \frac{1}{k})$-approximation polynomial-time algorithm of $O$.

Further, if there is a common polynomial $p(n,k)$ such that for each $x$, the time of $A_k(x)$ is bounded by $p(n,k)$, where $n = |x|$, the problem is called to have fully polynomial approximation scheme.

**Definition 2.5** An optimization problem $O$ is said to be hard to approximate within a factor of $r$ if the existence of an $r$-approximation algorithm for $O$ implies $P=NP$.

Usually, we can divide the approximation results into three classes:

1. For any constant $r > 1$, it is NP-hard to approximate $O$ with approximation ratio $r$.
2. There is a constant $r_0 > 1$ such that for $r > r_0$, $O$ has a polynomial-time $r$-approximation algorithm while for $1 < r < r_0$, it is NP-hard to approximate $O$ with approximation ratio $r$.
3. For any constant $r > 1$, $O$ has a polynomial-time $r$-approximation algorithm.

The PCP theorem in the next section is widely used in the research of the second class.

3. PCP Theorem

3.1. PCP Theorem

PCP is short for Probabilistically Checkable Proofs, and the PCP Theorem provides an interesting new characterization for NP.

Let a verifier be a polynomial-time probabilistic Turing machine containing an input tape, a work tape, a tape that contains a random string, and a tape called the proof string denoted by $\pi$. The proof string should be thought of as an array of bits, out of which the verifier will examine a few. The verifier works as follows. First, it reads the input and the random string, and writes down on its work tape some addresses of locations in the proof string $\pi$. Next, it examines the bits in those locations in $\pi$. The process of reading a bit from $\pi$ is called a query. Finally, the verifier decides to accept or reject, based on what the input, the random string, and the queried bits of $\pi$ were.
Definition 3.1 A verifier is \((r(n), q(n))-\)restricted if on each input of size \(n\) it uses at most \(O(r(n))\) random bits for its computation, and queries at most \(O(q(n))\) bits of the proof.

In other words, an \((r(n), q(n))-\)restricted verifier has two associated integers \(c, k\). The random string has length \(c \cdot r(n)\). The verifier operates as follows on an input of size \(n\). It reads the random string \(R\), computes a sequence of \(k \cdot q(n)\) locations \(i_1(R), i_2(R), \ldots, i_{k \cdot q(n)}(R)\), and queries those locations in \(\pi\). Depending on what these bits were, it accepts or rejects.

Define \(M^\pi(x, R)\) to be 1 if \(M\) accepts input \(x\), with access to the proof \(\pi\), using a string of random bits \(R\), and 0 otherwise.

Definition 3.2 A verifier \(M\) probabilistically checks membership proofs for language \(L\) if

1. (completeness) For every input \(x \in L\), there is a proof \(\pi_x\) that causes \(M\) to accepts for every random string, that is, \(\exists \pi \Pr_R[M^\pi(x, R) = 1] = 1\).

2. (soundness) For any input \(x \notin L\), every proof \(\pi\) is rejected with probability at least \(\frac{1}{2}\), that is, \(\forall \pi, \Pr_R[M^\pi(x, R) = 1] \leq \frac{1}{2}\).

Definition 3.3 A language \(L\) is in \(\text{PCP}(r(n), q(n))\) if there is an \((r(n), q(n))-\)restricted verifier \(M\) that probabilistically checks membership proofs for \(L\).

According to the usual definition, the complexity class \(\text{NP}\) is the class of languages that can be verified by a deterministic polynomial-time Turing machine \(M\) (called "verifier"). That is,

- \(x \in L \Rightarrow \exists \pi \text{s.t.} M^\pi(x, \pi) = 1\);
- \(x \notin L \Rightarrow \forall \pi \text{s.t.} M^\pi(x, \pi) = 0\).

Theorem 3.4 (PCP) \(\text{NP} = \text{PCP}(\log n, 1)\)

The theorem gives a novel characterization of \(\text{NP}[2]\). Its proof is complicated, and we will not give it here.

The soundness of the PCP verifier in definition 2.2 is \(1/2\), which can be reduced to an arbitrary small positive constant by increasing the number of queries. If the original soundness is \(1 - \delta\) with \(q\) queries, then we can obtain soundness \((1 - \delta)^l\) by making \(lq\) queries. However, we need small soundness with small number of queries sometimes. For example, in the following Theorem 4.2, we need only three queries to the proof string. The above verifier called sequential verifier does not work. Raz has shown that it can be achieved by parallel repetition of PCPs at the expanse of increasing the size of the alphabet of proof string.

Definition 3.5 Let \(S\) be a finite set. Let \(V\) be a PCP verifier using alphabet \(S\) and \(l \in N\). The \(l\)-times parallel repeated \(V\) is the verifier \(M\) as follows:

1. \(M\) uses the alphabet \(S^l\) and we denote the input proof string of \(M\) by \(\hat{\pi}\).

2. Let \(q\) denote the number of queries \(V\) makes. On input \(x\), \(M\) chooses \(l\) independent random strings \(r^1, \ldots, r^l\) for \(V\), and runs \(V\) on the input and the random strings to obtain \(l\)
sets of \( q \) queries as follows:

\[
i_1^1, i_2^1, \ldots, i_q^1
\]
\[
i_1^2, i_2^2, \ldots, i_q^2
\]
\[
\ldots
\]
\[
i_1^l, i_2^l, \ldots, i_q^l.
\]

3. \( M \) makes \( q \) queries \( i_1, \ldots, i_q \) to the proof string \( \hat{\pi} \) where \( i_j \) is \((i_j^1, \ldots, i_j^l)\).

4. For \( j \in [q] \), denote \((a_j^1, \ldots, a_j^l) = \hat{\pi}(i_j)\). The verifier \( M \) accepts if and only if for every \( k \in [l] \), the verifier \( V \) on random string \( r_k \) accepts when given the responses \( a_1^k, \ldots, a_q^k \).

In the following text, we denote the above verifier \( V_{RAZ} \).

From the definition we can see that the parallel verifier accepts with probability one when the original always accepts. Raz gives the following theorem about the soundness parameter.

**Theorem 3.6** There exist constants \( a \) and \( b \) such that the soundness parameter of \( M \) is at most \((1 - \delta^a)^b\).

### 3.2. The equivalent formulation of PCP Theorem

**Definition 3.7** Let \( q, W \) be natural numbers. A \( q\text{-CSP}_W \) instance \( \varphi \) is a collection of functions \( \varphi_1, \ldots, \varphi_m \) (called constraints) from \( \{0, \ldots, W - 1\}^n \) to \( \{0, 1\} \) such that each function \( \varphi_i \) depends on at most \( q \) of its input locations. That is, for every \( i \in [m] \), there exist \( j_1, \ldots, j_q \in [n] \) and \( f: \{0, \ldots, W - 1\}^q \rightarrow \{0, 1\} \) such that \( \varphi_i(u) = f(u_{j_1}, \ldots, u_{j_q}) \) for every \( u \in \{0, \ldots, W - 1\}^n \), where \( [m] = \{1, 2, \ldots, m\} \) and \( [n] = \{1, 2, \ldots, n\} \).

We say that an assignment \( u \in \{0, \ldots, W - 1\}^n \) satisfies constraint \( \varphi_i \) if \( \varphi_i(u) = 1 \). The fraction of constraints satisfied by \( u \) is \( \frac{\sum_{i=1}^m \varphi_i(u)}{m} \), and we let \( \text{opt}( \varphi ) \) denote the maximum of this value over all \( u \in \{0, \ldots, W - 1\}^n \). We say that \( \varphi \) is satisfiable if \( \text{opt}( \varphi ) = 1 \).

We call \( q \) the arity of \( \varphi \) and \( W \) the alphabet size. If \( W = 2 \) we call \( \varphi \) a \( q\text{-CSP}-instance \).

It is easy to see that 3SAT is a special case of constraint satisfaction problems.

**Definition 3.8** Let \( \rho \in (0, 1) \), the \( \rho\text{-GAPqCSP}_W \) problem is to determine, given an instance \( \varphi \) of \( q\text{-CSP}_W \) whether:

1. \( \text{opt}(\varphi) = 1 \), in which case we say \( \varphi \) is a "yes" instance of \( \rho\text{-GAPqCSP}_W \).
2. \( \text{opt}(\varphi) \leq \rho \), in which case we say \( \varphi \) is a "no" instance of \( \rho\text{-GAPqCSP}_W \).

Also, we can define the problem \( \rho\text{-GAP3SAT} \). It is a special case of \( \rho\text{-GAP3CSP} \).

**Definition 3.9** Let \( \rho \in (0, 1) \) and \( L \) be a language. If there is a polynomial-time computable function \( f \) such that

\( x \in L \Rightarrow f(x) \) is a "yes" instance of \( \rho\text{-GAPqCSP}_W \);
\( x \notin L \Rightarrow f(x) \) is a "no" instance of \( \rho\text{-GAPqCSP}_W \).

We say that \( L \) is polynomial-time reducible to \( \rho\text{-GAPqCSP}_W \), denoted by \( L \leq_p \rho\text{-GAPqCSP}_W \).

If for every language \( L \in \text{NP} \), \( L \leq_p \rho\text{-GAPqCSP}_W \), We say that \( \rho\text{-GAPqCSP}_W \) is \( \text{NP-hard} \).
In fact, the PCP Theorem is equivalent to the NP-hardness of a certain gap version of qCSP.

**Theorem 3.10** There exist constant $q \in \mathbb{N}$ and $\rho \in (0, 1)$ such that $\rho$-GAP$q$CSP is NP-hard.

**Proof:** Assume that $NP \subseteq PCP(\log n, 1)$. We will show that $1/2$-GAP$q$CSP is NP-hard for some constant $q$. Since 3SAT $\in$ NP, 3SAT has a PCP system in which the verifier $M$ makes a constant number of queries, which we denote by $q$, and uses $c \cdot \log n$ random bits for some constant $c$. Given every input $x$ and $r \in \{0, 1\}^c \cdot \log n$, define $M_{x,r}$ to be the function that on input a proof $\pi$ outputs 1 if the verifier $M$ on input $x$ and random bits $r$. Note that $M_{x,r}$ depends on at most $q$ locations. Thus for every $x \in \{0, 1\}^n$, the collection $\varphi = \{M_{x,r}\}_{r \in \{0, 1\}^c \cdot \log n}$ is a polynomial-sized $q$CSP instance. And it is easy to see the transformation can be carried in polynomial-time. By the completeness and soundness of the PCP system, if $x \in 3$SAT then $\varphi$ will satisfy $\text{opt}(\varphi) = 1$, while $x /\in 3$SAT then $\text{opt}(\varphi) \leq 1/2$. \[\square\]

**Theorem 3.11** If there exist $\rho \in (0, 1)$ and $q > 0$ such that $\rho$-GAP$q$CSP is NP-hard, then $NP = PCP(\log n, 1)$.

**Proof:** Assume there exist $\rho \in (0, 1)$ and $q > 0$ such that $\rho$-GAP$q$CSP is NP-hard, then for every language $L \in NP$, there is a polynomial-time computable function $f$ such that $x \in L \Rightarrow \text{opt}(\varphi) = 1$; $x /\in L \Rightarrow \text{opt}(\varphi) \leq \rho$.

Given an input $x$, the verifier will run the reduction $f(x)$ to obtain a $q$CSP instance $\varphi = \{\varphi_i\}_{i=1}^m$, where $\varphi_i = \varphi_i(x_{i_1}, \ldots, x_{i_q})$, and $i_1, \ldots, i_q \in [n], m = n^a$.

A PCP verifier $M_0$ of $\rho$-GAP$q$CSP is as follows:
1. Input an instance $x$ and proof string $\pi_0$;
2. Transform $x$ to $f(x) = \varphi$;
3. Choose $i \in [m]$ randomly;
4. Query $q$ bits from $\pi_0$ to obtain $a_{i_1}, \ldots, a_{i_q}$;
5. Output $\varphi_i(a_{i_1}, \ldots, a_{i_q})$.

Since $\rho \in (0, 1)$, there exists $k \geq 1$ such that $\rho^k \leq 1/2$. Then we can construct another verifier $M$ based on $M_0$:
1. Input an instance $x$ and proof string $\pi$;
2. Transform $x$ to $f(x) = \varphi$;
3. Repeat the following operation for $k$ times:
   (1). Choose $i \in [m]$ randomly;
   (2). Query $q$ bits from $\pi_0$ to obtain $a_{i_1}, \ldots, a_{i_q}$;
   (3). Output 0 if $\varphi_i(a_{i_1}, \ldots, a_{i_q}) = 0$ and halt;
4. Output 1 and halt.

$M$ simulates $M_0$ for $k$ times in step 3, and it outputs 1 if and only if the operations in step 3 all output 1. Thus,

\[
\begin{align*}
x \in L & \Rightarrow \exists \pi \text{s.t. } \Pr_R[M^\pi(x, R) = 1] = 1; \\
x /\in L & \Rightarrow \forall \pi \text{s.t. } \Pr_R[M^\pi(x, R) = 1] \leq \rho^k \leq 1/2.
\end{align*}
\]
M uses $ka \cdot \log n$ random bits and queries at most $kq$ locations of the proof string. Therefore, $L \in PCP(\log n, 1)$, and then $NP \subseteq PCP(\log n, 1)$. The inverse inclusion is obvious, so the PCP theorem holds.

Since 3SAT is a special case of 3CSP instances, that $\rho$-GAP3SAT is NP-hard implies that $\rho$-GAP$q$CSP is NP-hard. The following theorem shows that the former can also be reduced from the later.

**Theorem 3.12** There exists constant $\rho \in (0, 1)$ such that $\rho$-GAP3SAT is NP-hard. That is, there is a polynomial-time computable function $f$ such that $\forall L \in NP$,

$$x \in L \Rightarrow opt(f(x)) = 1;$$

$$x \notin L \Rightarrow opt(f(x)) \leq \rho.$$ 

Where $f(x)$ is an instance of $\rho$-GAP3SAT.

**Proof:** $1/2$-GAP$q$CSP is NP-hard by theorem 3.10 and it suffices to show a reduction from $1/2$-GAP$q$CSP to $\rho$-GAP3SAT where $\rho \in (0, 1)$. That is, we will show a polynomial-time function mapping "yes" instances of $1/2$-GAP$q$CSP to "yes" instances of $\rho$-GAP3SAT and "no" instances of $1/2$-GAP$q$CSP to "no" instances of $\rho$-GAP3SAT.

Let $\varphi = \{\varphi_1, \ldots, \varphi_m\}$ with variables $x_1, \ldots, x_n$ be an instance of $1/2$-GAP$q$CSP, then $\varphi \in q$CSP $\Rightarrow opt(\varphi) = 1$ and $\varphi \notin q$CSP $\Rightarrow opt(\varphi) \leq 1/2$.

For every $i \in [m]$, $\varphi_i$ can be expressed as at most $2^i$ clauses, where each clause is CNF with at most $q$ variables. So, $\varphi_i = \bigwedge_{j \in [2^i]} C_{i,j}$. Assume $q > 3$, we can transform each clause to $q-2$ clauses, each of which contains at most 3 variables, by introducing $q - 3$ additional auxiliary variables. For example, for $C = L_1 \vee \cdots \vee L_q$, introduce new variables $y_1, \ldots, y_{q-3}$ to create a collection of clauses: $(L_1 \vee y_1), (\neg y_1 \vee y_2), \ldots, (\neg y_{q-4} \vee L_{q-2} \vee y_{q-3}), (\neg y_{q-3} \vee L_{q-1} \vee L_q)$. So, $\varphi' = \bigwedge_{i \in [m]} \bigwedge_{j \in [2^i]} \bigwedge_{k \in [q-2]} C_{i,j,k}$, and there are $m(q - 2)2^i$ clauses with $n + m(q - 3)2^i$ variables in $\varphi'$.

It is easy to check that $opt(\varphi') = 1$ when $opt(\varphi) = 1$ and $opt(\varphi') \leq 1 - \frac{1}{(q-2)2^i}$ when $opt(\varphi) \leq 1/2$. Let $\rho = 1 - \frac{1}{(q-2)2^i}$, then the conclusion follows. □

As stated above, the three theorems (Theorem 3.4, Theorem 3.10 and Theorem 3.12) are equivalent.

Theorem 3.12 means that it is NP-hard to tell whether $opt(\varphi) = 1$ or $opt(\varphi) \leq \rho$ where $\varphi$ is an instance of MAX3SAT. Hence it is NP-hard to approximate MAX3SAT with approximation factor $\frac{1}{\rho}$. From this, using expander graphs it can be shown that it is NP-hard to tell whether $opt(\psi) = 1$ or $opt(\psi) \leq c$ where $\psi$ is an instance of MAX3SAT with restriction that each variable is in exactly 5 clauses. We say that $c - GAP3SAT(5)$ is NP-hard.

Next we will show instances of MAX3SAT(5) can be reduced to instances of Label Cover. An instance of Label Cover is denoted by $\langle L(G(V,W,E), [M], [N], \{\pi_{vw}\}_{v,w \in E}\rangle$ where $1.G(V,W,E)$ is a regular bipartite graph.

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


A labeling \( l : V \rightarrow [N], W \rightarrow [M] \) satisfies an edge \((v, w)\) if \( \pi_{vw}(l(w)) = l(v) \). Given an instance \( L \), the goal is to find a labeling which satisfies the maximum fraction of edges, denoted by \( \text{opt}(L) \).

**Theorem 3.13** It is \( \text{NP}-\text{hard} \) to tell whether \( \text{Max-LIN-2} \) is \( \text{NP complete} \).

**Proof:** Given an instance \( \psi \) of \( \text{MAX3SAT}(5) \) we construct an instance \( L_\psi \) of Label Cover as follows. Let \( \psi \) have variables \( \{x_1, \ldots, x_n\} \) and clauses \( \{C_1, \ldots, C_m\} \). Then the graph \( G_\psi(V, W, E) \) has \( V := \{x_1, \ldots, x_n\} \) and \( W := \{C_1, \ldots, C_m\} \). Also \( (x_i, C_j) \in E \) if and only if \( x_i \in C_j \). Note that the degree of a vertex in \( V \) is 3 while that in \( W \) is 3. Hence it is a regular bipartite graph.

Since \( V \) gets labels from \([2]\) we can assume it to be from the set \( \{0, 1\} \). Let \([7]\) denote the seven satisfying assignments that a clause \( C_j \in W \) can have in some order and \( \pi_{x_i,C_j} : [7] \rightarrow [2] \) be the assignment of variable \( x_i \) be the satisfying assignment. For instance, if the clause \( C_j \) were \( x_1 \lor x_2 \lor x_3 \) and the third satisfying assignment were \( x_1 = 1, x_2 = 0, x_3 = 1, \), then \( \pi_{x_1,C_j}(3) = 1 \) while \( \pi_{x_2,C_j}(3) = 0 \).

We will see that if \( \text{opt}(\psi) = 1 \), then \( \text{opt}(L_\psi) = 1 \), and if \( \text{opt}(\psi) \leq c \) then \( \text{opt}(L_\psi) \leq 1 - \frac{1-c}{3} \).

If \( \text{opt}(\psi) = 1 \), then there is an assignment which satisfies all the clauses. Thus labeling \( x_i \) as per the satisfying assignment, and giving \( C_j \) the label corresponding to the satisfying assignment, we see that we can satisfy all the edges. Hence \( \text{opt}(L_\psi) = 1 \).

When \( \text{opt}(\psi) \leq c \), consider any labeling \( l \) of \( L \). Since the labeling of \( V \) can be thought of as an assignment, at least \( 1 - c \) fraction of clauses \( C_j \) are unsatisfied by this assignment. Pick one such clause \( C_j = x_1 \lor x_2 \lor x_3 \). Since \( C_j \) is unsatisfied, whatever label it might have, one of its three neighbors must be violating the constraints, otherwise the clause would be satisfied. Thus at least \( \frac{1-c}{3} \) of the edges are violated and hence \( \text{opt}(L_\psi) \leq 1 - \frac{1-c}{3} \). Let \( c' = 1 - \frac{1-c}{3} \) and we complete the proof. \( \square \)

4. \text{MAX-SATISFY over \( F_2 \)}

Given a system of \( m \) linear equations with \( n \) variables over \( F_2 \), find a solution satisfying the maximal number of equations. We call it \text{MAX-SATISFY over \( F_2 \)}, denoted by \text{Max-LIN-2}. If each equation contains exactly \( k \) variables, we denote it by \text{Max-Ek-LIN-2}.

**Theorem 4.1** \text{Max-LIN-2} is \( \text{NP complete} \).

**Proof:** \text{MAX2SAT} is \( \text{NP complete} \) and it suffices to show that there is a reduction from \text{MAX2SAT} to \text{MAX-LIN-2} in polynomial time. Let \( C = C_1 \land C_2 \land \cdots \land C_l \), where \( C_i = \alpha_i \lor \beta_i, i = 1, \ldots, l \). For each clause \( C_i = \alpha_i \lor \beta_i \), we construct three equations \( C'_i \):

\[
\begin{align*}
\alpha_i + \beta_i &= 1, \\
\alpha_i &= 1, \\
\beta_i &= 1, \quad (1)
\end{align*}
\]

It is easy to see that \( C_i = 1 \) if and only if two equations of \( C'_i \) hold. Hence \( k \) equations of \( C \) are satisfied if and only if \( 2k \) equations of \( C'_i \) are satisfied. This reduction can be done in polynomial time and so the conclusion follows. \( \square \)
The theorem shows that it is hard to find exact solutions of MAX-LIN-2 unless \( P = NP \), and so it is natural and important to consider the approximation algorithm of the problem.

The PCP Theorem can be used to prove that many NP optimization problems cannot be approximated in polynomial time within some precision with the assumption \( P \neq NP \). But it does not imply how well can we approximate the problems. From theorem 3.12, we can see that it is \( \text{NP-hard} \) to approximate MAX3SAT with approximation ratio \( \frac{1}{(q-2)2q^{-2}} \), which is close to one. And we know that there is a random polynomial-time \( 8/7 \)-approximation algorithm for MAX3SAT. So, a natural question is: is there a \( \rho \)-approximation polynomial-time algorithm, where \( \rho \) lies between the two values? Therefore, we need some stronger PCP Theorems. The following 3-bit PCP Theorem due to Johan H˚ astad gives some tight inapproximability results about the problems MAX-SATISFY over finite fields and MAX3SAT.

**Theorem 4.2** For every \( \epsilon > 0 \) and every language \( L \in NP \) there is a PCP-verifier for \( L \) making three queries having completeness at lest \( 1 - \epsilon \) and soundness at most \( 1/2 + \epsilon \). Moreover, given a proof \( \pi \), \( V \) chooses a triple \((i_1, i_2, i_3)\) and \( b \in \{0,1\} \) according to some distribution and accepts if and only if \( \pi_{i_1} + \pi_{i_2} + \pi_{i_3} = b \pmod{2} \).

**Corollary 4.3** It is \( \text{NP-hard} \) to approximate Max-E3-LIN-2 with approximation ratio \( 2 - \epsilon \) for every \( \epsilon > 0 \).

The result is tight since a random assignment is expected to satisfy half of the equations. Moreover, we have the following corollaries.

**Corollary 4.4** It is \( \text{NP-hard} \) to approximate Max-Ek-LIN-2 with approximation ratio \( 2 - \epsilon \) for every \( \epsilon > 0 \) and \( k \geq 3 \).

**Proof:** We will reduce instances of Max-E3-LIN-2 to instances of Max-Ek-LIN-2. Given a system of equations with variables \( (x_i)_{i=1}^{n} \) and 3 variables in each equation. Add the same \( k - 3 \) new variables \( (y_i)_{i=1}^{k-3} \) in every equation to make them all have \( k \) variables. If \( \prod_{i=1}^{k-3} y_i = 1 \), then the assignment to the larger system satisfies the same equations in the smaller system while if \( \prod_{i=1}^{k-3} y_i = -1 \) then it satisfies exactly the equations not satisfied in the larger system. Changing every \( x_i \) to its negation, now the assignment satisfies the equations that are satisfied in the larger system. Therefore, the maximal number of equations is preserved and it is easy to translate a solution of the larger system to a solution of the smaller system. And it is done in polynomial time. \( \square \)

**Corollary 4.5** \((8/7 - \eta)\)-approximation for MAX3SAT is \( \text{NP-hard} \) for every \( \eta > 0 \).

**Proof:** We reduce instances of Max-E3-LIN-2 to instances of MAX3SAT. The linear equation over \( F_2 \) \( a + b + c = 0 \) is equivalent to four clauses \( (\bar{a} \lor b \lor c), (a \lor \bar{b} \lor c), (a \lor b \lor \bar{c}) \) and \( (\bar{a} \lor b \lor \bar{c}) \). It is similar for the equation \( a + b + c = 1 \). We can see that the linear equation is satisfied if and only if all of the clauses are satisfied, and the equation is unsatisfied then three of the clauses are unsatisfied. Therefore, if the fraction of the satisfied equations is at least \( 1 - \epsilon \) so are the clauses and if the fraction of satisfied equations is at most \( 1/2 + \epsilon \), the fraction for the clauses is at most \( 1 - (1/2 - \epsilon) \times 1/4 = 7/8 - \epsilon/4 \). \( \frac{1-\epsilon}{7/8-\epsilon/4} = 8/7 - \eta \), where
η > 0 and is arbitrary. Then the conclusion holds.

Now it suffices to prove Theorem 4.2.

We can construct a PCP verifier starting from an instance of Label Cover \( L(G(V, W, E), [M], [N], \{\pi_{vw}\}_{(v, w) \in E}) \). Let the labeling to instances \( L \) as the proof string and the verifier works as follows: pick one edge \((v, w) \in E\) at random and read the labels \( l(v), l(w) \) from the proof. It accepts if and only if \( \pi(l(w)) = l(v) \). This PCP is called Outer PCP. We can see from Theorem 3.13 that the completeness is 1 and soundness is \( c' \). In fact, we can make a parallel verifier which make the soundness parameter arbitrary small. We call it \( V_{RAZ} \).

Suppose the new alphabet for \( V \) is \([n]\) and for \( W \) is \([m]\), obviously \( m > n \), and \( m, n \) are both polynomials of the input. Hence it is NP-hard to tell whether \( \text{opt}(L) = 1 \) or \( \text{opt}(L) \leq \epsilon \) for arbitrary \( \epsilon > 0 \).

Clearly, we use \( O(\log mn) \) random bits and two queries. One query is from the alphabet \([n]\) and one from \([m]\), and hence the total number of bits queried is \( \log m + \log n \). This is far from the desired goal of three query bits. We need to construct a new PCP verifier based on the Outer PCP \( V_{RAZ} \). The new proof string contains the encodings of the labels of the vertices instead of the labels of the vertices.

It is convenient for us to to work over \( \{1, -1\} \) instead of the standard set \( \{0, 1\} \). We use the mapping \( b \mapsto (-1)^b \) (that is \( 0 \mapsto 1 \) and \( 1 \mapsto -1 \)). It maps the addition operation in \( \{0, 1\} \) to multiplication operation in \( \{1, -1\} \). The operation over the set of functions from \( \{1, -1\} \) to \( \{1, -1\} \) is defined as follows: \((f + g)(x) = f(x) + g(x), (af)(x) = af(x) \) and the inner product \( \langle f, g \rangle = E_{x \in \{1, -1\}}^n[f(x)g(x)] \). And the Fourier basis of the function space is the set \( \{\chi_\alpha\}_{\alpha \subseteq [n]} \) where \( \chi_\alpha(x) = \Pi_{i \in \alpha}x_i (\chi_\emptyset = 1) \). These correspond to the linear equations over \( F_2 \) and the basis is an orthonormal basis since \( \langle \chi_\alpha, \chi_\beta \rangle = \delta_{\alpha, \beta} \). Every function in the above function set can be represented as \( f = \sum_{\alpha \subseteq [n]} \hat{f}_\alpha \chi_\alpha \). We call \( \hat{f}_\alpha \) Fourier coefficient of \( f \).

It is easy to see that \( \langle f, g \rangle = \sum_\alpha \hat{f}_\alpha \hat{g}_\alpha \). Expand the left of the formula \( \langle f, g \rangle = \langle \Sigma_\alpha \hat{f}_\alpha \chi_\alpha, \Sigma_\beta \hat{g}_\beta \chi_\beta \rangle = \sum_{\alpha, \beta} \hat{f}_\alpha \hat{g}_\beta \delta_{\alpha, \beta} = \sum_\alpha \hat{f}_\alpha \hat{g}_\alpha \).

Next, we give the definition of long code which is used in the new verifier.

**Definition 4.6** We say that \( f : \{\pm 1\}^m \to \{\pm 1\} \) is a coordinate function if there is some \( i \in [m] \) such that \( f(x_1, x_2, \ldots, x_m) = x_i \), that is, \( f = \chi_i \).

**Definition 4.7** The long code for \([m]\) encodes each \( i \in [m] \) by the table of all values of the function \( \chi_i : \{\pm 1\}^m \to \{\pm 1\} \).

Hence the long code for \( l(v) \in [n] \) is the truth table of \( \chi_{l(v)} : \{\pm 1\}^n \to \{\pm 1\} \), and the long code for \( l(w) \in [m] \) is the truth table of \( \chi_{l(w)} : \{\pm 1\}^m \to \{\pm 1\} \).

Given a function \( f : \{\pm 1\}^m \to \{\pm 1\} \), we want to know whether it is long code, that is whether \( f \) has good agreement with \( \chi_i \) for some \( i \). By the definition of the inner product, \( \langle f, g \rangle = \epsilon \) if and only if \( f \) has agreement \( 1/2 + \epsilon/2 \) with \( g \). Since \( \langle f, \chi_\alpha \rangle = \hat{f}_\alpha \), \( f \) has better agreement with the linear equation if it has larger Fourier coefficient. As to the proof of Theorem 4.2, we need add some noise to the test.
Long code test:
Input: a string of length $2^m$ as a function $f : \{\pm 1\}^m \to \{\pm 1\}$.
Desired property: The function $f$ should be a long code, that is, there exists $i \in m$ such that $f = \chi_i$.
Verifier:
1. Pick two random vectors $x, y \in \{\pm 1\}^m$ with uniform probability;
2. Pick a vector $z \in \{\pm 1\}^m$ according to the following distribution: for every $i \in [m]$ choose $z_i = 1$ with probability $1 - \rho$ and choose $z_i = -1$ with probability $\rho$;
3. It accepts iff $f(x)f(y) = f(xyz)$.

Note: For every two vectors $x, y \in \{\pm 1\}^m$, denote by $xy$ their componentwise multiplication.
Suppose $f = \chi_i$, then
$$f(x)f(y)f(xyz) = x_iy_i(x_iy_iz_i) = z_i$$
Hence the test accepts iff $z_i = 1$ which has probability $1 - \rho$.

**Lemma 4.8** If the test accepts with probability at least $1/2 + \epsilon$ then $\sum_{X, f} f^3(x)(1 - 2\rho)^{|x|} \geq 2\epsilon$.

**Proof:** Since the acceptance probability is $1/2 + \epsilon$ then $E[f(x)f(y)f(xyz)] \geq (1/2 + \epsilon) - (1/2 - \epsilon) = 2\epsilon$. Replacing $f$ by its Fourier expansion, we obtain
$$E_{x,y,z}[(\sum_{X} f\hat\alpha(x)) \cdot (\sum_{X} f\hat\beta(y)) \cdot (\sum_{X} f\hat\gamma(z))] = E_{x,y,z}[(\sum_{X} f\hat\alpha(x)\chi_\alpha(x)\chi_\beta(y)\chi_\gamma(z))] = \sum_{X, \alpha, \beta, \gamma} f\hat\alpha f\hat\beta f\hat\gamma E_{x,y,z}(\chi_\alpha(x)\chi_\beta(y)\chi_\gamma(z)).$$
It equals 0 unless $\alpha = \beta = \gamma$. So the above formula equals $\sum_{X, \alpha} f\hat\alpha^3 E_{z}([\chi_\alpha(z)])$, where $E_{z}([\chi_\alpha(z)]) = E_{z}[\Pi_{i \in \alpha} z_i] = \Pi_{i \in \alpha} E_{z}[z_i] = (1 - 2\rho)^{|\alpha|}$. Hence we get $\sum_{X, \alpha} f\hat\alpha^3 (1 - 2\rho)^{|\alpha|} \geq 2\epsilon$. □

The new verifier due to Håstad is based on this and denoted by $V_H$. Instead of using alphabet $\{1, \ldots, m\}$ or $\{1, \ldots, n\}$, $V_H$ expects each entry in the proof string to be encoded by the long code. Assume the encodings $f$ is bifolded, that is, for all $X \in \{\pm 1\}^m$, $f(-X) = -f(X)$.

**Lemma 4.9** If $f : \{\pm 1\}^m \to \{\pm 1\}$ is bifolded and $\hat f_\alpha \neq 0$ then $|\alpha|$ must be an odd number.

**Proof:**
$$\hat f_\alpha = \langle f, \chi_\alpha \rangle = \frac{1}{2^m} \sum_{X} f(X) \prod_{i \in \alpha} X_i$$
If $|\alpha|$ is even then $\prod_{i \in \alpha} X_i = \prod_{i \in \alpha} (-X_i)$. So if $f$ is bifolded, the sum is 0. □

New verifier $V_H$:
Input: a proof string $\pi$ encoded by long code.
1. Choose $(v, w) \in E$ at random, $l(w) = i, l(v) = j$ is the labels in the proof string $\pi$ of $V_{RAZ}$;
2. $\pi$ contains in the locations $i$ and $j$ two functions respectively (which may or may not the long code of each entry of $\pi$), denoted by $f$ and $g$, where $f$ and $g$ are bifolded;
3. Choose \( X \in \{\pm 1\}^n, Y \in \{\pm 1\}^m \) at random;
4. Choose a vector \( \mu \in \{\pm 1\}^m \) according to the following distribution: for every \( i \in [m] \) choose \( \mu_i = 1 \) with probability \( 1 - \rho \) and choose \( \mu_i = -1 \) with probability \( \rho \);
5. It accepts iff \( f(Y)g(X)f((X \circ \pi)Y\mu) = 1 \), where \( (X \circ \pi)_i = X_{\pi(i)}, i = 1, \ldots, m \).

Our goal is to prove the new verifier has completeness 1 and soundness \( \frac{1 + \delta}{2} \).

If \( \text{opt}(L) = 1 \), then there exists a correct labeling \( l(w) = i, l(v) = j \) for \( v \in V, w \in W \) such that for \( (v, w) \in E, \pi_{v,w}(l(w)) = l(v) \). That is, \( \pi(i) = j \). Hence there exists proof string \( \hat{\pi} \) which contains \( f = \chi_i \) corresponding to \( i \) and \( g = \chi_j \) corresponding to \( j \). Therefore, \( f(Y)g(X)f((X \circ \pi)Y\mu) = Y_iX_j((X \circ \pi)Y\mu)_i = Y_iX_jX_{\pi(i)}Y_i\mu_i = \mu_i \). Hence \( \hat{\pi} \) accepts iff \( \mu_i = 1 \), which happens with probability \( 1 - \rho \).

If \( \hat{\pi} \) accepts with probability at least \( \frac{1 + \delta}{2} \), then we can construct a verifier \( \hat{\text{Raz}} \) such that its soundness is at least \( 4\rho \delta^2 \), which contradicts with the arbitrary small soundness of \( \text{Raz} \).

The details are as follows:

Consider the expectation of the acceptance condition. First,

\[ E_{r,X,Y,\mu}(f(Y)g(X)f((X \circ \pi)Y\mu)) \geq \frac{1 + \delta}{2} - \frac{1 - \delta}{2} = \delta. \]

\[ E_{r,X,Y,\mu}(f(Y)g(X)f((X \circ \pi)Y\mu)) = E_{r,X,Y,\mu}[(\Sigma f\hat{a}\chi_\alpha(Y)) \cdot (\Sigma g\hat{b}\chi_\beta(X)) \cdot (\Sigma \chi_\gamma((X \circ \pi)Y\mu))], \]

\[ = E_{r,X,Y,\mu}[\Sigma \alpha, \beta, \gamma \hat{a}\hat{b}\hat{\chi}_\alpha(Y)\chi_\beta(X)\chi_\gamma((X \circ \pi)Y\gamma(\mu))]. \]

Since \( \chi_\gamma(X \circ \pi) = \prod_{i \in \gamma}(X \circ \pi)_i = \prod_{i \in \gamma}X_{\pi(i)} = \prod_{j \in \pi_2(\gamma)}X_j = \chi_{\pi_2(\gamma)}(X) \), where \( \pi_2(\gamma) = \{ j \in [n] : |\pi^{-1}(j) \cap \gamma| \text{ is odd} \} \), the above formula can be written as:

\[ E_{r,X,Y,\mu}[\Sigma \alpha, \beta, \gamma \hat{a}\hat{b}\hat{\chi}_\alpha(\gamma)\chi_\beta(\gamma)(X)\chi_\gamma(\mu)] = E_{r,X,Y,\mu}[\Sigma \alpha, \beta, \gamma \hat{a}\hat{b}\hat{\chi}_\alpha(\gamma)(1 - 2\rho)^{|\gamma|}\chi_\beta(\gamma)(X)\chi_\gamma(\mu)] \]

Since \( E(\chi_\alpha\Delta_\gamma(Y)) = 0 \) unless \( \alpha = \gamma \) and \( E(\chi_\beta\Delta_\gamma(\gamma)(X)) = 0 \) unless \( \beta = \pi_2(\gamma) \), the above formula equals

\[ E_r[\Sigma \alpha, \beta, \gamma \hat{a}\hat{b}\hat{\chi}_\alpha(\gamma)(1 - 2\rho)^{|\alpha|}] \geq \delta. \]

Next, we will use \( \hat{\pi} \) to construct a proof \( \pi \) for \( \text{Raz} \):

1. Select \( \alpha \subseteq [m] \) with probability \( \hat{f}_\alpha^2 \) and then select \( i \) at random from \( \alpha \);
2. Select \( \beta \subseteq [n] \) with probability \( \hat{g}_\beta^2 \) and then select \( j \) at random from \( \beta \);
3. \( \text{Raz} \) accepts if and only if \( \pi(i) = j \).

The selection is reasonable because \( \langle f, f \rangle = \Sigma \alpha \hat{f}_\alpha^2 = 1 \) and \( \langle g, g \rangle = \Sigma \beta \hat{g}_\beta^2 = 1 \).

If \( \beta = \pi_2(\alpha) \), then for all \( j \in \beta \) there exists at least one \( i \in \alpha \) such that \( \pi(i) = j \). Hence, the verifier \( \text{Raz} \) accepts with probability at least \( \Sigma \alpha \hat{f}_\alpha^2 \hat{g}_{\pi_2(\alpha)}^2 |\alpha|^{-1} \).

Using Cauchy equality, we have

\[ \Sigma \alpha \hat{f}_\alpha^2 \hat{g}_{\pi_2(\alpha)}^2 |\alpha|^{-1} \leq (\Sigma \alpha \hat{f}_\alpha^2 \hat{g}_{\pi_2(\alpha)}^2 |\alpha|^{-1})^{1/2} (\Sigma \alpha \hat{f}_\alpha^2)^{1/2} = (\Sigma \alpha \hat{f}_\alpha^2 \hat{g}_{\pi_2(\alpha)}^2 |\alpha|^{-1})^{1/2} = \Sigma \alpha \hat{f}_\alpha^2 \hat{g}_{\pi_2(\alpha)}^2 |\alpha|^{-1/2} \leq \Sigma \alpha \hat{f}_\alpha^2 \hat{g}_{\pi_2(\alpha)}^2 |\alpha|^{-1/2} \]
The long code for \([m]\) encodes each \(g\) and we define the coefficients by \(\hat{Z}\) function \(\chi\). Verifier forming \(1\). We can see that \(\delta\) \(\geq\) \(\frac{1+\delta}{2}\). Transforming \(\{\pm 1\}\) back to \(\{0, 1\}\), the acceptance condition becomes linear equation over \(F_2\). Let the bits in the proof to be unknown boolean variables and tests of the verifier (linear equation) as constraints, the task of finding a proof that maximizes acceptance probability is the same as the task of finding an assignment that satisfies the maximum fraction of constraints. It is easy to see that theorem 4.2 holds.

5. MAX-SATISFY over \(F_p\)

We can extend the results in section 4 to \(F_p\), where \(p\) is a prime. We also start from the verifier \(V_{Raz}\) of Label Cover, the difference is the definition of the long code.

Assume \(U_p\) is the set of \(p\)th roots of unity. Denote \(U_p = \{\eta^0, \ldots, \eta^{p−1}\}\) and \(Z_p = \{0, 1, \ldots, p−1\}\). Consider the map: \(Z_p \rightarrow U_p\) which maps \(a \in Z_p\) to \(\eta^a \in U_p\). It maps the addition operation in \(Z_p\) to multiplication operation in \(U_p\). We work over \(U_p\) and then the result can be transformed back to \(Z_p\).

**Definition 5.1** The long code for \([m]\) encodes each \(i \in [m]\) by the table of all values of the function \(\chi_i: U_p^m \rightarrow U_p\).

Hence the long code for \(l(v) \in [n]\) is the truth table of \(\chi_{l(v)}: U_p^m \rightarrow U_p\), and the long code for \(l(w) \in [m]\) is the truth table of \(\chi_{l(w)}: U_p^m \rightarrow U_p\).

For \(f, g: U_p^m \rightarrow \mathbb{C}\), their inner product is defined by

\[
\langle f, g \rangle = \frac{1}{p^m} \sum_{X \in U_p^m} f(X)\overline{g(X)}
\]

Where \(\overline{g(X)}\) denotes complex conjugation.

The Fourier basis function becomes \(\chi_\alpha, \chi_\alpha(X) = \prod_{i \in [m]} X_i^{\alpha_i}\), where \(\alpha = (\alpha_1, \ldots, \alpha_m), \alpha_i \in Z_p\). We can see that \(\langle \chi_\alpha, \chi_\beta \rangle = \delta_{\alpha, \beta}\). The basis functions form a complete orthonormal system and we define the coefficients by \(\hat{f}_\alpha = \langle f, \chi_\alpha \rangle\) and \(f(X) = \sum_\alpha \hat{f}_\alpha \chi_\alpha(X)\).

\[
\langle f, f \rangle = \langle \sum_\alpha \hat{f}_\alpha \chi_\alpha, \sum_\beta \hat{f}_\beta \chi_\beta \rangle = \sum_\alpha |\hat{f}_\alpha|^2
\]
and
\[ \langle f, f \rangle = \frac{1}{p^m} \sum_X |f(X)|^2 = 1 \]

Hence, \( \sum_a |\hat{f}_a|^2 = 1 \).

When working with long code, we need to fold over \( U_p \).

**Definition 5.2** Given a function \( f : U_p^m \to U_p \). The function \( f_{U_p} \), folding over \( U_p \), is defined by for each set of \( (\gamma X)_{\gamma \in U_p} \) selecting one. If \( \gamma_0 X \) is selected, then \( f_{U_p}(\gamma X) = \gamma \gamma_0^{-1} f(\gamma_0 X) \) for all \( \gamma \in U_p \).

It is easy to see that \( f_{U_p}(\gamma X) = \gamma f_{U_p}(X) \), and we all denote \( f_{U_p} \) by \( f \).

Now we will describe the verifier \( V_H \):

Input: a proof string \( \pi' \) encoded by long code.
1. Choose \( (v, w) \in E \) at random, \( l(w) = i, l(v) = j \) is the labels in the proof string \( \pi \) of \( V_{RAZ} \):
2. \( \pi' \) contains the locations \( i \) and \( j \) two functions respectively (which may or may not be the long code of each entry of \( \pi \) ), denoted by \( f \) and \( g \), where \( f : U_p^m \to U_p \), \( g : U_p^m \to U_p \) and they are both folded over \( U_p \).
3. Choose \( X \in U_p^m, Y \in U_p^m \) at random;
4. Choose a vector \( \mu \in U_p^m \) according to the following distribution: for every \( i \in [m] \) choose \( \mu_i = 1 \) with probability \( 1 - \rho \) and otherwise \( \mu_i = \gamma \) where \( \gamma \) is chosen randomly and uniformly in \( U \);
5. It accepts iff \( f(Y)g(X)f((X \circ \pi)Y\mu)^{-1} = 1 \), where \( (X \circ \pi)_{i} = X_{\pi(i)}, i = 1, \ldots, m \).

If \( opt(L) = 1 \), then there exists a correct labeling \( l(w) = i, l(v) = j \) for \( v \in V, w \in W \) such that for \( (v, w) \in E \), \( \pi_{v,w}(l(w)) = l(v) \). That is, \( \pi(i) = j \). Hence there exists proof string \( \pi' \) which contains \( f = \chi_i \) corresponding to \( i \) and \( g = \chi_j \) corresponding to \( j \). Therefore, \( f(Y)g(X)f((X \circ \pi)Y\mu)^{-1} = Y_iX_j((X \circ \pi)^{-1}Y^{-1}\mu^{-1})_i = Y_iX_j^{-1}Y^{-1}\mu^{-1}_i = \mu^{-1}_i \). Hence \( V_H \) accepts iff \( \mu^{-1}_i = 1 \), which happens with probability at least \( 1 - \rho \).

If \( V_H \) accepts with probability at least \( \frac{1 - \rho}{p^2} \), then we can construct a verifier \( V_{RAZ} \) such that its soundness is at least \( 2p^2/\rho^2 \), which contradicts with the arbitrary small soundness of \( V_{RAZ} \). The details are as follows:

**Lemma 5.3** Suppose \( \gamma \in U_p, \gamma^* \in Z_p \),

\[ \sum_{\gamma^* \in Z_p, \gamma^* \neq 0} \gamma^* = \begin{cases} p - 1 & \gamma = 1 \\ -1 & \text{otherwise} \end{cases} \]

**Proof:** If \( \gamma = 1 \), \( \sum_{\gamma^* \in Z_p, \gamma^* \neq 0} \gamma^* = (p - 1) \cdot 1 = p - 1 \).

If \( \gamma \neq 1 \), as we vary \( \gamma^* \) over \( Z_p \), \( \gamma^* \) varies over a complete set of roots of unity. Hence, \( \sum_{\gamma^* \in Z_p} \gamma^* = 0 \), and then \( \sum_{\gamma^* \in Z_p, \gamma^* \neq 0} \gamma^* = -\gamma^0 = -1 \). \( \Box \)
Consider the following formula:

\[ E_{r,X,Y,\mu}[\sum_{\gamma^* \neq 0} (f(Y)g(X)f(((X \circ \pi)Y\mu)^{-1}))^{\gamma^*}] \]

\[ \geq (p - 1) \cdot \frac{1 + \delta}{p} + (-1) \cdot \left(1 - \frac{1 + \delta}{p}\right) \]

\[ = \delta \]

Denote \( G = g^{\gamma^*}, F = f^{\gamma^*} \),

\[ E_{X,Y,\mu}[F(Y)G(X)F(((X \circ \pi)Y\mu)^{-1})] \]

\[ = E_{X,Y,\mu}[\Sigma_{\alpha} \hat{F}_{\alpha} \chi_{\alpha}(Y) \cdot \Sigma_{\beta} \hat{G}_{\beta} \chi_{\beta}(X)] \cdot \Sigma_{\gamma} \hat{F}_{\gamma} \chi_{\gamma}(((X \circ \pi)Y\mu)^{-1})] \]

\[ = E_{X,Y,\mu}[\Sigma_{\alpha,\beta,\gamma} \hat{F}_{\alpha} \hat{G}_{\beta} \hat{F}_{\gamma} \chi_{\alpha}(Y) \chi_{\beta}(X) \chi_{\gamma}((X \circ \pi)^{-1}) \chi_{\gamma}(Y^{-1}) \chi_{\gamma}(\mu^{-1})] \]

\[ E_\mu(\chi_{\gamma}(\mu^{-1})) = E_\mu(\prod_{i \in [m]} \mu_i^{-\gamma_i}) = (1 - \rho)^{s(\gamma)} \]

Where \( s(\gamma) = \{|i : \gamma_i \neq 0\}| \) and \( \pi_p(\gamma) = \{\theta : \theta_j = \sum_{i : \pi(i) = j} \gamma_i\} \).

Hence,

\[ E_{X,Y,\mu}[F(Y)G(X)F(((X \circ \pi)Y\mu)^{-1})] = \sum_{\alpha} \hat{F}_{\alpha} \hat{G}_{\mu(\alpha)}^2 \rho_{\alpha} (1 - \rho)^{s(\alpha)} \]

So, there exists some \( \gamma_0^* \) such that \( E_{r}[\sum_{\alpha} \hat{F}_{\alpha} \hat{G}_{\mu(\alpha)}^2 \rho_{\alpha} (1 - \rho)^{s(\alpha)}] \geq \frac{\delta}{p} \).

**Lemma 5.4** Given \( f \) as above, and \( \gamma^* \in \mathbb{Z}_p, F = f^{\gamma^*} \). Then for all \( \hat{F}_\alpha \neq 0 \), we have \( \sum_i \alpha_i = \gamma^* \). In particular if \( \gamma^* \) is nonzero, there exists some \( i \) such that \( \alpha_i \neq 0 \).

**Proof:** Assume \( \sum_i \alpha_i \neq \gamma^* \), and take some \( \gamma \in \mathbb{U}_p \) with \( \gamma^{\gamma^* - \sum_i \alpha_i} \neq 1 \).

We have

\[ \hat{F}_\alpha = \frac{1}{p^m} \sum_{X \in \mathbb{U}_p^m} F(X)\overline{\chi_{\alpha}(X)} \]

\[ = \frac{1}{p^m} \sum_{X \in \mathbb{U}_p^m} F(\gamma X)\overline{\chi_{\alpha}(\gamma X)} \]

\[ = \frac{1}{p^m} \sum_{X \in \mathbb{U}_p^m} \gamma^{\gamma^*} F(x) \prod_{i \in [m]} (\gamma X_i)^{\alpha_i} \]

\[ = \frac{1}{p^m} \sum_{X \in \mathbb{U}_p^m} \gamma^{\gamma^* - \sum_{i \in [m]} \alpha_i} F(X)\overline{\chi_{\alpha}(X)} \]

\[ = \gamma^{\gamma^* - \sum_{i \in [m]} \alpha_i} \hat{F}_\alpha \]

We conclude that \( \hat{F}_\alpha = 0 \). □
Next, we will use $\tilde{\pi}'$ to construct a proof $\pi$ for $V_{Raz}$:
1. Select $\alpha \subseteq U_p^m$ with probability $|\tilde{F}_\alpha|^2$ and then select $i \in [m]$ at random such that $\alpha_i \neq 0$;
2. Select $\beta \subseteq U_p^n$ with probability $|\hat{G}_\beta|^2$ and then select $j \in [n]$ at random such that $\beta_j \neq 0$;
3. $V_{Raz}$ accepts if and only if $\pi(i) = j$.

The selection is reasonable because $\sum_\alpha |\tilde{F}_\alpha|^2 = 1$ and $\sum_\beta |\hat{G}_\beta|^2 = 1$, and Lemma 5.4.

If $\beta = \pi_p(\alpha)$, then for all $j \in [n]$, $\beta_j \neq 0$ there exists at least one $i \in [m]$, $\alpha_i \neq 0$ such that $\pi(i) = j$. Hence, the verifier $V_{Raz}$ accepts with probability at least $\sum_\alpha |\tilde{F}_\alpha|^2|\hat{G}_{\pi_2(\alpha)}|^2s(\alpha)^{-1}$.

Using Cauchy equality, we have
\[
\sum_\alpha |\tilde{F}_\alpha|^2|\hat{G}_{\pi_2(\alpha)}|^2s(\alpha)^{-1/2} \\
\leq (\sum_\alpha |\tilde{F}_\alpha|^2|\hat{G}_{\pi_2(\alpha)}|^2s(\alpha)^{-1})^{1/2} (\sum_\alpha |\tilde{F}_\alpha|^2)^{1/2} \\
= (\sum_\alpha |\tilde{F}_\alpha|^2|\hat{G}_{\pi_2(\alpha)}|^2s(\alpha)^{-1})^{1/2}
\]
Hence,
\[
E_{r}[\sum_\alpha |\tilde{F}_\alpha|^2|\hat{G}_{\pi_2(\alpha)}|^2s(\alpha)^{-1}] \geq E_{r}[\sum_\alpha |\tilde{F}_\alpha|^2|\hat{G}_{\pi_2(\alpha)}|^2s(\alpha)^{-1/2}] \\
\geq E_{r}(\sum_\alpha |\tilde{F}_\alpha|^2|\hat{G}_{\pi_2(\alpha)}|^2s(\alpha)^{-1/2})^2
\]
Since $(2\rho s(\alpha))^{-1/2} \geq \exp(-\rho s(\alpha)) \geq (1 - \rho)^{s(\alpha)}$,
\[
E_{r}[\sum_\alpha |\tilde{F}_\alpha|^2|\hat{G}_{\pi_2(\alpha)}|^2s(\alpha)^{-1}] \geq 2\rho E_{r}[\sum_\alpha |\tilde{F}_\alpha|^2|\hat{G}_{\pi_2(\alpha)}|(1 - \rho)^{s(\alpha)}] \geq 2\rho \delta^3/p^2
\]

Therefore, the soundness of the verifier $V_{Raz}$ is $2\rho \delta^3/p^2$. Choose $\rho = \delta$, and choose $\delta$ such that $2\delta^3/p^2 \geq \epsilon$, then the soundness of $V_{Raz}$ we construct is at least $\epsilon$, which is a contradiction with our assumption.

The conclusion is that the verifier $V_H$ has completeness $1 - \delta$ and soundness $\frac{1+\delta}{\rho}$.

Similar to the case of $F_2$, we can get the following theorem.

**Theorem 5.5** For any $\epsilon > 0$, it is NP-hard to approximate Max-E3-LIN-$p$ with a factor of $p - \epsilon$.

**References**


