1 Plan

In this lecture, we will consider two different formulations of the PCP theorem and prove that they are equivalent. In the second half of the lecture, we will develop the tools needed to prove a weaker form of the theorem where the size of the proof is allowed to be exponential.

2 PCP Theorem: Equivalent Formulations

Recall that in the last class, we defined the notion of a PCP verifier. Formally,

Definition 1 (PCP Verifier) For two positive integer valued functions $q, r$ on the set of integers, and a language $L$, we say that $L$ has a $(r(n), q(n))$ PCP verifier if there exists a probabilistic polynomial time Turing machine $V$ which, when given an input $x$ and random access to a string $\pi \in \{0, 1\}^{q(n)2^{r(n)}}$, called a proof, uses at most $r(n)$ random bits and reads at most $q(n)$ locations of $\pi$ and then outputs a $V^\pi(x) \in \{0, 1\}$ satisfying,

- Completeness: $x \in L \Rightarrow \exists \pi \text{ such that } Pr[V^\pi(x) = 1] = 1$
- Soundness: $x \notin L \Rightarrow \forall \pi, Pr[V^\pi(x) = 1] \leq \frac{1}{2}$

The class $\text{PCP}(r(n), q(n))$ is the class of all languages $L$ such that there are constants $c, d$ such that $L$ has a $(r(n), q(n))$ PCP verifier.

We are now all set to state the first version of the PCP theorem.

Theorem 1 (PCP theorem: Formulation 1) $\text{NP} = \text{PCP}(\log n, 1)$

The other view of the PCP theorem, is the view from the point of view of hardness of approximation. Before we state this version of the theorem, we will set up some more notation.

For a boolean formula in 3CNF $\phi$, we define its value, $\text{val}(\phi)$ as the maximum fraction of clauses satisfied by any assignment to the variables. Clearly, $\text{val}(\phi) = 1$ if and only in $\phi$ is satisfiable. Recall that in the last class we defined the problem of $\text{MAX3SAT}$ where the input is a boolean formula in 3CNF form and the goal is to output an assignment which satisfies the maximum number of clauses. Clearly, the problem is NP hard. Let us now define what we mean by a $\rho$ approximation algorithm for $\text{MAX3SAT}$.

Definition 2 ($\rho$ approximation algorithm) For any $0 \leq \rho \leq 1$, we say that an algorithm $A$ is a $\rho$ approximation algorithm for $\text{MAX3SAT}$ if $A$ takes as input a boolean formula $\phi$, which is a 3CNF and outputs an assignment which satisfies at least $\rho \text{val}(\phi)$ fraction of all the clauses of $\phi$.

From what we have seen so far, it is conceivable that there is a $\rho$ approximation algorithm for $\text{msat}$ for any constant $0 \leq \rho \leq 1$. We will eventually show that this is not the case. More formally, the following statement is an alternative formulation of the PCP theorem.

Theorem 2 (PCP theorem: Formulation 2) There exists a constant $\rho$ such that $0 \leq \rho \leq 1$, and for all languages $L$ in $\text{NP}$, there is a polynomial time computable function $f_L: \{0, 1\}^* \rightarrow \{0, 1\}^*$ (we will think of the output of $f_L$ as a 3CNF), such that

- $x \in L \Rightarrow \text{val}(f_L(x)) = 1$
- $x \notin L \Rightarrow \text{val}(f_L(x)) \leq \rho$
As a corollary, it follows that unless $P = NP$, there is a $\rho$ such that $0 \leq \rho \leq 1$ and there is no $\rho$ approximation algorithm for MAX3SAT. Hence, we will say that it is NP hard to approximate MAX3SAT for this $\rho$.

Having stated the two versions of the PCP theorem, our goal now is to show that Theorem 1 and Theorem 2 are equivalent to each other. To show this, we will define a slightly more general looking version of Theorem 2 and then show that all the three of these are equivalent.

To this end, let the define the notion of constraint satisfaction problems.

**Definition 3** ($q$-CSP) A $q$-CSP instance $\phi$ on $n$ variables and $m$ constraints, is a collection of constraints $\phi_1, \phi_2, \ldots, \phi_m$ such that for each $i \in [m],$

- $\phi_i$ is a function from $\{0,1\}^n \rightarrow \{0,1\}$
- There exist locations $i_1, i_2, \ldots, i_q \in [n]$ and a function $f_i : \{0,1\}^q \rightarrow \{0,1\}$, such that for each $u \in \{0,1\}^n$, $\phi_i(u) = f_i(u_{i_1}, u_{i_2}, \ldots, u_{i_q})$. Here, $u_{i_j}$ is the $j^{th}$ bit of $u$.

An assignment $u$ is said to satisfy $\phi$ if for every $i \in [m]$, $\phi_i(u) = 1$. $val(\phi)$ is the maximum fraction of constraints satisfiable by any assignment to the variables. Let us now define the gap version of a CSP problem formally. This will serve as the link between the two formulations of the PCP theorem that we have seen so far.

**Definition 4** For a natural number $q$ and a $\rho \in [0,1]$, the $\rho$-Gap $q$-CSP problem takes as input a $q$-CSP instance $\phi$ and the goal is to determine whether $val(\phi) = 1$ or $val(\phi) < \rho$.

Note that it is assumed that the input $\phi$ is either satisfiable or its value is at most $\rho$. We will assume that the input is promised to be of one of these two types. We will now state the third version of the PCP theorem, which we will then argue to be equivalent to the other two versions stated above.

**Theorem 3** (PCP theorem: Formulation 3) There exists a natural number $q$, and a $\rho \in [0,1]$ such that $\rho$-Gap $q$-CSP problem is NP hard.

We will now argue that all the three formulations of the theorem are equivalent. First, let us show that Theorem 2 and Theorem 3 are equivalent. Observe that the problem MAX3SAT is a special instance of the 3-CSP problem. Hence, Theorem 2 immediately implies Theorem 3. We will now prove the other direction.

**Claim 4** Theorem 3 $\Rightarrow$ Theorem 2.

**Proof** Theorem 3 implies that there exists a natural number $q$ and a constant $\rho \in [0,1]$, such that $\rho$-Gap $q$-CSP is NP hard. Let us consider an instance $\phi$ of $q$-CSP on $n$ variables and with $m$ constraints. We will now reduce it to a 3 CNF instance while preserving the gap. The ideas are similar although simpler to some of what we used in the proof of the Cook-Levin Theorem. Let us consider a constraint $\phi_i$ for some $i \in [m]$. Let us look at the corresponding function $f_i$ on $q$ bits. This function can be represented as a $q$-CNF with $2^q$ clauses. Attaching appropriate lables to the variables, we will have a $q$-CNF instance $\phi_i$ such that for any assignment $u$, $\phi_i(u) = 1$ if and only if $\phi_i(u)$ equals 1. In particular, if $\phi(u) = 1$, then all $2^q$ clauses of $\phi_i$ are satisfiable, else at most $2^q - 1$ of them are satisfiable. Applying this transformation to all the constraints $\phi_i$ and taking their conjunction, we will have an instance of $q$-CNF with $2^q m$ clauses, such that if for any assignment $u$, if $u$ satisfies $\phi$, then $u$ satisfies $\phi'$ and if $u$ leaves $\epsilon$ fraction of constraints unsatisfied in $\phi$, then $u$ leaves at least $\frac{\epsilon}{2^q}$ fraction of clauses unsatisfied in $\phi'$. For a constant $q$, this is a gap preserving reduction from $q$-CSP to $q$-CNF. To get to 3-CNF, we will use a natural strategy to convert a $q$-CNF instance to a 3-CNF instance, by introducing at most $q$ new variables for each clause. We will map each clause in $q$ literals to a conjunction of $q$ clauses each in 3 literals, while preserving satisfiability. We construct an instance of 3-CNF $\phi$ from $\phi'$ by this transformation. An argument similar to the one above, shows that if $\phi'$ was satisfiable, then so was
\[ \hat{\phi}. \] Similarly, if an assignment leaves \( \epsilon \) fraction of all clauses unsatisfied in \( \phi' \), then it leaves at least \( \frac{\epsilon}{7} \) fraction of all clauses unsatisfied in \( \hat{\phi} \). Hence, if there is no polynomial time \( \rho \) approximation algorithm for \( q \)-CSP, then there is a constant \( \delta \), dependent upon \( \rho \) such that it is \( \text{NP} \) hard to approximate \( \text{MAX3SAT} \) within a factor \( \delta \). ■

We will now claim that the Theorem 1 is equivalent to Theorem 3.

**Claim 5** Theorem 1 \( \iff \) Theorem 3.

**Proof** First we show Theorem 1 \( \implies \) Theorem 3. It is enough to show a reduction from 3-SAT to \( 1/2 \text{GAP}q \)-CSP.

We assume \( 3 \text{-SAT} \in \text{PCP}(\log(n), 1) \). Then \( \exists \) poly-time \( V \) that makes \( q \) queries into \( T \) using \( c \log(n) \) random bits. For fixed \( x, r \), let \( V_{x,r} \) be the function that takes as input the proof \( T \), and outputs 1 iff \( V \) accepts \( T \) on input \( x, r \).

First, note that \( V_{x,r} \) depends on \( \leq q \) locations of \( T \). Now \( \forall x \) consider \( \{V_{x,r}\}_{r \in (0,1)^{c \log(n)}} \) \( (q \text{-CSP}) \).

If \( x \in L \), \( \exists T \), s.t. \( \forall r \) \( V \) accepts \( \Rightarrow T \) satisfies all \( V_{x,r} \) \( \Rightarrow \) \( \exists \) Val(\{\{V_{x,r}\}\}) = 1

If \( x \notin L \), \( \forall T \) at least 1/2 of all comparisons in \( \{V_{x,r}\} \) are unset \( \Rightarrow \) Val(\{\{V_{x,r}\}\}) \leq 1/2.

Now we show Theorem 3 \( \Rightarrow \) Theorem 1. We assume \( \exists \rho, q \) s.t. \( \rho \text{-GAP} q \text{-CSP} \) is \( \text{NP} \)-hard.

\( \forall L \in \text{NP}, \exists \) poly-time \( f \) s.t.

\[
\begin{align*}
x \in L & \Rightarrow \text{Val}(f(x)) = 1 \\
x \notin L & \Rightarrow \text{Val}(f(x)) < \rho
\end{align*}
\]

We want to construct a \((c \log(n), q)\)-PCP verifier for \( L \). \( V \) will expect \( T \) to be a satisfying assignment for \( f(x) \) (which contains \( m \) clauses and \( n \) variables), and \( V \) will pick \( i \in [m] \) at random and query the \( q \) variable locations in \( T \) that \( \phi_i \) depends on. It will output 1 if the assignment satisfies \( \phi_i \) and 0 otherwise.

If \( x \in L \), \( \exists T \) that makes \( V \) accept with probability \( = 1 \)

If \( x \notin L \), \( \forall T \), \( V \) accepts with probability \( < \rho \)

Therefore completeness \( = 1 \), and soundness \( \leq \rho \). We can make the soundness \( \leq 1/2 \) by repeating. ■

## 3 Proving a weaker PCP-Theorem

While all statements of the PCP-Theorem are equivalent, it is still difficult to prove any of them. Therefore, we will prove a much weaker version of Theorem 1.

**Theorem 6** \( \text{NP} \subseteq \text{PCP}(\text{poly}(n), 1) \)

**Corollary 7** \( \exists \) exponential length PCP’s for 3-SAT

We will prove the corollary, which will provide a proof for the theorem. But that will come next week.

First, some definitions and claims.

## 4 Hadamard codes

**Definition 8 (Hadamard Code)** \( H : \{0,1\}^n \rightarrow \{0,1\}^{2^n} \), where \( H(u) = \langle u, x \rangle_{x \in \{0,1\}^n} \). In other words, \( H(u) \) returns the truth table of the linear function \( f \) where the \( i \)th bit of \( u \) for the \( i \)th coefficient of \( f \), and each value of \( x \) can take on a different value in \( \{0,1\}^n \).

If \( u_1 \neq u_2 \), then \( H(u_1) \) and \( H(u_2) \) differ on at least half their coordinates. So \( \delta(H(u_1), H(u_2)) \geq 1/2 \).

To see this, note that since \( u_1 \neq u_2 \), at least one bit of \( u_1 \) differs from \( u_2 \). So for every place where the bits differ, the values of \( H(u_1) \) and \( H(u_2) \) will differ where the \( x \)s corresponding to the differing bits are 1, which is half the time for each \( x \).
4.1 Local decoding

**Definition 9 (ε-close/far)** Suppose \( f : \{0,1\}^n \rightarrow \{0,1\}, g : \{0,1\}^n \rightarrow \{0,1\} \). \( \delta(f,g) = \Pr_x[f(x) \neq g(x)] \). If \( \delta(f,g) < \epsilon \), we say \( f \) and \( g \) are \( \epsilon \)-close. If \( \delta(f,g) \geq \epsilon \), we say \( f \) and \( g \) are \( \epsilon \)-far.

**Claim 10** Suppose \( f : \{0,1\}^n \rightarrow \{0,1\}^n \) s.t. \( f \) is \( 1/4 \)-close to a codeword of the Hadamard code. This uniquely determines that codeword from \( f \).

Our next issue is the efficiency of determining the codeword. Given \( f \) that is \( \epsilon \)-close to the Hadamard code (i.e. \( \exists \) some linear function \( g \) s.t. \( \delta(f,g) < \epsilon \)), can we efficiently recover \( g(x) \) with very high probability for any values of \( x \) with access to only \( f \)? For all the \( x \) values where \( f(x) = g(x) \), there’s not any work for us to do. But wherever \( f(x) \neq g(x) \), we need to query other random points in \( f \) to try and determine \( g(x) \).

**Claim 11** We can determine any \( g(x) \) with constantly many queries into \( f \).

**Proof** We will use the following simple algorithm to try and find \( g(x) \):

1. Pick \( y \in \{0,1\}^n \)
2. Query \( f(y), f(x+y) \)
3. Output \( f(y) + f(x+y) \)

What can we say about our algorithm? Since \( f \) and \( g \) are \( \epsilon \)-close, \( f(y) = g(y) \) with probability \( \geq 1 - \epsilon \). Likewise, \( f(x+y) = g(x+y) \) with probability \( \geq 1 - \epsilon \). So with probability \( \geq 1 - 2\epsilon \), \( f(y) + f(x+y) = g(y) + g(x+y) = g(x) \). Obviously, we can boost the probability of success by repeating this test and taking the majority result.

4.2 Local testing

Given \( f : \{0,1\} \rightarrow \{0,1\} \), we want to “test” if \( f \) is the truth table of a linear function (if \( f \) is a Hadamard code). Alternatively, we want to ”test” if \( f \) is linear or far from linear with only a few number of queries.

We want completeness = 1 (if \( f \) linear ⇒ accept w.p. = 1) and soundness \( \leq \rho \) (if \( f \) \( \epsilon \)-far from linear ⇒ accept w.p. \( \leq \rho \)) with only \( O(1) \) queries.

One suggestion is to pick random \( x, y \) and check if \( f(x) + f(y) = f(x+y) \). It is an exercise to verify that this holds for all \( x, y \) iff \( f \) is linear. So if \( f \) is linear, we accept with probability = 1. But if \( f \) is \( \epsilon \)-far from linear, how often will the test accept?

**Theorem 12** This test works.