1 Hardness of Approximation

Theorem 1 (PCP Theorem). Every language $L$ in $\text{NP}$ has a PCP verifier using $O(\log n)$ random coin tosses and making $q = O(1)$ queries to its certificate.

Last time we showed that the PCP Theorem is equivalent to the following statement: There exists a constant $q$ such that $\text{Gap}_{1/2} \text{MAX}_q \text{CSP}$ is $\text{NP}$-hard.

Using the idea behind the standard $\text{NP}$-hardness reduction from $\text{SAT}$ to $\text{3SAT}$, this in turn is equivalent to the statement: There exists a constant $\rho < 1$ such that $\text{Gap}_{\rho} \text{MAX}_3 \text{SAT}$ is $\text{NP}$-hard.

Just as how we used the $\text{NP}$-hardness of the exact version of $\text{3SAT}$ to prove a host of other $\text{NP}$-hardness results, so too can we use the hardness of the approximation version. For instance:

Theorem 2. For every constant $\rho \in (0, 1)$, the problem $\text{MAX} - \text{INDSET}$ is $\text{NP}$-hard to approximate within a factor of $\rho$.

The quantifier “for every” $\rho$ here is interesting, because the approximation problem becomes easier as $\rho \to 0$.

Proof. First, we’ll use the standard $\text{NP}$-hardness reduction from $\text{3SAT}$ to $\text{INDSET}$ to show this is true for some $\rho < 1$. Then we’ll “amplify” the approximation gap to show that the statement is true for every $\rho$.

**Part 1:** Let $\rho < 1$ be such that $\text{Gap}_{\rho} \text{MAX}_3 \text{SAT}$ is $\text{NP}$-hard. Let $f$ be the standard $\text{NP}$-hardness reduction from $\text{3SAT}$ to $\text{INDSET}$. This reduction has the property that, for every $3\text{CNF}$ formula $\varphi$, we have

$$\text{val}(\varphi) \geq k \iff f(\varphi) \text{ has an independent set of size } \geq k.$$ 

Thus, if $\varphi$ is a formula with $m$ clauses, we have

$$\text{val}(\varphi) = m \implies \text{IS}(f(\varphi)) = m$$
$$\text{val}(\varphi) < \rho m \implies \text{IS}(f(\varphi)) < \rho m$$

where $\text{IS}(G)$ is the size of the largest independent set in $G$. So it is $\text{NP}$-hard to approximate $\text{IS}$ to within a factor of $\rho$. 

1
Part 2: Now we reduce the problem of $\rho$-approximating the largest independent set in a graph $G$ to that of $\rho^k$-approximating the largest independent set in a new graph $G^k$. The new graph $G^k$ is constructed (in poly-time) from $G$ as follows:

Vertices of $G^k$: All subsets of $k$ vertices of $G$

Edges of $G^k$: $S \sim T$ iff $S \cup T$ is not independent in $G$.

\[
G = \begin{array}{c}
1 & 2 \\
3 & 4
\end{array}
\quad
G^2 = \begin{array}{c}
1, 2 & 1, 3 \\
2, 3 & 2, 4 \\
1, 4 & 3, 4
\end{array}
\]

Let $T$ be a maximum-size independent set in $G$. Then one can check that the largest independent set in $G^k$ is:

$$\{S \subset T \mid |S| = k\}.$$ 

Thus we have

$$\text{IS}(G^k) = \left(\text{IS}(G)\right)^k \approx \left(\text{IS}(G)\right)^k.$$ 

Thus, we have

$$\text{IS}(G) = m \implies \text{IS}(G^k) \approx m^k$$

$$\text{IS}(G) < \rho m \implies \text{IS}(G^k) \lesssim (\rho m)^k = \rho^k m^k.$$ 

We can make the gap $\rho^k$ smaller than any constant by taking $k$ to be a sufficiently large constant.

2 PCP Mini

“The” PCP Theorem says $\text{NP} = \text{PCP}(r(n) = \log n, q(n) = 1)$, where $r(n)$ represents the number of verifier coin tosses, and $q(n)$ is the number of queries. Recall that WLOG, we can always take the length of the proof to be $2^{q(n)} \cdot r(n) = \text{poly}(n)$ in this statement.

Today, we’ll prove the following “mini” (or maybe “mega” depending on how you look at it) version of the PCP Theorem:

Theorem 3 (PCP Mini). $\text{NP} \subseteq \text{PCP}(\text{poly}(n), 1)$. 

That is, every language in \( \text{NP} \) has a PCP system with a exponentially long proofs. It suffices to design such a PCP system for the \( \text{NP} \)-complete problem

\[ \text{QUAD} = \{ \text{satisfiable systems of quadratic equations over } \mathbb{Z}_2 \} \]

**Example 4.** An instance of QUAD looks like the following:

\[
Q = \begin{cases} 
    x_1 x_2 + x_3 x_4 & = 1 \\
    x_2 x_3 & = 0 \\
    x_1 x_2 + x_2 x_4 & = 0.
\end{cases}
\]

This system is satisfiable, say, with satisfying assignment \((x_1, x_2, x_3, x_4) = (1, 1, 0, 1)\).

To see this is \( \text{NP} \)-complete, we can reduce from CKT−SAT as follows: Label each gate with a distinct variable, and enforce consistency with the gates feeding into it using a quadratic equation. For instance, if a gate requires \( z = x \vee y \), add the constraint \((1 - x)(1 - y) + z = 1\).

### 2.1 PCP for QUAD

**Random subset sum principle:** If \( u \neq v \in \mathbb{Z}_2^n \), then

\[
\Pr_{x \sim \mathbb{Z}_2^n}[\langle u, x \rangle = \langle v, x \rangle] = \frac{1}{2}.
\]

That is, if \( u \) and \( v \) are distinct bit vectors, then the probability that the XOR of the same random subset of bits from \( u \) and from \( v \) agree is 1/2.

**Example 5.** Let \( u = 1011 \), \( v = 1001 \). Then \( \langle u, x \rangle = \langle v, x \rangle \) if and only if \( x_3 = 0 \), which happens with probability 1/2.

We’ll use this principle in a few places in our PCP construction, but the main use is as follows: If \( u \) fails to solve a system \( Q \), then it fails to solve a random linear combination of the constraints with probability 1/2.

The honestly generated proof \( \pi \) for an instance \( Q \) of QUAD will consist of the values of all \( 2^n \) quadratic functions of a satisfying assignment to \( Q \). That is,

\[
\pi = \left( \sum_{i,j} A_{ij} u_i u_j \right)_{A \in \mathbb{Z}_2^n} = \left( \langle A, uu^\top \rangle \right)_{A \in \mathbb{Z}_2^n} =: \text{WH}(uu^\top)
\]

which is called the “Walsh-Hadamard” encoding of the \( n \times n \) matrix \( uu^\top \).

Here’s the gameplan for probabilistically verifying an alleged proof \( \pi^* \):

1. **Linearity Test:** Check (in a manner we’ll describe later) that \( \pi^* \) is “close” to \( \pi := \text{WH}(vv^\top) \) for some \( v \in \mathbb{Z}_2^n \).
2. **Random Subset Sum:** Take a random linear combination of the constraints in \( Q \) to obtain a single quadratic equation \( Ax = b \).
3. **Local Decoding:** Compute \( \pi(A) \) using a constant number of probes to \( \pi^* \), and check that it equals \( b \).
2.2 Linearity Testing

**Definition 6.** For a vector \( v \in \mathbb{Z}_2^m \), the Walsh-Hadamard encoding \( WH(v) \) is the truth table of the linear function \( f : \mathbb{Z}_2^m \to \mathbb{Z}_2 \) defined by \( f(x) = \langle x, v \rangle \). (This is just a \( 2^m \)-bit vector.)

In the linearity testing problem, we are given query access to a function \( \hat{f} : \mathbb{Z}_2^m \to \mathbb{Z}_2 \), and our goal is to test whether it is “close” to some linear function \( f(x) = \langle x, v \rangle \).

Note that a function \( \hat{f} \) is linear if and only if \( \hat{f}(x + y) = \hat{f}(x) + \hat{f}(y) \) for all \( x, y \in \mathbb{Z}_2^m \). The BLR (Blum-Luby-Rubinfeld) Test checks the global property of linearity of a function \( \hat{f} \) by just checking whether this identity holds for a random pair \( x, y \):

**BLR Test:** Pick random \( x, y \leftarrow \mathbb{Z}_2^m \) and check that \( \hat{f}(x + y) = \hat{f}(x) + \hat{f}(y) \).

This test has the following guarantees:

**Definition 7.** Two functions \( f, g : \mathbb{Z}_2^m \to \mathbb{Z}_2 \) are \( \delta \)-close if \( \Pr_{x \in \mathbb{Z}_2^m} [f(x) = g(x)] \geq 1 - \delta \).

**Completeness:** If \( \hat{f} \) is linear, then \( \Pr[\hat{f}(x + y) = \hat{f}(x) + \hat{f}(y)] = 1 \).

**Soundness:** If \( \hat{f} \) is not \( \delta \)-close to any linear function, then \( \Pr[\hat{f}(x + y) = \hat{f}(x) + \hat{f}(y)] \leq 1 - \Omega(\delta) \).

Note that this test requires evaluating \( \hat{f} \) at only 3 random locations. Moreover, soundness can be amplified through repetition, at the expense of increasing the number of queries to \( \hat{f} \).

2.3 Local Decoding

Suppose we know that \( \hat{f} \) is \( \delta \)-close to some linear function \( f \). (Note that if \( \delta < 1/4 \), then this function \( f \) is unique.)

**Claim 8.** There is an (efficient) algorithm that computes \( f(x) \) (with high probability) using \( O(1) \) probes to \( \hat{f} \).

**Decoding Procedure:** Pick a random \( y \leftarrow \mathbb{Z}_2^m \) and compute \( \hat{f}(x + y) - \hat{f}(x) \).

**Analysis:** Observe that for every \( x \), the point \( x + y \) (for uniformly random \( y \)) is itself uniformly random. Therefore, by a union bound,

\[
\Pr[\hat{f}(x + y) - \hat{f}(y) \neq f(x)] \leq \Pr[\hat{f}(x + y) \neq f(x + y)] + \Pr[\hat{f}(y) \neq f(y)] \leq 2\delta.
\]

4
2.4 Fixing a Lie

The linearity test we described is able to determine (whp) whether $\pi^*$ is close to $\text{WH}(M)$ for some $M \in \mathbb{Z}_2^{n \times n}$. But how do we ensure that this $M$ takes the form $uu^\top$ for some $u \in \mathbb{Z}_2^n$?

**Solution:** We’ll enable the verifier to check this by also including an encoding $g$ of $u$ itself as part of the proof.

**Test:** Given $f$ and $g$ (alleged to encode $uu^\top$ and $u$, respectively), pick $r, s \leftarrow \mathbb{Z}_2^n$ uniformly and test if $f(rs^\top) = g(r)g(s)$.

**Completeness:** If $f = \text{WH}(uu^\top)$ and $g = \text{WH}(u)$, then

$$f(rs^\top) = \sum_{i,j} (rs^\top)_{ij}u_iu_j$$

$$= \sum_{i,j} (r_iu_i)(s_ju_j)$$

$$= \langle r, u \rangle \cdot \langle s, u \rangle$$

$$= g(r)g(s).$$

**Soundness:** If $f = \text{WH}(M)$ and $g = \text{WH}(u)$ for some $M \neq uu^\top$, then

$$\Pr[f(rs^\top) = g(r)g(s)] = \Pr[\langle M, rs^\top \rangle = \langle u, r \rangle \cdot \langle u, s \rangle]$$

$$= \Pr[\langle s, Mr \rangle = \langle s, uu^\top r \rangle]$$

$$= \frac{1}{2} + \frac{1}{2} \Pr[Mr \neq uu^\top r]$$

$$= \frac{3}{4}.$$