**E0 224 Computational Complexity Theory** **Indian Institute of Science**
**Fall 2014** **Department of Computer Science and Automation**

# Lecture 27: Nov 17, 2014

*Lecturer: Chandan Saha* <chandan@csa.iisc.ernet.in> *Scribe: Bibaswan & Sarath*

In the previous few lectures, we were introduced to the PCP theorem and were shown an alternative hardness of approximation view of the theorem. The PCP theorem implies the hardness of approximation results for many problems. In the last lecture we saw a strong hardness of approximation result for MAX-INDSET. In this lecture we will see a weaker result for MIN-VERTEX-COVER. We will also start the proof of a weaker PCP theorem.

## 27.1 Hardness of approximation of MIN-VERTEX-COVER

**Theorem 27.1.** *There is a constant $\rho' > 1$ s.t. $\rho'$-approximation of MIN-VERTEX-COVER is NP-hard.*

*Proof.* We already know from PCP theorem that $\rho$-approximation of MAX-3SAT is NP-hard where $\rho$ is some constant $< 1$. Now similar to the proof of hardness of approximation for MAX-INDSET given in the previous lecture, given an instance $\varphi$ of MAX-3SAT, we construct a graph $G_\varphi$ as follows:
Assuming $\varphi$ has $m$ clauses and $n$ variables, where each clause has three literals, for each clause we will create a cluster of 7 vertices, each vertex represents one of the 7 possible satisfying assignments for that clause. Thus we will have a total of $t = 7m$ vertices in $G_\varphi$. Within each cluster, there will be edges between each of the 7 vertices among each other. For any two clusters, there will be an edge between the vertex in one cluster to a vertex in another cluster if and only if the two assignments the vertices represent are conflicting.
As already stated in the previous lecture, $val(\varphi)$ is the maximum fraction of clauses that can be satisfied by any assignment for $\varphi$ iff $G_\varphi$ has a MAX-INDSET of size $val(\varphi)m$
Now, if $G_\varphi$ has a MAX-INDSET of size $val(\varphi)m$ or $val(\varphi)\frac{t}{7}$ then it has a MIN-VERTEX-COVER of size $(t - val(\varphi)\frac{t}{7})$.
Consider a YES instance of $\varphi$. In that case $val(\varphi) = 1$ and hence, MIN-VERTEX-COVER($G_\varphi$) $= t - \frac{t}{7} = \frac{6t}{7}$.
Consider a NO instance of $\varphi$. In that case $val(\varphi) < \rho$ and hence, MIN-VERTEX-COVER($G_\varphi$) $\geq t - \frac{\rho t}{7} = \frac{(7-\rho)t}{7}$.
Now, suppose we have a $\rho'$-approximation algorithm for MIN-VERTEX-COVER. In that case to distinguish between the YES and NO instances of $\varphi$, we would want $\rho' \frac{6t}{7} \leq \frac{(7-\rho)t}{7}$ which implies $\rho' \leq \frac{(7-\rho)}{6}$.
Thus we obtain a $\rho'$ for which $\rho'$-approximation of MIN-VERTEX-COVER is NP-hard. □

Due to the *unique games conjecture* some people believe, the best efficient approximation algorithm for MIN-VERTEX-COVER is a 2-approximation [SKhot08].

## 27.2 Proof of PCP theorem (weaker version)

**Theorem 27.2.** ***(Exponential size PCP system for NP)****[ALM+]* $NP \subseteq PCP(poly(n), 1)$

**Proof idea:** We already know that language QUADEQ is NP-Complete (given in assignment 1). If we can show that QUADEQ $\subseteq PCP(poly(n), 1)$ then we are done. To show this we show that there exists a PCP system for QUADEQ.

The PCP $(poly(n), 1)$-verifier expects the proof to contain an encoded version of the original proof, which is a satisfying assignment for the input instance of QUADEQ. The verifier checks the such an encoded certificate by simple probabilistic tests.
**Note:** The size of the proof can be *exponential* in size of the problem size. As long as the verifier makes only a constant number of queries to the proof and uses $O(poly(n))$ random bits, we are fine.

### 27.2.1 Some preliminaries:

**Definition 27.3.** ***Tensor Product of two vectors:*** *Let $x = (x_1, x_2, \ldots, x_n) \in \mathbb{F}_2^n$, $y = (y_1, y_2, \ldots, y_n) \in \mathbb{F}_2^n$ be two vectors. Then $x \otimes y$ is defined as*

$$x \otimes y = \begin{bmatrix} x_1y_1 & x_1y_2 & \ldots & x_1y_j & \ldots & x_1y_n \\ x_2y_1 & x_2y_2 & \ldots & x_2y_j & \ldots & x_2y_n \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ x_iy_1 & x_iy_2 & \ldots & x_iy_j & \ldots & x_iy_n \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ x_ny_1 & x_ny_2 & \ldots & x_ny_j & \ldots & x_ny_n \end{bmatrix}$$

*$x \otimes y$ can be thought of as a vector of size $n^2$.*

**Definition 27.4.** ***Walsh-Hadamard code:*** *Let $z \in \{0,1\}^n$. Then W-H code of z is denoted by the $2^n$ length vector $f_z$*

$$f_z = (z.r)_{r \in \{0,1\}^n}$$
$$where\ (z.r) = \sum_{i=1}^{n} z_i r_i$$

*Since $r \in \{0,1\}^n$, each $r$ denotes a subset $S \subseteq [n]$ s.t. the $i^{th}$ element in $[n]$ is selected in $S$ if the bit $r_i = 1$. Thus $(z.r) = \sum_{i \in S} z_i$, $S \subseteq [n]$ defined by r.*

### 27.2.2 Proof construction:

Now we explain the construction of the PCP proof system for QUADEQ. Let us consider an instance of QUADEQ as a model example:

$$\begin{aligned} u_1u_2 + u_3u_5 + u_4 &= 1 \\ u_1 + u_3u_4 &= 0 \\ u_2 + u_5u_4 &= 1 \end{aligned} \tag{27.1}$$

The proof of satisfiablity of the above QUADEQ instance will consist of a vector $u \in \{0,1\}^5$ denoting a satisfying assignment of the variables $u_1, u_2, \ldots, u_5$. A polytime DTM can simply check whether each of the clauses is satisfied by putting the appropriate values to the variables. However this will require the verifier to read the entire certificate. But we want to read only a constant number of locations of the proof and verify the proof with high probability.
So in our case the prover has to supply the verifier a special encoded version of vector $u$ such that the verifier with $p(n)$ random coins can read only a constant number of locations of the encoded proof and can decide with high probability if the proof is correct or not as required by the PCP theorem. So we will show that such an encoded proof does exist and demonstrate how the verifier will check the proof.

**PCP certificate description:** The certificate $\pi$ will consist of two parts $g_1$ and $g_2$, where $g_1 = f_u \in \{0,1\}^{2^n}$ and $g_2 = f_{u \otimes u} \in \{0,1\}^{2^{n^2}}$.

| $f_u$ | $f_{u \otimes u}$ |
|---|---|
| $g_1$ | $g_2$ |

That is, the proof essentially consists of Walsh-Hadamard encoding of the string $u$ and the string $u \otimes u$ (note that the tensor product can be interpreted as just a vector instead of a matrix). Thus the total length of the proof will be $2^n + 2^{n^2}$.

The main challenge in proving this weaker version of PCP theorem is to verify the proof using *constant* number of queries. There are two parts in verifying the given proof. First, the verifier needs to verify that the first part of given proof is indeed a Walsh-Hadamard encoding of some string $u$ and second part of the proof is an encoding of $u \otimes u$ (of course, with constant number of queries to the proof). After this, the verifier needs to verify that $u$ is a satisfying assignment of the given QUADEQ instance.

First, we describe how we can do the second part. That is, we assume that the given proof is valid (i.e., we assume that the first part of the given proof is a Walsh-Hadamard encoding of some string $u$ and the second part is an encoding of $u \otimes u$), and we describe a PCP verifier.

Observe that the Walsh-Hadamard encoding of $u \otimes u$ essentially consists of sums of the form $\sum_S u_i u_j$ where the set $S$ is a subset of $[n] \times [n]$. Also, note that the LHS of the equations in the QUADEQ instance are of the same form. In other words, the LHS of equations are present at various locations of Walsh-Hadamard encoding of $u \otimes u$. However, we cannot read each one of those locations in the given proof and verify with the RHS of equations, since we are allowed to read constantly many locations in the proof.

More interestingly, observe that any linear combination (over $\mathbb{F}_2$) of LHS of the given set of equations is again in the form of LHS of a quadratic equation, and hence will be present at some location in the Walsh-Hadamard encoding of $u \otimes u$. This gives an idea as to how to read the proof using constantly many queries.

For the sake of explanation, let us consider our QUADEQ instance 27.1. Let

$$w = \begin{pmatrix} u_1u_2 + u_3u_5 + u_4 \\ u_1 + u_3u_4 \\ u_2 + u_5u_4 \end{pmatrix} \tag{27.2}$$

and

$$b = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} \tag{27.3}$$

Then the verifier need to check whether $w = b$ by making constant number of queries to the proof. Let $m$ denote the number of equations in the QUADEQ instance.

First, we show the following lemma:

**Lemma 27.5.** *(Random Subsum Lemma) If $w \neq b$ then*

$$\mathbb{P}_{r \in_R \{0,1\}^m}(w.r = b.r) = \frac{1}{2} \tag{27.4}$$

*Proof.* Observe that

$$\mathbb{P}_{r \in_R \{0,1\}^m}(w.r = b.r) = \mathbb{P}_{r \in_R \{0,1\}^m}((w+b).r = 0)$$

Now, consider $w + b$. If $w \neq b$, then $w + b \neq 0$. Let $v = w + b$. Then

$$(w + b).r = \sum_{i=1}^{m} v_i r_i$$

In the above expression, the RHS will be a sum (over $\mathbb{F}_2$) of $r_i$'s depending on the non-zero positions of the string $w + b$. Let the above sum be $r_{i_1} + \cdots + r_{i_k}$ (where $k$ is the number of terms in the sum). Now, the total number of $m$-length strings is $2^m$. We need to find the fraction of strings (i.e., the probability) that makes the above expression $0$. Note that, the degrees of freedom in choosing the $r_i$'s that make the above sum zero is $(m - 1)$, i.e., if we fix $(k - 1)$ number of terms in the sum, then the $k$ th term automatically gets fixed to make the sum $0$. Therefore,

$$\begin{aligned}\mathbb{P}((w + b).r = 0) &= \frac{2^{m-1}}{2^m} \\ &= \frac{1}{2}\end{aligned}$$

□

Now, to check whether $w = b$, the verifier picks a random string $r \in \{0, 1\}^m$, and computes $w.r$ and $b.r$. Since $w.r$ is again in the form of LHS of a quadratic equation, it queries the proof corresponding to the location $w.r$ (note that $w.r$ will be present in some location in the Walsh-Hadamard encoding of $u \otimes u$). If the given system of equations is satisfied by $u$, then clearly $w.r = b.r$ and the verifier accepts. If not, then we have $w \neq b$ and by the above lemma, the verifier rejects with probability $1/2$.

**Remark:** We can reduce the probability of making an error by increasing the number of queries (but constantly many queries) to the proof by using independently picked random strings $r$.

Thus, assuming that the given proof is indeed valid, we have a PCP verifier for QUADEQ that uses polynomially many random bits and constant number of queries to the proof. The remaining task is to verify that the proof is indeed valid. That is, we need to do the following:

1. First, the verifier needs to ensure that $g_1 = f_u$ for some $u \in \{0, 1\}^n$ and $g_2 = f_w$ for some $w \in \{0, 1\}^{n^2}$.
2. Second, the verifier needs to ensure that $w = u \otimes u$.

Towards this task, we make the following definitions:

**Definition 27.6.** *(Linear functions on q-length strings) A function $h : \{0, 1\}^q \to \{0, 1\}$ is a linear function if for every $x, y \in \mathbb{F}_2^q$, $h(x + y) = h(x) + h(y)$, where addition $(+)$ is over $\mathbb{F}_2$.*

**Definition 27.7.** *(Equivalent definition of a linear function on q-length strings) A function $h : \{0, 1\}^q \to \{0, 1\}$ is a linear function if $\exists S \subseteq [q]$ such that for every $x = (x_1, \ldots, x_n) \in \mathbb{F}_2^q$,*

$$h(x) = \sum_{i \in S} x_i \tag{27.5}$$

First, we show the following claim:

**Claim 27.8.** *The above definitions of linear functions are equivalent*

*Proof.* It is straightforward to show that Definition 27.7⇒ Definition 27.6. By Definition 27.7, there exists $S \subseteq [q]$ such that

$$h(x) = \sum_{i \in S} x_i \tag{27.6}$$

for all $x \in \mathbb{F}_2^q$. Therefore, for all $x, y \in \mathbb{F}_2^q$, we have

$$\begin{aligned} h(x+y) &= \sum_{i \in S} (x_i + y_i) \\ &= \sum_{i \in S} x_i + \sum_{i \in S} y_i \\ &= h(x) + h(y) \end{aligned} \tag{27.7}$$

where the last equality makes use of Definition 27.7. Note that this is precisely Definition 27.6. To show the converse, we represent a vector in terms of standard basis vectors, i.e., we write $x$ as

$$x = x_1(1, 0, \ldots, 0)^T + x_2(0, 1, 0, \ldots, 0)^T + \cdots + x_q(0, 0, \ldots, 0, 1)^T \tag{27.8}$$

Then, by using Definition 27.6, for any $x \in \mathbb{F}_2^q$, we have

$$h(x) = \sum_{i=1}^{q} x_i h(e_i) \tag{27.9}$$

where $e_i$ is the standard basis vector with a "1" at $i$ th position. Note that the above is true since, if $x_i = 0$ then $h(x_i e_i) = 0$, and if $x_i = 1$ then $h(x_i e_i) = h(e_i)$. Observe that the above expression can be written as a sum of a subset of $x_i$'s depending on the value of $h(\cdot)$ for the basis vector $e_i$'s. That is, there exists a set $S \subseteq [q]$ such

$$h(x) = \sum_{i \in S} x_i \tag{27.10}$$

which is precisely Definition 27.7. □

Having defined linear functions, we show the following:

**Claim 27.9.** *Walsh-Hadamard codewords of length $2^q$ are precisely linear functions on $q$-length strings*

**Remark:** We say that a $2^q$ length string is a Walsh-Hadamard *codeword* if it is a Walsh-Hadamard encoding of some $q$-length string.

*Proof.* For a fixed $q$-length string (say $v$), the Walsh-Hadamard codeword is obtained by taking the dot product of $v$ with all $q$-length strings. That is, the codeword is given by

$$\begin{aligned} ((v.r))_{r \in \{0,1\}^q} &= \left( \sum_{i=1}^{q} v_i r_i \right)_{r \in \{0,1\}^q} \\ &= \left( \sum_{i \in S_v} r_i \right)_{r \in \{0,1\}^q} \end{aligned}$$

where $S_v \subseteq [q]$ depending on $v$. But, using Definition 27.7 of linear functions, we see that the sum in the above expression is precisely a linear function on $q$-length string $r$ that is defined by $S_v$. Thus, we conclude that Walsh-Hadamard codewrords of length $2^q$ are precisely linear functions on $q$-length strings. □

Thus, we have a nice characterization of Walsh-Hadamard codewords of length $2^q$ in terms of linear functions on $q$-length strings. Now, let us recall our PCP verifier's objective. The first step is to check whether the first part of the proof is a valid Walsh-Hadamard codeword, that is, to check if $g_1 = f_u$ for some $u \in \{0,1\}^q$. But, by Claim 27.9, it suffices to check whether the given $2^q$-length string is a truth table for a linear function on $q$-length strings.

But, it is not immediately clear how to do this by using constant number of queries to the proof. Let us first try a naive strategy. Suppose we pick two strings $x, y \in \{0,1\}^q$ randomly and query the three locations in the proof corresponding to $h(x), h(y)$ and $h(x+y)$. We accept the proof if we find that $h(x) + h(y) = h(x+y)$ and reject if not. Now, if $h$ is a linear function, we accept with probability 1. However, if $h$ is not linear, we need not reject with high probability. To see this, suppose that $h$ is indeed a linear function and we define another function $h'$ by just flipping a single bit of $h$. Then, it is clear that $h'$ is no longer a linear function. But, $h'$ satisfies the condition in Definition 27.6 for every pair of $x$ and $y$ if none of $x$ and $y$ correspond to the bit that is flipped. Therefore, for randomly picked strings $x$ and $y$, the probability that $h'(x) + h'(y) \neq h'(x+y)$ is not high. Hence, this strategy does not server our purpose.

However, it turns out that we need not check whether the given $2^q$ length string is exactly the truth table of a linear function, but it suffices to check whether the given string is *close* to a linear function. In this case, we can pretend the given proof as a slightly corrupted version of a Walsh-Hadamard codeword, and we can do local decoding of this codeword to verify whether $u$ is a satisfying assignment for the QUADEQ instance. Note that even if the given proof is close to a linear function, we can make use of our random subsum lemma. Also, it turns out that one can test whether the given proof is close to a linear function by using constant number of queries to the proof. We will study the details in the next lecture.

# References

[SKhot08] KHOT, SUBHASH and ODED REGEV 'Vertex cover might be hard to approximate to within 2 - $\epsilon$.' *Journal of Computer and System Sciences 74.3 (2008): 335-349.* 2008

[ALM+] ARORA, SANJEEV, CARSTEN LUND, RAJEEV MOTWANI, MADHU SUDAN and MARIO SZEGEDY 'Proof verification and the hardness of approximation problems.' *Journal of the ACM (JACM) 45, no. 3 (1998): 501-555.* 1998

[AB09] ARORA, SANJEEV, and BARAK, BOAZ. *Computational Complexity: A Modern Approach*, Cambridge University Press, 2009.