Computational Complexity November 15, 2018

Lecture 12 (Notes)

Lecturer: Ola Svensson *Scribes: Ola Svensson*

Disclaimer: These notes were written for the lecturer only and may contain inconsistent notation, typos, and they do not cite relevant works. They also contain extracts from the two main inspirations of this course:

1. The book *Computational Complexity: A Modern Approach* by Sanjeev Arora and Boaz Barak;
2. The course *http://theory.stanford.edu/ trevisan/cs254-14/index.html* by Luca Trevisan.

1 Introduction

Recall last lecture:

- Probabilistic Checkable Proofs. **PCP:** A language L is in PCP(r, q) if there exists a probabilistic polynomial time verifier V which, when given an instance x and a proof Π of length $poly(|x|)$, proceeds as follows
  1. Reads x and $O(r)$ random bits.
  2. Based on x and $O(r)$ random bits, queries $O(q)$ bits from Π.
  3. Computes and either accepts or rejects with following properties,
     - **completeness:** If $x \in L$ then $\exists$ proof Π such that $V^{\Pi}(x)$ accepts with probability 1.
     - **Soundness:** If $x \notin L$ the $\forall$ proof Π, the verifier $V^{\Pi}(x)$ accepts with probability at most $\frac{1}{2}$.
- PCP-Theorem $\mathbf{NP} = \mathbf{PCP}(\log n, 1)$.
- Walsh-Hadamard Code: The WH code encodes n bits into 2^n bits. The encoding function WH: $\{0,1\}^n \to \{0,1\}^{2^n}$ maps the input u $\in \{0,1\}^n$ into the truth table of the function $x \odot u = \sum_{i=1}^{n} x_i u_i$ mod 2.
  - Local testability: there is a test that reads $O(1)$ bits that always accept a code word (linear function) and rejects any function that is not 99.99% close to a linear function with probability at least 99.99%.
  - Local decodability: Given a function $f : \{0,1\}^n \to \{0,1\}$ that is 99.99% close to a linear function $\hat{f}$, we can decode $\hat{f}(x)$, for any $x \in \{0,1\}^n$, correctly with probability 99.98% by reading two bits of f.

Today:

- Proof that $\mathbf{NP} \subseteq \mathbf{PCP}(poly(n), 1)$ by giving a $\mathbf{PCP}(poly(n), 1)$ verifier for QUADEQ.

2 Reformulation of The QUADEQ Problem

In order to prove that $\mathbf{NP} \subset \mathbf{PCP}(poly(n), 1)$, we will prove that an **NP**-complete problem has a proof-verifier that uses $O(poly(n))$ random bits and queries $O(1)$ bits from the proof.

Definition 1 *(QUADEQ) Given a system of m quadratic equations over n variables in $\mathbb{F}_2$, decide whether there exists an assignment of the variables such that all equations are satisfied.*

Example 1 *Given the following system:*

$$u_1u_2 + u_3u_1 + u_2u_3 = 1$$
$$u_2u_2 + u_1 = 0$$
$$u_3u_2 + u_2u_1 + u_1u_1 = 1$$

The system is satisfied when $u_1 = u_2 = u_3 = 1$.

Since we are working in $\mathbb{F}_2$, we have that $x = x^2$, hence by replacing any linear term by its square value, we will only have quadratic terms in our equations. For n variables, there are n^2 different possible quadratic terms (There are actually $\frac{n(n+1)}{2}$ different quadratic terms, if we consider that $u_iu_j = u_ju_i$, but it won't make a difference for what we are going to do). Let U be a vector containing the n^2 possible quadratic terms:

$$U = \begin{bmatrix} u_1u_1 \\ u_1u_2 \\ u_1u_3 \\ \dots \\ \\ u_nu_n \end{bmatrix} =: u \otimes u\,.$$

We can then represent the problem with a linear system of n^2 variables $\{U_{1,1}, U_{1,2}, ..., U_{n,n}\}$ with the added constraint that $U_{i,j} = u_iu_j$, i.e., that $U = u \otimes u$.

We can now reformulate our problem QUADEQ with matrices: Given $A \in \mathbb{F}_2^{m \times n^2}$ and $b \in \mathbb{F}_2^m$, we want to find $U \in \mathbb{F}_2^{n^2}$ satisfying: (1) $AU = b$ and (2) $U = u \otimes u$ for some $u \in \{0,1\}^n$.

Example 2 *We can rewrite our previous example as:*

$$\begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} U_{1,1} \\ U_{1,2} \\ U_{1,3} \\ U_{2,1} \\ U_{2,2} \\ U_{2,3} \\ U_{3,1} \\ U_{3,2} \\ U_{3,3} \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} \textit{ and } U = u \otimes u \textit{ for some } u \in \{0,1\}^n.$$

3 PCP($poly(n), 1$) verifier for QUADEQ

3.1 Intuition

The first verifier that we can think of is a one that gets access to a string $(U, u) \in \mathbb{F}_2^{n^2} \times \mathbb{F}_2^n$ and accepts the string as a valid proof if $AU = b$ and $U = u \otimes u$. Clearly, if a QUADEQ instance is satisfiable by a certain assignment $u_1, \dots, u_n$, then the verifier accepts the proof (U, u), where $u = (u_1, \dots, u_n)$ and $U = u \otimes u$. On the other hand, if a QUADEQ instance is not satisfiable, then the verifier rejects any string (U, u). This shows that the suggested verifier is an **NP** verifier for the QUADEQ problem.

Unfortunately, the proof (U, u) is not robust enough in order to use it in a $\mathbf{PCP}(poly(n), 1)$ verifier. Remember that a $\mathbf{PCP}(poly(n), 1)$ verifier reads poly(n) random bits based on which (and on the input) it reads a constant number of bits from the proof, and then it finally decides whether or not to accept the proof as a valid one. Therefore, since we are basing our decision on a constant number of bits of the proof, we have to robustify it. One way to do this is by requiring that a large portion of the proof has to depend on a large number of bits of the original proof (U, u), so that a constant number of bits that is randomly chosen from the proof will be likely to be "representative" of the whole original proof (U, u).

Walsh-Hadamard codes is an excellent tool that can help us achieve this requirement. Therefore, instead of getting access to an (n^2+n)-long string (U, u), our PCP verifier will get access to a $(2^{n^2}+2^n)$-long string $(g, f) \in \mathbb{F}_2^{2^{n^2}} \times \mathbb{F}_2^{2^n}$ (g, resp. f, can be thought of as the truth table a function from $\mathbb{F}_2^{n^2}$, resp. from $\mathbb{F}_2^n$, to $\mathbb{F}_2$, we will identify g and f with those functions). A valid proof will consist of a pair of Walsh-Hadamard codes $g = WH(U)$ and $f = WH(u)$, where $(U, u) \in \mathbb{F}_2^{n^2} \times \mathbb{F}_2^n$ satisfies $AU = b$ and $U = u \otimes u$. Thus, the PCP verifier will check (g, f) in 3 steps:

- Step 1: Check that f and g are codewords (i.e., linear functions).
- Step 2: Check that $g = WH(u \otimes u)$, where $f = WH(u)$.
- Step 3: Check that $U = u \otimes u$ satisfies $AU = b$.

Of course, we have to do these steps by making only a constant number of random queries from (f, g), and we have to do it in such a way that the verifier accepts a valid proof with probability 1, and if the problem instance is not satisfiable then the verifier must reject any proof with a probability of at least $\frac{1}{2}$.

3.2 The Verifier

Given $A \in \mathbb{F}_2^{m \times n^2}$, $b \in \mathbb{F}_2^m$

decide whether there exists a pair (u, U) such that

1. $AU = b$
2. $U = u \otimes u$

The verifier expects as a proof the Walsh-Hadamard encoding of U and u, $WH(U), WH(u)$.

Our verifier will treat the proof as two functions:

$$f : \mathbb{F}_2^{n^2} \to \mathbb{F}_2 \ (WH(U))$$

$$g : \mathbb{F}_2^n \to \mathbb{F}_2 \ (WH(u))$$

Step 1

Do a linearity test that only accepts with probability 0.01 if one of the functions is not 99.99% close to linear.

Step 1 applies this linearity test twice to each of f and g. We have the following:

- If f and g are linear functions, (f, g) passes Step 1 with probability 1.
- If either f or g is not 99%-close to a linear function, (f, g) passes Step 1 with probability of at most $0.01 \leq \frac{1}{2}$, i.e., (f, g) is rejected with a probability of at least $\frac{1}{2}$.
- Step 1 makes $O(1)$ random queries from f and $O(1)$ random queries from g. As the description of f and g contains 2^n and 2^{n^2} bits respectively, we need $O(n^2 + n)$ random bits for Step 1.

Step 2

As f and g are 99%-close to linear functions we assume (between friends) that $g = WH(U)$ and $f = WH(u)$ for some U and u. We need to check that $U = u \otimes u$.

Claim 2 *If $U = u \otimes u$ then $g(r \otimes r') = f(r)f(r')$*

Proof

$$g(r \otimes r') = \sum_{ij} r_i r'_j U_{ij}$$

$$f(r)f(r') = \sum_i u_i r_i \sum_j u_j r'_j = \sum_{ij} r_i r'_j u_i u_j$$

■

Claim 3 *If $U \neq u \otimes u$ then $Pr[g(r \otimes r') = f(r)f(r')] \leq 3/4$*

Proof

$$g(r \otimes r') = \sum_{ij} r_i r'_j U_{ij} = r^t U r'$$

$$f(r)f(r') = \sum_i u_i r_i \sum_j u_j r_j = \sum_{ij} r_i r'_j u_i u_j = r^t (uu^t) r'$$

Now since $U \neq uu^t$, there exist at least one column c_1 in U, and one column c_2 in uu^t such that $c_1 \neq c_2$, and they have the same position in U and uu^t respectively. From the random subsum principle, we know that $r^t c_1 \neq r^t c_2$ for exactly half the possibility of r. Fixing an r such that $r^t U \neq r^t (uu^t)$, we get that for half the possibilities of r', we have that $r^t U r' \neq r^t (uu^t) r'$, hence we get that for at least one fourth of the possible values of (r, r'), $(r \otimes r') \neq f(r)f(r')$. ■

Motivated by the previous two claims, Step 2 of the verifier will repeat the following 3 times:

- Choose $r \in \mathbb{F}_2^n$ and $r' \in \mathbb{F}_2^n$ uniformly and independently.
- Using local decodability, decode $f(r)$, $f(r')$ and $g(r \otimes r')$.
- Check if $f(r) \cdot f(r') = g(r \otimes r')$.

We have the following:

- If $f = WH(u)$ and $g = WH(u \otimes u)$, then according to claim 2, (f, g) passes Step 2 with probability 1.
- If $f = WH(u)$ and $g = WH(U)$ respectively, such that $U \neq u \otimes u$, then:
  - The probability that (f, g) is rejected in this iteration is at least $\frac{1}{4}$ according to claim 3.
  - Thus, the probability that (f, g) is rejected in at least one iteration is at least $1 - \left(1 - \frac{1}{4}\right)^3 > \frac{1}{2}$.
- In each iteration, we need n random bits for the choice of each of r and r'. Moreover, in order to decode $f(r)$, $f(r')$ and $g(r \otimes r')$, we need n, n and n^2 random bits respectively, and we query two bits for each one of them. Therefore, Step 2 uses a total of $3(4n + n^2) = 12n + 3n^2$ random bits, and queries a total of $3(2 + 2 + 2) = 18$ bits from (f, g).

Step 3

Suppose now that $g = WH(U)$ and $f = WH(u)$, where $U = u \otimes u$.

Claim 4 *Let y be a random vector uniformly chosen in $\mathbb{F}_2^m$. If $AU = b$ then $y^t AU = y^t b$ with probability 1. On the other hand, if $AU \neq b$ then $y^t AU \neq y^t b$ with probability $\frac{1}{2}$.*

Proof The claim is trivial for $AU = b$. We simply apply the random subsum property for the case where $AU \neq b$. ■

Notice that $y^t AU = (A^t y)^t U = (A^t y) \odot U = g(A^t y)$ and so $y^t AU$ can be obtained by decoding $g(A^t y)$. Motivated by the previous claim, Step 3 repeats the following twice:

- Choose y uniformly in $\mathbb{F}_2^m$.
- Decode $g(A^t y)$.
- Check if $g(A^t y) = y^t b$.

We have the following:

- If $g = WH(U)$ and $AU = b$, then $g(A^t y) = y^t b$ for all y and (f, g) passes Step 3 with probability 1.
- If $g = WH(U)$ and $AU \neq b$, then:
  - There is a $\frac{1}{2}$ probability that (f, g) will be rejected by this iteration.
  - Therefore, (f, g) will be rejected by Step 3 with a probability of at least $1 - \left(1 - \frac{1}{2}\right)^2 > \frac{1}{2}$.
- Each iteration requires m random bits in order to choose $y \in \mathbb{F}_2^m$, and we need n^2 additional random bits in order to decode $g(A^t y)$. Moreover, two queries from g is needed in order to decode $\hat{g}(A^t y)$. Therefore, Step 3 uses a total of $2(m + n^2) = 2m + 2n^2$ random bits and queries a total of $2(2) = 4$ bits from g.

Completeness of the proposed PCP verifier Suppose that the instance of the QUADEQ problem is satisfiable, there must exist a vector $u \in \mathbb{F}_2^n$ such that $A(u \otimes u) = b$. If we let $f = WH(u)$ and $g = WH(u \otimes u)$, the proof (f, g) will pass all the three steps of the verifier with probability 1.

Soundness of the proposed PCP verifier Suppose that the instance of the QUADEQ problem is not satisfiable, then for any vector $u \in \mathbb{F}_2^n$ we must have $A \cdot (u \otimes u) \neq b$. Let (f, g) be any $(2^n + 2^{n^2})$-long string in $\mathbb{F}_2^{2^n} \times \mathbb{F}_2^{2^{n^2}}$. We have:

- If either f or g is not 99%-close to a linear function, then (f, g) will be rejected by Step 1 with a probability of at least $\frac{1}{2}$.
- If both $f = WH(u)$ and $g = WH(U)$ respectively with $U \neq u \otimes u$, then (f, g) will be rejected by Step 2 with a probability of at least $\frac{1}{2}$.
- If $f = WH(u)$ and $g = WH(U)$ respectively with $U = u \otimes u$, then we must have $AU \neq b$. Therefore, (f, g) will be rejected by Step 3 with a probability of at least $\frac{1}{2}$.
- We conclude that in all cases, there is a probability of at least $\frac{1}{2}$ that (f, g) will be rejected by at least one step of the verifier.
- The total number of random bits that are used is $poly(n, m)$ which is polynomial in the size of the input.
- The total number of queries from the proof (f, g) is $O(1)$.

This shows that QUADEQ $\in \mathbf{PCP}(poly(n), 1)$, which proves that $\mathbf{NP} \subseteq \mathbf{PCP}(poly(n), 1)$.