Lecture 18

Lecturer: Madhu Sudan *Scribe: Jinwoo Shin*

In this lecture, we will discuss about PCP (*probabilistically checkable proof*). We first overview the history followed by the formal definition of PCP and then study its connection to inapproximability results.

1 History of PCP

Interactive proofs were first independently investigated by Goldwasser-Micali-Rackoff and Babai. Goldwasser-Micali-Wigderson were focused on the cryptographic implications of this technique; they are interested in *zero-knowledge* proofs, which prove that one has a solution of a hard problem without revealing the solution itself. Their approach is based on the existence of one-way functions.

Another development on this line was made by BenOr-Goldwasser-Kilian-Wigderson who introduced several provers. $2IP$ has two provers who may have agreed on their strategies before the interactive proof starts. However, during the interactive proof, no information-exchange is allowed between two provers. The verifier can them questions in any order. The authors showed that with 2 provers, zero-knowledge proofs are possible without assuming the existence of one-way functions.

Clearly, $2IP$ is at least as powerful as IP since the verifier could just ignore the second prover. Furthermore, one can define $3IP$, $4IP$, $\ldots, MIP$ and naturally ask whether those classes get progressively more powerful. It turns out that the answer is no by Fortnow-Rompel-Sipser. They proved that $2IP = 3IP = \ldots = MIP = OIP$, where OIP is called *oracle interactive proof.* OIP is an interactive proof system with a memoryless oracle; all answers are committed to the oracle before the interactive proof starts and the oracle just answer them (without considering the previous query-history) when the verifier query them.

Now we consider the difference between OIP and $NEXPTIME$. They both have an input x of n bits and a proof(certificate) π of length 2^{n^c} for some constant c. The difference relies on the verifier V. If $L \in NEXPTIME$, there exists a deterministic-exptime V such that

- If $x \in L$, then there exists a π such that $V(x, \pi)$ accepts,
- If $x \notin L$, then $\forall\, \pi$, $V(x, \pi)$ rejects.

Similarly, if $L \in OIP$, there exists a randomized-polytime V such that

- If $x \in L$, then there exists a π such that $V(x, \pi)$ accepts with high probability,
- If $x \notin L$, then $\forall\, \pi$, $V(x, \pi)$ accepts with low probability.

Hence the relation $OIP \subseteq NEXPTIME$ is obvious since one can build the verifier for $NEXPTIME$ by simulating all random coins of V in OIP. More interestingly, Babai-Fortnow-Lund showed the other direction $OIP \supseteq NEXPTIME$. Using this relation $MIP = OIP = NEXPTIME$, Feige-Goldwasser-Lovasz-Safra-Szegedy proved that the maximum clique-size of a graph is hard to approximate and later Arora-Lund-Motwani-Sudan-Szegedy found the more general class of optimization problems, called *MAX-SNP* problems, which are hard to approximate.

Note that in the description of OIP, V checks only poly-many places of exponentially large-sized proof π. One can generalized this concept, and consider an oracle interactive proof system with $\gamma(n)$ random bits used by the verifier V and $q(n)$ bits queried from the oracle by V. Here n is the number of bits in the input x. Arora-Safra first introduce this notation as the probabilistically checkable proof class $PCP[\gamma(n), q(n)]$, which can be defined formally as follows.

Definition 1 *The language L is in $PCP_{c,s}[\gamma(n), q(n)]$ if there exists a PCP-verifier V, which uses $\gamma(n)$ random bits and queries $q(n)$ bits from the oracle, such that*

- *If $x \in L$, then there exists a π such that $V(x, \pi)$ accepts with probability $\geq c$,*

- *If $x \notin L$, then $\forall \pi$, $V(x, \pi)$ accepts with probability $\leq s$.*

Otherwise we mention c and s in the above definition, we implicitly assume $c = 1$ and $s < 1 - \varepsilon$ for $\varepsilon > 0$. Hence, in this notation, $NEXPTIME = OIP = PCP[poly(n), poly(n)]$. Now, it is natural to ask what other complexity classes we can get by varying γ and q. On the line of this question, Babai-Fortnow-Levin-Szegedy proves that $NP \subseteq PCP[poly(\log n), poly(\log n)]$. Arora-Safra improved this by showing that $NP \subseteq PCP[O(\log n), O(\sqrt{\log n})]$, and finally Arora-Lund-Motwani-Sudan-Szegedy (and later Dinur) proved that

$$NP \subseteq PCP[O(\log n), O(1)],$$

which is called PCP-Theorem. This was improved even further by Hastad and Moshkovitz-Raz who gave proofs of $NP \subseteq PCP_{c,s}[O(\log n), 3]$ and $NP \subseteq PCP_{c,s}\left[(1 + o(1)) \log n, 3\right]$ respectively, where $c = 1$ and $s = 1/2 + o(1)$. In this lecture, we will study the proof of PCP-Theorem presented by Dinur.

2 Inapproximability of MAX-3SAT

Before we look at the proof of PCP-Theorem, we present its connection to the inapproximability of MAX-3SAT problem. First we define MAX-3SAT problem. For given m clauses on n Boolean variables with each clause of length at most 3, the goal of this problem can be stated as follows depending on which types of problems we consider.

- (Search) Find an assignment which satisfies the maximum number of clauses.
- (Optimization) Find the maximum number OPT in above Search problem.
- (Decision) Decide whether $OPT \geq t$ for a given t.

Similarly, the approximation versions of these goals can be stated for a given approximation-ratio $\alpha \geq 1$.

- (Search) Find an assignment which satisfies more than $\frac{OPT}{\alpha}$ clauses.
- (Optimization) Find a number x such that $\frac{OPT}{\alpha} \leq x \leq OPT$.
- (Decision) Decide whether $OPT \geq t$ or $OPT \leq \frac{t}{\alpha}$ for a given t.

We note that the search problem is the hardest and the decision problem is the easiest. In 1976, Johnson gave an algorithm with $\alpha = 2$ for this problem. (His algorithm is somewhat trivial by choosing a random assignment and its complement, but he settles the concept of the approximation ratio.) Later in 1992, Yannakakis developed a highly nontrivial algorithm with $\alpha = \frac{4}{3}$. The current best algorithm which was given by Zwick in 1998 has $\alpha = \frac{8}{7}$. Using PCP-Theorem, Hastad in 1998 showed that this $\frac{8}{7}$ is best one can achieve under assuming $P \neq NP$. Today, we will prove a weaker version that $\alpha < \frac{16}{15}$ cannot be achievable if $P \neq NP$.

To prove this, for any NP-complete language L, it is sufficient to construct a poly-time reduction f such that $f(x) = (\phi, t)$ and it satisfies the following:

- If $x \in L$, then $OPT(\phi) \geq t$,
- If $x \notin L$, then $OPT(\phi) < \frac{15}{16}t$.

Here, ϕ is a 3SAT formula and t is a positive integer.

From PCP-Theorem, we have a verifier V which uses $O(\log n)$ random bits and query 3 bits from the oracle (or the proof π). For a fixed random bits R, let i_1 be the index of the bit of π where V queries at the first time. Depending on the value of π_{i_1}, V may query the different index of π at the second time; i_2 if $\pi_{i_1} = 0$ and i_3 if $\pi_{i_1} = 1$. Similarly, when V query at the third time, its query-index is one of i_4, i_5, i_6 and i_7. Now, there is no more query and it is decided whether V accepts or not based on these three queries. If we consider each bit of the proof π as a boolean variables π_i, the decision of V can be expressed as a 3SAT

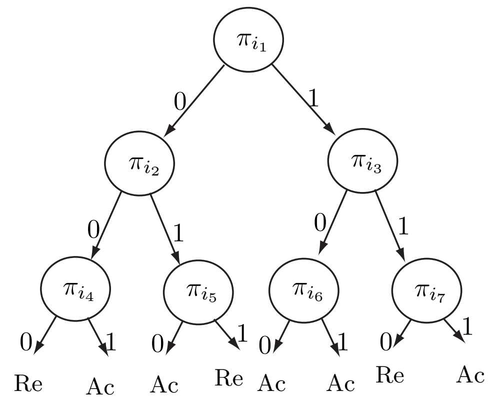

Figure 1: This figure illustrates an example of the decision-tree of V for a fixed random bits R. 'Re' and 'Ac' stand for 'Reject' and 'Accept' respectively.

formula ϕ^R. For example, if V rejects when $(\pi_{i_1}, \pi_{i_2}, \pi_{i_5}) = (0, 1, 1)$, $\pi_{i_1} \vee \bar{\pi}_{i_2} \vee \bar{\pi}_{i_5}$ should be one of clauses in ϕ^R. Figure 1 illustrates this example. In this case,

$$\phi^R = (\pi_{i_1} \vee \pi_{i_2} \vee \pi_{i_4}) \wedge (\pi_{i_1} \vee \bar{\pi}_{i_2} \vee \bar{\pi}_{i_5}) \wedge (\bar{\pi}_{i_1} \vee \bar{\pi}_{i_3} \vee \pi_{i_7}).$$

Hence, ϕ^R consists of at most 8 clauses. The reduction f chooses $\phi = \wedge_R \phi^R$ and t as the total number of clauses in ϕ. From the definition of PCP, if $x \in L$, then there exists a π such that V accepts. In other words, if $x \in L$, ϕ is satisfiable. On the other hand, if $x \notin L$, then $\forall \pi$, V accepts π with probability $\approx \frac{1}{2}$. Thus, if $x \notin L$, the number of unsatisfying ϕ^R is at least $\approx m/2$ for all π, where m is the total number of R. Since $t \leq 8m$, the desired inequality follows: $OPT(\phi) \lesssim t - m/2 \leq \frac{15}{16}t$.

3 Dinur's Proof of PCP-Theorem

Before stating the Dinur's main theorem, we first define the Generalized Graph k-coloring problem, say $GGC(k)$. For a given graph $G = (V, E)$ and a function $\pi : E \times \{1, \ldots, k\} \times \{1, \ldots, k\} \to \{0, 1\}$, the Generalized Graph k-coloring problem is finding a coloring $\chi : V \to \{1, \ldots, k\}$ such that $\pi(e, \chi(u), \chi(v)) = 1$ for all $e = (u, v) \in E$. Also, define

$$\begin{aligned} UNSAT(G, \chi) &= \frac{\#\text{ of edges } e = (u, v) \text{ s.t. } \pi(e, \chi(u), \chi(v)) = 0}{|E|}, \\ UNSAT(G) &= \min_{\chi} UNSAT(G, \chi). \end{aligned}$$

Now we state the Dinur's main theorem.

Theorem 2 *For any NP-complete language L, there exist $k, \varepsilon > 0$ and a poly-time reduction f to the instance of $GGC(k)$ such that $f(x) = (G, \pi)$ and it satisfies the following:*

- *If $x \in L$, then $UNSAT(G) = 0$.*
- *If $x \in L$, then $UNSAT(G) \geq \varepsilon$.*

It is not hard to see that Theorem 2 is equivalent to PCP-Theorem. The key lemma behind Theorem 2 is following.

Lemma 3 $\exists k$[1]$, \varepsilon > 0, f$ *such that*

[1] In fact, Lemma 3 is true for all $k \geq 3$. However, to show Theorem 2, it is enough to consider a specific k.

- *f is a linear-time reduction with $GGC(k) \xrightarrow{f} GGC(k)$ and $G \xrightarrow{f} \widehat{G}$*
- $UNSAT(G) = 0 \;\Rightarrow\; UNSAT(\widehat{G}) = 0$
- $UNSAT(\widehat{G}) \geq \min\{\varepsilon, 2 \cdot UNSAT(G)\}$.

We call Lemma 3 as *Amplifying Lemma* since the reduction f amplifies $UNSAT$ while preserving the order of the graph i.e. $|\widehat{G}| = O(|G|)$. Initially, G may have $UNSAT(G) \in \left\{0, \frac{1}{|E|}\right\}$ in the worst case. Apply f to G to get a new graph $G \leftarrow \widehat{G}$ such that $UNSAT(G) \in \left\{0, \frac{2}{|E|}\right\}$. Similarly, we iteratively apply f to G until we obtain $UNSAT(G) \in \{0\} \cup [\varepsilon, 1]$. This procedure needs at most $O(\log |E|\varepsilon) = O(\log |G|\varepsilon)$ iterations. Since the size of G blows up by a constant factor at each iteration, the size of G which we finally obtained is $O(1)^{O(\log |G|\varepsilon)} = \text{poly}(|G|)$. Therefore, using the fact that $GGC(k)$ is NP-complete, we can completes the proof of Theorem 2.

Now we sketch the proof of Lemma 3. Its proof needs the following two lemmas.

Lemma 4 $\forall k, C\ \exists K, \varepsilon_0 > 0, f$ *such that*

- *f is a linear-time reduction with $GGC(k) \xrightarrow{f} GGC(K)$ and $G \xrightarrow{f} \widehat{G}$*
- $UNSAT(G) = 0 \;\Rightarrow\; UNSAT(\widehat{G}) = 0$
- $UNSAT(\widehat{G}) \geq \min\{\varepsilon_0, C \cdot UNSAT(G)\}$.

Lemma 5 $\exists k, \delta > 0\ \forall K\ \exists f$ *such that*

- *f is a linear-time reduction with $GGC(K) \xrightarrow{f} GGC(k)$ and $G \xrightarrow{f} \widehat{G}$*
- $UNSAT(G) = 0 \;\Rightarrow\; UNSAT(\widehat{G}) = 0$
- $UNSAT(\widehat{G}) \geq \delta \cdot UNSAT(G)$.

Choose k, δ from Lemma 5 and then let $C = 2/\delta$ for Lemma 4. Hence, naturally we choose the reduction f in Lemma 3 as $f = f_2 \circ f_1$ where f_1 and f_2 are linear reductions in Lemma 4 and Lemma 5 respectively. If $G \xrightarrow{f_1} \widehat{G} \xrightarrow{f_2} \widetilde{G}$,

$$\begin{aligned}
UNSAT(G) = 0 \quad &\Rightarrow \quad UNSAT(\widehat{G}) = 0 \quad \Rightarrow \quad UNSAT(\widetilde{G}) = 0, \\
UNSAT(\widetilde{G}) \quad &\geq \quad \delta \cdot UNSAT(\widehat{G}) \\
&\geq \quad \delta \cdot \min\{\varepsilon_0, \frac{2}{\delta} \cdot UNSAT(G)\} \\
&\geq \quad \min\{\varepsilon_0 \delta, 2 \cdot UNSAT(G)\}.
\end{aligned}$$

Hence, we can choose $\varepsilon = \varepsilon_0 \delta$ for Lemma 3. In the next lecture, we will study the proof of Lemma 5.