6.841/18.405J: Advanced Complexity Theory April 10, 2002

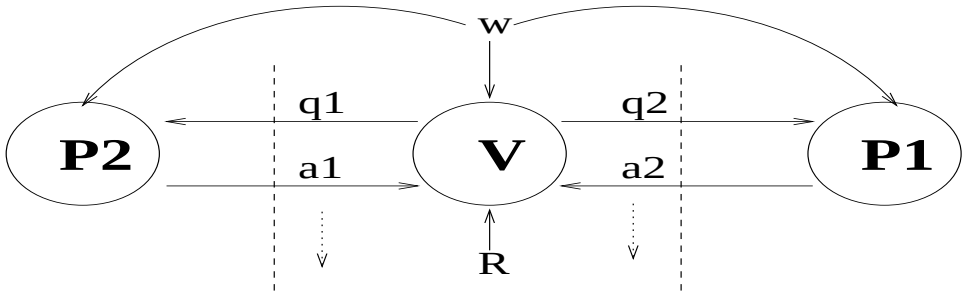
Lecture 15

Lecturer: Madhu Sudan *Scribe: David Woodruff*

In this lecture we will cover Multiprover Interactive Proofs (MIPs), Oracle Interactive Proofs (OIPs), and Probabilistically Checkable Proofs (PCPs). We will see that 2IP, the complexity class of languages with interactive proofs with two provers, is stronger than IP in a certain cryptographic sense. We then introduce OIP to show that 2IP is equivalent to MIP, the set of languages having a polynomial number of provers. Finally, we introduce the class PCP and relate it to the previous complexity classes we have studied.

Multiprover Interactive Proofs (MIP)

What happens if we allow the verifier to interact with more than one prover in an interactive proof? The provers have unbounded computational resources, but cannot interact with each other. If the prover wants to cheat the verifier, he has to make sure he will not be detected when the verifier interacts with the other provers. The complexity class of multiprover interactive proofs (MIP) was defined by Ben-Or, Goldwasser, Kilian and Wigderson. In the case of 2 provers we have the complexity class 2IP, shown in the following diagram:



where $\{q_i\}$ is the set of questions asked by the verifier V, $\{a_i\}$ is the set of answers given by provers P_1 and P_2, w is the common input string, and R is the string of V's random coin tosses. After interacting with the provers, V is required to produce a boolean verdict, $\text{Verdict}(w, R, a_1, \ldots, a_k)$. As in the definition of IP, V is restricted to probabilistic polynomial time in the length of w.

Formally, a language $L \in$ 2IP if

- (completeness) $w \in L$ implies $\exists P_1, P_2$ such that $\Pr[P_1 \leftrightarrow V \leftrightarrow P_2 \text{ accepts }] = 1$.
- (soundness) $w \notin L$ implies $\forall P_1, P_2$, we have $\Pr[P_1 \leftrightarrow V \leftrightarrow P_2 \text{ accepts }] \leq 1/2$.

Clearly IP $\subseteq$ MIP because the verifier can choose to interact with just one prover. Moreover, it seems that by limiting the cheating possibilities of the prover in MIP, we should be able to construct more verifiers for more statements and hence MIP should be a larger class of languages than IP. In fact Babai, Fortnow, and Lund showed that MIP = NEXPTIME.

We note that 2IP is robust with respect to error in the sense that one can convert any two-sided error verifier V into a one-sided error verifier V'. Also, one can amplify the error by repeating the above proof sequentially. In general, however, one cannot repeat the above proof in parallel without compromising soundness.

An Application of 2IP: Zero-knowledge Proof

We give a cryptographic application of a two-prover interactive proof. Informally a *Zero Knowledge Proof* is a proof by which the prover can convince the verifier whether or not a string x is in a language L without

giving the verifier any other information whatsoever, such as certain properties of x which the verifier could not compute otherwise. Here is a two-prover protocol for graph 3-colorability:

1. V picks a random edge $e = (u, v)$ in the input graph G. V then picks a random endpoint w of e, i.e., $w \in \{u, v\}$.

2. Next V sends e to P_1 who responds with the pair of colors (c_1, c_2), where $c_1 = color(u)$ and $c_2 = color(v)$. If $c_1 = c_2$, then V immediately rejects G.

3. V then sends w to P_2 who responds with $c_3 = color(w)$.

4. If $c_1 \neq c_3$ and $w = c_1$, then V rejects. Similarly, if $c_2 \neq c_3$ and $w = c_2$, then V rejects. Otherwise, V accepts G.

We argue completeness, soundness, and zero-knowledge. Completeness is clear - if a graph G is 3-colorable, the honest prover convinces the verifier of this with probability 1.

As for soundness, consider a graph $G = (V, E)$ which is not 3-colorable. Then there exists an edge e of G whose endpoints have the same color for any coloring. Suppose P_1 and P_2 have agreed upon a coloring of the vertices of G before the protocol begins. Fix P_2. This effectively fixes a coloring of G. Then V will choose an edge e' to send to P_1 which will equal e with probability $1/|E|$. In order for V not to reject G immediately, P_1 will have to give different colorings for the two endpoints of e, despite the coloring scheme P_1 and P_2 have agreed upon that assigns the same color to both endpoints. Then, when V queries P_2 for the color of one of the randomly chosen endpoints of e, the color P_2 returns for at least one of the endpoints of e will differ from the assignment given by P_1. Hence, the probability that V will accept G is at most $1 - 1/(2|E|)$. Repeating the above protocol sequentially a polynomial number of times, one can achieve a soundness of $1/2$. Note that if $|E|^2$ rounds will be run sequentially, the provers need to agree upon a sequence of $|E|^2$ color relabellings beforehand. The provers can collude with the knowledge of G, but must collude before V's random coins are tossed.

Finally, we informally argue that the protocol is zero-knowledge. Before the beginning of the protocol the provers agree upon a random coloring of the vertices. That is to say, if $G = (V, E)$ is 3-colorable, then there exists a coloring assignment $f : V \to \{1, 2, 3\}$ such that if $(u, v) \in E, f(u) \neq f(v)$. Then if $\pi : \{1, 2, 3\} \to \{1, 2, 3\}$ is a permutation on three letters, we see that π composed with f will give another 3-coloring. Hence there are 6 random colorings of G corresponding to f. The provers agree upon f and the permutation π before the beginning of the protocol. When V learns a coloring $color(u)$ and $color(v)$ for an edge (u, v), he doesn't learn anything other than the fact that $color(u) \neq color(v)$ since any combination of two colors for u and v is equally likely. Hence, the protocol is zero-knowledge.

Oracle Interactive Proofs (OIP)

Clearly we have the inclusions IP $\subseteq$ 2IP $\subseteq$ 3IP $\subseteq \cdots \subseteq$ MIP, since the verifier can simply choose not to interact with some of the provers. But are polynomially many provers more powerful than 2 provers? To answer this question, Fortnow, Rompel and Sipser introduced the complexity class of Oracle Interactive Proofs (OIP).

The key difference between MIP and OIP is that the provers in Oracle Interactive Proofs are oracles, i.e., they are memoryless provers. Without loss of generality we can think of an oracle O as a function from $\{0, 1\}^*$ to $\{0, 1\}$. Hence, the obvious definition:
$L \in$ OIP iff

1. (completeness) $w \in L$ implies $\exists O$ s.t. $\Pr[V \leftrightarrow O \text{ accepts }] = 1$

2. (soundness) $w \notin L$ implies $\forall O$ $\Pr[V \leftrightarrow O \text{ accepts }] \leq 1/2$

We first show MIP $\subseteq$ OIP. Intuitively, in OIP we are restricting to a smaller class of provers so the verifier is more likeley to accept more statements, so a language is more likely to meet the soundness criterion, so OIP is likely to be a larger class of languages than MIP.

Formally, we can simulate any Multiprover Interactive Proof with an Oracle Interactive Proof. Suppose there are p provers in the Multiprover Interactive Proof. We simply convert each prover into a lookup table. If V asks prover i question q_k with history $q_1, q_2, \ldots, q_{k-1}$, then there is an entry in the table that maps $(i, q_1, \ldots, q_k)$ to the answer a_k that prover i would respond with. We can create one table for all provers mapping any possible set of questions the verifier could ask to the answer given by each prover. But this is just an oracle O and hence MIP $\subseteq$ OIP.

We will prove the reverse inclusion for the case when V is a non-adaptive verifier, even though OIP $\subseteq$ MIP for adaptive verifiers as well. We will give a reduction but we will not formally argue completeness and soundness. Specifically, we shall argue that OIP $\subseteq$ 2IP, and together with MIP $\subseteq$ OIP, it will follow that 2IP = 3IP = $\cdots$ = MIP.

Suppose we have an Oracle Interactive Proof with verifier V asking questions $q_1, \ldots, q_m$ to oracle O. Then we construct a verifier V' in the two-prover setting which behaves as follows. It first asks the same questions $q_1, \ldots, q_m$ to prover P_1 and receives a sequence of answers $a_1, \ldots, a_m$. It then randomly chooses an index j and sends q_j to P_2. P_2 then responds with an answer b. Finally, V' accepts if and only if V would accept given answers $a_1, \ldots, a_m$ from O and if $a_j = b$. Intuitively, although P_1 has more room to cheat than O since he is not memoryless, P_2's answer is used to ensure P_1 is not using the history of questions to base his answers on. Completeness of this protocol is clear. To see that it is sound, note both that the original oracle O can cheat only with low probability and that if P_1 tries to cheat, he will be detected by V's interaction with P_2 with nonnegligible probability.

Now let's compare IP and OIP = MIP. We saw in the last lecture that IP = PSPACE, developed by Lund, Fortnow, Karloff, and Nisan, and later proven by Shamir. On the other hand, Babai, Fortnow, and Lund showed that OIP = NEXPTIME. NEXPTIME can be thought of as the complexity class of languages of short theorems with long proofs and polynomial-time verification in the length of the proof. Note that the inclusion OIP $\subseteq$ NEXPTIME is pretty clear. For languages in OIP we have a probabilistic polynomial-time verifier whereas for languages in NEXPTIME we are allowed a deterministic exponential-time verifier. Moreover, we can write down the oracle as a table of (question, answer) pairs in NEXPTIME. We would like to "scale down" the equality MIP = NEXPTIME to obtain an equality of the form MIP' = NP, where verifiers in MIP' run in logarithmic time (in the length of w), but this is not possible since the verifier would not even be able to read the entire input.

Hence, we see that IP $\neq$ OIP unless PSPACE = NEXPTIME.

Probabilistically Checkable Proofs (PCP)

We now parameterize Oracle Interactive Proofs more precisely and give them a new name, Probabilistically Checkable Proofs (PCP). In particular, we keep track of the number of coins V tosses, $r(|w|)$, and the number of queries V makes to O, $q(|w|)$. As before, we restrict V to run in polynomial time. Formally,

$L \in \text{PCP}_{c,s}[r, q]$ iff $\exists$ an OIP for L with an (r, q)-restricted verifier V with completeness c and soundness s.

The following identities are immediate:

- $\text{PCP}_{2/3,1/3}[\text{poly}(n), 0] = \text{BPP}$.
- GNI $\in$ $\text{PCP}_{1,1/2}[\text{poly}(n), 1]$ as shown in class.
- $\text{PCP}_{1,1/2}[\text{poly}(n), \text{poly}(n)] = \text{OIP}$.
- NP = $\text{PCP}_{1,0}[0, \text{poly}(n)]$ = $\text{PCP}_{1,0}[O(\log(n)), \text{poly}(n)]$
- $\text{PCP}_{1,1/2}[O(\log(n)), O(\log(n))] \subseteq \text{NP}$.

What if we try to "scale down" from NEXPTIME to NP to get proofs for NP languages? Work by Babai, Fortnow, Levin, and Szegedy, and by Feige, Goldwasser, Lovász, Safra, and Szegedy showed that NP $\subseteq$ $\mathrm{PCP}_{1,1/2}[\mathrm{poly}(\log(n)), \mathrm{poly}(\log(n))]$. Later work by Arora and Safra showed that NP = $\mathrm{PCP}_{1,1/2}[\log(n), \sqrt{(\log(n))}]$, and work by Arora, Lund, Motwain, Sudan, and Szegedy showed that NP = $\mathrm{PCP}_{1,1/2}[\log(n), k]$ for a constant k. Work by Håsted reduced the number of query bits to 3, showing that $\forall \epsilon > 0$ NP = $\mathrm{PCP}_{1-\epsilon,1/2}[\log(n), 3]$, and finally Guruswami, Lewin, Sudan, and Trevisin got perfect completeness by showing $\forall \epsilon > 0$ NP = $\mathrm{PCP}_{1,1/2}[\log(n), 3]$. In other words, a proof of any statement in NP can be written in such a way that it can be verified by looking at only 3 bits of the proof.

We will now use these results to show that if NP $\neq$ P, then even approximating an NP-hard problem is very hard. Let V be a verifier for a $\mathrm{PCP}_{1,1/2}[\log(n), 3]$ language L. For each possible sequence of the verifier's random coins R_i, we shall construct a decision tree corresponding to oracle f's responses to V's 3 queries. The decision tree is a balanced binary tree of depth 3. We start at the root node q_1. If $f(q_1) = 0$, we examine q_1's left child, otherwise we examing q_1's right child. Based on our response $f(q_1)$ and R_i, we can compute the next question q_2 asked by V. We then branch according to whether $f(q_2) = 0$ or $f(q_2) = 1$. We continue branching in this way until we reach a leaf node of the tree, which is either an Accept node or a Reject node, depending on whether V, given toin cosses R_i, accepts or rejects based on f's responses to his questions. Each path in this decision tree can be written as a 3-CNF ϕ_i formula of at most 8 clauses. We do this for all polynomially many random strings R_i. We then define $\Phi = \bigwedge_{i=1}^{poly} \phi_i$. Completeness of L implies Φ is satisfiable. Soundness implies that if input x is not in L, then at least $1/2$ of all formulae ϕ_i are not satisfied for any assignment of oracle answers. For a formula to not be satisfied, at least 1 of 8 clauses must not be satisfied. Hence, $1/16$ of all clauses of Φ are not satisfied. This says that unless P = NP, even approximating hard problems is very hard.