CMPT 710 - Complexity Theory: Lecture 19

Valentine Kabanets

November 6, 2003

1 Interactive Protocols

The *interactive protocols (IP)* are also protocols between a probabilistic polytime Verifier and an all-powerful Prover, where after a certain number of rounds of communication, the Verifier accepts or rejects. The difference from Arthur-Merlin protocols is that the Verifier in an Interactive Protocol *does not* reveal its randomness to the Prover. Sometimes, these protocols are called *private-coin* protocols, as opposed to public-coin Arthur-Merlin protocols.

In interactive protocols, the Verifier moves first. An IP protocol with k rounds of communication is the protocol where at most k messages are exchanged in a conversation between Verifier and Prover.

We say that a language $L \in \mathsf{IP}[k]$ if there is a probabilistic polytime verifier such that, for every $x \in L$, there is a Prover that convinces the Verifier to accept with probability $\geq 3/4$ after at most k-rounds; and for every $x \notin L$, each Prover can convince Arthur to accept with probability at most $1/4$ after k rounds.

Here's an example of an IP protocol for the Graph Nonisomorphism Problem (NISO). Define $NISO = \{(G_1, G_2) \mid \ G_1 \text{ and } G_2 \text{ are not isomorphic}\}$.

Theorem 1. $NISO \in \mathsf{IP}[2]$

Proof. Here's a protocol. Given an input (G_1, G_2) of two graphs on n vertices, the Verifier will randomly pick $i \in \{1, 2\}$, and a random permutation π of the set $\{1, 2, \ldots, n\}$. The verifier will send $\pi(G_i)$ to the Prover (i.e., the Verifier sends a randomly permuted copy of a randomly chosen graph in (G_1, G_2)). The Prover sends back $j \in \{1, 2\}$. The Verifier accepts iff $j = i$.

Analysis (1) If G_1 and G_2 are non-isomorphic, the computationally unbounded Prover can always find a correct $j = i$ by checking which of G_1 and G_2 is isomorphic to the graph received from the Verifier. So, in this case, the Verifier can be made to accept with probability 1.

(2) If G_1 and G_2 are isomorphic, then a graph sent to the Prover by the Verifier in case $i = 1$ is from the same distribution as the graph sent in the case $i = 2$. Hence, the Prover has no way of determining i, and his j will be equal to i with probability $1/2$. (This can be shown formally after some simple probability calculations; it's left as an exercise.)

So, if the graphs are non-isomorphic, the Verier accepts with probability 1. If the graph are isomorphic, the Verifier accepts with probability at most $1/2$. (It is possible to reduce the error probability to $1/4$.) □

2 AM and IP

Let's define $\mathsf{AM}[k]$ to be the class of languages decided by an Arthur-Merlin protocol with at most k rounds of communication. Note that $\mathsf{AM} = \mathsf{AM}[2]$.

From the definition of $\mathsf{IP}[2]$, it seems that $\mathsf{IP}[2]$ is more powerful than AM. It is easy to simulate Arthur-Merlin protocol in $\mathsf{IP}[2]$, but it's not at all clear how to simulate $\mathsf{IP}[2]$ with AM. The surprising result of Goldwasser and Sipser shows that in fact the two classes are the same!

Theorem 2 (Goldwasser-Sipser). *For any k, $\mathsf{IP}[k] = \mathsf{AM}[k]$.*

The proof of this theorem is too involved to be presented here. To get a glimpse of the proof technique used in the proof, we'll construct an AM protocol for NISO.

Theorem 3 (Goldwasser-Sipser). $NISO \in \mathsf{AM}$

For the proof, we'll need some terminology. An *automorphism* of a graph G is an isomorphism between G and G. A *trivial automorphism* of G is the identity function. Note that a graph G on n vertices without any non-trivial automorphism has exactly $n!$ distinct isomorphic graphs. (In general, if G has k non-trivial automorphisms, then G has exactly $n!/k$ distinct isomorphic graphs.)

We'll use universal hash function families. A family $H = H_n$ of hash functions $h : U \to M$ is called universal if it has two properties:

1. (uniformity) for any $u \in U$ and any $a \in M$, $\mathbf{Pr}_h[h(u) = a] = 1/|M|$,
2. (pairwise independence) for any u, u', with $u \neq u'$, and for any $a \in M$, $\mathbf{Pr}_h[h(u) = a \wedge h(u') = a] = 1/|M|^2$.

   We assume that our family of hash functions is efficient in the sense that each h can be evaluated efficiently, and we can efficiently sample a random h from the family H.

   Now we are ready for the proof.

Proof of Theorem 3. With loss of generality, assume that given input graphs (G_1, G_2) have non-trivial automorphisms.

Define the set $W = \{G' \mid G' \text{ is isomorphic to } G_1 \text{ or to } G_2\}$. Observe that, if G_1 and G_2 are isomorphic, then $|W| = n!$. If G_1 and G_2 are not isomorphic, then $|W| = 2(n!)$. Define $Y = W \times W$ be the cross-product of W with itself. Then, for isomorphic G_1 and G_2, $|Y| = (n!)^2$, whereas for non-isomorphic G_1 and G_2, $|Y| = 4(n!)^2$.

Let $M = \{0, 1, 2, \ldots, 4(n!)^2 - 1\}$ be a set of size $4(n!)^2$. Let H be a family of universal hash functions $h : U \to M$, where $Y \subset U$ (i.e., U is a set of binary strings that contains binary encodings of all elements in Y.)

Arthur will use a randomly chosen hash function to test if Y is large or small. More formally, Arthur picks a random hash function h and sends it to Merlin. Merlin sends back a string $y \in Y$ with a proof that $y \in Y$ (note that such a proof is short: it's just a pair of isomorphisms). Arthur checks that Merlin's proof of $y \in Y$ is correct, and he checks that $h(y) = 0$. If the checks pass, then Arthur accepts; otherwise, he rejects.

Analysis In case G_1 and G_2 are isomorphic, we have $|Y| = (n!)^2$, and so

$$\begin{aligned}\mathbf{Pr}_h[\exists y \in Y :\ h(y) = 0] &\leq \sum_{y \in Y} \mathbf{Pr}_h[h(y) = 0] \\ &= |Y|/|M| = 1/4,\end{aligned}$$

where the first inequality is by the "union bound", and the second equality by uniformity property of our hash function family.

In case G_1 and G_2 are non-isomorphic, we have $|Y| = 4(n!)^2 = |M|$. So,

$$\begin{aligned}\mathbf{Pr}_h[0 \in h(Y)] &= \mathbf{Pr}_h[h(y_1) = 0 \vee h(y_2) = 0 \vee \cdots \vee h(y_{|Y|}) = 0] \\ &\geq \sum_{y \in Y} \mathbf{Pr}_h[h(y) = 0] - \sum_{\{y,z\} \in Y} \mathbf{Pr}_h[h(y) = 0 \wedge h(z) = 0] \\ &= |Y|/|M| - |Y| * (|Y| - 1)/(2|M|^2) \\ &\geq 1 - 1/2 = 1/2,\end{aligned}$$

where the first inequality is by the Inclusion-Exclusion Principle, and the second equality uses uniformity and pairwise independence of our family of hash functions.

So, if G_1 and G_2 are not isomorphic, Arthur can be made to accept with probability at least $1/2$. If, on the other hand, G_1 and G_2 are isomorphic, Arthur accepts with probability at most $1/4$. (Correctness probability can be amplified by repeating the protocol several times in parallel.) So, we have an AM protocol for NISO.

It remains to show how to deal with our assumption that the input graphs have no non-trivial automorphisms. Let us define
$W = \{(G', \pi) \mid\ G'$ is isomorphic to G_1 or G_2, and π is an automorphism of $G'\}$. It is left as exercise to check that, for isomorphic G_1 and G_2, $|W| = n!$; and for non-isomorphic G_1 and G_2, $|W| = 2(n!)$. The rest of the proof is the same as before. □

3 More about AM and IP

If we allow polynomial number of rounds, then we get the entire class PSPACE.

Theorem 4 (Shamir). $\mathsf{IP}[\mathsf{poly}] = \mathsf{AM}[\mathsf{poly}] = \mathsf{PSPACE}$.

For any constant k, $\mathsf{AM}[k] = \mathsf{AM}[2] = \mathsf{AM}$.

Theorem 5 (Babai). *For any constant k,* $\mathsf{AM}[k] = \mathsf{IP}[k] = \mathsf{IP}[2] = \mathsf{AM}$.

Finally, we have $\mathsf{AM} \subseteq \Pi_2^p$. So AM is in the second level of the polytime hierarchy. If we allow a polynomial number of rounds, we get the entire PSPACE.

It is believed that $\mathsf{AM} = \mathsf{MA} = \mathsf{NP}$. The reason is similar to the case of "BPP vs P": if EXP contains a language of exponential SAT-oracle circuit complexity, then $\mathsf{AM} = \mathsf{NP}$. Here, our circuits have additional gates, SAT-oracle gates, that compute the SAT function. A simple counting argument shows that most Boolean functions require SAT-circuits of exponential size. If we could construct a particular truth table of such hard function in time polynomial it its size, then we could derandomize AM. However, proving SAT-oracle circuit lower bounds is even harder than proving regular circuit lower bounds.