CS6840: Advanced Complexity Theory Feb 13, 2012

Lecture 20 : Interactive protocol for Permanent

Lecturer: Jayalal Sarma M.N. *Scribe: Anup Joshi*

THEME: Between P and PSPACE.
LECTURE PLAN: Proof of correctness of GNI protocol, Historical Aspects of IP, $P^{\#P}$, and final proof. The interactive protocol for the permanent (outline).

1 Proof of correctness of GNI protocol

In the last lecture we have seen an interactive protocol for GNI, here we argue about the correctness of the interactive protocol. If G_1 and G_2 are the two graphs for which we want to check if they are isomorphic or not, lets assume that both of them are not isomorphic, i.e.

$$\begin{aligned} G_1 \ncong G_2, &\Rightarrow \nexists \tau : G_1 \to G_2 \\ &\Rightarrow \exists ! b' \text{ such that } \sigma(G_{b'}) \cong G' \\ &\Rightarrow \text{Return } b' \end{aligned}$$

Now, lets consider the case when G_1 and G_2 are isomorphic, i.e.

$$\begin{aligned} G_1 \cong G_2, &\Rightarrow \exists \tau : G_1 \to G_2 \\ &\Rightarrow G' \cong \tau(G_1) \\ &\Rightarrow G' \cong \tau(G_2) \end{aligned}$$

Since it is isomorphic, b' can be any of 1 or 2. Hence, with probability 1/2, prover can convince the verifier by replying randomly. Here, in order to get a very small error probability say ϵ, we can repeat the process on the same input, and take the majority vote. Hence, in summary we have the following situation:

$$\begin{aligned} \text{If } (G_1, G_2) \in GNI &\Rightarrow \Pr[\text{Verifier accepts}] = 1 \\ (G_1, G_2) \notin GNI &\Rightarrow \Pr[\text{Verifier accepts}] \leq 1/2 \end{aligned}$$

2 NP versus IP

In NP, the verifier deterministically verifies the guessed path. We also call this guessed path or string as the *certificate* if the computation is accepted on that path.

In IP, however, the verifier is non-deterministic. The proofs are big, and the only way to solve it is to pick random places in the proof, and run multiple protocols on them. We can say that IP is the randomized relaxation of the class NP.

3 Historical aspects of IP

The class IP has an interesting history behind it. On November 27, 1989, Noam Nisan proved a weaker result showing that $Perm \in \mathsf{IP}$. He sent this result written on an e-mail to several of his colleagues. From then on all the people in the mailing-list started their own research on the topic, and in just 2 weeks, i.e. on December 13, 1989, Lance Fortnow sent an e-mail to all the mailing-list members that $P^{\#\mathsf{P}}$ was in IP. And again in just 2 weeks time, i.e. on December 26, 1989, Adi Shamir announced his findings. He showed that $\mathsf{PSPACE} \subseteq \mathsf{IP}$, and thus $\mathsf{PSPACE} = \mathsf{IP}$. The reverse containment, i.e., $\mathsf{IP} \subseteq \mathsf{PSPACE}$ was showed earlier by Papadimitriou in 1987.

Theorem 1. $\mathsf{P}^{\#\mathsf{P}} \subseteq \mathsf{IP}$

Proof. It is enough if we show that the permanent computation can be done by a prover-verifier pair. This proof was left as an exercise. □

4 Interactive protocol for Permanent

Lets recall that the permanent $\mathsf{perm}(M)$ of an $n \times n$ matrix $M = (m_{ij})$ is defined as

$$\mathsf{perm}(M) = \Sigma_\sigma \Pi_{i=1}^n m_{i,\sigma(i)}.$$

Obtaining the permanent of a matrix is not easy to compute, but we have an interactive protocol for Permanent. Let $\tilde{\mathrm{M}}_{ij}$ be the minor of M obtained by deleting i^{th} row and j^{th} column. The protocol is as given below:

Protocol:

1. Prover commit on a value of $\mathsf{perm}(M)$.
2. Verifier asks for permanent of $q_j = \tilde{\mathrm{M}}_{ij}$, for $1 \leq i \leq n$.

3. Prover provides permanent
4. Verifier checks if $\Sigma_{j=1}^{n} q_j M[i,j] = q$. If yes, repeat step 2, else *reject*

According to the protocol, the prover initiates the task by first committing on a permanent of a $n \times n$ matrix. After that a sequence of interaction between the prover and verifier follows in which the prover provides the permanent of smaller matrices. The verifier verifies that the permanent of the smaller matrices are indeed correct by going through some consistency checks on the received values. If the prover has not cheated even once, then the consistency checks are always satisfied, and eventually it is accepted. If the prover cheats then it has to cheat only once to fool the verifier. If the verifier catches the cheating prover it rejects, and if the prover was honest then the verifier accepts. As we see, this protocol requires exponential time to check the values received from the prover. There will be $n(n-1)...(n-n+1) = n!$ checks needed, which the verifier cannot do.

Since the verifier cannot check all q_j, instead of this we randomly pick a q_j and verify it recursively. But even in this method the probability of catching a cheating prover is very low, since the prover has to cheat only once. A solution is to consider the matrix $D(x) = xA_1 + (1-x)A_2$, where A_1 and A_2 are two submatrices, and there permanent is p_1 and p_2 respectively. Here we observe that $D(1) = A_1$ and $D(0) = A_2$. Let $f(x) = Perm(D(x))$, then $f(0) = Perm(D(0)) = Perm(A_2) = p_2$, and $f(1) = Perm(D(1)) = Perm(A_1) = p_1$. We note that the $Perm(D(x))$ is a polynomial of degree $\leq n$, since the entries of $D(x)$ are linear functions of x. The prover has to cheat on one of A_1 or A_2, in order to meet the consistency check of $D(0) = A_2$ and $D(1) = A_1$, hence once the prover has cheated it has to repeatedly cheat on the subsequent interactions in order to fool the verifier. This leaves a very low margin of error for the prover. This is the LFKN protocol, which we will see in detail in the next lecture.