CS221: Computational Complexity Prof. Salil Vadhan

Lecture 28: Interactive Proof Systems, Graph Nonisomorphism

12/02 *Scribe: Ruggiero Cavallo*

Contents

1 Announcements

- PS5 - $\textsc{Majority} = \begin{cases} 1 & \text{if number of } n \text{ inputs equal to } 1 \geq n/2 \\ 0 & \text{otherwise} \end{cases}$
- section today at 6pm in MD-323
- this Friday's class will be in MD-G125

2 Agenda

- Interactive Proofs

3 Classical Proof Systems

In classical proof systems, we write down mathematical assertions as strings. Then, we have some language L that equals the set of "true" assertions. For example,

$$\textsc{Sat} \leftrightarrow \text{ true statements of form "}\phi \text{ is satisfiable."}$$

A proof system for a language L is a verification algorithm V that has the following properties:

1. Completeness: $x \in L \Rightarrow \exists \text{ proof } \; V(x, \text{proof}) = \textit{accept}$
   That is, "true assertions have proofs."

2. Soundness: $x \in L \Rightarrow \forall$ proof* $V(x, \text{proof}^\star) = \mathit{reject}$

   That is, "false assertions have no proofs."

3. Efficiency: $\forall x,$ proof $V(x, \text{proof})$ runs in time $poly(|x|)$

   Efficiency is natural - the notion of a proof is meaningless if it can't help you verify the claim.

Note that **NP** = languages with classical proof systems (think of V as checking the relation of the **NP** language).

4 Interactive Proof Systems

For interactive proof systems, we add two new ingredients:

1. Randomization - V can toss coins, err with some small probability.
2. Interaction - we replace static written proofs with a dynamic, computationally unbounded "prover."

Why introduce these two ingredients?

- it's philosophically interesting. Outside of math, in most areas of human thought this is how people think of proof- conversational, interactive, randomized(?).
- perhaps we can prove a larger class of assertions (i.e., maybe we can capture more languages). This is interesting from a complexity point of view.
- cryptography (this was the original motivation for interactive proofs).
- efficiency in verification. Any problem in NP has a probabilistic proof system where the verification algorithm only reads a constant number of bits of the proof.
- novel properties: "zero knowledge" (proof reveals nothing other than trueness of assertion; this is very useful in cryptography).
- unexpected applications: inapproximability.

Example: interactive proof system for Graph Nonisomorphism.

$G = ([n], E)$, permutation $\pi : [n] \to [n]$ (where $[n] = \{1, \ldots, \text{n}\}$)

$\pi(G) \triangleq ([n], \{(\pi(u), \pi(v)) \; (u, v) \in E\})$

G and H are isomorphic (written $G \cong H$) if $\exists \pi$ such that $\pi(G) = H$

Graph Isomorphism is the language GI $= \{(G_0, G_1) : G_0 \cong G_1\}$.

- $\in$ **NP**
- not known to be in **P**

- not known to be **NP**-complete

GRAPH NONISOMORPHISM is the language GNI $= \{(G_0, G_1) : G_0 \not\cong G_1\}$.

- not known to be in **NP** (we don't know a classical proof system for it, but we'll see a nice interactive one).

Interactive Proof for GNI

G_0 and G_1 are given to both the verifier V and the prover P.

V:

- choose $b \xleftarrow{\text{R}} \{0,1\}$ at random
- Choose $\pi \xleftarrow{\text{R}} S_n$ (permutations on $[n]$) at random
- Let $H = \pi(G_b)$ (H is a graph)
- Send H to all-powerful "prover" P

P:

- Let c equal 0 if $H \cong G$, 1 otherwise
- Send c to V

V:

- Accept if $c = b$, reject otherwise

Proposition 1 *If $G_0 \not\cong G_1$, then V accepts in $(P, V)(G_0, G_1)$ [the interaction] with probability 1. [I.e., if the statement is true, then there is a strategy for prover which makes verifier always accept.]*

Proof: If $G_0 \not\cong G_1$, then H will be isomorphic to exactly one of the input graphs (namely G_b), so P will always guess correctly. ■

Proposition 2 *If $G_0 \cong G_1$, then V accepts with probability $\leq 1/2$ in $(P^*, V)(G_0, G_1)$. This is true no matter what strategy P^* the prover follows. [So it holds even if P is trying to trick you.]*

Proof: If $G_0 \cong G_1$, then $\pi(G_0)$ and $\pi(G_1)$ have the same distribution (for random permutation π), so P^* has no information about b. [This is like the "Coke/Pepsi challenge".] ■

Now we're ready for a formal definition of interactive proofs.

Definition 3 (Goldwasser, Micali, Rackoff '85) *An interactive proof system for a language L is an interactive protocol (P, V) with the following properties:*

1. *Completeness - If $x \in L \Rightarrow Pr[V$ accepts in $(P, V)(x)] \geq 2/3$.*

2. *Soundness* - *If* $x \notin L \Rightarrow \forall P^*$ $Pr[V$ *accepts in* $(P^*, V)(x)] \leq 1/3$.

3. *Efficiency* - $\forall x$ *the total computation time of* V *in* $(P, V)(x)$ *is at most* $poly(|x|)$.

Note that you can make 2/3 in (**1**) and 1/3 in (**2**) arbitrarily close to 1, 0, respectively, by the usual method of repeating many times and having the verifier rule by majority. Also note that there is no constraint on P's efficiency.

We define **IP** to be the class of languages with interactive proof systems.

Theorem 4 GNI $\in$ **IP**

• *proved above.*

How big is **IP**?

Theorem 5 (Lund, Fortnow, Karloff, Nisan '92) co-NP $\subseteq$ **IP**, *in fact* $\mathbf{P}^{\#\mathbf{P}} \subseteq$ **IP**.

Proof Sketch: Convince poly-time verifier that a formula is UNsatisfiable.

$$\text{E\#SAT} = \{(\phi, k) : \phi \text{ has exactly } k \text{ satisfying assignments}\}$$

Observation: ϕ has exactly k satisfying assignments iff $\exists k_0, k_1$ such that:

1. $k_0 + k_1 = k$
2. $\phi_0(x_2, \ldots, x_n) = \phi(0, x_2, \ldots, x_n)$ has exactly k_0 satisfying assignments.
3. $\phi_1(x_2, \ldots, x_n) = \phi(1, x_2, \ldots, x_n)$ has exactly k_1 satisfying assignments.

ϕ and k are given to both the verifier V and the prover P.

P:

- Send k_0 and k_1 to V.

V:

- choose $b \in \{0, 1\}$ at random
- Send b to P.

The prover will recursively prove that $(\phi_b, k_b) \in$ E#SAT. Each time we're eliminating one variable; at the end the verifier can just check for itself.

1. If $(\phi, k) \in$ E#SAT $\Rightarrow V$ accepts with probability 1.

2. If $(\phi, k) \notin$ E#SAT $\Rightarrow V$ accepts with probability < 1. Either 1, 2, or 3 of above will be violated.

But actually, if $(\phi, k) \notin$ E#SAT, V might accept with probability exponentially close to 1 (i.e., $1 - 2^{-n}$). We'll fix this next time.

$\square$