DD2446 COMPLEXITY THEORY

LECTURE I

LAST TIME

Randomized computation

PTM: TM that flips fair coin at each step

BPP Poly-time, gets answer right with prob $\geq 2/3$

(Can be boosted to exponentially close to 1)

One-sided error

RP: No false positives – answer $x \in L$ always correct

coRP: No false negatives

ZPP: Expected poly-time

if terminates with answer, then always correct

$P \subseteq BPP \subseteq EXP$ [even $\subseteq PSPACE$ as noted in class]

$BPP \subseteq P/poly$

boosting success prob $\Rightarrow$ "golden" random string $\Rightarrow$ advice

$BPP \subseteq \Sigma_2^P \cap \Pi_2^P$ $(\subseteq PH)$ – didn't show

Complete problems? } seems hard; BPP
Hierarchy theorems? } semantic class

Randomized reductions

$L \leq L'$ if $\exists$ poly-time PTM M s.t.

$\forall x \in \{0,1\}^*$ $\Pr[L'(M(x)) = L(x)] \geq 2/3$

If $L' \in BPP$ and $L \leq_r L'$, then $L \in BPP$

But these reductions not transitive

$BP \cdot NP = \{L \mid L \leq_r 3\text{-SAT}\}$

What is a proof?

Key insight: something that is (computationally) easy to check.

Other aspects:

- Interactivity — Proof static, written object? Or dialogue between prover & verifier?
- Correctness — Always 100% correct? Or OK with high probability?

Different combinations possible — give different results

TODAY: INTERACTIVE PROOFS

Example: F unsatisfiable CNF formula

Static (NP-style) proof? Seems hard

Efficient interactive proof? YES!

Interactive proofs important for

- cryptography (next lecture)
- hardness of approximation (hopefully later during course)
- indications of non-NP-completeness (will mention, but probably only briefly)

Set-up

Prover — all-powerful, but not necessarily trustworthy

Verifier — poly-time bounded; can flip coins (privately and hide results from prover)

Verifier computes function f of

- input x
- randomness r
- previous messages

Prover computes function g of

- input
- previous messages

<u>k-round interaction</u> sequence of message strings

$$
\begin{aligned}
a_1 &= f(x, r) \\
a_2 &= g(x, a_1) \\
a_3 &= f(x, r, a_1, a_2) \\
a_4 &= g(x, a_1, a_2, a_3) \\
&\vdots
\end{aligned}
$$

define <u>output</u> $\text{out}_f\langle f, g\rangle(x)$ to be

$f(x, r, a_1, \ldots, a_k)$

Announced by <u>verifier</u>

Assumed to be either $\begin{cases} 1 & \text{verifier happy that } x \in L \\ 0 & \text{verifier } \underline{\text{not convinced}} \ x \in L \end{cases}$

<u>Transcript</u> sequence of messages sent
random variable over r

<u>DEF 1</u> (IP) (input length/size)

For $k = k(n)$, $L \subseteq \{0,1\}^*$ is in $\boxed{IP[k]}$ if

$\exists$ PTM V that is poly-time (in size of <u>input</u> x)

and can have k-round interaction with some $P: \{0,1\}^* \to \{0,1\}^*$

s.t.

COMPLETENESS $\quad x \in L \Rightarrow \exists P \ \ Pr[\text{out}_V\langle V, P\rangle(x) = 1] \geq 2/3$

SOUNDNESS $\quad x \notin L \Rightarrow \forall P' \ Pr[\text{out}_V\langle V, P'\rangle(x) = 1] \leq 1/3$

where all probabilities are over the choice of V's random string r.

$$IP = \bigcup_{c \in \mathbb{N}^+} IP[n^c]$$

The constants are again arbitrary.

LEMMA 2 We can set completeness parameter to $1 - 2^{-n^s}$ and soundness error parameter to 2^{-n^s} for any fixed $s > 0$ without changing IP

Proof Boosting as in our error reduction for BPP in Lecture 8.

PROP 3 The class of languages in IP decided by a deterministic verifier is exactly NP.

Proof See Arora-Barak.

- Allowing prover randomness does not help
  Can think of probabilistic prover as random choice between deterministic provers.
  If high success prob on average, then ∃ some deterministic prover with high success prob

- Prover is in principle unbounded - can even solve undecidable problems. But can be shown that given verifier V can compute optimal prover for x using poly(|x|) space
  Thus IP ⊆ PSPACE
  [More details needed for a formal proof, of course.]
  Could be nice pset problem.

- Possible to replace completeness 2/3 with exactly 1. (perfect completeness) Nontrivial fact.

- By contrast, reducing soundness error to exactly 0 makes verifier deterministic ⇒ collapse to NP.

- Randomness hidden from prover: private coin interactive proofs. Also ∃ public coin interactive proofs as we shall see soon.

- Boosting success probability by sequential repetition in Lem 2. Can also do parallel repetition by asking questions in parallel and running protocol in parallel
  But then prover can correlate answers
  However, success probability still decreases quickly.
  [We'll skip this.]

Ex 4 Interactive proof for GRAPH ISOMORPHISM

Now

GRAPH ISOMORPHISM = Notation $G_1 \cong G_2$

$$\left\{ \langle G_1, G_2 \rangle \;\middle|\; \begin{array}{l} \exists\, \pi : V(G_1) \to V(G_2) \text{ one-to-one \& onto} \\ \text{s.t. } (u,v) \in E(G_1) \Leftrightarrow (\pi(u), \pi(v)) \in E(G_2) \end{array} \right\}$$

GRAPH ISOMORPHISM $\in$ NP (certificate: π)
Not known to be in P
Probably not NP-complete (unless PH collapses)

Important component in this proof:

Interactive proof for GRAPH NON ISOMORPHISM (GNI for short)

Protocol for GNI

V: Pick $i \in_R \{1,2\}$

Apply random permutation π^* to G_i to get graph H with $V(H) = \{\pi^*(v) \mid v \in V(G_i)\}$

$E(H) = \{(\pi^*(u), \pi^*(v)) \mid (u,v) \in E(G_i)\}$

Send H to P.

P: Decide which G_j, $j \in \{1,2\}$, was used to produce H. Send j to P.

V: Accept if $i = j$; reject otherwise

Analysis

$G_1 \not\cong G_2$ Then all-powerful prover can decide whether $H \cong G_1$, or $H \cong G_2$ and answer accordingly

$G_1 \cong G_2$ Then prover just sees a uniformly random permutation of G_1, all the time

Statistically totally impossible to tell the difference regardless of computational resources

(Random coin flip best guess)

For any prover, success prob = 1/2.

Repeat twice to get prob below 1/3

Interesting aspect: what does verifier learn?

Nothing except whether $G_1 \cong G_2$

Zero-knowledge proof

What does is mean to "learn nothing"??

Show that verifier can produce transcript with correct distribution in poly time without prover

So if this conversation was helpful for something else, verifier could have talked to him-/herself...

Useful in cryptography

Ex Prove you know your password without revealing it.

More about this next time...

Public coins

What if prover sees coinflips?

Then all verifier needs to send are these coinflips — prover can compute questions him-/herself

DEF 5 Arthur-Merlin protocols

Arthur - king
Merlin - court magician

$AM[k] = IP[k]$ except verifier only sends random bits

Notation

$AM[2]$ Arthur first, then Merlin [$= AM$]

$AM[3] = AMA$

$AM[4] = AMAM$

MA Merlin (prover) goes first, then Arthur (verifier) flips coins and accepts/rejects.

LEMMA 6 $AM[2] = BP \cdot NP$

Proof A little bit of work. Could also be nice pset problem.

LEMMA 7 $\forall k \geq 2$, k constant, $AM[k] = AM[2]$

THEOREM 8 [Goldwasser, Sipser '87]
For every $k = k(n)$ computable in poly (n)

$$IP[k] \subseteq AM[k+2]$$

... Won't prove any of this ...

But can get $GNI \in AM[2]$, which in turn shows GI unlikely to be NP-complete

[Read Sec 8.2 in Arora-Barak if interested — there's just no way we can cover this ...]

We've argued $NP \overset{(1)}{\subseteq} IP \overset{(2)}{\subseteq} PSPACE$

(1) Probably strict — consider GNI protocol

What about (2)?

What does interaction buy you? Nothing, IP with deterministic ~~prover~~ verifier $= NP$

What does randomization buy you? Probably nothing; believe $P = BPP$

So what does interaction + randomization buy you? Not too much, right? Wrong

THEOREM 9 [Lund, Fortnow, Karloff, Nisan '90]
[Shamir '90]

$IP = PSPACE$

$IP \subseteq PSPACE$ not hard, as mentioned before.
Sufficient to show $TQBF \in IP[poly(n)]$

We'll prove something weaker that gives flavour of main ideas

THEOREM 10 $coNP \subseteq IP$

Want protocol for $\overline{3\text{-SAT}} = \{F \mid F \text{ unsatisfiable 3-CNF}\}$
Design more general protocol to prove # satisfying assignments

$$\#SAT_D = \{\langle \phi, K \rangle \mid \phi \text{ 3-CNF formula with exactly } K \text{ satisfying assignments}\}$$

[$K = 0$ gives $\overline{3\text{-SAT}}$ as special case]

Trick ARITHMETIZATION of CNF formula

Fix some suitable finite field $\mathbb{F}$
(e.g., computing modulo some prime p)

1 = truth = field element $\in \mathbb{F}$
0 = falsity = (other) element $\in \mathbb{F}$

$\neg x_i$ true iff $1 - x_i = 1$
$x_i \wedge x_j$ true iff $x_i \cdot x_j = 1$

Rewrite every 3-clause as degree-3 poly

$$x_i \vee \overline{x_j} \vee x_k \rightsquigarrow 1 - (1 - X_i) X_j (1 - X_k)$$

Evaluates to 1 iff clause true

Let clause C_j correspond to $p_j(X_1, X_2, \ldots, X_n)$ X

Represent $\phi = \bigwedge_{j=1}^{m} C_j$ as

$$P_\phi = \prod_{j=1}^{m} p_j(X_1, \ldots, X_m) \qquad (*)$$

Degree $\leq 3m$

Represent as product $(*) \Rightarrow$ representation in size $O(m)$

α satisfying assignment to $\phi \Rightarrow P_\phi(\alpha) = 1$

α falsifying $-\!\!-\!\!-$ $\Rightarrow P_\phi(\alpha) = 0$.

Hence # satisfying assignments is

$$K = \#\phi = \sum_{b_1 \in \{0,1\}} \sum_{b_2 \in \{0,1\}} \cdots \sum_{b_n \in \{0,1\}} P_\phi(b_1, \ldots, b_n) \qquad (2)$$

In fact, do calculation in (2) modulo some prime $p > 2^n \geq K$ (so that even mod p answer is correct)

Prover starts by sending prime $p \in (2^n, 2^{2n}]$

Verifier can check in poly time that p is prime.

Then run SUMCHECK protocol to verify that (2) holds (mod p). (To be described shortly)

So we want protocol to verify

$$K = \sum_{b_1 \in \{0,1\}} \cdots \sum_{b_n \in \{0,1\}} g(b_1, \ldots, b_n) \pmod{p} \qquad (3)$$

for g (Polynomial) with efficient representation so that it can be evaluated in poly time (holds for P_ϕ).

Note that plugging in values for $x_2, \ldots, x_n$

$$g(X_1, b_2, \ldots, b_n)$$

is univariate polynomial in X_1, and so is

$$h(X_1) = \sum_{b_2 \in \{0,1\}} \cdots \sum_{b_n \in \{0,1\}} g(X_1, b_2, \ldots, b_n) \qquad (4)$$

The equality (3) holds iff $h(0) + h(1) = K$

SUMCHECK (g, K, n) ← total degree $\leq d$

V: If $n = 1$, check $g(0) + g(1) = K$ and accept / reject

If $n \geq 2$, ask prover to send (efficient representation of) $h(X_1)$ in (4)

↑ easy, this is just univariate poly

P: Reply with $s(X_1)$ [supposedly $h(X_1)$]

V: Check $s(0) + s(1) = K$, reject otherwise.

Pick random $a \in [0, p-1]$

Let $K' = s(a)$

Let $g' = g(a, X_2, \ldots, X_n)$

Run SUMCHECK protocol for $(g', K', n-1)$

[to check that

$$s(a) = \sum_{b_2 \in \{0,1\}} \cdots \sum_{b_n \in \{0,1\}} g(a, b_1, \ldots, b_n) \;]$$

LEMMA 11

SUMCHECK (g, K, n) $(\text{mod } p)$ has
completeness 1
soundness error $\leq \frac{dn}{p}$

In our case if ϕ has m clauses then
P_ϕ has degree $\leq 3m$ (why?)

m clauses over n variables $\Rightarrow m \leq \binom{n}{3} 3^3 \leq 27 n^3$

Pick $p > 2^n$

Hence get soundness error

$$\leq \frac{dn}{p} < \frac{81 n^4}{2^n} \to 0 \text{ as } n \to \infty$$

So we just need to prove Lem 11

Completeness Obvious. Prover will just respond correctly with $h(X_1)$ and verifier will accept

Soundness By induction over n

Base case $(n=1)$ Want to ~~verify~~ detect if $\sum_{b \in \{0,1\}} g(b) \neq K$

verifier will just compute $g(0) + g(1)$
0% probability of being fooled

Induction step
In general case, want to detect if

$$\sum_{b_1 \in \{0,1\}} \cdots \sum_{b_n \in \{0,1\}} g(b_1, \ldots, b_n) \neq K.$$

Assume soundness error of
SUMCHECK $(g', K', n-1)$ is $\leq \frac{d}{p}(n-1)$

Two cases:

(a) Prover (honestly) returns $h(X_1)$ as $h(4)$
But then $h(0) + h(1) \neq K$ and
verifier rejects

(b) Prover returns $s(X_1) \neq h(X_1)$
$\deg(s(X_1) - h(X_1)) \leq d$
$\Rightarrow$ at most d roots
$\Rightarrow$ at most d values for a s.t. $s(a) = h(a)$

(i) If prover is lucky and verifier picks such an a, then verifier is fooled

(ii) Otherwise, we get a sumcheck instance over $n-1$ variables with wrong value

$\Pr[\text{verifier fooled}] =$

$\Pr[\text{verifier fooled in case (i)}] + \Pr[V \text{ fooled in case (ii)}]$

$\leq \frac{d}{p} + \frac{d}{p}(n-1) = \frac{dn}{p}$ QED

Proves $coNP \subseteq IP$.

This is actually most of what is needed for $PSPACE \subseteq IP$ also, but some extra twists needed that we didn't have time to do.

SUMMARY

Interactive proofs

Verifier computationally bounded (poly time)
can use randomness (private)

Prover all-powerful but capricious

Want protocols s.t.

COMPLETENESS $x \in L \Rightarrow \exists$ Prover convinces verifier

SOUNDNESS $x \notin L \Rightarrow$ No prover can fool verifier

Some protocol for GRAPH NONISOMORPHISM

Interactive proofs with public coins
Arthur-Merlin

Surprisingly, IP = PSPACE

Proved weaker result coNP $\subseteq$ IP,
but main ingredients there

Key idea: ARITHMETIZATION
CNF formula $\rightsquigarrow$ polynomial