Probabilistically Checkable & Interactive Proofs **Spring 2020**

Lecture 1: Interactive Proofs, dIP, GNI, AM/MA

Scribe: Max Ovsiankin 2/30/2020

Welcome to the UGTCS PCPIP reading group! The prerequisites for this reading group are formally only CS 170, although it would definitely help if you have some background in complexity theory. Nevertheless, we will try to introduce the relevant complexity-theoretic notions and definitions as we go along.

This reading group will also cover some pretty mathematically difficult ideas, so be prepared! Confusion is a normal and expected part of the process.

1.1 Complexity Theory Background

Complexity theory as we know it from CS 170 is about measuring the difficulty of problems by the efficiency of the algorithms that solve them. As a shorthand, we say efficient algorithms are exactly those that are polynomial-time (ignoring details such as leading constants or polynomial degree). Why this 'extended Church-Turing thesis' seems to hold is a story for another time, but the class of problems that have poly-time algorithms are called P.

Languages that are in P are not very problematic. From both a complexity and cryptography perspective, you could say the information in P languages can be 'known' by any party with access to reasonable computational power.

But now, let's say that in a complexity-theoretic setting, you 'know' something I don't. You want to convince me that this thing you know is true. We already know from CS 170 a complexity class that encodes this idea:

Definition 1.1. *$L \in \mathsf{NP}$ if there exists a polynomial $q\colon \mathbb{N} \to \mathbb{N}$ and Turing machine $V(x, \pi)$ (for verifier) such that V runs in time $\mathrm{poly}(|x|)$, and satisfies the following two properties:*

- ***Completeness**: If $x \in L$, then there exists proof string $\pi \in \{0,1\}^{q(|x|)}$ such that $V(x, \pi) = 1$*
- ***Soundness**: If $x \notin L$, then for all $\tilde{\pi}$, $V(x, \tilde{\pi}) = 0$.*

This may not be the definition of NP that you recognize, but we've written it in a suggestive manner because we're about to modify it. You can think of NP as encoding the notion of a 'proof system', where you give me a short proof, and I do an efficient check on the proof to see if it's correct. If something is true, you should be able to convince me it is true with some proof (this is completeness). If something is false, you shouldn't be able to convince me it is true, no matter what proof you give me (this is soundness).

We know that $\mathsf{P} \subseteq \mathsf{NP}$. It is a fundamental open problem whether $\mathsf{P} \subsetneq \mathsf{NP}$, i.e. if there is truly a gap between things that are poly-time 'knowable' and things that you can convince me are true with a short proof string.

Maybe the fact that it only takes a short proof string to convince me means I should be able to know it by myself. Regardless, it's interesting to explore what happens when we push this notion of 'short proof string'. A great many rich innovations have been made in the 30 years since this thought.

In real life, we have more available to us than just you giving me a piece of paper and me verifying it. What happens if we have a conversation, instead?

1.2 Deterministic Interactive Proofs

Imagine connecting two parties such that they share a tape, and can talk to each other. One will be the prover P, and the other will be the verifier V.

Definition 1.2. *The variable $\langle P, V\rangle(x)$ respresents the following interaction (which we call a k-*round interaction*):*

(0.) V and P both have access to input x

1. *Round 1: V sends message a_1 to P, $V \xrightarrow{a_1} P$*
2. *Round 2: $P \xrightarrow{a_2} V$*
3. *Round 3: $V \xrightarrow{a_3} P$*

   $\vdots$

k. *Round k: $P \xrightarrow{a_k} V$*

(n+1). set $\langle P, V\rangle(x) = V(x, a_1, \ldots, a_k)$, i.e. V's view after the interaction finishes

Usually V is a poly-time Turing machine. P can be a bounded prover, or an arbitrary function that can even evaluate things that are uncomputable.

Lets define a class dIP, which is exactly NP extended to allow interaction. We restrict the verifier to be poly-time just like in the definition of NP. However, just as there's a 'magic' proof string π, we allow the prover P unlimited power.

Definition 1.3. $L \in \mathsf{dIP}$ *if there exists a Turing machine $V(x)$ (for verifier) such that V runs in time* $\text{poly}(|x|)$ *for each round, and satisfies the following properties:*

- ***Completeness**: If $x \in L$, then there exists unbounded prover P such that $\langle P, V\rangle(x) = 1$*
- ***Soundness**: If $x \notin L$, then for all unbounded provers $\tilde{P}$, $\langle \tilde{P}, V\rangle(x) = 0$*
- ***Complexity**: The number of rounds of $\langle P, V\rangle(x)$ is polynomial in $|x|$*

How is dIP related to NP? Well, $\mathsf{NP} \subseteq \mathsf{dIP}$, as that is exactly a 1-round interaction from the prover to the verifier. However, it turns out that allowing interaction has not given us anything more!

Claim 1.4. $\mathsf{dIP} = \mathsf{NP}$

.

Proof. $\mathsf{NP} \subseteq \mathsf{dIP}$ as remarked above.

To show $\mathsf{dIP} \subseteq \mathsf{NP}$, let $L \in \mathsf{dIP}$. Consider the poly-time verifier V', which takes a transcript $(a_1, \ldots, a_k)$ and checks that $a_1 = V(x)$, $a_3 = V(x, a_1, a_2)$, and so on, until finally it checks that $V(x, a_1, \ldots, a_k) = 1$. We will use this transcript as the NP-certificate.

Completeness: If $x \in L$, then there is some transcript $(a_1, \ldots, a_k)$ produced by P that causes V to accept. But then V' accepts this transcript.
Soundness: (we show the contrapositive) If there exists some transcript $(a_1, \ldots, a_k)$ that V' accepts, then there is a prover P that emits this transcript, so that $\langle P, V \rangle(x) = 1$. But then $x \in L$. □

1.3 Graph Non-Isomorphism

Before we relax the notion of IPs and hopefully bring them more power, let's explore an interesting problem we believe to be beyond NP. Consider two graphs G_1, G_2 with $V_1 = V_2 = \{1, \ldots, n\}$. We say these two graphs are *isomorphic* (written $G_1 \cong G_2$) if there exists some permutation $\pi \colon \{1, \ldots, n\} \to \{1, \ldots, n\}$ such that $G_2 = \pi(G_1)$. In essence, the two graphs are isomorphic if their vertices are relabelings of each other, or if they 'look the same'.

GI is the language consisting of all pairs (G_1, G_2) such that $G_1 \cong G_2$. We can see that $\mathsf{GI} \in \mathsf{NP}$, as π is our short certificate, and π can be easily verified as a permutation that modifies the graphs in the correct way. It is not known if $\mathsf{GI} \in \mathsf{P}$.

GNI is the exactly the complement: the language consisting of all pairs (G_1, G_2) such that $G_1 \ncong G_2$. GNI is in coNP, but is not believed to be in NP. In fact, it's not immediately clear how one can give a short proof, or even an interaction as a witness to the fact that two particular graphs G_1, G_2 are *not* isomorphic. However, it turns out that if we allow the poly-time verifier to use random coin flips, there is an interactive protocol for GNI!

Let us discuss one last thing about graph isomorphism, that we will use to construct the interactive protocol:

Claim 1.5. *Let $G_1 \cong G_2$. If S_n is all permutations on n vertices, then for all graphs H:*

$$\Pr_{\pi \leftarrow S_n}[\sigma(G_1) = H] = \Pr_{\sigma \leftarrow S_n}[\sigma(G_2) = H]$$

where $\sigma \leftarrow S_n$ means that σ is chosen uniformly at random from the set S_n. In other words, $\sigma(G_1)$ and $\sigma(G_2)$ have exactly the same distribution.

Proof. Let π be the permutation such that $\pi(G_1) = G_2$. For each H the set of permutations $\{\sigma \colon \sigma(G_1) = H\}$ is in one-to-one correspondence with the set $\{\sigma \colon \sigma(G_2) = H\}$ by the mapping $\sigma \mapsto \sigma \circ \pi$. As the distribution of σ is uniform over S_n, that means the probability of these events is the same. □

In other words, we've shown that a random permutation applied to G_1 has exactly the same distribution as one applied to G_2. The implication is that it is impossible for anyone, even a computationally unbounded prover, to distinguish between $\sigma(G_1)$ and $\sigma(G_2)$.

1.4 Interactive Proofs

We are now ready to define interactive proofs:

Definition 1.6. $L \in \mathsf{IP}$ *if there exists a (probabilistic) polynomial-time* V*, such that:*

- ***Completeness****: If* $x \in L$*, then there exists a* P *where* $\Pr[\langle P, V\rangle(x) = 1] = 1$*.*
- ***Soundness****: If* $x \notin L$*, then for any* $\tilde{P}$*, we have* $\Pr[\langle P, V\rangle(x) = 1] \leq \frac{1}{2}$*.*
- ***Complexity****: The number of rounds of* $\langle P, V\rangle(x)$ *is polynomial in* $|x|$*.*

In this definition, both probabilities are now taken over the random coin flips of V. Note also that we can make soundness exponentially decay with sequential repetitions of the protocol. More explicitly, if we repeat the protocol $|x|^c$ times so that the soundness error now becomes negligible in $|x|$, that is still in IP.

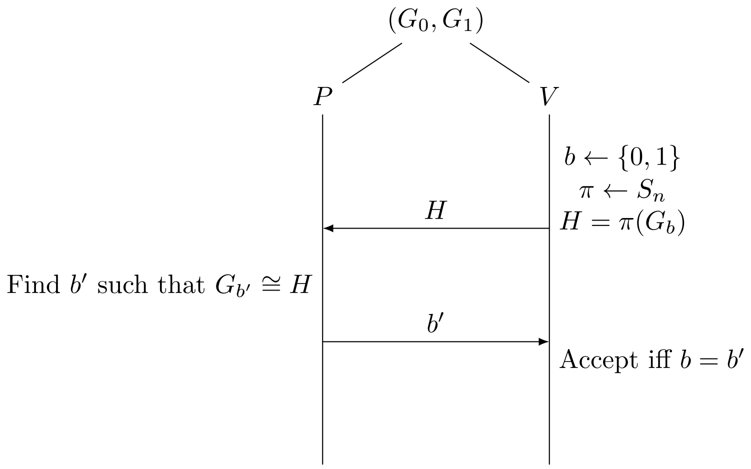
As hinted in the last section, there is an interactive protocol for GNI under this definition:

Theorem 1.7. [GMW91] $\mathsf{GNI} \in \mathsf{IP}$*.*

Proof. Consider the following protocol:

0. P and V are both given access to $x = (G_0, G_1)$.
1. V picks $b \leftarrow \{0, 1\}$ and $\pi \leftarrow S_n$, and sends $H = \pi(G_b)$ to P
2. P uses its unlimited resources to find the b' such that $G_{b'} \cong H$ (for example, by trying all permutations of G_0 and G_1), and sends b' to the verifier
3. V accepts iff $b = b'$

We can visualize the protocol with the following sequence diagram:



To show that $\mathsf{GNI} \in \mathsf{IP}$, we need to establish complexity, completeness, and soundness. The verifier is clearly polynomial-time on each step.

Completeness: Let $G_0 \ncong G_1$. Then P can, by emuerating all permutations of G_0 and G_1, discover which H is isomorphic to. Then $\Pr[b = b'] = 1$, so $\Pr[\langle P, V\rangle(G_0, G_1) = 1] = 1$.

Soundness: Let $G_0 \cong G_1$. By Claim 1.5, $H \mid b = 0$ has the same distribution as $H \mid b = 1$. Therefore, b' must be independent of b (regardless of what the prover does), so $\Pr[b = b'] = \frac{1}{2}$.

□

This is our first indication that IP brings some power just beyond regular proofs in the NP sense!

1.5 Round Complexity, AM/MA

We can consider number of rounds to be measure of the efficiency of a protocol, although IP does not allow us to differentiate this beyond some polynomial number of rounds. Let $k \colon \mathbb{N} \to \mathbb{N}$ count the round complexity as a function of input size. Then we let $\mathsf{IP}[k]$ denote the class of languages L that have interactive proofs with $k(|x|)$ rounds for any $x \in L$.

An essential feature of Theorem 1.7 appeared to be that the verifier was able to give a 'hidden' challenge to the prover, and the only way the prover could reliably complete this challenge was if $G_0 \neq\cong G_1$. What happens if we restrict the verifier from giving 'hidden' challenges to the prover?

Definition 1.8. *Let $k \colon \mathbb{N} \to \mathbb{N}$. Then we define:*

- $\mathsf{AM}[k]$ *is the subset of* $\mathsf{IP}[k]$ *where the verifier must send all its random bits as messages, and all its messages can only be random bits. Like for* $\mathsf{IP}[k]$*, the verifier goes first.*

- $\mathsf{MA}[k]$ *is exactly like* $\mathsf{AM}[k]$*, except the prover goes first.*

Some quick remarks on these definitions. AM and MA refer to *Arthur-Merlin* and *Merlin-Arthur*, where *Arthur* is a probabilistic polynomial-time verifier, and *Merlin* is the unbounded prover. Because we now involve round complexity explicitly in the definitions, it matters more who goes first (as for some protocols who goes first could change how many rounds are needed). Proof systems of the form deliniated by AM/MA are called *public-coin*, as all the randomness used by V is known by the prover.

The following theorem, which we will cover in later weeks, shows us that it wasn't the private randomness that gave IP the power to prove GNI:

Theorem 1.9. [GS86] *For all poly-time computable* k*,* $\mathsf{IP}[k] \subseteq \mathsf{AM}[k+2]$.

In other words, by adding only two rounds we can turn any private-coin protocol into a public-coin one! This simple fact turns out to have major implications in both cryptography and complexity.

References

[GMW91] Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity or all languages in np have zero-knowledge proof systems. *Journal of the ACM (JACM)*, 38(3):690–728, 1991.

[GS86] Shafi Goldwasser and Michael Sipser. Private coins versus public coins in interactive proof systems. In *Proceedings of the eighteenth annual ACM symposium on Theory of computing*, pages 59–68, 1986.