Interactive Proof Systems

JONATHAN XUE

ABSTRACT. Interactive proof systems are a method of using two agents, a prover and a verifier, to determine the truth of a certain statement. We introduce the concept of probabilistic and deterministic verifiers for such systems and evaluate them on metrics of completeness and soundness. Finally, we use an interactive proof system to show that graphs are non-isomorphic and finish with the concept of public coins.

1 Introduction

We begin with an informal explanation of interactive proof systems before establishing an explicit definition. To begin, we use the standard definition for a proof in **NP**, which states that a proof is a sequence of symbols, such that only provable statements have a corresponding valid sequence. For our interactive proof system, we have two agents, a prover and a verifier. The prover attempts to convince the verifier of the validity of a statement. The verifier attempts to determine whether the prover's argument definitively proves the statement's truth. Both prover and verifier alternate sending the other messages, which are strings, to achieve their respective goals, until the verifier finally reaches a conclusion as to whether the proof is sufficient or not. In their communication, both agents have the ability to keep track of all previous messages. We formally define a deterministic interactive proof system and its messages in the section below.

2 Deterministic Systems

Definition 2.1 (Messages of Deterministic Interactive Proof System)**.** Given a prover p and a verifier v, let them be functions such that $p, v : A^* \to A^*$, where A is an alphabet. Then, given a statement $x \in A^*$, we define the messages $m_1, \dots m_{2n} \in A^*$ sent between p and v of an interactive proof system with $2n$ messages to be as follows:

$$\begin{aligned} m_1 &= v(x) \\ m_2 &= p(x, m_1) \\ m_3 &= v(x, m_1, m_2) \\ &\dots \\ m_{2n-1} &= v(x, m_1, m_2, \dots, m_{2n-2}) \\ m_{2n} &= p(x, m_1, m_2, \dots, m_{2n-1}) \end{aligned} \tag{1}$$

After the last message sent, the verifier must either accept or reject the proof. We denote this output of the interaction as $\text{out}_v\langle p, v\rangle$. If $\text{out}_v\langle p, v\rangle = 1$, the proof is accepted. Otherwise, it is rejected.

Definition 2.2 (Deterministic Interactive Proof System)**.** A language $L = A^*$ has a *deterministic interactive proof system* if there's a deterministic Turing machine V that on input x, m_1, m_2, m_k runs in polynomial time which is both *complete* and *sound*:

$$(\text{Completeness}) x \in L \implies \exists P : L \to L \mid \text{out}_V\langle P, V\rangle = 1$$

$$(\text{Soundness}) x \notin L \implies \forall P : L \to L \mid \text{out}_V\langle P, V\rangle = 1$$

Informally, completeness means that all true statements will have a valid proof that the interactive proof system accepts, and that no false statement will have a valid proof. For an interactive proof system to be recognized, it must be both complete and sound.

2.1 dIP and NP Equivalence

Now that we have established a definition for deterministic interactive proof systems, we can examine their relationship with the **NP** class.

Theorem 1. The class **dIP** that consists of all languages with a deterministic interactive proof system is equivalent to the class **NP**.

Proof. By definition, every language in **NP** has a valid proof, and will have a deterministic interactive proof system that consists of one message. Thus, we only need to show that if a language L has such an interactive proof system, then $L \in$ **NP**. Since L has an interactive proof system, it must have a sequence of messages $m_1, m_2, \dots, m_k$ produced by the prover and verifier. To show that these messages produce a proof, we can verify that $V(x) = m_1, V(x, m_1, m_2) = m_3, \dots V(x, m_1, \dots, m_{k-1}) = m_k$. Since we know that such a sequence is the valid basis for an interactive proof system, we can then create a prover P such that $P(x, m_1) = m_2, P(x, m_1, m_2, m_3) = m_4, \dots,$ which means that $\text{out}_V\langle P, V\rangle = 1$, and thus $x \in L$. □

3 Probabilistic Verifiers

However, there are certain problems that cannot be proved with deterministic interactive proof systems. For these, we turn to interactive proof systems with a probabilistic verifier. While using a probabilistic verifier means that the system is not entirely complete and sound, it will have a high enough probability of success. We find that using randomness makes the system more robust, allowing for the verifier to achieve high accuracies regardless of the prover's methods. For the sake of interactive proof systems with probabilistic verifiers, we redefine completeness and soundness.

Definition 3.1 (Interactive Proof System)**.** . If we have a language L, we say that that (P, V) is an interactive proof system on L if:

$$\text{(Completeness) } \forall x \in L, \Pr(\text{out}_V \langle P, V \rangle = 1) \geq \frac{2}{3}$$

$$\text{(Soundness) } \forall x \notin L, \Pr(\text{out}_V \langle P, V \rangle = 1) \leq \frac{1}{3}$$

where V is a Turing machine that runs in polynomial time.

We provide an example to illustrate the added utility of having a probabilistic verifier over a deterministic one. Consider the following: Alice claims she can distinguish between black and white socks with her eyes closed. To verify this, Bob picks a random sock and asks Alice to distinguish it. If Bob repeats this process 40 times and Alice is able to determine the color of the sock each time, Bob can be fairly confident that Alice was initially telling the truth. Each time, without any particular skill, Alice would be able to guess the correct color with a $\frac{1}{2}$ probability, but given the amount of trials, guessing correctly all 40 times would be unlikely. Interactive proof systems with probabilistic verifiers employ the same logic, where by querying the prover many times, the verifier can be convinced of a statement. We now show the following two theorems.

Theorem 2. The class **IP** of all languages with an interactive proof system will not change if a prover is allowed to be *probabilistic*, that is the message sent m_i is determined by a random string used by the prover.

Proof. If we have some language L, whose probabilistic prover P causes the verifier V to accept with probability p, by producing statements based off a random function, if we average all the probabilities of acceptance by V, we know there is some deterministic prover P', which has the same probability of acceptance as P. Thus, any language that has a deterministic proof system can also be expressed as a languaged which has a proof system with a probabilistic prover, and **IP** will remain the same. □

Theorem 3. Given an interactive proof system with probabilistic verifier, we can design a method such that given some arbitrary positive value $\epsilon < 1$, the probability of correctly classifying an input is greater than $1 - \epsilon$.

Proof. If we run our interactive proof system some large number of times k, we can take the majority outcome, and cause our probability of correct classification to become arbitrarily close to 1. For a repetition of k times, the probability that we will correctly classify an input by majority, assuming k is odd, is

$$1 - \frac{2^{\lceil \frac{k}{2} \rceil} - 1}{3^k}$$

. Thus, so long as $\frac{2^{\lceil \frac{k}{2} \rceil}-1}{3^k} \leq \epsilon$, our condition is satisfied, and thus we are done since we can always choose a k that satisfies the given constraint. □

3.1 Proof of Graph Non-Isomorphism

We now give an example of a language not in **NP** that is in **IP**. If we use a numbering of vertices to represent a graph, we say that two graphs G_1, G_2 are isomorphic if they are the same, only with a reordering of the vertices. There is a problem GI that given two graphs, seeks to determine if they are isomorphic. In this proof, we show that the complement of GI, NGI, which decides whether given two graphs if they are non-isomorphic, is in **IP**, but not **NP**.

Proof. We begin with graphs G_1, G_2. To show that NGI is in **IP**, but not **NP**, it suffices to provide a interactive proof system with a probabilistic verifier and show that it is impossible to provide such a system with a deterministic verifer. First, we show that an IPS with a probabilistic verifier exists with the following protocol:

V: choose $i \in 1, 2$ at random and perform a random permutation of the vertices on G_i. Call this new graph H and send it to P
P: identify whether G_1 or G_2 was used to produce H. If G_j is determined to have been used, send j to V.
V: if $i = j$, accept. Reject otherwise.

We know this system is completel, as for any two graphs which are non-isomorphic, the all-knowing prover can determine which graph the permutation is derived from. However, if the graphs are isomorphic, then the probability of selecting which graph is $\frac{1}{2}$, meaning it is not sound. Given that it is impossible to create a perfectly sound system due to the nature of the problem, we have shown that NGI **NP**, and our protocol shows that NGI $\in$ **IP**, so we are done. □

4 Public Coins

So far, we have discussed proof systems with probabilistic verifiers, where the prover cannot access the verifier's random string. These are called *private coin* proofs. We now discuss systems where the prover has full access of the verifier's random string, leading to the model of interactive proof systems with *public coins.*

Definition 4.1 (Public Coin Proof)**.** . We denote by **AM**, the class of languages that can be decided by an interactive proof where the verifier's messages consist of sending a random string and are decided by tossing coins, whose result is made public to the prover. Such a proof is known as a public coin proof or an Arthur-Merlin proof.

We finish the discussion earlier about graph non-isomorphism with the following claim.

Theorem 4. GNI $\in$ **AM**.

Proof. We define the set $S = \{H : H \equiv G_1 \cup H : H \equiv G_2\}$, where $\equiv$ represents isomorphism. We know by definition that $H \in S$, because it's a permutation of some graph G_1, G_2, and is thus isomorphic to one of them. If $G_1 \equiv G_2$, then $|S| = n!$, where G_1, G_2 have n vertices. Otherwise, if $G_1, \not\equiv G_2$, then they will share no isomorphic graphs and $|S| = 2n!$. Thus, now to convince the verifier of non-isomorphism, we need only show that $|S| = 2n!$. By applying the Goldwasser-Sipser Lower-Bound Protocol [3] with $p = \frac{K}{2^k}$, we need only show that if $|S| \geq K$, then the verifier will accept with greater probability than $\frac{p}{2}$. Previously, we've shown that by iterating a interactive proof system with probabilistic verifier multiple times, we can get a higher accuracy. Since we deal in public coins, we run through the system an arbitrary amount of times, then accept if the fraction of acceptances was greater than $0.6p$. To achieve this, it is clear that the completeness probability must be greater than $\frac{2}{3}$ and the soundness probability must be less than 13, so we are done. $\square$

5 References

[1] Florian Tramer.*Lecture 3: Interactive Proofs.* 2019

[2] Princeton. *Chapter 9: Interactive Proofs.*

[3] Yale. *The Goldwasser-Sipser Lower-Bound Protocol*