Prover and Verifier

- There are two parties to a proof.
  - The **prover** (**Peggy**).
  - The **verifier** (**Victor**).
- Given an assertion, the prover's goal is to convince the verifier of its validity (**completeness**).
- The verifier's objective is to accept only correct assertions (**soundness**).
- The verifier usually has an easier job than the prover.

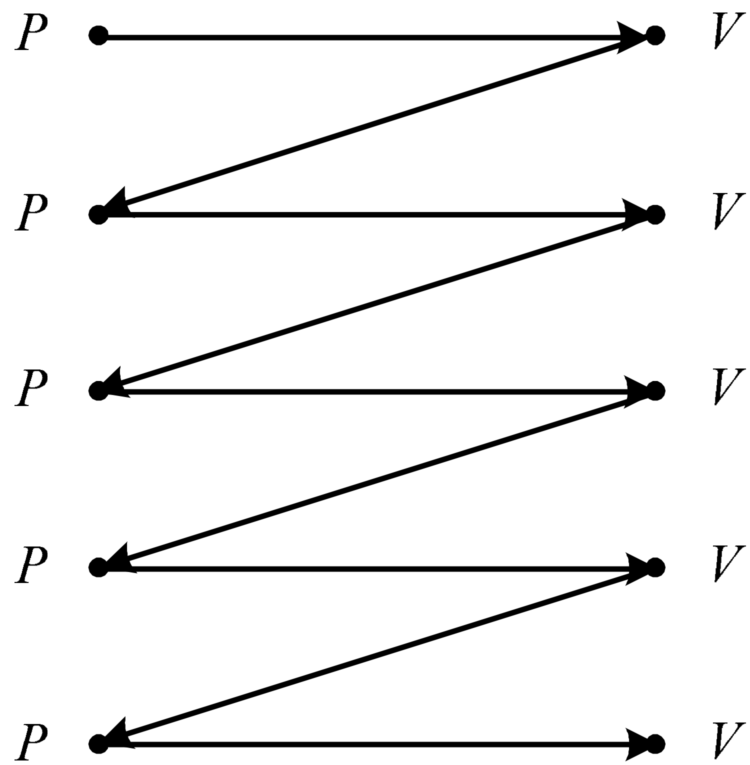
Interactive Proof Systems

- An **interactive proof** for a language L is a sequence of questions and answers between the two parties.
- At the end of the interaction, the verifier decides based on the knowledge he acquired in the proof process whether the claim is true or false.
- The verifier must be a probabilistic polynomial-time algorithm.
- The prover runs an exponential-time algorithm.

Interactive Proof Systems (continued)

- The system decides L if the following two conditions hold for any common input x.
  - If $x \in L$, then the probability that x is accepted by the verifier is at least $1 - 2^{-|x|}$.
  - If $x \notin L$, then the probability that x is accepted by the verifier with *any* prover replacing the original prover is at most $2^{-|x|}$.
- Neither the number of rounds nor the lengths of the messages can be more than a polynomial in $|x|$.

An Interactive Proof



IP[a]

- **IP** is the class of all languages decided by an interactive proof system.
- When $x \in L$, the completeness condition can be modified to require that the verifier accepts with certainty without affecting IP.
- Similar things cannot be said of the soundness condition when $x \notin L$.

[a] Goldwasser, Micali, Rackoff, 1985.

The Relations of IP with Other Classes

- NP $\subseteq$ IP.
  - IP becomes NP when the verifier is deterministic.
- BPP $\subseteq$ IP.
  - IP becomes BPP when the verifier ignores the prover's messages.
- IP actually coincides with PSPACE.[a]

[a] Shamir, 1990.

Graph Nonisomorphism

- Two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are isomorphic if there exists a π which is a one-one and onto mapping of the nodes set V_1 to V_2 so that $(u, v) \in E_1$ if and only if $(\pi(u), \pi(v)) \in E_2$.
- $V_1 = V_2 = \{1, 2, \ldots, n\}$.
- The task is to answer if $G_1 \not\cong G_2$.
- Little is known about the complexity of the problem except that it is in coNP (how about NP? NP-complete? BPP?).
- No known polynomial-time algorithms.

A 2-Round Algorithm

Victor selects a random $i \in \{1, 2\}$;
Victor selects a random permutation π on $\{1, 2, \ldots, n\}$;
Victor applies π on graph G_i to obtain graph H;
Victor sends (G_1, H) to Peggy;
if $G_1 \cong H$ **then**
    Peggy sends $j = 1$ to Victor;
else
    Peggy sends $j = 2$ to Victor;
end if
if $j = i$ **then**
    Victor accepts;
else
    Victor rejects;
end if

Analysis

- Victor runs in probabilistic polynomial time.
- Suppose the two graphs are no isomorphic.
  - Peggy is able to tell which G_i is isomorphic to H.
  - Hence Victor always accepts.
- Suppose the two graphs are isomorphic.
  - No matter which i is picked by Victor, Peggy or anybody always sees identical graphs.
  - Peggy or anybody with exponential power has only probability one half of guessing i correctly.
  - Hence Victor accepts with probability $1/2$.
- Repeat the algorithm to obtain the desired probabilities.

Zero Knowledge Proofs[a]

- An interactive proof protocol (P, V) for language L has the perfect **zero-knowledge** property if:
  - For every verifier V', there is a probabilistic algorithm M with expected polynomial running time.
  - M on any input $x \in L$ generates the same probability distribution as the one that can be observed on the communication channel of (P, V') on input x.
- Whatever a verifier can learn from the specified prover P via the communication channel could as well be computed from the verifier alone.

[a]Goldwasser, Micali, Rackoff, 1985.

Comments

- The verifier does not learn anything except "$x \in L$."
- For all practical purposes "whatever" can be done after interacting with a zero-knowledge prover can be done when just believing that the assertion he claims is indeed valid.
- Zero knowledge is a property of the prover.
  - It is the robustness of the prover against attempts of the verifier to extract knowledge via interaction.
  - The verifier may deviate arbitrarily (but in polynomial time) from the predetermined program.

Comments (continued)

- Zero-knowledge proofs yield no knowledge in the sense that they can be constructed by the verifier who believes the statement, and yet these proofs do convince him.
- The "paradox" is resolved by noting that it is not the text of the conversation that convinces the verifier, but the fact that this conversation was held "on line."
- There is no zero-knowledge requirement when $x \notin L$.
- *Computational* zero-knowledge proofs are based on complexity assumptions.

Zero-Knowledge Proof of 3 Colorability

for $i = 1, 2, \ldots, |E|^2$ **do**

  Peggy chooses a random permutation π of the 3-coloring ϕ;

  Peggy encrypts it as $\pi(\phi(1)), \pi(\phi(2)), \ldots, \pi(\phi(|V|))$ and sends it to Victor;

  Victor chooses at random an edge $e \in E$ and sends it to Peggy for the coloring of the endpoints of e;

  if $e = (u, v) \in E$ **then**

    Peggy reveals the coloring of u and v and "proves" that they correspond to their encryption;

  else

    Peggy stops;

  end if

10: **if** the "proof" provided in Line 6 is not valid **then**
11:     Victor rejects and stops;
12: **end if**
13: **if** $\pi(\phi(u)) = \pi(\phi(v))$ or $\pi(\phi(u)), \pi(\phi(v)) \notin \{1, 2, 3\}$ **then**
14:     Victor rejects and stops;
15: **end if**
16: **end for**
17: Victor accepts;

The algorithm is due to Goldreich, Micali, Wigderson, 1986.

Analysis

- If the graph is 3-colorable and both prover and verifier follow the protocol, then the verifier always accepts.
- If the graph is not 3-colorable and the verifier follows the protocol, then no matter how the prover plays, the verifier will accept with probability at most $(1 - m^{-1})^{m^2} \leq e^{-m}$.
- Thus, the protocol is valid.
- This protocol yields no knowledge to the specified verifier, since all he gets is a sequence of random pairs.
- The proof that the protocol is indeed zero-knowledge (with respect to *any* verifier) is much more complex.

Tackling Intractable Problems

- Many important problems are NP-complete or worse.
- **Heuristics** have been developed to attack them.
- They are **approximation algorithms**.
- How good are the approximations?
  - We are looking for theoretically *guaranteed* bounds, not "empirical" bounds.
- Are there problems that cannot be approximated well?

Some Definitions

- Given an **optimization problem**, each problem instance x has a set of **feasible solutions** $F(x)$.
- Each feasible solution $s \in F(x)$ has a cost $c(s) \in \mathbb{Z}^+$.
- The **optimum cost** is $\text{OPT}(x) = \min_{s \in F(x)} c(s)$ for a minimization problem.
- It is $\text{OPT}(x) = \max_{s \in F(x)} c(s)$ for a maximization problem.

Approximation Algorithms

- Let algorithm M on x returns a feasible solution.
- M is an **ϵ-approximation algorithm**, where $\epsilon > 0$, if for all x,

$$\frac{|c(M(x)) - \text{OPT}(x)|}{\max(\text{opt}(x), c(M(x)))} \leq \epsilon.$$

  - For a minization problem,

$$\frac{c(M(x)) - \min_{s \in F(x)} c(s)}{c(M(x))} \leq \epsilon.$$

  - For a maximization problem,

$$\frac{\max_{s \in F(x)} c(s) - c(M(x))}{\max_{s \in F(x)} c(s)} \leq \epsilon.$$

Lower and Upper Bounds

- For a minization problem,

$$c(M(x)) \leq \frac{\min_{s \in F(x)} c(s)}{1-\epsilon}.$$

- For a maximization problem,

$$(1-\epsilon) \times \max_{s \in F(x)} c(s) \leq c(M(x)).$$

Comments

- ϵ takes values between 0 and 1.
- For maximization problems, an ϵ-approximation algorithm returns solutions that are never smaller than $1 - \epsilon$ times the optimum.
- For minimization problems, an ϵ-approximation algorithm returns solutions that are never more than $\frac{1}{1-\epsilon}$ times the optimum.
- For each NP-complete optimization problem, we shall be interested in determining the *smallest* ϵ for which there is a polynomial-time ϵ-approximation algorithm.
- Sometimes, ϵ has no minimum value.

Approximation Thresholds

- The **approximation threshold** is the greatest lower bound of all $\epsilon > 0$ such that there is a polynomial-time ϵ-approximation algorithm.
- The approximation threshold of an optimization problem can be anywhere between 0 (approximation to any desired degree) and 1 (no approximation is possible).
- If P = NP, then all optimization problems in NP have approximation threshold 0.

NODE COVER

- NODE COVER seeks the smallest $C \subseteq V$ in graph $G = (V, E)$ such that for each edge in E, at least one of its endpoints is in C.
- A heuristic to obtain a good node cover is to iteratively move a node with the highest degree to the cover.
- This turns out to produce $c(M(x))/\text{OPT}(x) = \Theta(\log n)$.
- It is not an ϵ-approximation algorithm for any $\epsilon < 1$.

A 0.5-Approximation Algorithm

$C := \emptyset$;
while $G \neq \emptyset$ **do**
    Delete any edge $[u, v]$ from G;
    Add u and v to C;
end while
return C;

The Analysis

- C contains $|C|/2$ edges.
- No two edges of C share a node.
- *Any* node cover must contain at least one node from each of these edges.
- This means that $\text{OPT}(G) \geq |C|/2$.
- So

$$\frac{|C| - \text{OPT}(G)}{|C|} \leq 1/2.$$

Maximum Satisfiability

- Given a set of clauses, MAXSAT seeks the truth assignment that satisfies the most.
- MAX2SAT is already NP-complete (p. 197).
- Consider the more general k-MAXGSAT.
  - Given a set of boolean expressions $\Phi = \{\phi_1, \phi_2, \ldots, \phi_m\}$ in n variables.
  - Each ϕ_i is a *general* expression involving k variables.
  - k-MAXGSAT seeks the truth assignment that satisfies the most expressions.

A Probabilistic Interpretation of an Algorithm

- Each ϕ_i involves k variables and is satisfied by t_i of the 2^k truth assignments.
- A random truth assignment $\in \{0,1\}^n$ satisfies ϕ_i with probability $p(\phi_i) = t_i / 2^k$.
- Hence a random truth assignment satisfies an expected number

$$p(\Phi) = \sum_{i=1}^{m} p(\phi_i)$$

of expressions ϕ_i.

The Search Procedure

- Clearly

$$p(\Phi) = \frac{1}{2} \{ p(\Phi[x_1 = \texttt{true}]) + p(\Phi[x_1 = \texttt{false}]) \}.$$

- Select the $t \in \{\texttt{true}, \texttt{false}\}$ such that $p(\Phi[x_1 = t])$ is the larger one.
- Note that $p(\Phi[x_1 = t]) \geq p(\Phi)$.
- Repeat with expression $\Phi[x_1 = t]$ until all variables have been given truth values and all ϕ_i are either true or false.
- At least $p(\Phi)$ expressions are satisfied because our expectation never decreased in the search process.

Approximation Threshold

- The optimum is at most the number of satisfiable ϕ_i—i.e., those with $p(\phi_i) > 0$.
- Hence the ratio of algorithm's output vs. the optimum is

$$\leq \frac{p(\Phi)}{\sum_{p(\phi_i)>0} 1} = \frac{\sum_i p(\phi_i)}{\sum_{p(\phi_i)>0} 1} \leq \min_{p(\phi_i)>0} p(\phi_i).$$

- The heuristic is a polynomial-time ϵ-approximation algorithm with $\epsilon = 1 - \min_{p(\phi_i)>0} p(\phi_i)$.
- Because $p(\phi_i) \geq 2^{-k}$, the heuristic is a polynomial-time ϵ-approximation algorithm with $\epsilon = 1 - 2^{-k}$.

Back to MAXSAT

- In MAXSAT, the ϕ_i's are clauses.
- Hence $p(\phi_i) \geq 1/2$.
- The heuristic becomes a polynomial-time ϵ-approximation algorithm with $\epsilon = 1/2$.
- If the clauses have at least distinct k distinct literals, then $p(\phi_i) \geq 1 - 2^{-k}$.
- The heuristic becomes a polynomial-time ϵ-approximation algorithm with $\epsilon = 2^{-k}$.