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.
- The setup is very much like the Turing test.[a]

[a] Turing (1950).

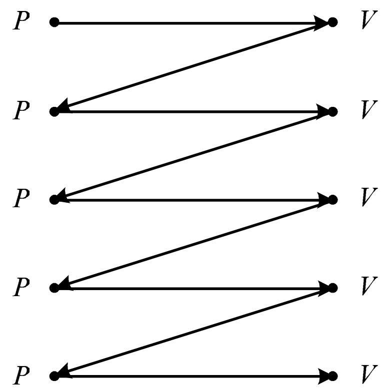
Interactive Proof Systems

- 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.
  - If the prover is not more powerful than the verifier, no interaction is needed.

Interactive Proof Systems (concluded)

- 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 of $|x|$.

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.[b]
- Similar things cannot be said of the soundness condition when $x \notin L$.
- Verifier's coin flips can be public.[c]

[a]Goldwasser, Micali, and Rackoff (1985).
[b]Goldreich, Mansour, and Sipser (1987).
[c]Goldwasser and Sipser (1989).

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 (see pp. 844ff for a proof).[a]

[a]Shamir (1990).

Graph Isomorphism

- $V_1 = V_2 = \{1, 2, \ldots, n\}$.
- Graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are **isomorphic** if there exists a permutation π on $\{1, 2, \ldots, n\}$ so that $(u, v) \in E_1 \Leftrightarrow (\pi(u), \pi(v)) \in E_2$.
- The task is to answer if $G_1 \cong G_2$ (**isomorphic**).
- No known polynomial-time algorithms.
- The problem is in NP (hence IP).
- But it is not likely to be NP-complete.[a]

[a]Schöning (1987).

GRAPH NONISOMORPHISM

- $V_1 = V_2 = \{1, 2, \ldots, n\}$.
- Graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are **nonisomorphic** if there exist no permutations π on $\{1, 2, \ldots, n\}$ so that $(u, v) \in E_1 \Leftrightarrow (\pi(u), \pi(v)) \in E_2$.
- The task is to answer if $G_1 \not\cong G_2$ (**nonisomorphic**).
- Again, no known polynomial-time algorithms.
  - It is in coNP, but how about NP or BPP?
  - It is not likely to be coNP-complete.
- Surprisingly, GRAPH NONISOMORPHISM $\in$ IP.[a]

[a]Goldreich, Micali, and Wigderson (1986).

A 2-Round Algorithm

```
Victor selects a random i ∈ { 1, 2 };
Victor selects a random permutation π on { 1, 2, . . . , n };
Victor applies π on graph G_i to obtain graph H;
Victor sends (G_1, H) to Peggy;
if G_1 ≅ 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 not isomorphic.
  - Peggy is able to tell which G_i is isomorphic to H.
  - So Victor always accepts.
- Suppose the two graphs are isomorphic.
  - No matter which i is picked by Victor, Peggy or any prover sees 2 identical graphs.
  - Peggy or any prover with exponential power has only probability one half of guessing i correctly.
  - So Victor erroneously accepts with probability 1/2.
- Repeat the algorithm to obtain the desired probabilities.

Knowledge in Proofs

- Suppose I know a satisfying assignment to a satisfiable boolean expression.
- I can convince Alice of this by giving her the assignment.
- But then I give her more knowledge than necessary.
  - Alice can claim that she found the assignment!
  - Login authentication faces essentially the same issue.
  - See `www.wired.com/wired/archive/1.05/atm_pr.html` for a famous ATM fraud in the U.S.

Knowledge in Proofs (concluded)

- Digital signatures authenticate *documents* but not *individuals*.
- They hence do not solve the problem.
- Suppose I always give Alice random bits.
- Alice's extracts no knowledge from me by any measure, but I prove nothing.
- Question 1: Can we design a protocol to convince Alice of (the knowledge of) a secret without revealing anything extra?
- Question 2: How to define this idea rigorously?

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 an 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.

[a] Goldwasser, Micali, and Rackoff (1985).

Comments

- 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.
  - A verifier cannot use the transcript of the interaction to convince a third-party of the validity of the claim.
  - The proof is hence not transferable.

Comments (continued)

- Whatever a verifier can "learn" from the specified prover P via the communication channel could as well be computed from the verifier alone.
- 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 by just believing that the claim is indeed valid.
- 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.

Comments (concluded)

- The "paradox" is resolved by noting that it is not the transcript 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.
- It is known that if one-way functions exist, then zero-knowledge proofs exist for every problem in NP.[a]

[a] Goldreich, Micali, and Wigderson (1986).

Will You Be Convinced?

- A newspaper commercial for hair-growing products for men.
  - A (for all practical purposes) bald man has a full head of hair after 3 months.
- A TV commercial for weight-loss products.
  - A (by any reasonable measure) overweight woman loses 10 kilograms in 10 weeks.

Quadratic Residuosity

- Let n be a product of two distinct primes.
- Assume extracting the square root of a quadratic residue modulo n is hard without knowing the factors.
- We next present a zero-knowledge proof for x being a quadratic residue.

Zero-Knowledge Proof of Quadratic Residuosity (continued)

1: **for** $m = 1, 2, \ldots, \log_2 n$ **do**
2:   Peggy chooses a random $v \in Z_n^*$ and sends $y = v^2 \bmod n$ to Victor;
3:   Victor chooses a random bit i and sends it to Peggy;
4:   Peggy sends $z = u^i v \bmod n$, where u is a square root of x; $\{u^2 \equiv x \bmod n.\}$
5:   Victor checks if $z^2 \equiv x^i y \bmod n$;
6: **end for**
7: Victor accepts x if Line 5 is confirmed every time;

Analysis

- Suppose x is a quadratic nonresidue.
  - Peggy can answer only one of the two possible challenges.
    - Reason: a is a quadratic residue if and only if xa is a quadratic nonresidue.
  - So Peggy will be caught in any given round with probability one half.

Analysis (continued)

- Suppose x is a quadratic residue.
  - Peggy can answer all challenges.
  - So Victor will accept x.
- How about the claim of zero knowledge?
- The transcript between Peggy and Victor when x is a quadratic residue can be generated without Peggy!
  - So interaction with Peggy is useless.

Analysis (continued)

- Here is how.
- Suppose x is a quadratic residue.[a]
- In each round of interaction with Peggy, the transcript is a triplet (y, i, z).
- We present an efficient algorithm Bob that generates (y, i, z) with the same probability *without* accessing Peggy.

[a] By definition, we do not need to consider the other case.

Analysis (concluded)

1: Bob chooses a random $z \in Z_n^*$;
2: Bob chooses a random bit i;
3: Bob calculates $y = z^2 x^{-i} \bmod n$;
4: Bob writes (y, i, z) into the transcript;

Comments

- Bob cheats because (y, i, z) is *not* generated in the same order as in the original transcript.
  - Bob picks Victor's challenge first.
  - Bob then picks Peggy's answer.
  - Bob finally patches the transcript.
  - So it is not the transcript that convinces Victor, but that conversation with Peggy is held "on line."
- The same holds even if the transcript was generated by a cheating Victor's interaction with (honest) Peggy, but we skip the details.

Zero-Knowledge Proof of 3 Colorability[a]

1: **for** $i = 1, 2, \ldots, |E|^2$ **do**
2: Peggy chooses a random permutation π of the 3-coloring ϕ;
3: Peggy samples an encryption scheme randomly and sends $\pi(\phi(1)), \pi(\phi(2)), \ldots, \pi(\phi(|V|))$ encrypted to Victor;
4: Victor chooses at random an edge $e \in E$ and sends it to Peggy for the coloring of the endpoints of e;
5: **if** $e = (u, v) \in E$ **then**
6: Peggy reveals the coloring of u and v and "proves" that they correspond to their encryption;
7: **else**
8: Peggy stops;
9: **end if**

[a]Goldreich, Micali, and Wigderson (1986).

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;

Analysis

- If the graph is 3-colorable and both Peggy and Victor follow the protocol, then Victor always accepts.
- If the graph is not 3-colorable and Victor follows the protocol, then however Peggy plays, Victor will accept with probability $\leq (1 - m^{-1})^{m^2} \leq e^{-m}$, where $m = |E|$.
- Thus the protocol is valid.
- This protocol yields no knowledge to Victor as all he gets is a bunch of random pairs.
- The proof that the protocol is zero-knowledge to *any* verifier is more intricate.

Approximability

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 NP problems that cannot be approximated well (assuming NP $\neq$ P)?
- Are there NP problems that cannot be approximated at all (assuming NP $\neq$ P)?

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 $\textsc{opt}(x) = \min_{s \in F(x)} c(s)$ for a minimization problem.
- It is $\textsc{opt}(x) = \max_{s \in F(x)} c(s)$ for a maximization problem.

Approximation Algorithms

- Let algorithm M on x returns a feasible solution.
- M is an **ϵ-approximation algorithm**, where $\epsilon \geq 0$, if for all x,
$$\frac{|c(M(x)) - \textsc{opt}(x)|}{\max(\textsc{opt}(x), c(M(x)))} \leq \epsilon.$$
  - For a minimization 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 minimization problem,
$$\min_{s \in F(x)} c(s) \leq c(M(x)) \leq \frac{\min_{s \in F(x)} c(s)}{1 - \epsilon}.$$
  - So **approximation ratio** $\frac{\min_{s \in F(x)} c(s)}{c(M(x))} \geq 1 - \epsilon$.
- For a maximization problem,
$$(1 - \epsilon) \times \max_{s \in F(x)} c(s) \leq c(M(x)) \leq \max_{s \in F(x)} c(s).$$
  - So approximation ratio $\frac{c(M(x))}{\max_{s \in F(x)} c(s)} \geq 1 - \epsilon$.
- The above are alternative definitions of ϵ-approximation algorithms.

Range Bounds

- ϵ takes values between 0 and 1.
- For maximization problems, an ϵ-approximation algorithm returns solutions within $[\,(1-\epsilon) \times \text{OPT}, \text{OPT}\,]$.
- For minimization problems, an ϵ-approximation algorithm returns solutions within $[\,\text{OPT}, \frac{\text{OPT}}{1-\epsilon}\,]$.
- 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 $\epsilon \geq 0$ 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.
- So we assume P $\neq$ NP for the rest of the discussion.

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 ratio $\frac{c(M(x))}{\text{OPT}(x)} = \Theta(\log n)$.
- It is not an ϵ-approximation algorithm for any $\epsilon < 1$.

A 0.5-Approximation Algorithm[a]

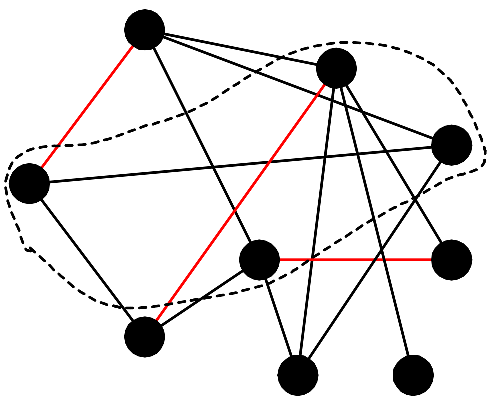
$C := \emptyset$;
while $E \neq \emptyset$ **do**
  Delete an arbitrary edge $[\,u, v\,]$ from E;
  Delete edges incident with u and v from E;
  Add u and v to C; {Add 2 nodes to C each time.}
end while
return C;

[a] Johnson (1974).

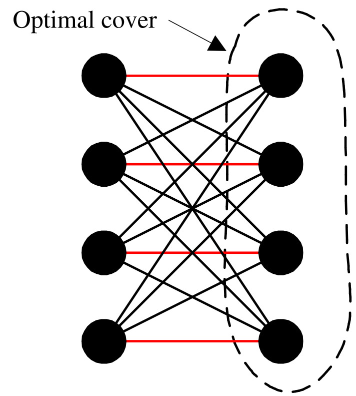
Analysis

- 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
$$\frac{\text{OPT}(G)}{|C|} \geq 1/2.$$
- The approximation threshold is ≤ 0.5.
- We remark that 0.5 is also the lower bound for any "greedy" algorithms.[a]

[a] Davis and Impagliazzo (2004).



The 0.5 Bound Is Tight for the Algorithm[a]



[a] Contributed by Mr. Jenq-Chung Li (R92922087) on December 20, 2003.

Maximum Satisfiability

- Given a set of clauses, MAXSAT seeks the truth assignment that satisfies the most.
- MAX2SAT is already NP-complete (p. 267).
- Consider the more general k-MAXGSAT for constant k.
  - 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 exactly 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 $p(\phi_i) = t_i/2^k$.
  - $p(\phi_i)$ is easy to calculate for a $k = O(\log n)$.
- 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 $t_1 \in \{\texttt{true}, \texttt{false}\}$ such that $p(\Phi[\,x_1 = t_1\,])$ is the larger one.
- Note that $p(\Phi[\,x_1 = t_1\,]) \geq p(\Phi)$.
- Repeat with expression $\Phi[\,x_1 = t_1\,]$ until all variables x_i have been given truth values t_i and all ϕ_i either true or false.

The Search Procedure (concluded)

- By our hill-climbing procedure,

$$\begin{aligned} & p(\Phi[\,x_1 = t_1, x_2 = t_2, \ldots, x_n = t_n\,]) \\ \geq\ & \cdots \\ \geq\ & p(\Phi[\,x_1 = t_1, x_2 = t_2\,]) \\ \geq\ & p(\Phi[\,x_1 = t_1\,]) \\ \geq\ & p(\Phi). \end{aligned}$$

- So at least $p(\Phi)$ expressions are satisfied by truth assignment $(t_1, t_2, \ldots, t_n)$.
- The algorithm is deterministic.

Approximation Analysis

- 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

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

- 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 $p(\phi_i) \geq 1/2$, which happens when ϕ_i contains a single literal.
- And the heuristic becomes a polynomial-time ϵ-approximation algorithm with $\epsilon = 1/2$.[a]
- If the clauses have k distinct literals, $p(\phi_i) = 1 - 2^{-k}$.
- And the heuristic becomes a polynomial-time ϵ-approximation algorithm with $\epsilon = 2^{-k}$.
  - This is the best possible for $k \geq 3$ unless P = NP.

[a] Johnson (1974).

MAX CUT Revisited

- The NP-complete MAX CUT seeks to partition the nodes of graph $G = (V, E)$ into $(S, V - S)$ so that there are as many edges as possible between S and $V - S$ (p. 289).
- **Local search** starts from a feasible solution and performs "local" improvements until none are possible.

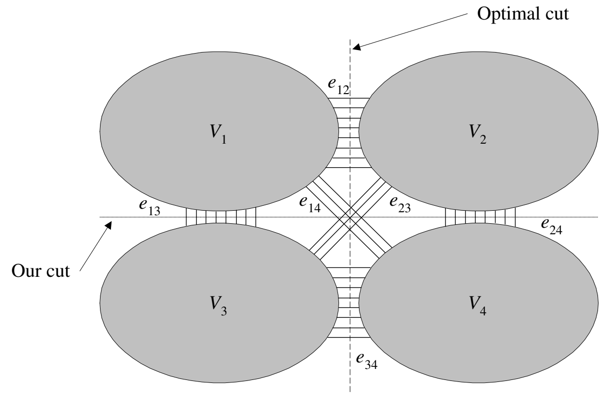
A 0.5-Approximation Algorithm for MAX CUT

$S := \emptyset$;
while $\exists v \in V$ whose switching sides results in a larger cut **do**
  $S := S \cup \{v\}$;
end while
return S;

- A 0.12-approximation algorithm exists.[a]
- 0.059-approximation algorithms do not exist unless NP = ZPP.

[a] Goemans and Williamson (1995).

Analysis



Analysis (continued)

- Partition $V = V_1 \cup V_2 \cup V_3 \cup V_4$, where our algorithm returns $(V_1 \cup V_2, V_3 \cup V_4)$ and the optimum cut is $(V_1 \cup V_3, V_2 \cup V_4)$.
- Let e_{ij} be the number of edges between V_i and V_j.
- Because no migration of nodes can improve the algorithm's cut, for each node in V_1, its edges to $V_1 \cup V_2$ are outnumbered by those to $V_3 \cup V_4$.
- Considering all nodes in V_1 together, we have $2e_{11} + e_{12} \leq e_{13} + e_{14}$, which implies

$$e_{12} \leq e_{13} + e_{14}.$$

Analysis (concluded)

- Similarly,

$$\begin{aligned} e_{12} &\leq e_{23} + e_{24} \\ e_{34} &\leq e_{23} + e_{13} \\ e_{34} &\leq e_{14} + e_{24} \end{aligned}$$

- Adding all four inequalities, dividing both sides by 2, and adding the inequality $e_{14} + e_{23} \leq e_{14} + e_{23} + e_{13} + e_{24}$, we obtain

$$e_{12} + e_{34} + e_{14} + e_{23} \leq 2(e_{13} + e_{14} + e_{23} + e_{24}).$$

- The above says our solution is at least half the optimum.