## 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)

- Suppose I always give Alice random bits.
- Alice extracts no knowledge from me by any measure, but I prove nothing.
- Question 1: Can we design a protocol to convince Alice (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 "$x \in L$."
- 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 (continued)

- 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.
  - $M$ only needs to generate a distribution that is computationally indistinguishable from the verifier's view of the interaction.

## Comments (concluded)

- It is known that if one-way functions exist, then zero-knowledge proofs exist for every problem in NP.[a]
- The verifier can be restricted to the honest one (i.e., it follows the protocol).[b]
- The coins can be public.[c]

---

[a] Goldreich, Micali, and Wigderson (1986).
[b] Vadhan (2006).
[c] Vadhan (2006).

## Are You 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 Residuacity

- 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 $x \in Z_n^*$ being a quadratic residue.

## Zero-Knowledge Proof of Quadratic Residuacity

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;

## A Useful Corollary

**Corollary 76** *Let $n = pq$ be a product of two distinct primes. (1) If $x$ and $y$ are both quadratic residues modulo $n$, then $xy \in Z_n^*$ is a quadratic residue modulo $n$. (2) If $x$ is a quadratic residue modulo $n$ and $y$ is a quadratic nonresidue modulo $n$, then $xy \in Z_n^*$ is a quadratic nonresidue modulo $n$.*

- Suppose $x$ and $y$ are both quadratic residues modulo $n$.
- Let $x \equiv a^2 \bmod n$ and $y \equiv b^2 \bmod n$.
- Now $xy$ is a quadratic residue as $xy \equiv (ab)^2 \bmod n$.

## The Proof (concluded)

- Suppose $x$ is a quadratic residue modulo $n$ and $y$ is a quadratic nonresidue modulo $n$.
- By Lemma 75 (p. 596), $(x \,|\, p) = (x \,|\, q) = 1$ but, say, $(y \,|\, p) = -1$.
- Now $xy$ is a quadratic nonresidue as $(xy \,|\, p) = -1$, again by Lemma 75 (p. 596).

## Analysis

- Suppose $x$ is a quadratic nonresidue.
  - Peggy can answer only one of the two possible challenges.
    - If $a$ is a quadratic residue, then $xa$ is a quadratic nonresidue by Corollary 76 (p. 623).
    - So $x^i y$ can be a quadratic residue (see Line 5) only when $i = 0$.
  - 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.
- Here is how.

## Analysis (continued)

- 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 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

- Assume $x$ is a quadratic residue.
- In both cases, for $(y, i, z)$, $y$ is a random quadratic residue, $i$ is a random bit, and $z$ is a random number.
- 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.

## Comments (concluded)

- 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.

## Does the Following Work, Too?[a]

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: Peggy sends $z = uv \bmod n$, where $u$ is a square root of $x$; $\{u^2 \equiv x \bmod n.\}$
4: Victor checks if $z^2 \equiv xy \bmod n$;
5: **end for**
6: Victor accepts $x$ if Line 4 is confirmed every time;

[a]Contributed by Mr. Chih-Duo Hong (R95922079) on December 13, 2006. It is like always choosing $i = 1$ in the original protocol.

## Does the Following Work, Too?[a] (concluded)

- Suppose $x$ is a quadratic nonresidue.
- But Peggy can mislead Victor into accepting $x$ as a quadratic residue.
- She simply sends $y = x$ and $z = x$ to Victor.
- This pair will satisfy $z^2 \equiv xy \bmod n$ by construction.
- The protocol is hence not even an IP protocol!

[a] Contributed by Mr. Chin-Luei Chang (D95922007) on June 16, 2008.

# Zero-Knowledge Proof of 3 Colorability[a]

1: **for** $i = 1, 2, \ldots, |E|^2$ **do**
2: Peggy chooses a random permutation $\pi$ of the 3-coloring $\phi$;
3: Peggy samples encryption schemes randomly, commits[b] them, 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 encryptions;
7: **else**
8: Peggy stops;
9: **end if**

---

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

[b] Contributed by Mr. Ren-Shuo Liu (D98922016) on December 22, 2009.

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 intricate.

## Comments

- Each $\pi(\phi(i))$ is encrypted by a different cryptosystem.[a]
  - Otherwise, all the colors will be revealed in Step 6.
- Each edge $e$ must be picked randomly.[b]
  - Otherwise, Peggy will know Victor's game plan and plot accordingly.

[a] Contributed by Ms. Yui-Huei Chang (R96922060) on May 22, 2008

[b] Contributed by Mr. Chang-Rong Hung (R96922028) on May 22, 2008

*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 $c(s) \in \mathbb{Z}^+$.
  - Here, cost refers to the quality of the feasible solution, not the time required to obtain it.
  - It is our **objective function**, e.g., total distance, satisfaction, or cut size.
- 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 **$\epsilon$-approximation algorithm**, where $\epsilon \geq 0$, if for all $x$,

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

  - For a minimization problem,

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

  - For a maximization problem,

$$\frac{\max_{s \in F(x)} c(s) - c(M(x))}{\max_{s \in F(x)} c(s)} \leq \epsilon. \tag{10}$$

## 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,

$$(1 - \epsilon) \times \max_{s \in F(x)} c(s) \leq c(M(x)) \leq \max_{s \in F(x)} c(s). \quad (11)$$

  - So approximation ratio $\frac{c(M(x))}{\max_{s \in F(x)} c(s)} \geq 1 - \epsilon$.

- They are alternative definitions of $\epsilon$-approximation.

## Range Bounds

- $\epsilon$ takes values between 0 and 1.
- For maximization problems, an $\epsilon$-approximation algorithm returns solutions within $[\,(1-\epsilon) \times \textsc{opt}, \textsc{opt}\,]$.
- For minimization problems, an $\epsilon$-approximation algorithm returns solutions within $[\,\textsc{opt}, \frac{\textsc{opt}}{1-\epsilon}\,]$.
- For each NP-complete optimization problem, we shall be interested in determining the *smallest* $\epsilon$ for which there is a polynomial-time $\epsilon$-approximation algorithm.
- Sometimes $\epsilon$ 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 $\epsilon$-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 an approximation threshold of 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

$$\frac{c(M(x))}{\text{OPT}(x)} = \Theta(\log n).$$

- Hence the approximation ratio is $\Theta(\log^{-1} n)$.
- It is not an $\epsilon$-approximation algorithm for any $\epsilon < 1$.

## A 0.5-Approximation Algorithm[a]

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

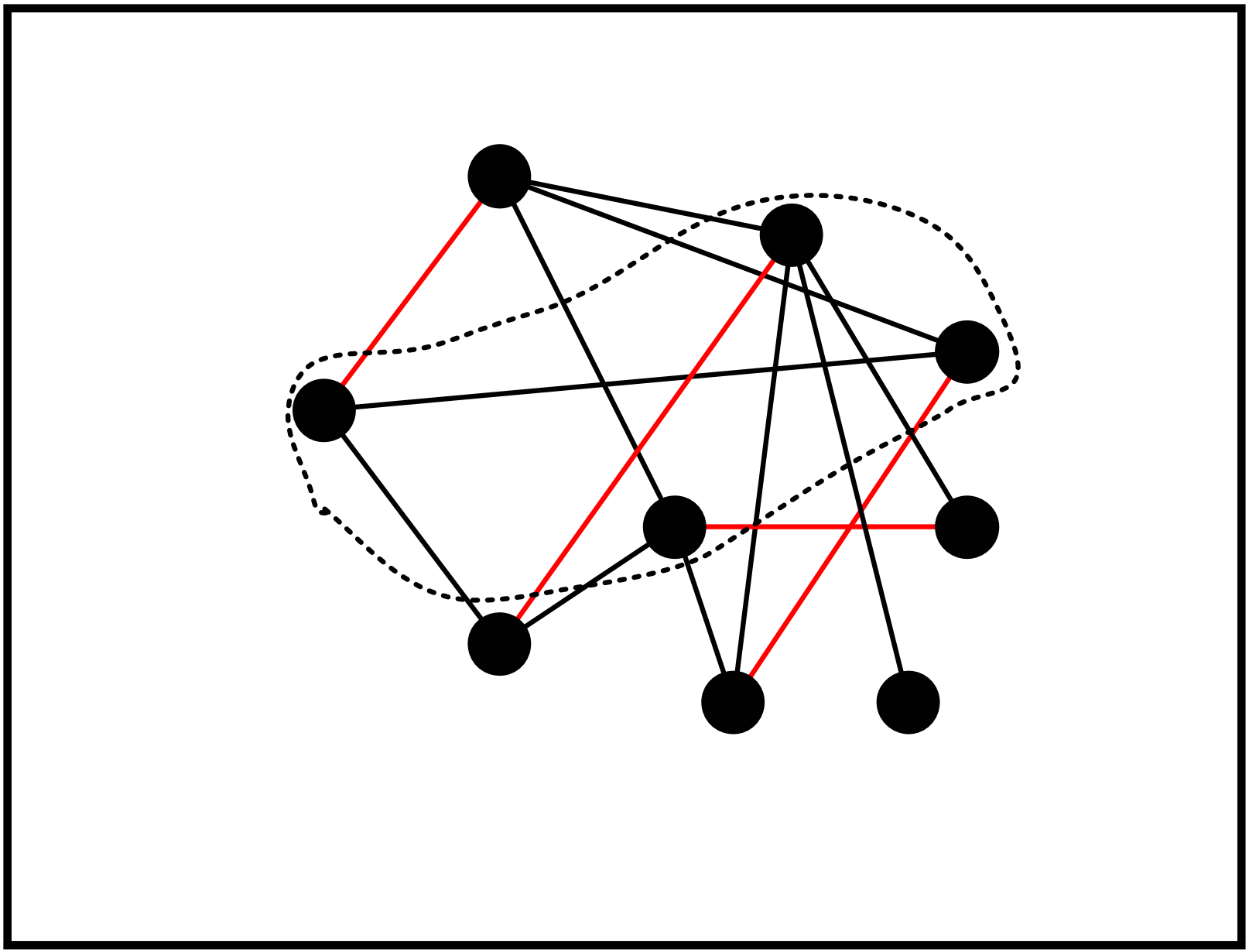
[a] Johnson (1974).

## Analysis

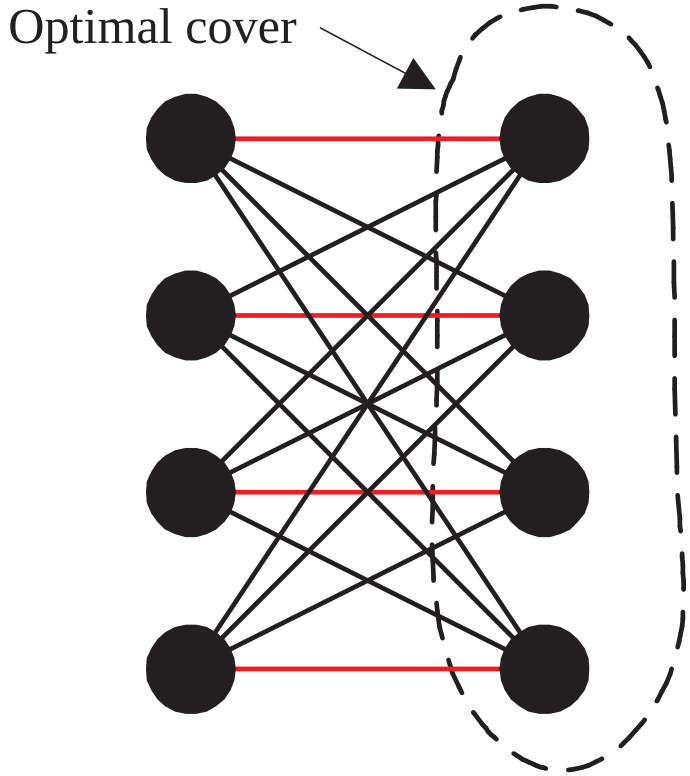
- $C$ contains $|C|/2$ edges.
- No two edges of $C$ share a node.[a]
- *Any* node cover must contain at least one node from each of these edges.
- This means that $\textsc{opt}(G) \geq |C|/2$.
- So
$$\frac{\textsc{opt}(G)}{|C|} \geq 1/2.$$
- The approximation threshold is $\leq 0.5$.[b]

[a] In fact, $C$ is a *maximal* matching.

[b] 0.5 is also the lower bound for any "greedy" algorithms (see 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. 290).
- 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 $\phi_i$ 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 $\phi_i$ involves exactly $k$ variables and is satisfied by $s_i$ of the $2^k$ truth assignments.
- A random truth assignment $\in \{0,1\}^n$ satisfies $\phi_i$ with probability $p(\phi_i) = s_i/2^k$.
  - $p(\phi_i)$ is easy to calculate as $k$ is a constant.
- Hence a random truth assignment satisfies an expected number

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

of expressions $\phi_i$.

## 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 $\phi_i$ either true or false.

## The Search Procedure (concluded)

- By our hill-climbing procedure,

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

- 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 $\phi_i$—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 $\epsilon$-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 $\epsilon$-approximation algorithm with $\epsilon = 1 - 2^{-k}$.