Zero-Knowledge Proof of 3 Colorability[a]

for $i = 1, 2, \ldots, |E|^2$ **do**
    Peggy chooses a random permutation π of the 3-coloring ϕ;
    Peggy samples encryption schemes randomly, commits[b] them, and sends $\pi(\phi(1)), \pi(\phi(2)), \ldots, \pi(\phi(|V|))$ encrypted 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 encryptions;
    else
        Peggy stops;
    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 π(φ(u)) = π(φ(v)) or π(φ(u)), π(φ(v)) ∉ {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 $\text{OPT}(x) = \min_{s \in F(x)} c(s)$ 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 $\epsilon \geq 0$, 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,

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

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,

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

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

- They are alternative definitions of ϵ-approximation.

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 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 ϵ-approximation algorithm for any $\epsilon < 1$.

A 0.5-Approximation Algorithm[a]

```
C := ∅;
while E ≠ ∅ 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;
```

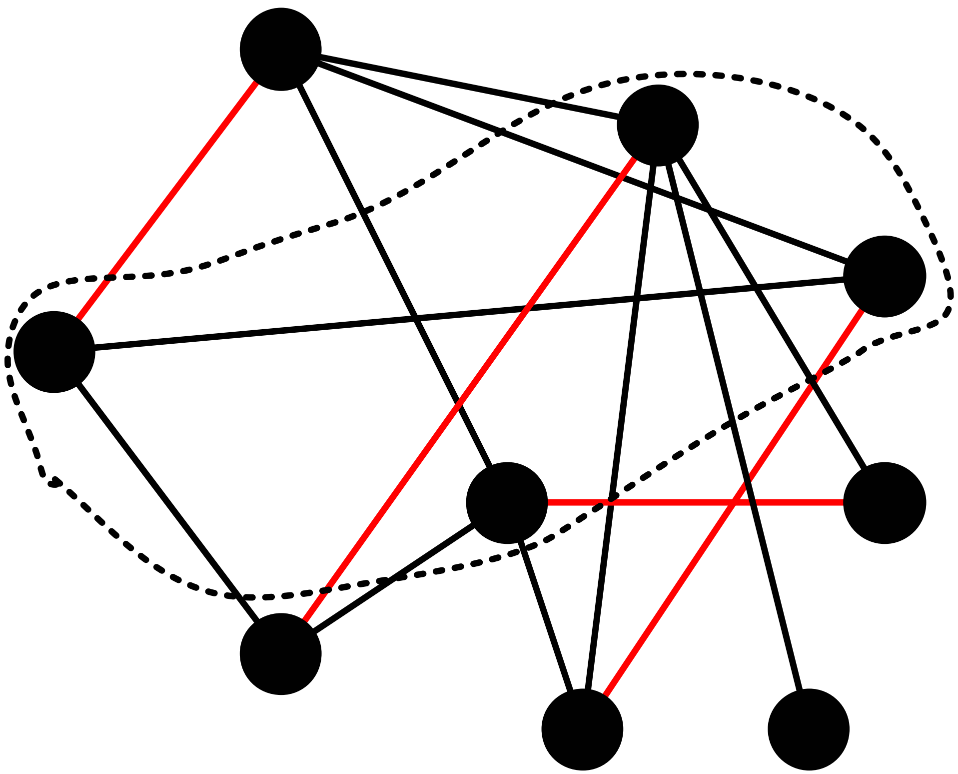
[a] Johnson (1974).

Analysis

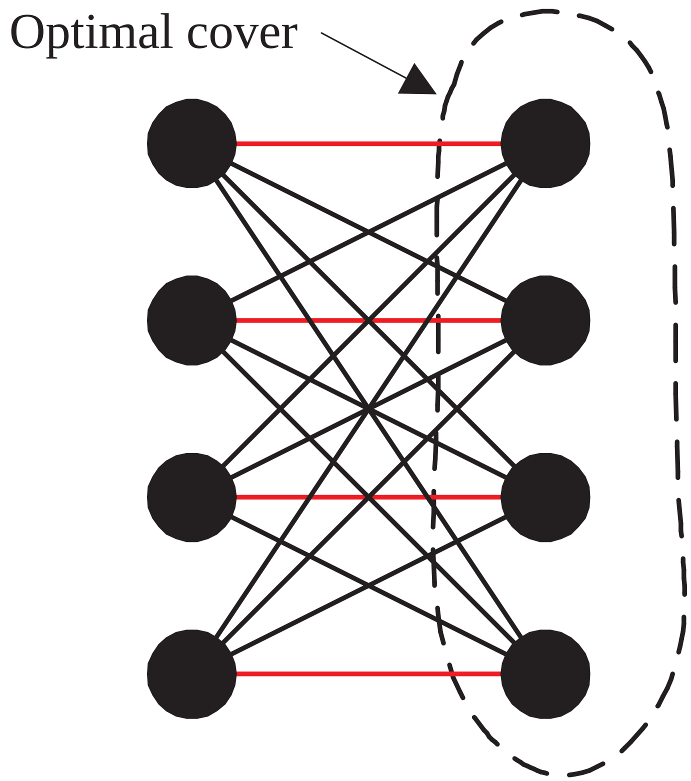
- C contains $|C|/2$ edges.
- No two edges of C share a node.[a]
- *Any* node cover must contain at least one node from each of these edges.
- This means that $\text{OPT}(G) \geq |C|/2$.
- So
$$\frac{\text{OPT}(G)}{|C|} \geq 1/2.$$
- The approximation threshold is ≤ 0.5.[b]

[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. 287).
- 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 s_i of the 2^k truth assignments.
- A random truth assignment $\in \{0, 1\}^n$ satisfies ϕ_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 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,

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

- 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 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. 315).
- **Local search** starts from a feasible solution and performs "local" improvements until none are possible.
- Next we present a local search algorithm for MAX CUT.

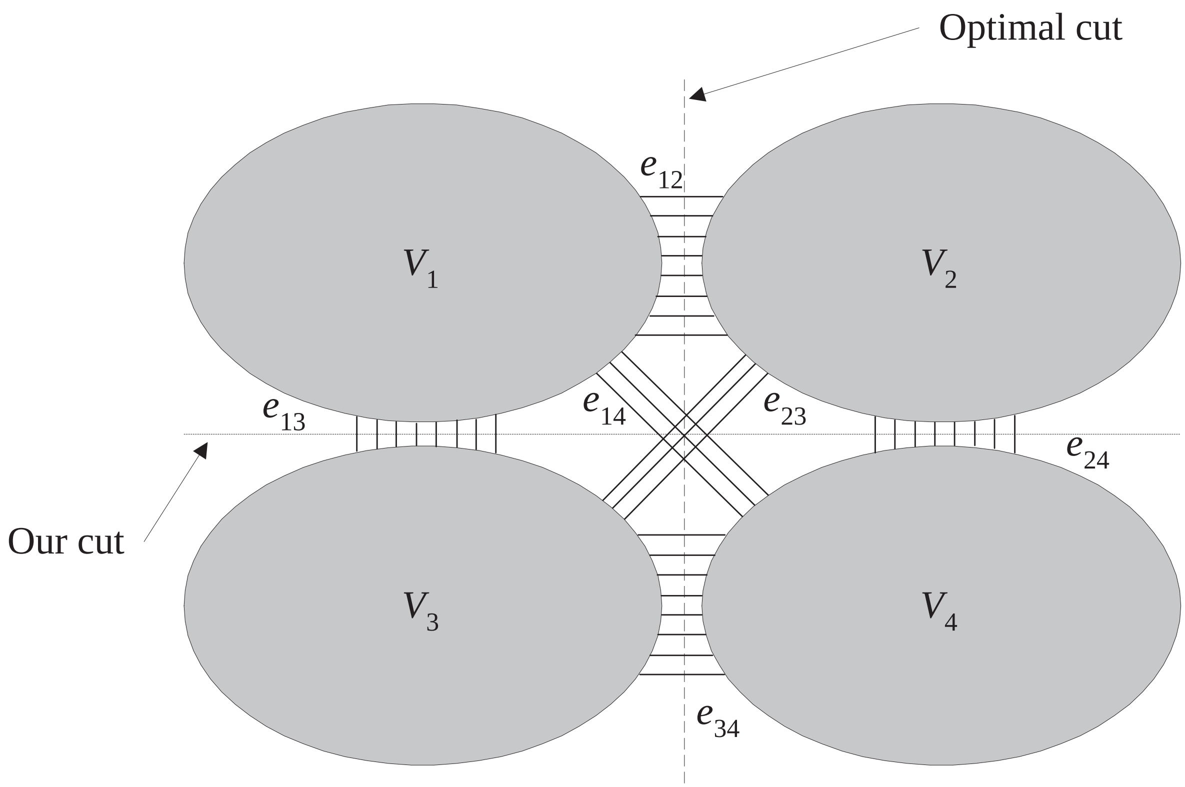
A 0.5-Approximation Algorithm for MAX CUT

```
S := ∅;
while ∃v ∈ V whose switching sides results in a larger cut do
    Switch the side of 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 $(V_1 \cup V_2, V_3 \cup V_4)$.
  - 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.
- For each node $v \in V_1$, its edges to $V_1 \cup V_2$ are outnumbered by those to $V_3 \cup V_4$.
  - Otherwise, v would have been moved to $V_3 \cup V_4$ to improve the cut.

Analysis (continued)

- Considering all nodes in V_1 together, we have $2e_{11} + e_{12} \leq e_{13} + e_{14}$
  - It is $2e_{11}$ is because each edge in V_1 is counted twice.
- The above inequality implies

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

Analysis (concluded)

- Similarly,

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

- Add all four inequalities, divide both sides by 2, and add the inequality $e_{14} + e_{23} \leq e_{14} + e_{23} + e_{13} + e_{24}$ to 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.

Approximability, Unapproximability, and Between

- KNAPSACK, NODE COVER, MAXSAT, and MAX CUT have approximation thresholds less than 1.
  - KNAPSACK has a threshold of 0 (see p. 663).
  - But NODE COVER and MAXSAT have a threshold larger than 0.
- The situation is maximally pessimistic for TSP: It cannot be approximated unless P = NP (see p. 661).
  - The approximation threshold of TSP is 1.
    - The threshold is 1/3 if the TSP satisfies the triangular inequality.
  - The same holds for INDEPENDENT SET.