Zero-Knowledge Proof of 3 Colorability

1: for $i = 1, 2, \ldots, |E|$ 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 colors $\pi(\phi(u))$ and $\pi(\phi(v))$ 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 $\pi(\phi(u)), \pi(\phi(v)) \not\in \{1, 2, 3\}$ 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.
- Suppose the graph is not 3-colorable and Victor follows the protocol.
- Let $e$ be an edge that is not colored legally.
- Victor will pick it with probability $1/m$, where $m = |E|$. 
- Then however Peggy plays, Victor will accept with probability $\leq 1 - (1/m)$ per round.
Analysis (concluded)

- So Victor 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 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 in Line 3.$^a$
  - Otherwise, all the colors will be revealed in Line 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
All science is dominated by the idea of approximation.

— Bertrand Russell (1872–1970)
Just because the problem is NP-complete does not mean that you should not try to solve it.

— Stephen Cook (2002)
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, number of satisfied expressions, or cut size.
Some Definitions (concluded)

- 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 (polynomial-time) 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,
    \[
    \frac{\max_{s \in F(x)} c(s) - c(M(x))}{\max_{s \in F(x)} c(s)} \leq \epsilon. \tag{17}
    \]
Lower and Upper Bounds

- For a minimization problem,
  \[
  \min_{s \in F(x)} c(s) \leq c(M(x)) \leq \frac{\min_{s \in F(x)} c(s)}{1 - \epsilon}.
  \]

- For a maximization problem,
  \[
  (1 - \epsilon) \times \max_{s \in F(x)} c(s) \leq c(M(x)) \leq \max_{s \in F(x)} c(s). \tag{18}
  \]
Range Bounds

- $\epsilon$ ranges between 0 (best) and 1 (worst).

- For minimization problems, an $\epsilon$-approximation algorithm returns solutions within
  \[
  \left[ \text{OPT}, \frac{\text{OPT}}{1 - \epsilon} \right].
  \]

- For maximization problems, an $\epsilon$-approximation algorithm returns solutions within
  \[
  \left[ (1 - \epsilon) \times \text{OPT}, \text{OPT} \right].
  \]
Approximation Thresholds

- 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.

- But sometimes $\epsilon$ has no minimum value.

- The **approximation threshold** is the greatest lower bound of all $\epsilon \geq 0$ such that there is a polynomial-time $\epsilon$-approximation algorithm.

- By a standard theorem in real analysis, such a threshold must exist.\(^a\)

\(^a\)Bauldry (2009).
Approximation Thresholds (concluded)

- 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.
Approximation Ratio

- $\epsilon$-approximation algorithms can also be defined via approximation ratio:

$$\frac{c(M(x))}{\text{OPT}(x)}.$$

- For a minimization problem, the approximation ratio is

$$1 \leq \frac{c(M(x))}{\min_{s \in F(x)} c(s)} \leq \frac{1}{1 - \epsilon}. \quad (19)$$

- For a maximization problem, the approximation ratio is

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

\(^{a}\text{Williamson and Shmoys (2011).}\)
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 an approximation ratio of

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

- So it is not an $\epsilon$-approximation algorithm for any constant $\epsilon < 1$ according to Eq. (19).

\(^\text{a}\)Chvátal (1979).
A 0.5-Approximation Algorithm\textsuperscript{a}

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

\textsuperscript{a}Johnson (1974).
Analysis

- It is easy to see that $C$ is a node cover.
- $C$ contains $|C|/2$ edges.$^a$
- No two edges of $C$ share a node.$^b$
- Any node cover must contain at least one node from each of these edges.
  - If there is an edge in $C$ both of whose ends are outside the cover, then that cover will not be a valid cover.

---

$^a$The edges deleted in Line 3.

$^b$In fact, $C$ as a set of edges is a maximal matching.
Analysis (concluded)

- This means that $\text{OPT}(G) \geq |C|/2$.
- So the approximation ratio

\[ \frac{|C|}{\text{OPT}(G)} \leq 2. \]

- So we have a 0.5-approximation algorithm.
- The approximation threshold is therefore $\leq 0.5$. 
The 0.5 Bound Is Tight for the Algorithm\textsuperscript{a}

Optimal cover

\textsuperscript{a}Contributed by Mr. Jenq-Chung Li (R92922087) on December 20, 2003. Recall that König’s theorem says the size of a maximum matching equals that of a minimum node cover in a bipartite graph.
Remarks

• The approximation threshold is at least\textsuperscript{a}

\[ 1 - \left(10\sqrt{5} - 21\right)^{-1} \approx 0.2651. \]

• The approximation threshold is 0.5 if one assumes the unique games conjecture.\textsuperscript{b}

• This ratio 0.5 is also the lower bound for any “greedy” algorithms.\textsuperscript{c}

\textsuperscript{a}Dinur and Safra (2002).
\textsuperscript{b}Khot and Regev (2008).
\textsuperscript{c}Davis and Impagliazzo (2004).
Maximum Satisfiability

- Given a set of clauses, MAXSAT seeks the truth assignment that satisfies the most.

- MAX2SAT is already NP-complete (p. 345), so MAXSAT is NP-complete.

- Consider the more general $k$-MAXGSAT for constant $k$.
  - Let $\Phi = \{\phi_1, \phi_2, \ldots, \phi_m\}$ be a set of boolean expressions 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 average of

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

expressions $\phi_i$. 
The Search Procedure

- Clearly

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

- Select the \( t_1 \in \{\text{true, false}\} \) such that \( p(\Phi[x_1 = t_1]) \) is the larger one.

- Note that \( p(\Phi[x_1 = t_1]) \geq p(\Phi) \).

- Repeat the procedure with expression \( \Phi[x_1 = t_1] \) until all variables \( x_i \) have been given truth values \( t_i \) and all \( \phi_i \) are either true or false.
The Search Procedure (continued)

- By our hill-climbing procedure,

\[ p(\Phi) \]
\[ \leq p(\Phi[x_1 = t_1]) \]
\[ \leq p(\Phi[x_1 = t_1, x_2 = t_2]) \]
\[ \leq \ldots \]
\[ \leq p(\Phi[x_1 = t_1, x_2 = t_2, \ldots, x_n = t_n]). \]

- So at least \( p(\Phi) \) expressions are satisfied by truth assignment \( (t_1, t_2, \ldots, t_n) \).
The Search Procedure (concluded)

- Note that the algorithm is *deterministic*!
- It is called the **method of conditional expectations**.\(^a\)

\(^a\)Erdős and Selfridge (1973); Spencer (1987).
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\(^a\)

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

- So this 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 $\epsilon = 1 - 2^{-k}$.

\(^a\)Recall that $(\sum_i a_i)/(\sum_i b_i) \geq \min_i a_i/b_i$. 

©2014 Prof. Yuh-Dauh Lyuu, National Taiwan University
In MAXSAT, the $\phi_i$’s are clauses (like $x \lor y \lor \neg z$).

Hence $p(\phi_i) \geq 1/2$, which happens when $\phi_i$ contains a single literal.

And the heuristic becomes a polynomial-time $\epsilon$-approximation algorithm with $\epsilon = 1/2$\textsuperscript{a}.

- Suppose we set each boolean variable to true with probability $(\sqrt{5} - 1)/2$, the golden ratio.
- Then follow through the method of conditional expectations to derandomize it.
- We will obtain a $[(3 - \sqrt{5})]/2$-approximation algorithm, where $[(3 - \sqrt{5})]/2 \approx 0.382$.\textsuperscript{b}

\textsuperscript{a}Johnson (1974).
\textsuperscript{b}Lieberherr and Specker (1981).
Back to MAXSAT (concluded)

- If the clauses have $k$ distinct literals,
  
  $$p(\phi_i) = 1 - 2^{-k}.$$ 

- And the heuristic becomes a polynomial-time $\epsilon$-approximation algorithm with $\epsilon = 2^{-k}$.
  
  - This is the best possible for $k \geq 3$ unless P = NP.
MAX CUT Revisited

- 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 $V - S$.

- It is NP-complete.\textsuperscript{a}

- 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.

\textsuperscript{a}Recall p. 375.
A 0.5-Approximation Algorithm for MAX CUT

1: \( S := \emptyset; \)
2: \[ \text{while} \ \exists v \in V \text{ whose switching sides results in a larger cut do} \]
3: \[ \text{Switch the side of } v; \]
4: \[ \text{end while} \]
5: \[ \text{return } S; \]

- A 0.12-approximation algorithm exists.\(^a\)
- 0.059-approximation algorithms do not exist unless \( \text{NP} = \text{ZPP}. \)

\(^a\)Goemans and Williamson (1995).
Analysis

Optimal cut

Our cut

\[ V_1 \quad e_{12} \quad e_{13} \quad V_2 \]

\[ e_{14} \quad e_{23} \quad e_{24} \]

\[ V_3 \quad e_{34} \quad V_4 \]
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 $V_j$.
- Our algorithm returns a cut of size
  $$e_{13} + e_{14} + e_{23} + e_{24}.$$
- The optimum cut size is
  $$e_{12} + e_{34} + e_{14} + e_{23}.$$
Analysis (continued)

- 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.

- 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,

\[ e_{12} \leq e_{23} + e_{24} \]
\[ e_{34} \leq e_{23} + e_{13} \]
\[ e_{34} \leq e_{14} + e_{24} \]

- 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 (p. 736).
  - But NODE COVER (p. 714) and MAXSAT have a threshold larger than 0.

- The situation is maximally pessimistic for TSP, which cannot be approximated (p. 734).
  - The approximation threshold of TSP is 1.
    * The threshold is 1/3 if TSP satisfies the triangular inequality.
  - The same holds for INDEPENDENT SET (see the textbook).
Unapproximability of TSP\textsuperscript{a}

**Theorem 78** *The approximation threshold of TSP is 1 unless $P = NP$.*

- Suppose there is a polynomial-time $\epsilon$-approximation algorithm for TSP for some $\epsilon < 1$.
- We shall construct a polynomial-time algorithm to solve the NP-complete HAMILTONIAN CYCLE.
- Given any graph $G = (V, E)$, construct a TSP with $|V|$ cities with distances

\[
d_{ij} = \begin{cases} 
1, & \text{if } \{i, j\} \in E \\
\frac{|V|}{1-\epsilon}, & \text{otherwise}
\end{cases}
\]

\textsuperscript{a}Sahni and Gonzales (1976).
The Proof (concluded)

- Run the alleged approximation algorithm on this TSP.
- Suppose a tour of cost $|V|$ is returned.
  - This tour must be a Hamiltonian cycle.
- Suppose a tour that includes an edge of length $\frac{|V|}{1-\epsilon}$ is returned.
  - The total length of this tour is $> \frac{|V|}{1-\epsilon}$.
  - Because the algorithm is $\epsilon$-approximate, the optimum is at least $1 - \epsilon$ times the returned tour’s length.
  - The optimum tour has a cost exceeding $|V|$.
  - Hence $G$ has no Hamiltonian cycles.
KNAPSACK Has an Approximation Threshold of Zero\textsuperscript{a}

**Theorem 79** For any $\epsilon$, there is a polynomial-time $\epsilon$-approximation algorithm for KNAPSACK.

- We have $n$ weights $w_1, w_2, \ldots, w_n \in \mathbb{Z}^+$, a weight limit $W$, and $n$ values $v_1, v_2, \ldots, v_n \in \mathbb{Z}^+$.\textsuperscript{b}

- We must find an $S \subseteq \{1, 2, \ldots, n\}$ such that $\sum_{i \in S} w_i \leq W$ and $\sum_{i \in S} v_i$ is the largest possible.

\textsuperscript{a}Ibarra and Kim (1975).

\textsuperscript{b}If the values are fractional, the result is slightly messier, but the main conclusion remains correct. Contributed by Mr. Jr-Ben Tian (B89902011, R93922045) on December 29, 2004.
The Proof (continued)

- Let
  \[ V = \max\{v_1, v_2, \ldots, v_n\}. \]

- Clearly, \( \sum_{i \in S} v_i \leq nV. \)

- Let \( 0 \leq i \leq n \) and \( 0 \leq v \leq nV. \)

- \( W(i, v) \) is the minimum weight attainable by selecting only from the first \( i \) items and with a total value of \( v. \)
  - It is an \((n + 1) \times (nV + 1)\) table.

- Set \( W(0, v) = \infty \) for \( v \in \{1, 2, \ldots, nV\} \) and \( W(i, 0) = 0 \) for \( i = 0, 1, \ldots, n. \)^a

---

^aContributed by Mr. Ren-Shuo Liu (D98922016) and Mr. Yen-Wei Wu (D98922013) on December 28, 2009.
The Proof (continued)

- Then, for $0 \leq i < n$,
  \[
  W(i + 1, v) = \min\{W(i, v), W(i, v - v_{i+1}) + w_{i+1}\}.
  \]

- Finally, pick the largest $v$ such that $W(n, v) \leq W$.\(^a\)

- The running time is $O(n^2V)$, not polynomial time.

- Key idea: Limit the number of precision bits.

\(^a\)Lawler (1979).
The Proof (continued)

- Define

\[ v'_i = 2^b \left\lfloor \frac{v_i}{2^b} \right\rfloor. \]

- This is equivalent to zeroing each \( v_i \)'s last \( b \) bits.

- Call the original instance

\[ x = (w_1, \ldots, w_n, W, v_1, \ldots, v_n). \]

- Call the approximate instance

\[ x' = (w_1, \ldots, w_n, W, v'_1, \ldots, v'_n). \]
The Proof (continued)

- Solving $x'$ takes time $O(n^2V/2^b)$.
  - The algorithm only performs subtractions on the $v_i$-related values.
  - So the $b$ last bits can be removed from the calculations.
  - That is, use $v_i'' = \lfloor \frac{v_i}{2^b} \rfloor$ and $V = \max(v_1'', v_2'', \ldots, v_n'')$ in the calculations.
  - Then multiply the returned value by $2^b$.
  - It is an $(n + 1) \times (nV + 1)/2^b$ table.
The Proof (continued)

- The solution $S'$ is close to the optimum solution $S$:
  \[
  \sum_{i \in S'} v_i \geq \sum_{i \in S'} v'_i \geq \sum_{i \in S} v'_i \geq \sum_{i \in S} (v_i - 2^b) \geq \sum_{i \in S} v_i - n2^b.
  \]

- Hence
  \[
  \sum_{i \in S'} v_i \geq \sum_{i \in S} v_i - n2^b.
  \]

- Without loss of generality, assume $w_i \leq W$ for all $i$.
  - Otherwise, item $i$ is redundant.

- $V$ is a lower bound on $\text{OPT}$.
  - Picking an item with value $V$ is a legitimate choice.
The Proof (concluded)

- The relative error from the optimum is:

\[
\frac{\sum_{i \in S} v_i - \sum_{i \in S'} v_i}{\sum_{i \in S} v_i} \leq \frac{\sum_{i \in S} v_i - \sum_{i \in S'} v_i}{V} \leq \frac{n 2^b}{V}.
\]

- Suppose we pick \( b = \lceil \log_2 \frac{\epsilon V}{n} \rceil \).

- The algorithm becomes \( \epsilon \)-approximate.\(^a\)

- The running time is then \( O(n^2 V/2^b) = O(n^3/\epsilon) \), a polynomial in \( n \) and \( 1/\epsilon \).\(^b\)

---

\(^a\)See Eq. (17) on p. 706.

\(^b\)It hence depends on the value of \( 1/\epsilon \). Thanks to a lively class discussion on December 20, 2006. If we fix \( \epsilon \) and let the problem size increase, then the complexity is cubic. Contributed by Mr. Ren-Shan Luoh (D97922014) on December 23, 2008.
Comments

- INDEPENDENT SET and NODE COVER are reducible to each other (Corollary 41, p. 368).

- NODE COVER has an approximation threshold at most 0.5 (p. 714).

- But INDEPENDENT SET is unapproximable (see the textbook).

- INDEPENDENT SET limited to graphs with degree \( \leq k \) is called \( k \)-DEGREE INDEPENDENT SET.

- \( k \)-DEGREE INDEPENDENT SET is approximable (see the textbook).