Algorithms

Lecture Topic: Approximation Algorithms (Part 2)

Anxiao (Andrew) Jiang

Roadmap of this lecture:

1. Understand approximation algorithms by solving TSP.
   1.1 General TSP has no constant-ratio approximation unless P=NP.
2. The "Linear Programming (LP) Technique" for approximation algorithms.
   2.1 Approximation algorithm for "Weighted Vertex Cover Problem" using the "LP technique".
   2.2 Analyze the approximation ratio of the algorithm.
3. Randomized Algorithm.
   3.1 Define "Randomized Algorithm".
   3.2 Understand "Randomized Algorithm" by solving the "Max 3-CNF SAT Problem"

Approximation Algorithms

Traveling Salesman Problem (TSP)

Input: An undirected complete graph G=(V,E), where every edge $(u,v) \in E$ has a non-negative integer weight $w(u,v)$.

Output: A Hamiltonian cycle of minimum weight.

TSP with Triangle Inequality

Input: An undirected complete graph G=(V,E), where every edge $(u,v) \in E$ has a non-negative integer weight $w(u,v)$.
The edge weights satisfy the triangle inequality.

Output: A Hamiltonian cycle of minimum weight.

Approximation Algorithms

Traveling Salesman Problem (TSP)

Input: An undirected complete graph G=(V,E), where every edge $(u,v) \in E$ has a non-negative integer weight $w(u,v)$.

Output: A Hamiltonian cycle of minimum weight.

➡ Does this general TSP have any approximation algorithm?

TSP with Triangle Inequality

Input: An undirected complete graph G=(V,E), where every edge $(u,v) \in E$ has a non-negative integer weight $w(u,v)$.
The edge weights satisfy the triangle inequality.

Output: A Hamiltonian cycle of minimum weight.

➡ The TSP with triangle inequality has a polynomial-time 2-approximation algorithm.

Theorem: If $P \neq NP$, then for any constant $\rho \geq 1$, there is NO polynomial-time ρ-approximation algorithm for the general TSP.

Even if $\rho = 999^{999^{999^{999}}}$ (or any other huge number),

there still cannot exist a polynomial-time ρ-approximation algorithm for TSP,

unless $P = NP$ (which most people consider to be unlikely).

Theorem: If $P \neq NP,$ then for any constant $\rho \geq 1,$ there is NO polynomial-time ρ-approximation algorithm for the general TSP.

How to prove it? A hint: if we let $\rho = 1,$

the theorem becomes:

If $P \neq NP,$ then TSP has no polynomial-time 1-approximation algorithm.

$\Updownarrow$

If $P \neq NP,$ then TSP is not polynomial-time solvable.

$\Uparrow$

$TSP \in NPC$.

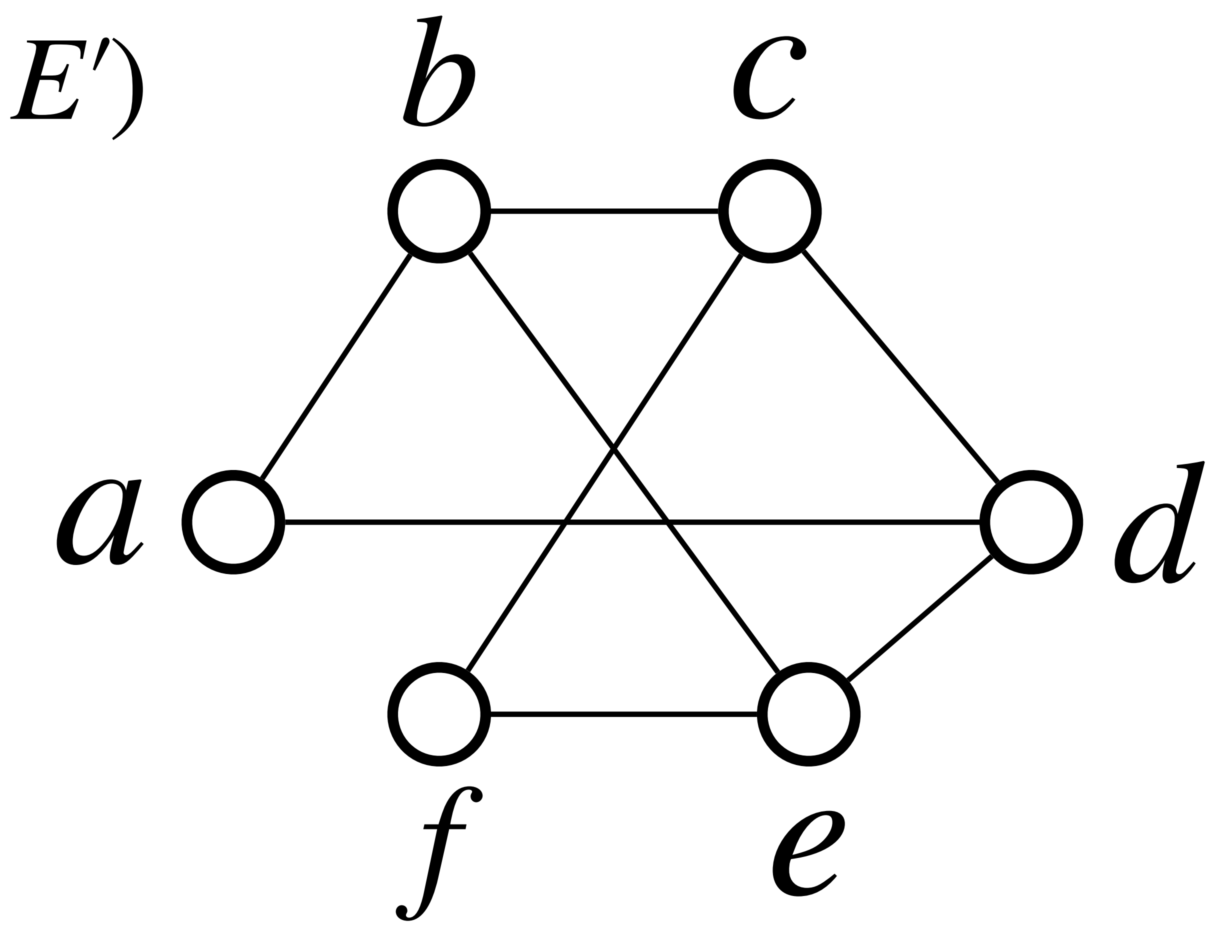
So the above theorem can actually be a generalization of proving TSP to be NP-Complete.

Theorem: If $P \neq NP$, then for any constant $\rho \geq 1$, there is NO polynomial-time ρ-approximation algorithm for the general TSP.

Proof:

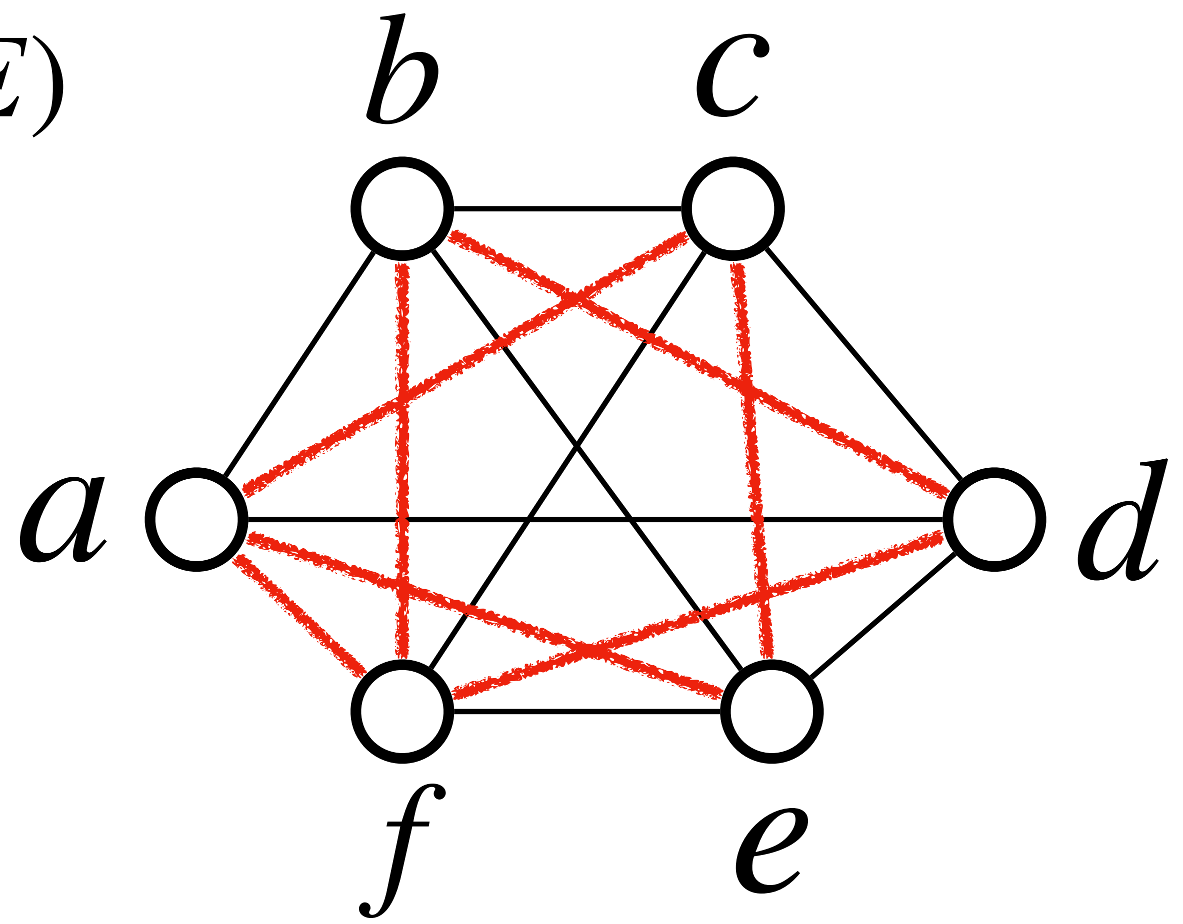
Hamiltonian Cycle Problem:

$G' = (V, E')$



Traveling Salesman Problem:

$G = (V, E)$



Black edges: weight 1

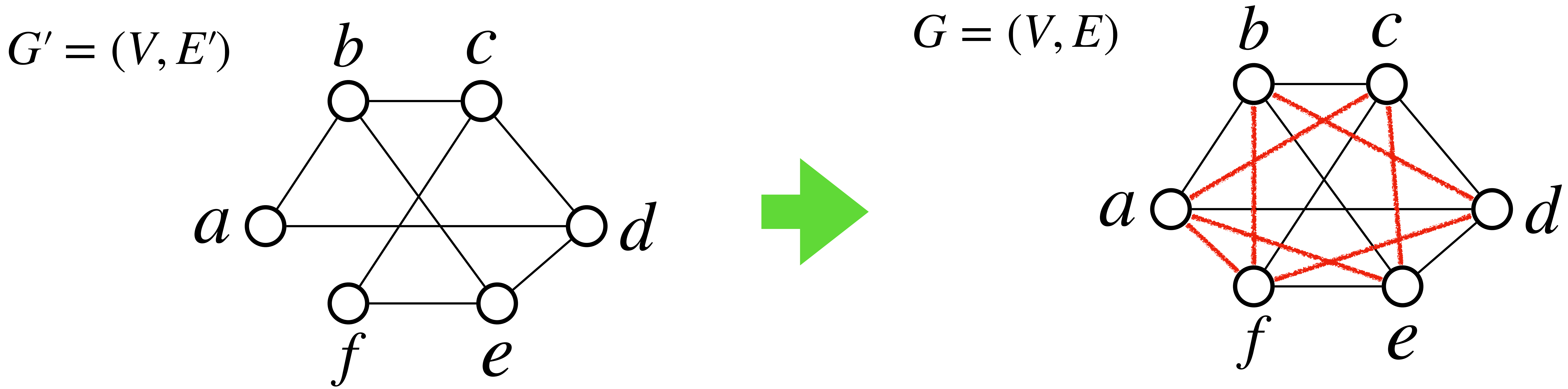
Red edges: weight $\rho|V| + 1$

Theorem: If $P \neq NP$, then for any constant $\rho \geq 1$, there is NO polynomial-time ρ-approximation algorithm for the general TSP.

Proof: By contradiction: assume the ρ-approximation algorithm exists.

Hamiltonian Cycle Problem:

Traveling Salesman Problem:



Black edges: weight 1

Red edges: weight $\rho |V| + 1$

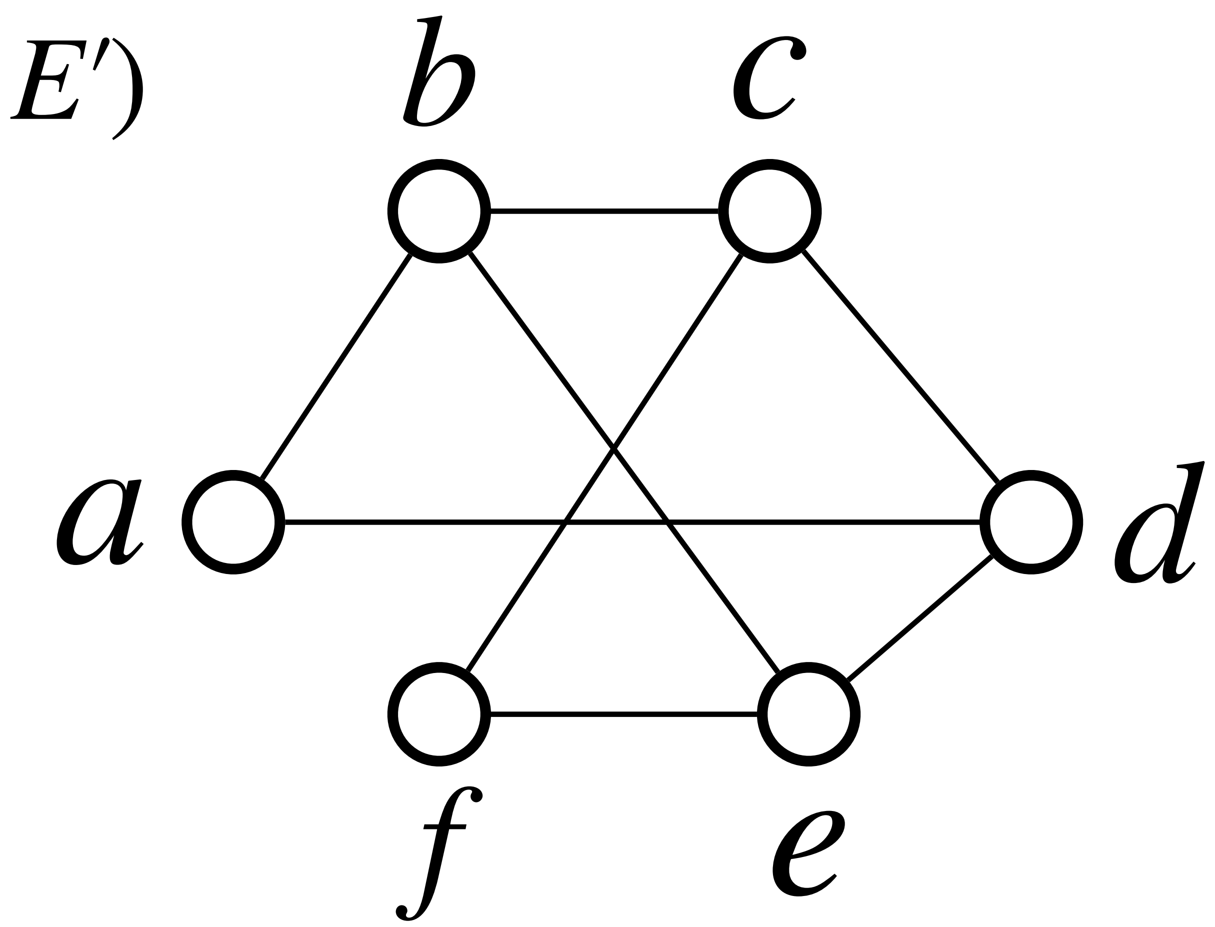
G' has a Hamiltonian cycle ➡ G has a Hamiltonian cycle of weight |V| ➡ Using the ρ-approximation algorithm, we can find a Hamiltonian cycle of weight $\leq \rho |V|$ in polynomial time.

Theorem: If $P \neq NP$, then for any constant $\rho \geq 1$, there is NO polynomial-time ρ-approximation algorithm for the general TSP.

Proof: By contradiction: assume the ρ-approximation algorithm exists.

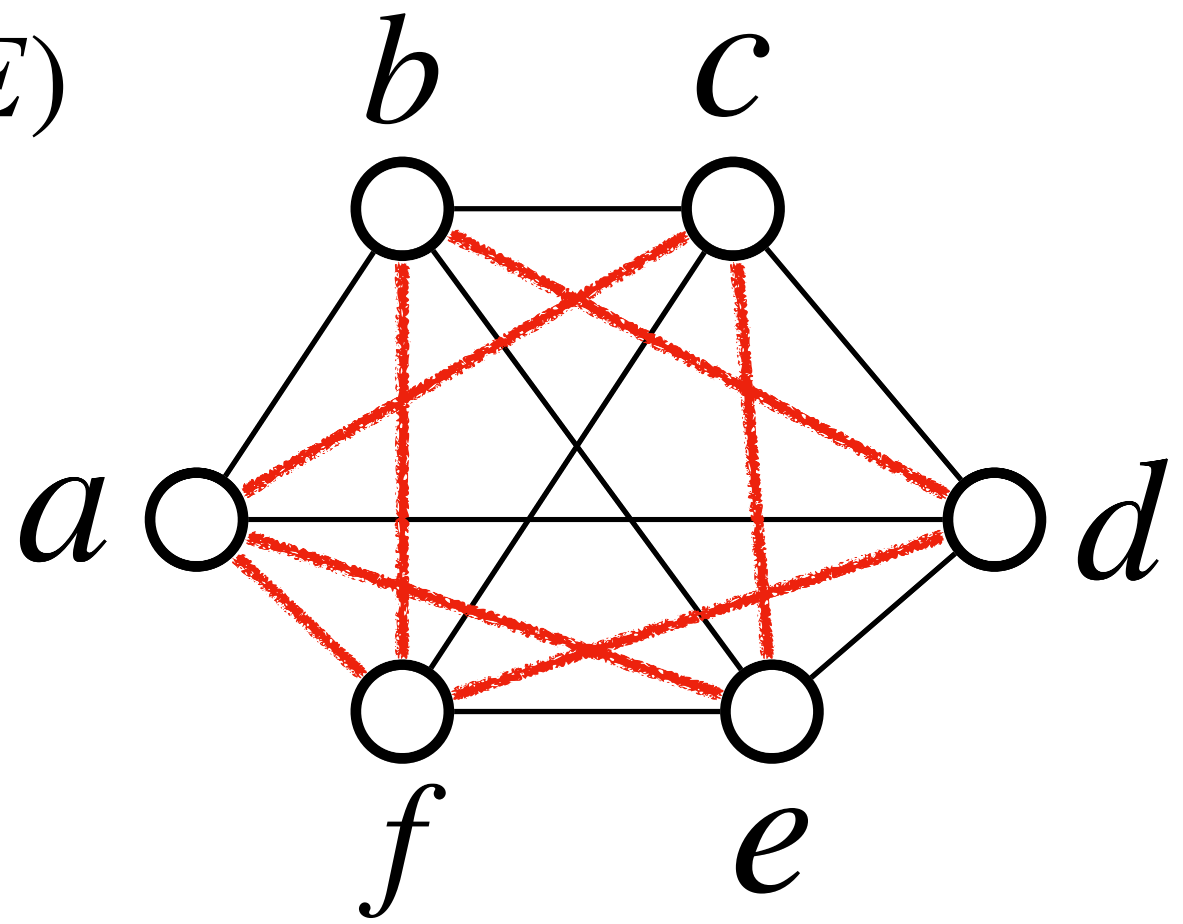
Hamiltonian Cycle Problem:

$G' = (V, E')$



Traveling Salesman Problem:

$G = (V, E)$



Black edges: weight 1

Red edges: weight $\rho|V| + 1$

G' has a Hamiltonian cycle ➡ G has a Hamiltonian cycle of weight |V| ➡ Using the ρ-approximation algorithm, we can find a Hamiltonian cycle of weight $\leq \rho|V|$ in G in polynomial time.

G' has no Hamiltonian cycle ➡ Any Hamiltonian cycle in G needs to use at least one red edge ➡ Any Hamiltonian cycle in G has weight $\geq \rho|V| + 1 + |V| - 1 = (\rho + 1)|V|$ ➡ Using the ρ-approximation algorithm, we find a Hamiltonian cycle of weight $\geq (\rho + 1)|V|$ in G in polynomial time.

Theorem: If $P \neq NP$, then for any constant $\rho \geq 1$, there is NO polynomial-time ρ-approximation algorithm for the general TSP.

Proof: By contradiction: assume the ρ-approximation algorithm exists.

Hamiltonian Cycle Problem: $G' = (V, E')$ → Traveling Salesman Problem: $G = (V, E)$

G' has a Hamiltonian cycle → G has a Hamiltonian cycle of weight |V| → Using the ρ-approximation algorithm, we can find a Hamiltonian cycle of weight $\leq \rho |V|$ in G in polynomial time.

G' has no Hamiltonian cycle → Any Hamiltonian cycle in G needs to use at least one red edge → Any Hamiltonian cycle in G has weight $\geq \rho |V| + 1 + |V| - 1 = (\rho + 1)|V|$ → Using the ρ-approximation algorithm, we find a Hamiltonian cycle of weight $\geq (\rho + 1)|V|$ in G in polynomial time.

Theorem: If $P \neq NP$, then for any constant $\rho \geq 1$, there is NO polynomial-time ρ-approximation algorithm for the general TSP.

Proof: By contradiction: assume the ρ-approximation algorithm exists.

Hamiltonian Cycle Problem: $G' = (V, E')$ → Traveling Salesman Problem: $G = (V, E)$

G' has a Hamiltonian cycle → • • • • • • → Using the ρ-approximation algorithm, we can find a Hamiltonian cycle of weight $\leq \rho |V|$ in G in polynomial time.

G' has no Hamiltonian cycle → • • • • • • → Using the ρ-approximation algorithm, we find a Hamiltonian cycle of weight $\geq (\rho + 1)|V|$ in G in polynomial time.

We can solve the Hamiltonian Cycle Problem "exactly" in polynomial time by solving TSP approximately using the polynomial-time ρ-approximation algorithm:

1) If we find a Hamiltonian cycle of weight $\leq \rho |V|$ in G → G' has a Hamiltonian cycle

2) If we find a Hamiltonian cycle of weight $\geq (\rho + 1)|V|$ in G → G' has no Hamiltonian cycle

Theorem: If $P \neq NP$, then for any constant $\rho \geq 1$, there is NO polynomial-time ρ-approximation algorithm for the general TSP.

Proof: By contradiction: assume the ρ-approximation algorithm exists.

Hamiltonian Cycle Problem: $G' = (V, E')$ ➡ Traveling Salesman Problem: $G = (V, E)$

G' has a Hamiltonian cycle ➡ • • • • • • ➡ Using the ρ-approximation algorithm, we can find a Hamiltonian cycle of weight $\leq \rho |V|$ in G in polynomial time.

G' has no Hamiltonian cycle ➡ • • • • • • ➡ Using the ρ-approximation algorithm, we find a Hamiltonian cycle of weight $\geq (\rho + 1)|V|$ in G in polynomial time.

We can solve the Hamiltonian Cycle Problem “exactly” in polynomial time by solving TSP approximately using the polynomial-time ρ-approximation algorithm:

1) If we find a Hamiltonian cycle of weight $\leq \rho |V|$ in G ➡ G' has a Hamiltonian cycle

2) If we find a Hamiltonian cycle of weight $\geq (\rho + 1)|V|$ in G ➡ G' has no Hamiltonian cycle

Theorem: If $P \neq NP,$ then for any constant $\rho \geq 1,$ there is NO polynomial-time ρ-approximation algorithm for the general TSP.

Proof: By contradiction: assume the ρ-approximation algorithm exists.

We can solve the Hamiltonian Cycle Problem “exactly” in polynomial time.

Theorem: If $P \neq NP,$ then for any constant $\rho \geq 1,$ there is NO polynomial-time ρ-approximation algorithm for the general TSP.

Proof: By contradiction: assume the ρ-approximation algorithm exists.

We can solve the Hamiltonian Cycle Problem “exactly” in polynomial time.

But we know the Hamiltonian Cycle Problem in NP-complete.

So it has to be

$$P = NP$$

which is a contradiction.

Q.E.D.

Quiz questions:

1. What method did we use to prove that general TSP has no constant-approximation unless P = NP?

Roadmap of this lecture:

1. Understand approximation algorithms by solving TSP.

   1.1 General TSP has no constant-ratio approximation unless P=NP.

2. The "Linear Programming (LP) Technique" for approximation algorithms.

   2.1 Approximation algorithm for "Weighted Vertex Cover Problem" using the "LP technique".

   2.2 Analyze the approximation ratio of the algorithm.

3. Randomized Algorithm.

   3.1 Define "Randomized Algorithm".

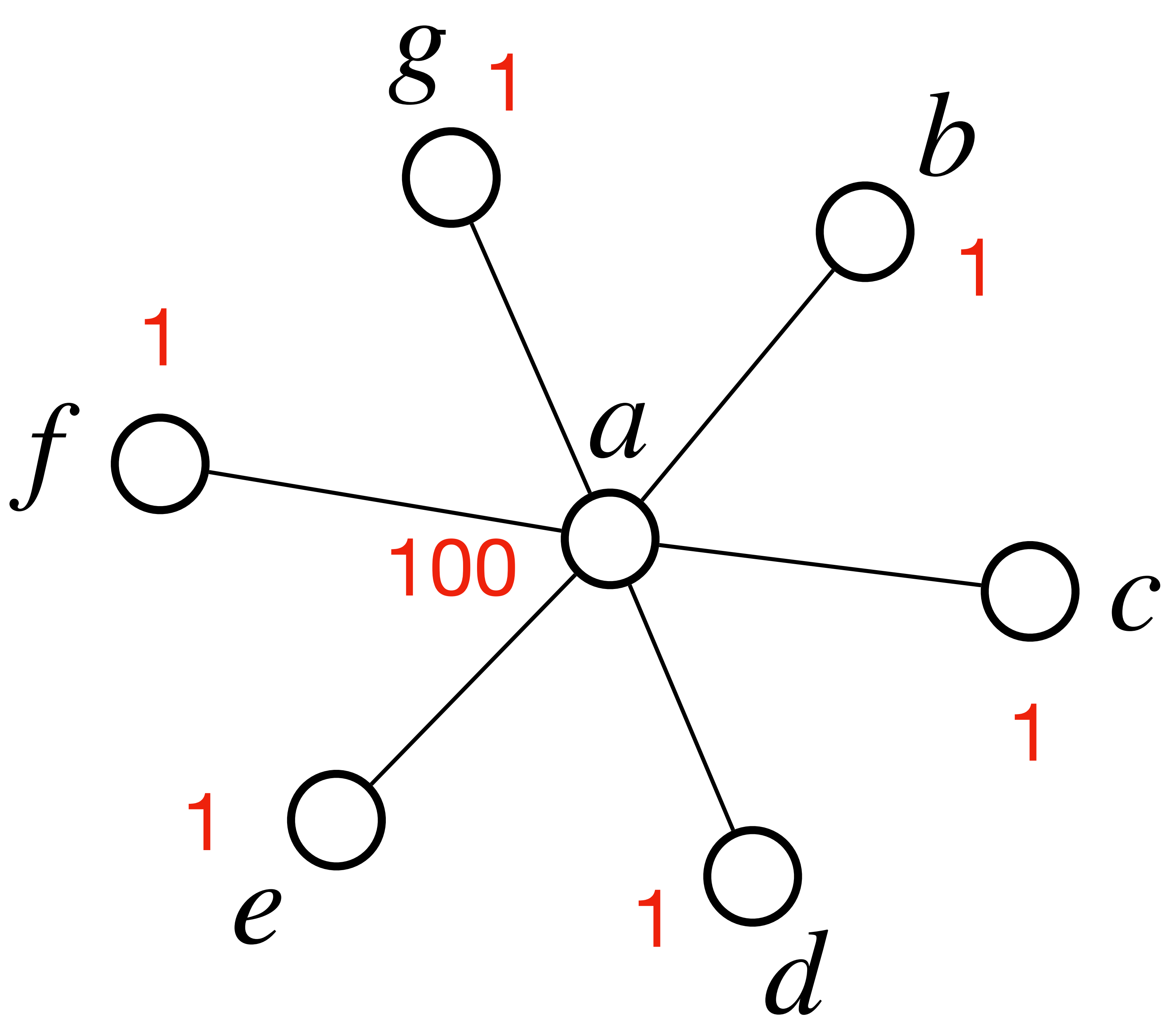
   3.2 Understand "Randomized Algorithm" by solving the "Max 3-CNF SAT Problem"

Technique: Linear Programming and Rounding for approximation

Weighted Vertex Cover Problem:

Input: An undirected graph G=(V,E), where every vertex $v \in V$ has a weight $w(v) > 0.$

Output: A vertex cover of minimum total weight.



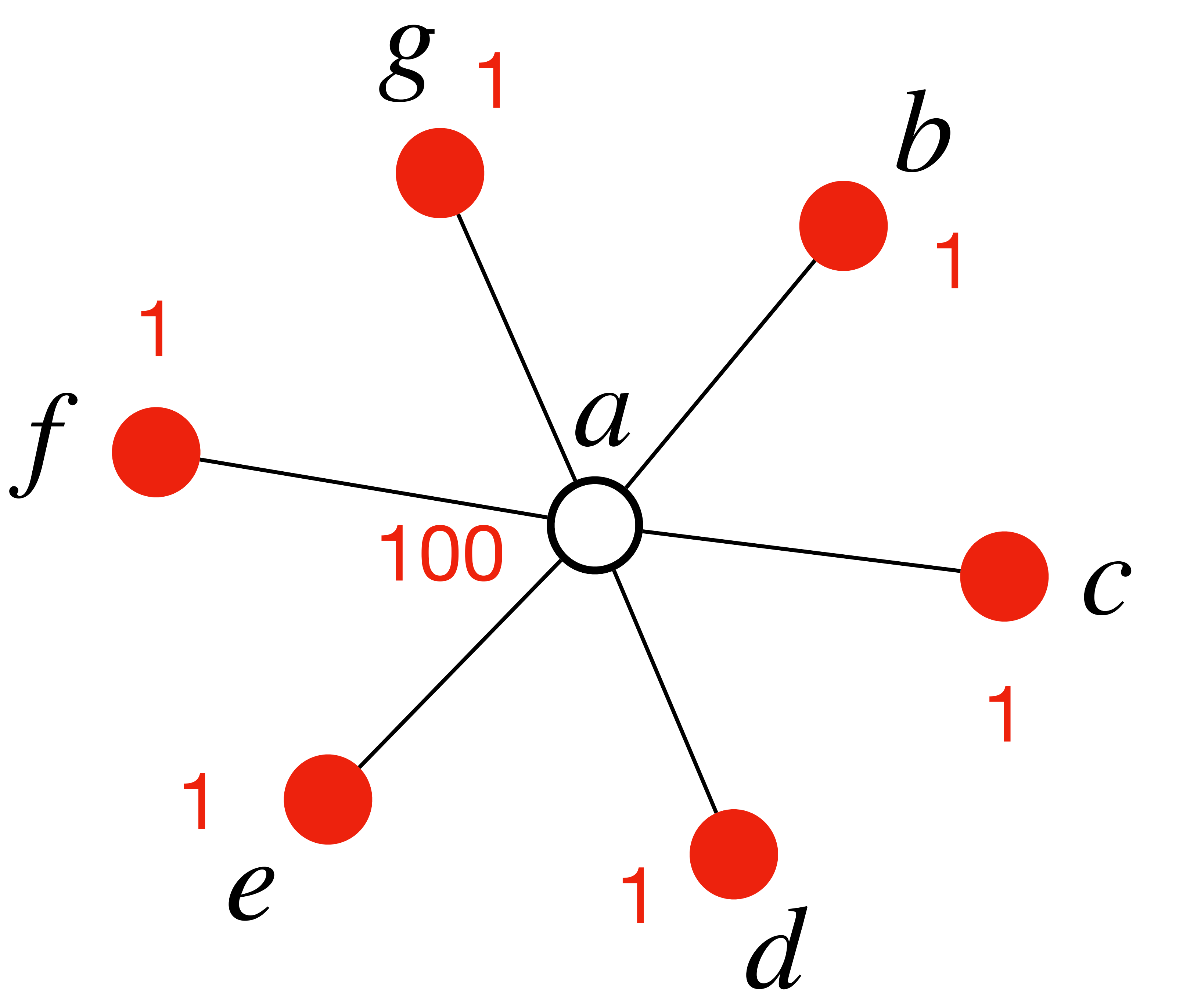
Technique: Linear Programming and Rounding for approximation

Weighted Vertex Cover Problem:

Input: An undirected graph G=(V,E), where every vertex $v \in V$ has a weight $w(v) > 0.$

Output: A vertex cover of minimum total weight.

Weight of vertex cover: 6



Technique: Linear Programming and Rounding for approximation

Weighted Vertex Cover Problem:

Input: An undirected graph G=(V,E), where every vertex $v \in V$ has weight $w(v) > 0$.

Output: A vertex cover of minimum total weight.

Our technique:

1. Formulate the problem as an integer programming problem.

Define variables:

For every node $v \in V$, define a variable

$$x(v) = \begin{cases} 1 & \text{if } v \text{ is in vertex cover} \\ 0 & \text{otherwise} \end{cases}$$

Technique: Linear Programming and Rounding for approximation

Weighted Vertex Cover Problem:

Input: An undirected graph G=(V,E), where every vertex $v \in V$ has weight $w(v) > 0$.

Output: A vertex cover of minimum total weight.

Our technique:

1. Formulate the problem as an integer programming problem.

Define variables:

For every node $v \in V$, define a variable $x(v) = \begin{cases} 1 & \text{if } v \text{ is in vertex cover} \\ 0 & \text{otherwise} \end{cases}$

Integer Programming Problem:

minimize $\sum_{v \in V} w(v)x(v)$

s.t. for every edge $(u, v) \in E$, $x(u) + x(v) \geq 1$

for every node $v \in V$, $x(v) \in \{0,1\}$

Technique: Linear Programming and Rounding for approximation

Weighted Vertex Cover Problem:

Input: An undirected graph G=(V,E), where every vertex $v \in V$ has weight $w(v) > 0$.

Output: A vertex cover of minimum total weight.

Our technique:

1. Formulate the problem as an integer programming problem.
2. Relax condition to turn it into an LP.

Integer Programming Problem:

$$\text{minimize} \quad \sum_{v \in V} w(v)x(v)$$
$$\text{s.t.} \quad \forall (u,v) \in E, \quad x(u) + x(v) \geq 1$$
$$\forall v \in V, \quad x(v) \in \{0,1\}$$

$\Rightarrow$

Linear Programming Problem:

$$\text{minimize} \quad \sum_{v \in V} w(v)x(v)$$
$$\text{s.t.} \quad \forall (u,v) \in E, \quad x(u) + x(v) \geq 1$$
$$\forall v \in V, \quad 0 \leq x(v) \leq 1$$

Technique: Linear Programming and Rounding for approximation

Weighted Vertex Cover Problem:

Input: An undirected graph G=(V,E), where every vertex $v \in V$ has weight $w(v) > 0$.

Output: A vertex cover of minimum total weight.

Our technique:

1. Formulate the problem as an integer programming problem.
2. Relax condition to turn it into an LP.
3. Solve the LP, then turn the LP solution to a solution of the original integer programming problem using "rounding".

Integer Programming Problem:

$$\text{minimize} \quad \sum_{v \in V} w(v)x(v)$$
$$\text{s.t.} \ \forall (u,v) \in E, \quad x(u) + x(v) \geq 1$$
$$\forall v \in V, \quad x(v) \in \{0,1\}$$

$\Rightarrow$

Linear Programming Problem:

$$\text{minimize} \quad \sum_{v \in V} w(v)x(v)$$
$$\text{s.t.} \ \forall (u,v) \in E, \quad x(u) + x(v) \geq 1$$
$$\forall v \in V, \quad 0 \leq x(v) \leq 1$$

Technique: Linear Programming and Rounding for approximation

Weighted Vertex Cover Problem:

Input: An undirected graph G=(V,E), where every vertex $v \in V$ has weight $w(v) > 0$.

Output: A vertex cover of minimum total weight.

Our technique:

1. Formulate the problem as an integer programming problem.
2. Relax condition to turn it into an LP.
3. Solve the LP, then turn the LP solution to a solution of the original integer programming problem using "rounding".

Integer Programming Problem:

$$\text{minimize} \quad \sum_{v \in V} w(v)x(v)$$
$$\text{s.t.} \ \forall (u,v) \in E, \quad x(u) + x(v) \geq 1$$
$$\forall v \in V, \quad x(v) \in \{0,1\}$$

Linear Programming Problem:

$$\text{minimize} \quad \sum_{v \in V} w(v)x(v)$$
$$\text{s.t.} \ \forall (u,v) \in E, \quad x(u) + x(v) \geq 1$$
$$\forall v \in V, \quad 0 \leq x(v) \leq 1$$

Optimal solution to LP: $\bar{x}(v) \quad \forall v \in V$

Technique: Linear Programming and Rounding for approximation

Weighted Vertex Cover Problem:

Input: An undirected graph G=(V,E), where every vertex $v \in V$ has weight $w(v) > 0$.

Output: A vertex cover of minimum total weight.

Our technique:

1. Formulate the problem as an integer programming problem.
2. Relax condition to turn it into an LP.
3. Solve the LP, then turn the LP solution to a solution of the original integer programming problem using "rounding".

Integer Programming Problem:

$$\text{minimize} \quad \sum_{v \in V} w(v)x(v)$$
$$\text{s.t. } \forall (u,v) \in E, \quad x(u) + x(v) \geq 1$$
$$\forall v \in V, \quad x(v) \in \{0,1\}$$

Linear Programming Problem:

$$\text{minimize} \quad \sum_{v \in V} w(v)x(v)$$
$$\text{s.t. } \forall (u,v) \in E, \quad x(u) + x(v) \geq 1$$
$$\forall v \in V, \quad 0 \leq x(v) \leq 1$$

Optimal solution to LP: $\bar{x}(v) \quad \forall v \in V$

Get a solution to the Integer Programming Problem:

$$\forall\ v \in V, \quad x(v) = \begin{cases} 1 & \text{if } \bar{x}(v) \geq 0.5 \\ 0 & \text{if } \bar{x}(v) < 0.5 \end{cases}$$

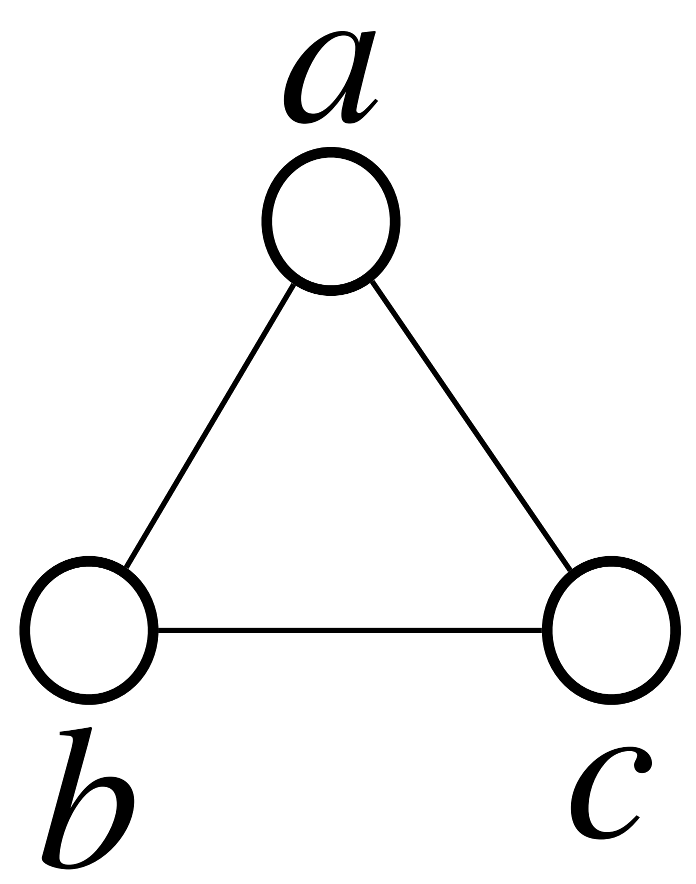
Technique: Linear Programming and Rounding for approximation

Weighted Vertex Cover Problem:

Input: An undirected graph G=(V,E), where every vertex $v \in V$ has weight $w(v) > 0$.

Output: A vertex cover of minimum total weight.

Example:



Integer Programming Problem:

$$\text{minimize} \quad \sum_{v \in V} w(v)x(v)$$
$$\text{s.t. } \forall (u,v) \in E, \quad x(u) + x(v) \geq 1$$
$$\forall v \in V, \quad x(v) \in \{0,1\}$$

Linear Programming Problem:

$$\text{minimize} \quad \sum_{v \in V} w(v)x(v)$$
$$\text{s.t. } \forall (u,v) \in E, \quad x(u) + x(v) \geq 1$$
$$\forall v \in V, \quad 0 \leq x(v) \leq 1$$

Optimal solution to LP: $\bar{x}(v) \quad \forall v \in V$

Get a solution to the Integer Programming Problem:

$$\forall\ v \in V, \quad x(v) = \begin{cases} 1 & \text{if } \bar{x}(v) \geq 0.5 \\ 0 & \text{if } \bar{x}(v) < 0.5 \end{cases}$$

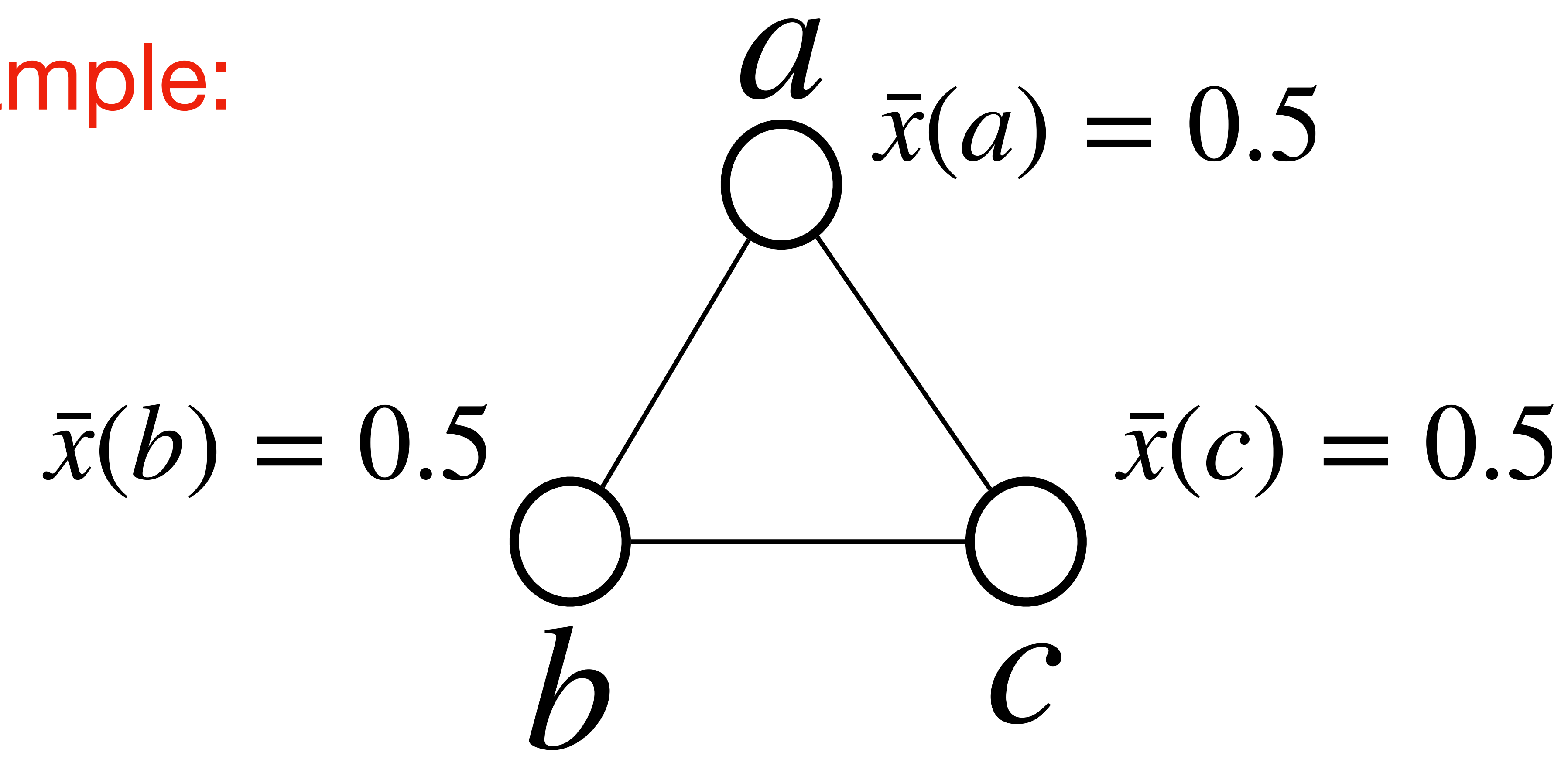
Technique: Linear Programming and Rounding for approximation

Weighted Vertex Cover Problem:

Input: An undirected graph G=(V,E), where every vertex $v \in V$ has weight $w(v) > 0$.

Output: A vertex cover of minimum total weight.

Example:



Integer Programming Problem:

$$\text{minimize} \quad \sum_{v \in V} w(v)x(v)$$
$$\text{s.t. } \forall (u,v) \in E, \quad x(u) + x(v) \geq 1$$
$$\forall v \in V, \quad x(v) \in \{0,1\}$$

Linear Programming Problem:

$$\text{minimize} \quad \sum_{v \in V} w(v)x(v)$$
$$\text{s.t. } \forall (u,v) \in E, \quad x(u) + x(v) \geq 1$$
$$\forall v \in V, \quad 0 \leq x(v) \leq 1$$

Optimal solution to LP: $\bar{x}(v) \quad \forall v \in V$

Get a solution to the Integer Programming Problem:

$$\forall\, v \in V, \quad x(v) = \begin{cases} 1 & \text{if } \bar{x}(v) \geq 0.5 \\ 0 & \text{if } \bar{x}(v) < 0.5 \end{cases}$$

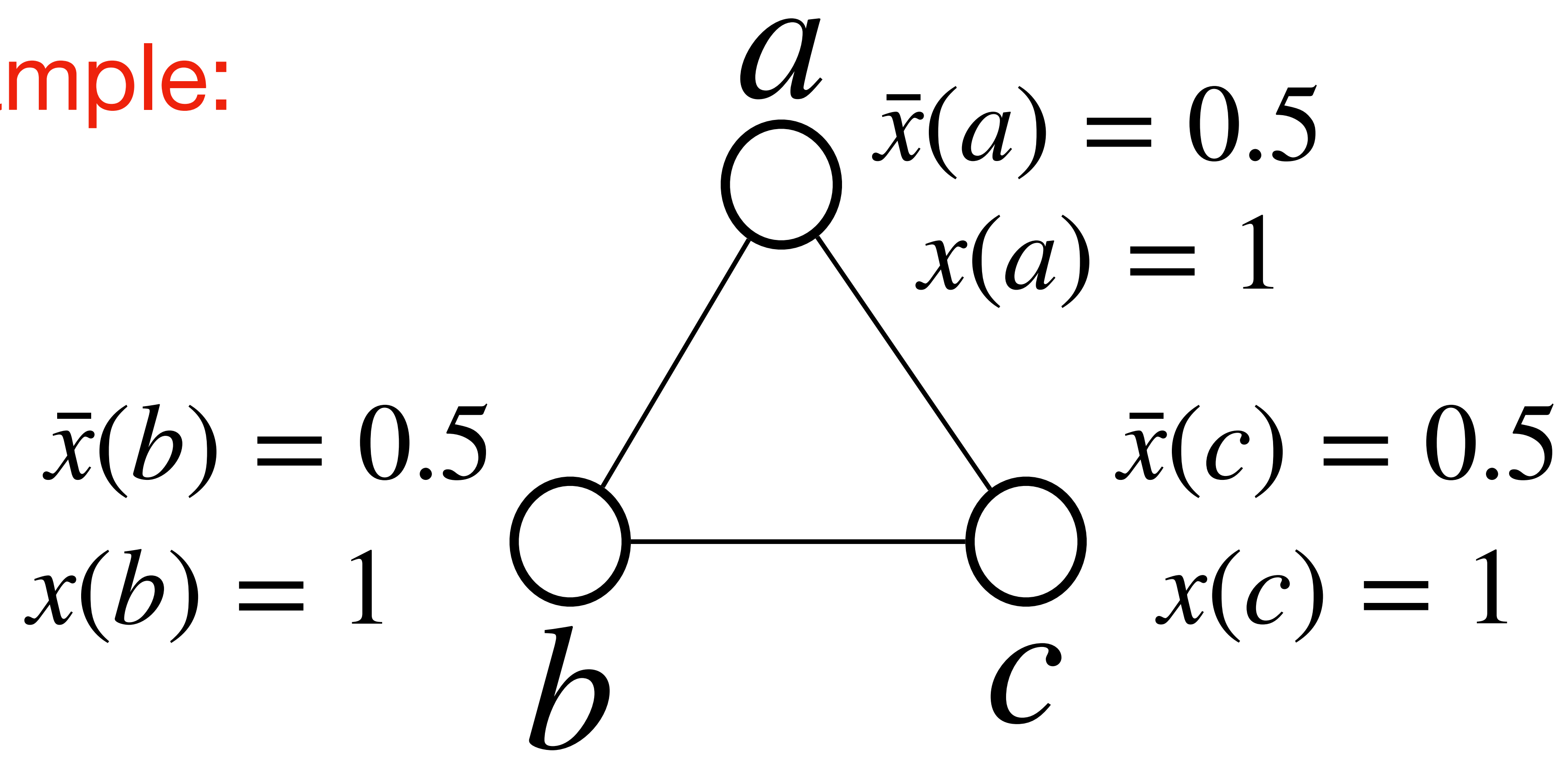
Technique: Linear Programming and Rounding for approximation

Weighted Vertex Cover Problem:

Input: An undirected graph G=(V,E), where every vertex $v \in V$ has weight $w(v) > 0$.

Output: A vertex cover of minimum total weight.

Example:



Integer Programming Problem:

$$\text{minimize} \quad \sum_{v \in V} w(v)x(v)$$
$$\text{s.t. } \forall (u,v) \in E, \quad x(u) + x(v) \geq 1$$
$$\forall v \in V, \quad x(v) \in \{0,1\}$$

Linear Programming Problem:

$$\text{minimize} \quad \sum_{v \in V} w(v)x(v)$$
$$\text{s.t. } \forall (u,v) \in E, \quad x(u) + x(v) \geq 1$$
$$\forall v \in V, \quad 0 \leq x(v) \leq 1$$

Optimal solution to LP: $\bar{x}(v) \quad \forall v \in V$

Get a solution to the Integer Programming Problem:

$$\forall\ v \in V, \quad x(v) = \begin{cases} 1 & \text{if } \bar{x}(v) \geq 0.5 \\ 0 & \text{if } \bar{x}(v) < 0.5 \end{cases}$$

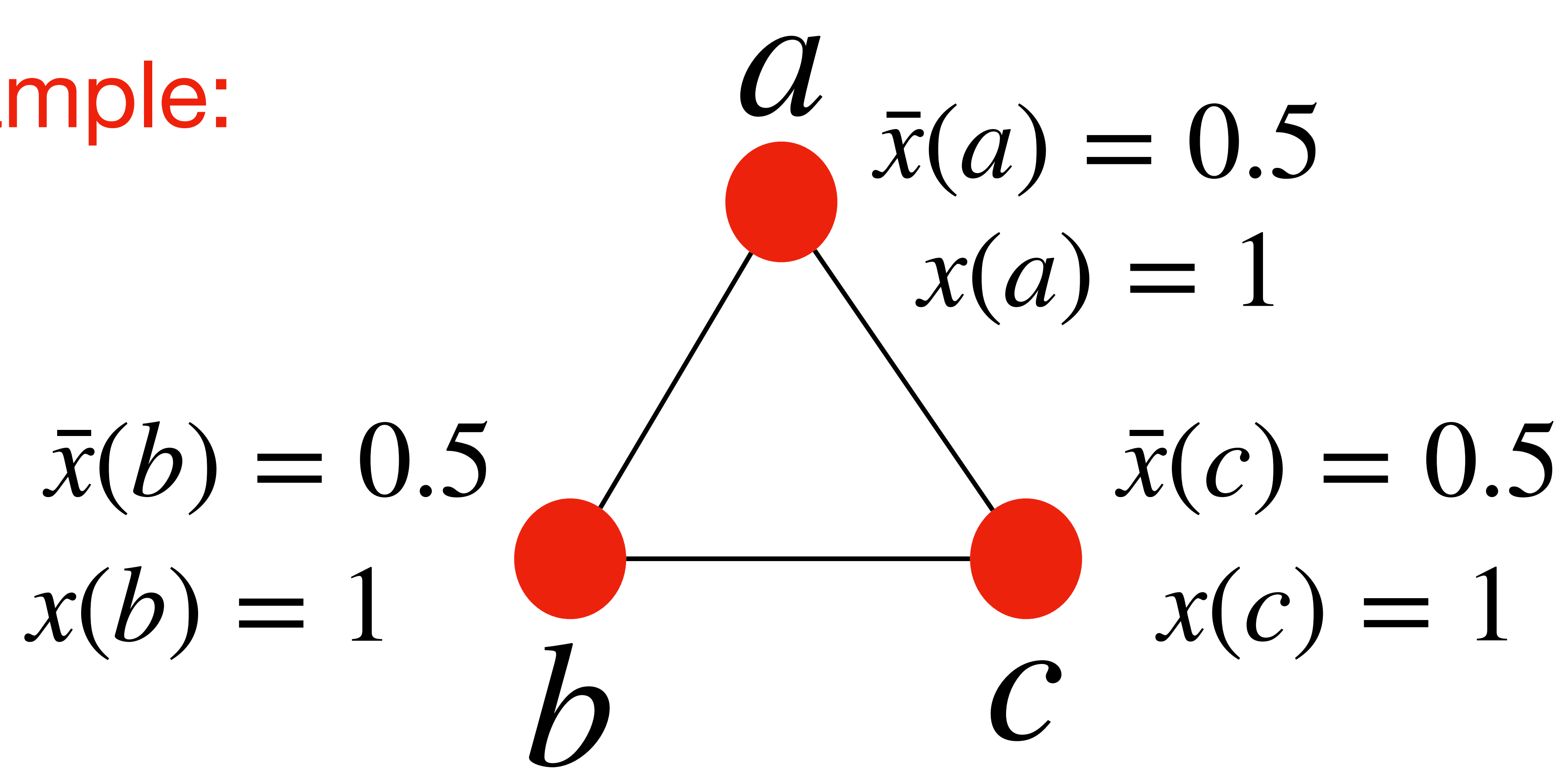
Technique: Linear Programming and Rounding for approximation

Weighted Vertex Cover Problem:

Input: An undirected graph G=(V,E), where every vertex $v \in V$ has weight $w(v) > 0$.

Output: A vertex cover of minimum total weight.

Example:



Integer Programming Problem:

$$\text{minimize} \quad \sum_{v \in V} w(v)x(v)$$
$$\text{s.t. } \forall (u,v) \in E, \quad x(u) + x(v) \geq 1$$
$$\forall v \in V, \quad x(v) \in \{0,1\}$$

Linear Programming Problem:

$$\text{minimize} \quad \sum_{v \in V} w(v)x(v)$$
$$\text{s.t. } \forall (u,v) \in E, \quad x(u) + x(v) \geq 1$$
$$\forall v \in V, \quad 0 \leq x(v) \leq 1$$

Optimal solution to LP: $\bar{x}(v) \quad \forall v \in V$

Get a solution to the Integer Programming Problem:

$$\forall\ v \in V, \quad x(v) = \begin{cases} 1 & \text{if } \bar{x}(v) \geq 0.5 \\ 0 & \text{if } \bar{x}(v) < 0.5 \end{cases}$$

Technique: Linear Programming and Rounding for approximation

Weighted Vertex Cover Problem:

Input: An undirected graph G=(V,E), where every vertex $v \in V$ has weight $w(v) > 0$.

Output: A vertex cover of minimum total weight.

Integer Programming Problem:

$$\begin{aligned} \text{minimize} \quad & \sum_{v \in V} w(v)x(v) \\ \text{s.t.} \quad & \forall (u,v) \in E, \ \ x(u) + x(v) \geq 1 \\ & \forall v \in V, \ \ x(v) \in \{0,1\} \end{aligned}$$

Linear Programming Problem:

$$\begin{aligned} \text{minimize} \quad & \sum_{v \in V} w(v)x(v) \\ \text{s.t.} \quad & \forall (u,v) \in E, \ \ x(u) + x(v) \geq 1 \\ & \forall v \in V, \ \ 0 \leq x(v) \leq 1 \end{aligned}$$

Optimal solution to LP: $\bar{x}(v) \quad \forall v \in V$

Get a solution to the Integer Programming Problem:

$$\forall \ v \in V, \quad x(v) = \begin{cases} 1 & \text{if } \bar{x}(v) \geq 0.5 \\ 0 & \text{if } \bar{x}(v) < 0.5 \end{cases}$$

Why is this indeed a vertex cover?

Quiz questions:

1. What is the main idea of the above LP-based approximation algorithm for "Weighted Vertex Cover"?

2. Can you think of an instance for which the above approximation algorithm outputs an optimal solution, and an instance for which it does not?

Roadmap of this lecture:

1. Understand approximation algorithms by solving TSP.

   1.1 General TSP has no constant-ratio approximation unless P=NP.

2. The "Linear Programming (LP) Technique" for approximation algorithms.

   2.1 Approximation algorithm for "Weighted Vertex Cover Problem" using the "LP technique".

   2.2 Analyze the approximation ratio of the algorithm.

3. Randomized Algorithm.

   3.1 Define "Randomized Algorithm".

   3.2 Understand "Randomized Algorithm" by solving the "Max 3-CNF SAT Problem"

Integer Programming Problem:

$$\begin{aligned}&\text{minimize} \quad \sum_{v\in V} w(v)x(v)\\ &\text{s.t. } \forall (u,v)\in E,\ \ x(u)+x(v)\geq 1\\ &\qquad \forall v\in V,\ \ x(v)\in\{0,1\}\end{aligned}$$

Linear Programming Problem:

$$\begin{aligned}&\text{minimize} \quad \sum_{v\in V} w(v)x(v)\\ &\text{s.t. } \forall (u,v)\in E,\ \ x(u)+x(v)\geq 1\\ &\qquad \forall v\in V,\ \ 0\leq x(v)\leq 1\end{aligned}$$

Optimal solution to LP: $\bar{x}(v) \quad \forall v\in V$

Get a solution to the Integer Programming Problem:

$$\forall\ v\in V,\quad x(v)=\begin{cases}1 & \text{if } \bar{x}(v)\geq 0.5\\ 0 & \text{if } \bar{x}(v)<0.5\end{cases}$$

Theorem: The above algorithm is a polynomial-time 2-approximation algorithm.

Integer Programming Problem:

$$\text{minimize} \quad \sum_{v \in V} w(v)x(v)$$
$$\text{s.t. } \forall (u,v) \in E, \quad x(u) + x(v) \geq 1$$
$$\forall v \in V, \quad x(v) \in \{0,1\}$$

Linear Programming Problem:

$$\text{minimize} \quad \sum_{v \in V} w(v)x(v)$$
$$\text{s.t. } \forall (u,v) \in E, \quad x(u) + x(v) \geq 1$$
$$\forall v \in V, \quad 0 \leq x(v) \leq 1$$

Optimal solution to LP: $\bar{x}(v) \quad \forall v \in V$

Get a solution to the Integer Programming Problem:

$$\forall\ v \in V, \quad x(v) = \begin{cases} 1 & \text{if } \bar{x}(v) \geq 0.5 \\ 0 & \text{if } \bar{x}(v) < 0.5 \end{cases}$$

Theorem: The above algorithm is a polynomial-time 2-approximation algorithm.

Proof: Weight of LP solution

$$\bar{C} = \sum_{v \in V} w(v)\bar{x}(v)$$

Integer Programming Problem:

$$\begin{aligned} &\text{minimize} \quad \sum_{v \in V} w(v)x(v) \\ &\text{s.t.} \ \forall (u,v) \in E, \ \ x(u) + x(v) \geq 1 \\ &\qquad \forall v \in V, \ \ x(v) \in \{0,1\} \end{aligned}$$

Linear Programming Problem:

$$\begin{aligned} &\text{minimize} \quad \sum_{v \in V} w(v)x(v) \\ &\text{s.t.} \ \forall (u,v) \in E, \ \ x(u) + x(v) \geq 1 \\ &\qquad \forall v \in V, \ \ 0 \leq x(v) \leq 1 \end{aligned}$$

Optimal solution to LP: $\bar{x}(v) \quad \forall v \in V$

Get a solution to the Integer Programming Problem:

$$\forall \ v \in V, \quad x(v) = \begin{cases} 1 & \text{if } \bar{x}(v) \geq 0.5 \\ 0 & \text{if } \bar{x}(v) < 0.5 \end{cases}$$

Theorem: The above algorithm is a polynomial-time 2-approximation algorithm.

Proof: Weight of LP solution

$$\bar{C} = \sum_{v \in V} w(v)\bar{x}(v)$$

Weight of optimal vertex cover: C^*

Integer Programming Problem:

$$\begin{aligned} &\text{minimize} \quad \sum_{v \in V} w(v)x(v) \\ &\text{s.t. } \forall (u,v) \in E, \quad x(u) + x(v) \geq 1 \\ &\quad \forall v \in V, \quad x(v) \in \{0,1\} \end{aligned}$$

Linear Programming Problem:

$$\begin{aligned} &\text{minimize} \quad \sum_{v \in V} w(v)x(v) \\ &\text{s.t. } \forall (u,v) \in E, \quad x(u) + x(v) \geq 1 \\ &\quad \forall v \in V, \quad 0 \leq x(v) \leq 1 \end{aligned}$$

Optimal solution to LP: $\bar{x}(v) \quad \forall v \in V$

Get a solution to the Integer Programming Problem:

$$\forall\ v \in V, \quad x(v) = \begin{cases} 1 & \text{if } \bar{x}(v) \geq 0.5 \\ 0 & \text{if } \bar{x}(v) < 0.5 \end{cases}$$

Theorem: The above algorithm is a polynomial-time 2-approximation algorithm.

Proof: Weight of LP solution

$$\bar{C} = \sum_{v \in V} w(v)\bar{x}(v)$$

Weight of optimal vertex cover: C^*

$$\boxed{\bar{C} \leq C^*}$$

Why?

Integer Programming Problem:

$$\begin{aligned}\text{minimize} \quad & \sum_{v \in V} w(v)x(v) \\ \text{s.t.} \quad & \forall (u,v) \in E, \;\; x(u) + x(v) \geq 1 \\ & \forall v \in V, \;\; x(v) \in \{0,1\}\end{aligned}$$

Linear Programming Problem:

$$\begin{aligned}\text{minimize} \quad & \sum_{v \in V} w(v)x(v) \\ \text{s.t.} \quad & \forall (u,v) \in E, \;\; x(u) + x(v) \geq 1 \\ & \forall v \in V, \;\; 0 \leq x(v) \leq 1\end{aligned}$$

Optimal solution to LP: $\bar{x}(v) \quad \forall v \in V$

Get a solution to the Integer Programming Problem:

$$\forall \; v \in V, \quad x(v) = \begin{cases} 1 & \text{if } \bar{x}(v) \geq 0.5 \\ 0 & \text{if } \bar{x}(v) < 0.5 \end{cases}$$

Theorem: The above algorithm is a polynomial-time 2-approximation algorithm.

Proof: Weight of LP solution

$$\bar{C} = \sum_{v \in V} w(v)\bar{x}(v)$$

Weight of optimal vertex cover: C^*

$$\boxed{\bar{C} \leq C^*}$$

Weight of integer program solution

$$C = \sum_{v \in V} w(v)x(v)$$

Integer Programming Problem:

$$\text{minimize} \quad \sum_{v \in V} w(v)x(v)$$
$$\text{s.t. } \forall (u,v) \in E, \quad x(u) + x(v) \geq 1$$
$$\forall v \in V, \quad x(v) \in \{0,1\}$$

Linear Programming Problem:

$$\text{minimize} \quad \sum_{v \in V} w(v)x(v)$$
$$\text{s.t. } \forall (u,v) \in E, \quad x(u) + x(v) \geq 1$$
$$\forall v \in V, \quad 0 \leq x(v) \leq 1$$

Optimal solution to LP: $\bar{x}(v) \quad \forall v \in V$

Get a solution to the Integer Programming Problem:

$$\forall\ v \in V, \quad x(v) = \begin{cases} 1 & \text{if } \bar{x}(v) \geq 0.5 \\ 0 & \text{if } \bar{x}(v) < 0.5 \end{cases}$$

Theorem: The above algorithm is a polynomial-time 2-approximation algorithm.

Proof: Weight of LP solution

$$\bar{C} = \sum_{v \in V} w(v)\bar{x}(v)$$

Weight of optimal vertex cover: C^*

$$\boxed{\bar{C} \leq C^*}$$

Weight of integer program solution

$$C = \sum_{v \in V} w(v)x(v) \leq 2 \sum_{v \in V} w(v)\bar{x}(v)$$

Why?

Integer Programming Problem:

$$\begin{array}{ll} \text{minimize} & \sum_{v \in V} w(v)x(v) \\ \text{s.t.} & \forall (u,v) \in E, \;\; x(u) + x(v) \geq 1 \\ & \forall v \in V, \;\; x(v) \in \{0,1\} \end{array}$$

Linear Programming Problem:

$$\begin{array}{ll} \text{minimize} & \sum_{v \in V} w(v)x(v) \\ \text{s.t.} & \forall (u,v) \in E, \;\; x(u) + x(v) \geq 1 \\ & \forall v \in V, \;\; 0 \leq x(v) \leq 1 \end{array}$$

Optimal solution to LP: $\bar{x}(v) \quad \forall v \in V$

Get a solution to the Integer Programming Problem:

$$\forall \; v \in V, \quad x(v) = \begin{cases} 1 & \text{if } \bar{x}(v) \geq 0.5 \\ 0 & \text{if } \bar{x}(v) < 0.5 \end{cases}$$

Theorem: The above algorithm is a polynomial-time 2-approximation algorithm.

Proof: Weight of LP solution

$$\bar{C} = \sum_{v \in V} w(v)\bar{x}(v)$$

Weight of optimal vertex cover: C^*

$$\boxed{\bar{C} \leq C^*}$$

Weight of integer program solution

$$C = \sum_{v \in V} w(v)x(v) \leq 2 \sum_{v \in V} w(v)\bar{x}(v) = 2\bar{C} \leq 2C^*$$

Vertex Cover Problem: 2-approximation.

TSP with triangle inequality: 2-approximation.

General TSP: no constant ratio approximation unless P=NP.

Set Covering Problem: $\rho(n)$ -approximation

$\rho(n) \to \infty$ as $n \to \infty$

Subset Sum Problem: $(1+\epsilon)$ -approximation

Time complexity poly$(n, \frac{1}{\epsilon})$

Quiz question:

1. How did we find the approximation ratio for the above algorithm for "Weighted Vertex Cover" without knowing the optimal cost?

Roadmap of this lecture:

1. Understand approximation algorithms by solving TSP.

    1.1 General TSP has no constant-ratio approximation unless P=NP.

2. The "Linear Programming (LP) Technique" for approximation algorithms.

    2.1 Approximation algorithm for "Weighted Vertex Cover Problem" using the "LP technique".

    2.2 Analyze the approximation ratio of the algorithm.

3. Randomized Algorithm.

    3.1 Define "Randomized Algorithm".

    3.2 Understand "Randomized Algorithm" by solving the "Max 3-CNF SAT Problem"

Randomized Algorithm

Consider a maximization problem.

Let $C^* > 0$ be the cost of an optimal solution.

Let $C > 0$ be the expected cost of the solution of a randomized algorithm.

If for all instances, we have $\dfrac{C^*}{C} \leq \rho,$

then the randomized algorithm is called a

ρ-approximation randomized algorithm.

Randomized Algorithm

Consider a minimization problem.

Let $C^* > 0$ be the cost of an optimal solution.

Let $C > 0$ be the expected cost of the solution of a randomized algorithm.

If for all instances, we have $\dfrac{C}{C^*} \leq \rho,$

then the randomized algorithm is called a ρ-approximation randomized algorithm.

Quiz questions:

1. What is a "Randomized Algorithm"?
2. What is the difference between the approximation ratio of a randomized algorithm and that of a deterministic algorithm?

Roadmap of this lecture:

1. Understand approximation algorithms by solving TSP.

   1.1 General TSP has no constant-ratio approximation unless P=NP.

2. The "Linear Programming (LP) Technique" for approximation algorithms.

   2.1 Approximation algorithm for "Weighted Vertex Cover Problem" using the "LP technique".

   2.2 Analyze the approximation ratio of the algorithm.

3. Randomized Algorithm.

   3.1 Define "Randomized Algorithm".

   3.2 Understand "Randomized Algorithm" by solving the "Max 3-CNF SAT Problem"

Max 3-CNF SAT Problem

Input: A 3-CNF Boolean formula of n variables and k clauses,
Where every clause is the OR of 3 literals.

The 3 variables involved in each clause are distinct.

Output: A solution to the variables that maximizes the number of satisfied clauses.

Randomized Algorithm

Max 3-CNF SAT Problem

Input: A 3-CNF Boolean formula of n variables and k clauses,
Where every clause is the OR of 3 literals.
The 3 variables involved in each clause are distinct.

Output: A solution to the variables that maximizes the number of satisfied clauses.

Randomized Algorithm:

For each variable, let it be 0 or 1 with probability 0.5 and 0.5.

Randomized Algorithm

Max 3-CNF SAT Problem

Input: A 3-CNF Boolean formula of n variables and k clauses,
Where every clause is the OR of 3 literals.

The 3 variables involved in each clause are distinct.

Output: A solution to the variables that maximizes the number of satisfied clauses.

Randomized Algorithm:

For each variable, let it be 0 or 1 with probability 0.5 and 0.5.

Theorem: The algorithm is a polynomial-time $\frac{8}{7}$-approximation randomized algorithm.

Randomized Algorithm

Max 3-CNF SAT Problem

Input: A 3-CNF Boolean formula of n variables and k clauses, Where every clause is the OR of 3 literals.

The 3 variables involved in each clause are distinct.

Output: A solution to the variables that maximizes the number of satisfied clauses.

Randomized Algorithm:

For each variable, let it be 0 or 1 with probability 0.5 and 0.5.

Theorem: The algorithm is a polynomial-time $\frac{8}{7}$-approximation randomized algorithm.

Proof: Consider any clause in the 3-CNF formula.

Example of a clause

$$x_1 \vee \bar{x}_2 \vee \bar{x}_3$$

Randomized Algorithm

Max 3-CNF SAT Problem

Input: A 3-CNF Boolean formula of n variables and k clauses, Where every clause is the OR of 3 literals.

The 3 variables involved in each clause are distinct.

Output: A solution to the variables that maximizes the number of satisfied clauses.

Randomized Algorithm:

For each variable, let it be 0 or 1 with probability 0.5 and 0.5.

Theorem: The algorithm is a polynomial-time $\frac{8}{7}$-approximation randomized algorithm.

Proof: Consider any clause in the 3-CNF formula.

Probability that the clause is satisfied = $\frac{7}{8}$

Example of a clause

$$x_1 \vee \bar{x}_2 \vee \bar{x}_3$$

Randomized Algorithm

Max 3-CNF SAT Problem

Input: A 3-CNF Boolean formula of n variables and k clauses, Where every clause is the OR of 3 literals.

The 3 variables involved in each clause are distinct.

Output: A solution to the variables that maximizes the number of satisfied clauses.

Randomized Algorithm:

For each variable, let it be 0 or 1 with probability 0.5 and 0.5.

Theorem: The algorithm is a polynomial-time $\frac{8}{7}$-approximation randomized algorithm.

Proof: Consider any clause in the 3-CNF formula.

Probability that the clause is satisfied = $\frac{7}{8}$

Expected number of satisfied clauses $C = \frac{7}{8} \cdot k$

Example of a clause

$$x_1 \vee \bar{x}_2 \vee \bar{x}_3$$

Randomized Algorithm

Max 3-CNF SAT Problem

Input: A 3-CNF Boolean formula of n variables and k clauses, Where every clause is the OR of 3 literals.

The 3 variables involved in each clause are distinct.

Output: A solution to the variables that maximizes the number of satisfied clauses.

Randomized Algorithm:

For each variable, let it be 0 or 1 with probability 0.5 and 0.5.

Theorem: The algorithm is a polynomial-time $\frac{8}{7}$-approximation randomized algorithm.

Proof: Consider any clause in the 3-CNF formula.

Example of a clause

$$x_1 \vee \bar{x}_2 \vee \bar{x}_3$$

Probability that the clause is satisfied = $\frac{7}{8}$

Expected number of satisfied clauses $C = \frac{7}{8} \cdot k$

Number of satisfied clauses for an optimal solution $C^* \leq k$

Randomized Algorithm

Max 3-CNF SAT Problem

Input: A 3-CNF Boolean formula of n variables and k clauses,
Where every clause is the OR of 3 literals.

The 3 variables involved in each clause are distinct.

Output: A solution to the variables that maximizes the number of satisfied clauses.

Randomized Algorithm:

For each variable, let it be 0 or 1 with probability 0.5 and 0.5.

Theorem: The algorithm is a polynomial-time $\frac{8}{7}$-approximation randomized algorithm.

Proof: Consider any clause in the 3-CNF formula.

Example of a clause

$$x_1 \vee \bar{x}_2 \vee \bar{x}_3$$

Probability that the clause is satisfied = $\frac{7}{8}$

Expected number of satisfied clauses $C = \frac{7}{8} \cdot k$

Number of satisfied clauses for an optimal solution $C^* \leq k$

$$\frac{C^*}{C} \leq \frac{k}{\frac{7}{8} \cdot k} = \frac{8}{7} \approx 1.14$$

Quiz questions:

1. What is the main idea of the above randomized algorithm?

2. If the number of literals in each clause increase, will the approximation ratio of the randomized algorithm increase or decrease?