Algorithm Analysis and Data Structures

CSCI 7432 - Fall 2022

Approximation Algorithms

Dr. Yao XU

Assistant Professor

Department of Computer Science

Georgia Southern University

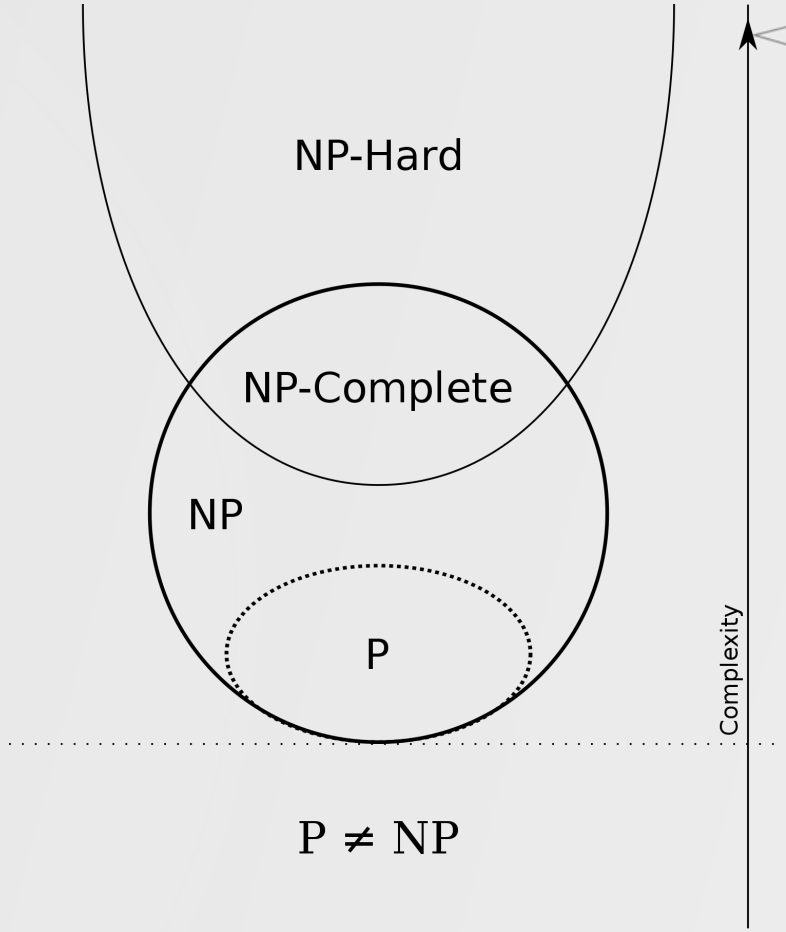
Email: yxu@georgiasouthern.edu

Table of Contents

Approximation Algorithms

Approximation Algorithms

- We call an algorithm that returns a near-optimal solution an ***approximation algorithm*.**
- **Approximation algorithms** must have provable performance.
- We talk about polynomial time **approximation algorithms** for **optimization problems** (solutions associated with values, assumed to be positive)
  - that are **NP-hard** (no **NP** membership)
  - whose corresponding decision versions are **NP-complete**



Approximation Ratio

- Consider an optimization problem P and an instance I of P.
  - An optimal solution with value C^*
  - A computed solution with value C
- Consider an **approximation algorithm** ALG designed for problem P
  - ALG has a $\rho(n)$ ***Approximation ratio*** if for any instance I of size n, $\rho(n) \geq \max\left\{\frac{C^*}{C}, \frac{C}{C^*}\right\}$
  - ALG is then called a $\rho(n)$***-approximation algorithm***.
  - For many problems, we can have approximation algorithms with $\rho(n) = \rho$ being a constant. That is, $\max\left\{\frac{C^*}{C}, \frac{C}{C^*}\right\} \leq \rho$ for any instance.
  - *Note:* $\rho(n) \geq 1$

The Vertex-Cover Problem

Vertex-Cover Problem

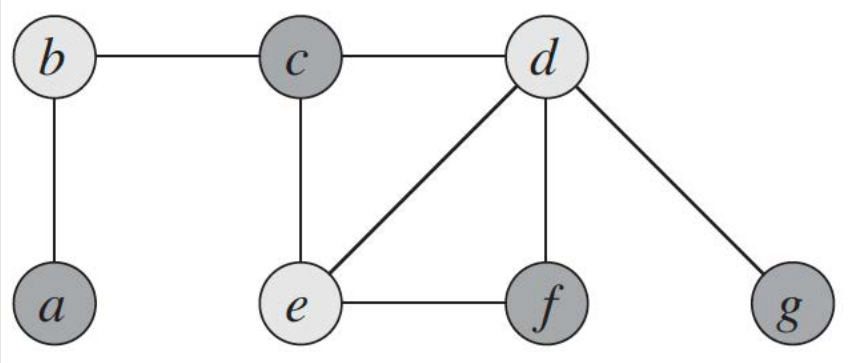
Recall: A ***vertex cover*** of an undirected graph $G = (V, E)$ is a subset $V' \subseteq V$ of vertices such that if $(u, v) \in E$, then $u \in V'$ or $v \in V'$ (or both).

The ***vertex-cover problem***:

- **Input**: An undirected graph $G = (V, E)$.
- **Output**: Find a minimum size vertex cover.

Its decision version **VC** is NP-complete.

Example:



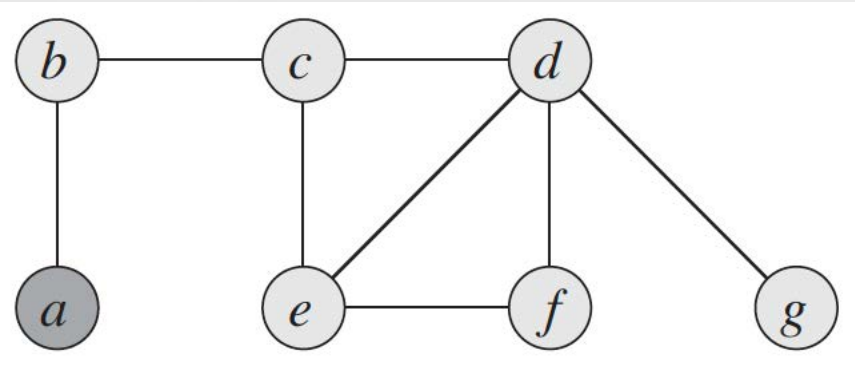
An optimal solution: $\{b, d, e\}$

An Approximation Algorithm

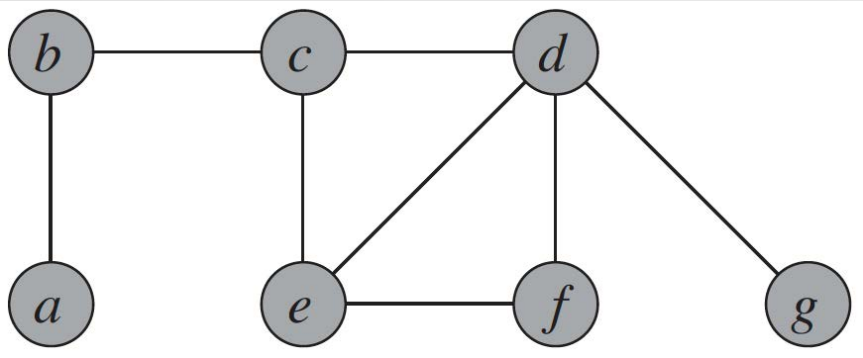
```
APPROX-VERTEX-COVER(G)
1  C = ∅
2  E' = G.E
3  while E' ≠ ∅
4      let (u, v) be an arbitrary edge of E'
5      C = C ∪ {u, v}
6      remove from E' every edge incident on either u or v
7  return C
```

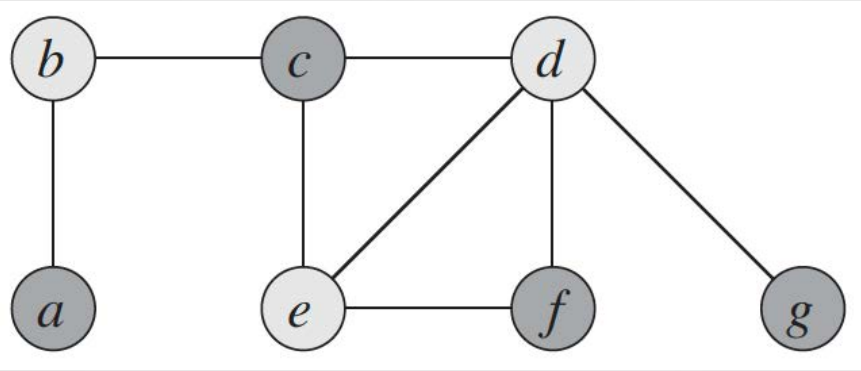
An approximate Solution:

- Edges added: $(b, c), (e, f), (d, g)$
- $C = \{b, c, d, e, f, g\}$



Example:





An optimal solution:

$C^* = \{b, d, e\}$

Polynomial Time Approximation

```
APPROX-VERTEX-COVER(G)
1  C = ∅
2  E' = G.E
3  while E' ≠ ∅
4      let (u, v) be an arbitrary edge of E'
5      C = C ∪ {u, v}
6      remove from E' every edge incident on either u or v
7  return C
```

APPROX-VERTEX-COVER runs in polynomial time:

- Line 2: $O(m)$ time
- **while** loop:
  - Line 4 and 6: every edge is either picked or removed - $O(m)$ time
  - Line 5: every vertex is added to C for at most once - $O(n)$ time

Total running time: $O(n + m)$

2-Approximation Proof

Claim: APPROX-VERTEX-COVER is a 2-approximation algorithm.

That is, the approximation ratio for any instance is $\frac{|C|}{|C^*|} \leq 2$.

Proof. Let A be the set of edges picked by the algorithm. We have

- $|C| = 2|A|$
- $|C^*| \geq |A|$ - Why?

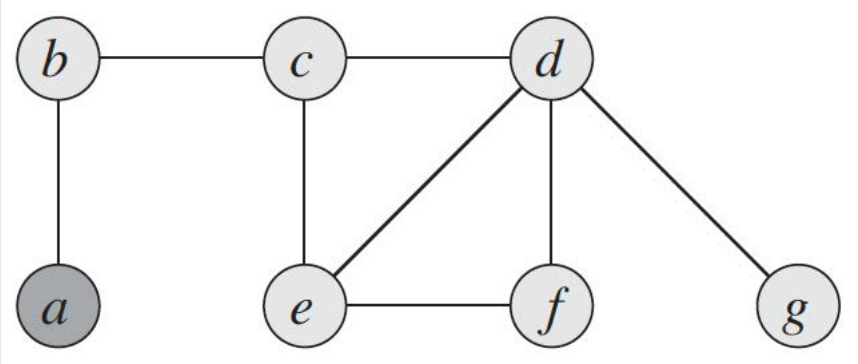
  As no two edges in A share an endpoint.

Therefore,

$$\frac{|C|}{|C^*|} \leq \frac{2|A|}{|A|} = 2$$

□

Example:



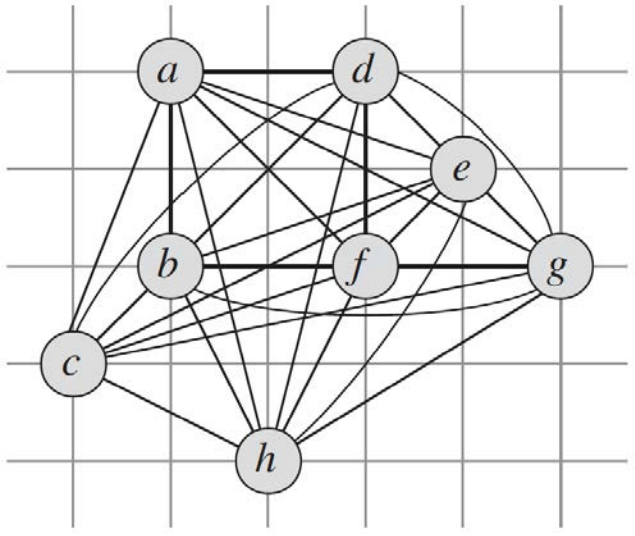
- $A = \{(b, c), (e, f), (d, g)\}$
- $C = \{b, c, d, e, f, g\}$
- $C^* = \{b, d, e\}$

The Traveling-Salesman Problem

Traveling-Salesman Problem (TSP)

- **Input**: A complete undirected graph $G = (V, E)$ with a nonnegative integer cost $c(u, v)$ associated with each edge $(u, v) \in E$.
- **Output:** Find a minimum-cost Hamiltonian cycle.

Example:



Assume ***triangle inequality***: for any three vertices $u, v, w \in V$,

$$c(u, w) \leq c(u, v) + c(v, w).$$

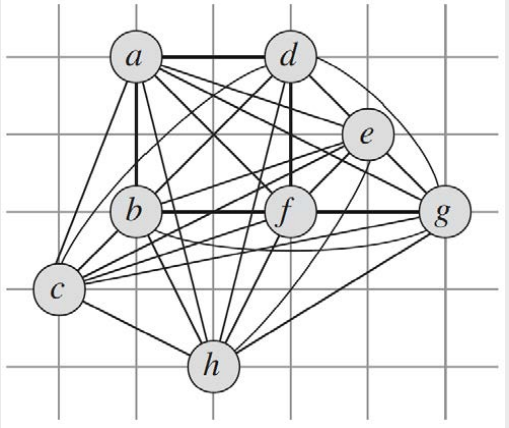
Minimum Spanning Tree (MST)

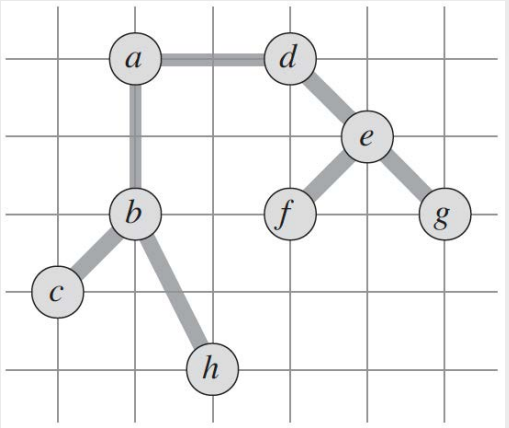
A closely related problem: ***Minimum Spanning Tree (MST)***

- A ***spanning tree*** of an undirected connected graph is its connected acyclic subgraph (i.e., a tree) that contains all the vertices of the graph.
- The ***minimum spanning tree (MST)*** problem:
  - **Input:** An undirected connected graph $G = (V, E)$, with weight $w(u, v)$ on each edge $(u, v) \in E$.
  - **Output:** Find a spanning tree $T = (V, E_T)$ of G such that the total weight of all edges in E_T, $w(T) = \sum_{(u,v)\in E_T} w(u, v)$, is minimized.

Both TSP and MST are to find a subgraph of minimum total weight that contains all the vertices.

Example:



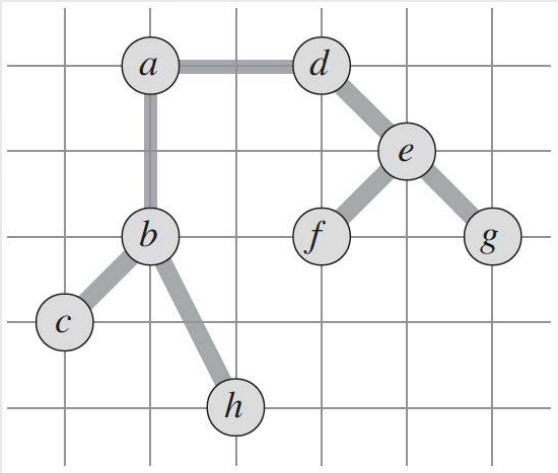


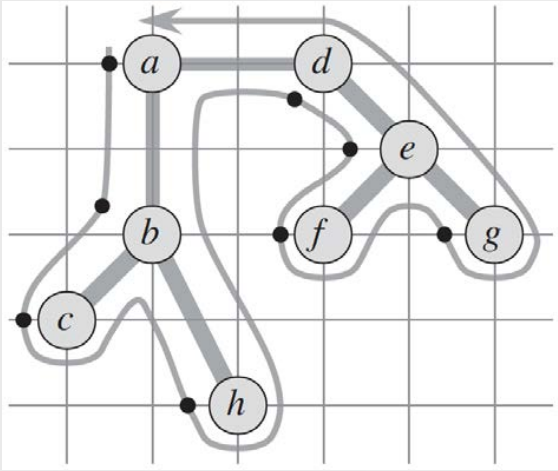
TSP vs. MST

- Find a subgraph

  TSP: a cycle vs. **MST**: a tree
  - minimum total weight
  - contains all the vertices
- **MST** can be solved in $O(m \log n)$ time by:
  - Kruskal's algorithm
  - Prim's algorithm
- To solve **TSP**:
  1. Find a MST T
  2. Transform the tree T into a cycle

Example:





The Traveling-Salesman Problem

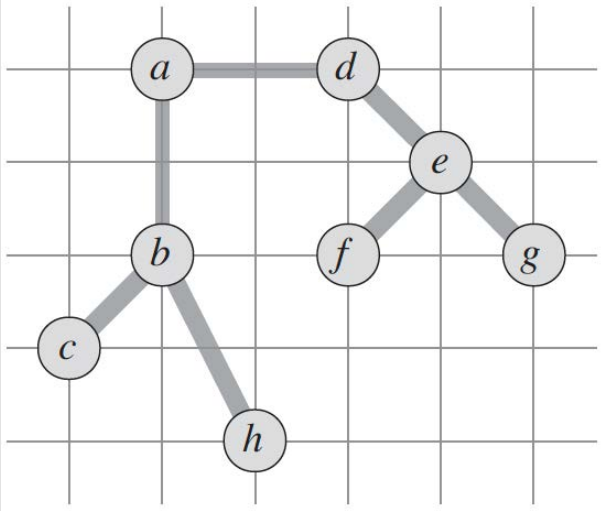
A 2-Approximation Algorithm

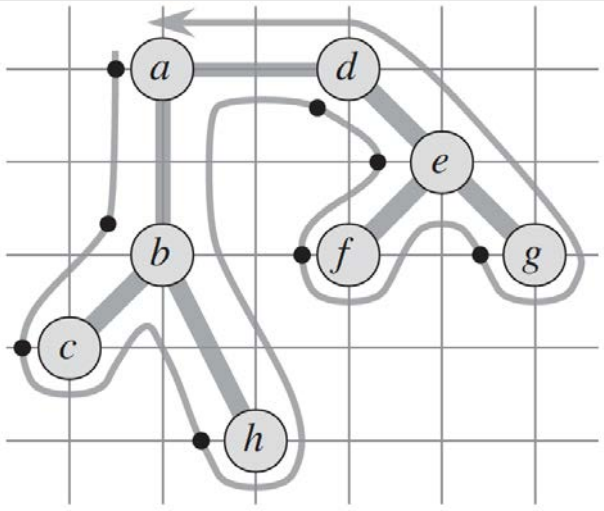
Idea of an Approximation Algorithm (1/2)

Idea of an **approximation algorithm**:

1. Find a MST T
2. Inflate it to be a spanning cycle C' (not simple) - a ***full walk*** of T
3. ***Short-cut*** C' to make it a Hamiltonian cycle (preorder tree walk)

Example:





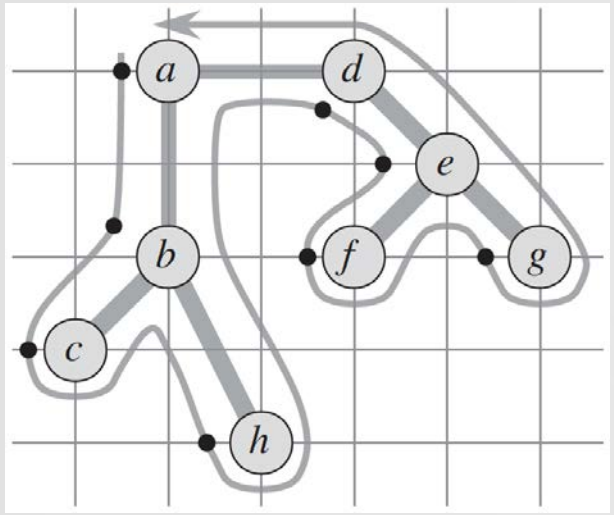

```
PREORDER-TREE-WALK(x)
1  if x ≠ NIL
2      print x.key
3      PREORDER-TREE-WALK(x.left)
4      PREORDER-TREE-WALK(x.right)
```

Idea of an Approximation Algorithm (2/2)

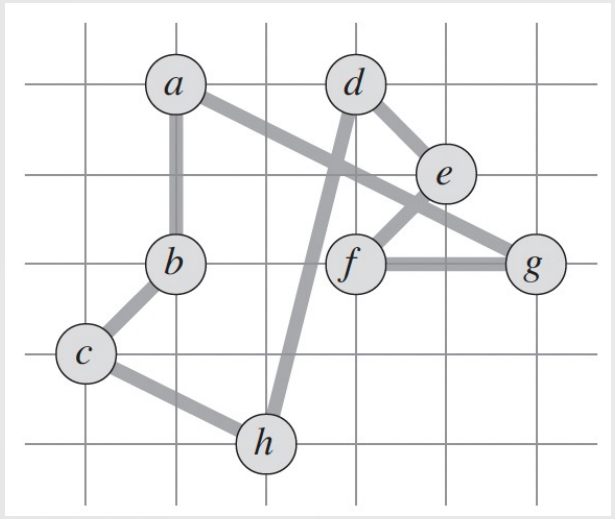
Idea of an **approximation algorithm**:

1. Find a MST T
2. Inflate it to be a spanning cycle C' (not simple) - a ***full walk*** of T
3. ***Short-cut*** C' to make it a Hamiltonian cycle (preorder tree walk)

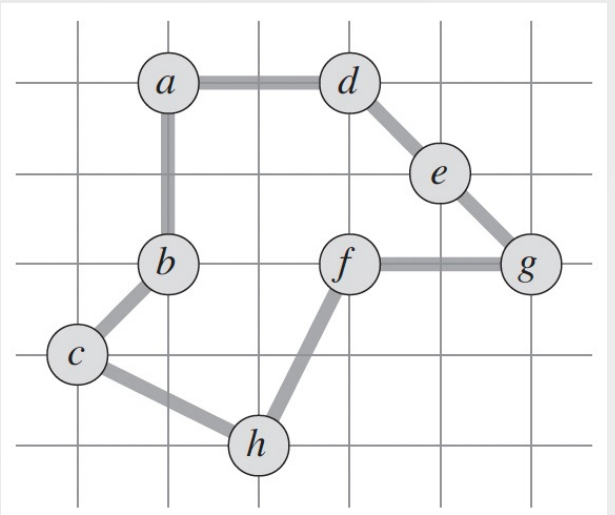
Example (cont'd):



Approximate Solution:



Optimal Solution:



Polynomial Time Approximation

- An approximation algorithm that finds a hamiltonian cycle:

```
APPROX-TSP-TOUR(G, c)
1  select a vertex r ∈ G.V to be a "root" vertex
2  compute a minimum spanning tree T for G from root r
       using MST-PRIM(G, c, r)
3  let H be a list of vertices, ordered according to when they are first visited
       in a preorder tree walk of T
4  return the hamiltonian cycle H
```

APPROX-TSP-TOUR runs in polynomial time.

- MST-Prim takes $O(m + n \log n)$ time.
- Preorder tree walk takes $\Theta(n)$ time.

2-Approximation Proof

Claim: APPROX-TSP-TOUR is a 2-approximation algorithm.

That is, we will prove: its approximation ratio for any instance is $\frac{c(H)}{c(H^*)} \leq 2$, where H^* is an optimal TSP tour.

Proof. Let $c(T)$ be the total cost of edges in the MST T.

- $c(T) \leq c(H^*)$
- W: a full walk of T. Each edge is visited twice $\Longrightarrow c(W) = 2c(T)$
- $c(H) \leq c(W)$ due to triangle inequality.

Therefore,

$$\frac{c(H)}{c(H^*)} \leq \frac{c(W)}{c(T)} = 2$$

□

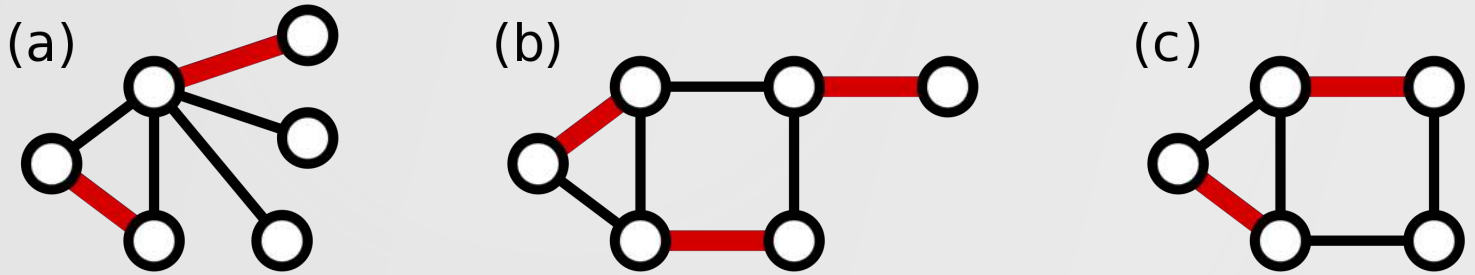
The Traveling-Salesman Problem

A 3/2-Approximation Algorithm

Perfect Matching

- *Recall:* In an undirected graph $G = (V, E)$, a ***matching*** is a subset of edges $M \subseteq E$ s.t. for all $v \in V$, at most one edge of M is incident on v.
  - A ***maximum matching*** is a **matching** of the maximum size.
- A ***perfect matching*** is a **matching** that *matches* all the vertices in V.
  - G can only contain a perfect matching when $|V|$ is even.
  - In the following examples, only the red edges in (b) is a perfect matching.

Examples:

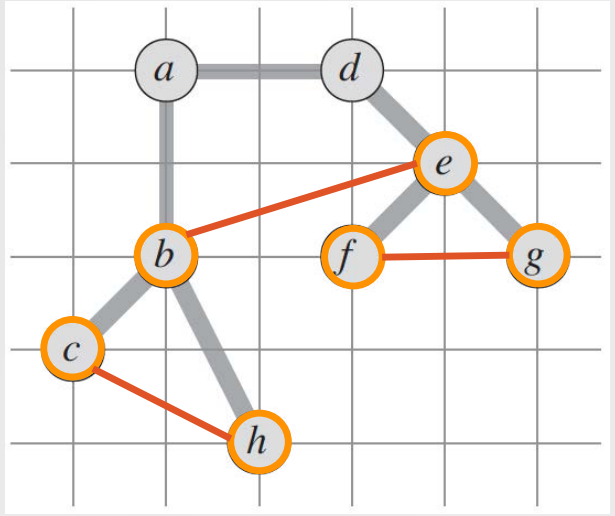


A 3/2-Approximation Algorithm

Idea for improvement (Christofides algorithm):

1. Find a MST T.
2. Let V' be the set of odd-degree vertices in T.
3. Find a minimum-weight **perfect matching** M of subgraph on V'. (Can be solved by **Blossom algorithm** by Edmonds in $O(n^2m)$ time.)
4. Combine T and M – Every vertex in this subgraph has even-degree, so it has a ***Eulerian walk*** W.
   - A walk that starts from a vertex and visits every edge exactly once and returns to the starting vertex. – $c(W) = c(T) + c(M)$
5. **Short-cut** W over repeated vertices to obtain a Hamiltonian cycle H.

Example:



3/2-Approximation Proof (1/2)

Need to prove: The approximation ratio for any instance is $\frac{c(H)}{c(H^*)} \leq \frac{3}{2}$, where H^* is an optimal TSP tour.

- W: a **Eulerian walk** on the subgraph $T + M$
- Note that $c(W) = c(T) + c(M)$
- $c(T) \leq c(H^*)$
- $c(H) \leq c(W)$ due to triangle inequality.
- **Claim:** $c(M) \leq \frac{1}{2}c(H^*)$. (Proved in the next slide)

Together, we have

$$\frac{c(H)}{c(H^*)} \leq \frac{2c(M) + c(M)}{2c(M)} \leq \frac{3}{2}$$

3/2-Approximation Proof (2/2)

Claim: $c(M) \leq \frac{1}{2} c(H^*)$.

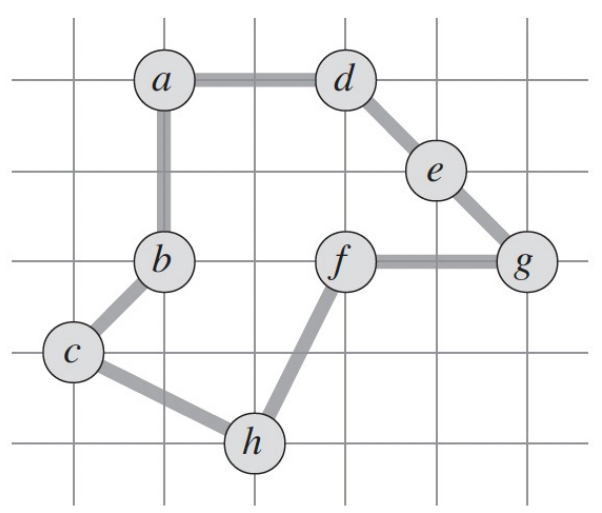
Proof.

- Consider V', the set of odd-degree vertices in T.
- Shortcut H^* over the vertices in V' to obtain a cycle C' that only contains vertices of V'.
- We must have $c(C') \leq c(H^*)$
- C' consists of two disjoint matching M_1 and M_2.
- $c(M) \leq c(M_1)$ and $c(M) \leq c(M_2)$

$\Rightarrow c(M) \leq \frac{1}{2} c(C') \leq \frac{1}{2} c(H^*)$

□

Example of H^*:



The Set-Covering Problem

The Set-Covering Problem

The Set-Covering problem (optimization problem):

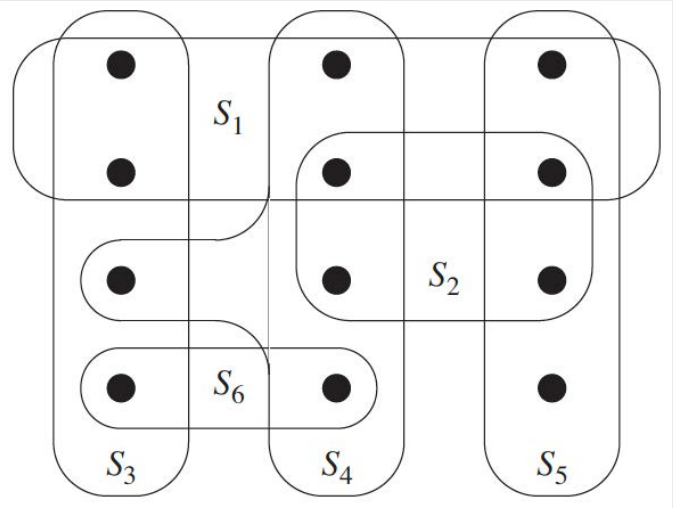
Input:

- A finite set $X = \{x_1, x_2, \cdots, x_n\}$ of n elements
- A family $\mathcal{F}$ of subsets of X s.t. every element of X belongs to at least one subset in $\mathcal{F}$.

Output: A minimum-size collection $\mathcal{C} \subseteq \mathcal{F}$ whose members cover all of X.

The decision problem (**SC**): Given X, $\mathcal{F}$, and an integer k, is there a collection $\mathcal{C} \subseteq \mathcal{F}$ of size at most k whose members cover all of X?

Example:



Optimal solution:

$\mathcal{C} = \{S_3, S_4, S_5\}$

The Set-Covering Problem

NP-Completeness Proof

NP-Completeness Proof of SC (1/2)

To prove that **SC** is NP-complete,

1. **Show that SC is in NP**

   Given $I = \langle X, \mathcal{F}, k \rangle$ and $\mathcal{C}$, verify if $\mathcal{C}$ is a solution to I in polynomial time.

   - Check whether $|\mathcal{C}| \leq k$
   - Check whether every element of X belongs to some subset in $\mathcal{C}$

2. **Show that VC $\leq_P$ SC**

   1) Given a graph $G = (V, E)$, we construct
      - A set $X = E$
      - A $\mathcal{F} = \{S_1, S_2, \cdots, S_n\}$ with each S_i corresponding to a vertex $v_i \in V$
      - Each S_i contains all edges incident to v_i

      All these can be done in polynomial time.

NP-Completeness Proof of SC (2/2)

2. Show that VC $\leq_P$ SC

2) G has a vertex cover of size k iff X can be covered by k subsets in $\mathcal{F}$

Prove "⇒": G has a vertex cover $V' \subseteq V$ of size k

- ⇒ Each edge is covered by a vertex in V'
- ⇒ Every element $e \in E$ is incident to at least one vertex in V'
- ⇒ $\exists \mathcal{C} = \{S_i | v_i \in V'\}$ with $|\mathcal{C}| = k$

Prove "⇐": X can be covered by $\mathcal{C} \subseteq \mathcal{F}$ with $|\mathcal{C}| = k$

- Say $\mathcal{C} = \{S_{j_1}, S_{j_2}, \cdots, S_{j_k}\}$
- ⇒ every edge is incident to some vertex from $\{v_{j_1}, v_{j_2}, \cdots, v_{j_k}\}$
- ⇒ $V' = \{v_{j_1}, v_{j_2}, \cdots, v_{j_k}\}$ is a vertex cover of G

The Set-Covering Problem

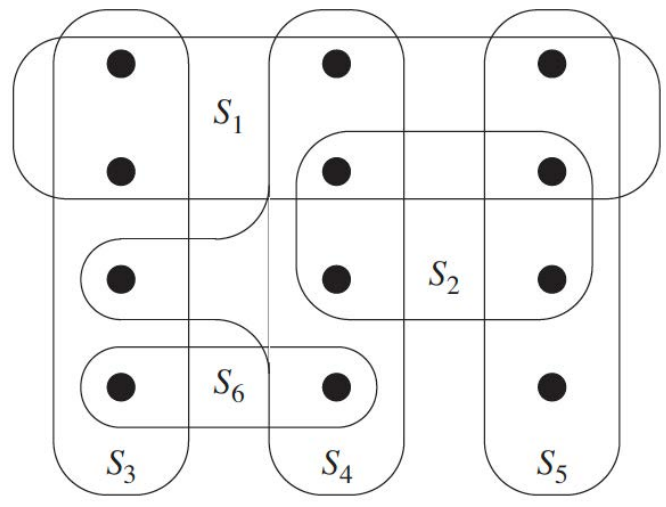
An Approximation Algorithm

A Greedy Approximation Algorithm

- Greedy choice: Pick the set that covers the greatest number of remaining elements that are uncovered.

```
GREEDY-SET-COVER(X, F)
1  U = X
2  C = ∅
3  while U ≠ ∅
4      select an S ∈ F that maximizes |S ∩ U|
5      U = U − S
6      C = C ∪ {S}
7  return C
```

Example:



Running time is polynomial.

- The number of iterations of the **while** loop is bounded by $\min\{n, k\}$
- Line 4: $O(nk)$ time

Greedy solution:

$\mathcal{C} = \{S_1, S_4, S_5, S_3\}$

H_n-Approximation Proof (1/3)

```
GREEDY-SET-COVER(X, F)
1  U = X
2  C = ∅
3  while U ≠ ∅
4      select an S ∈ F that maximizes |S ∩ U|
5      U = U − S
6      C = C ∪ {S}
7  return C
```

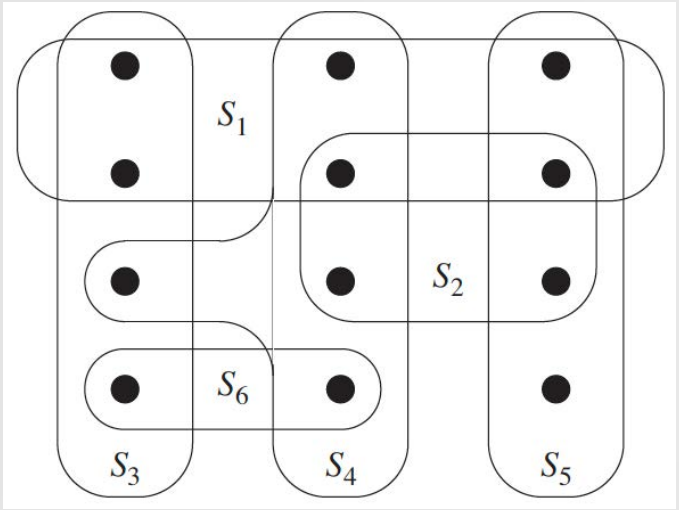
In each iteration of the **while** loop,

- Define $\alpha_j = \frac{1}{|S_j \cap U|}$ for each subset $S_j \in \mathcal{F}$.
- For each element $x \in S_i$, define $c(x) = \alpha_j$.

Suppose $x_1, x_2, \cdots, x_n$ is the order in which the elements are covered.

Observation: $\sum_{i=1}^{n} c(x_i) = |\mathcal{C}|$

Example:



H_n-Approximation Proof (2/3)

- Let $\mathcal{C}^*$ be an optimal solution (minimum-size subset $\mathcal{C} \subseteq \mathcal{F}$ whose members cover all of X).

Lemma: For any $1 \leq i \leq n$, $c(x_i) \leq \frac{|\mathcal{C}^*|}{n-i+1}$.

Proof. Consider the time we have covered $x_1, x_2, \cdots, x_{i-1}$, and we are to pick a set (which will cover x_i for the first time).

- If we pick all sets in $\mathcal{C}^*$ to cover the remaining $n - i + 1$ elements, then the average cost per element is $\frac{|\mathcal{C}^*|}{n-i+1}$.
- Since the greedy choice is to pick the set S_j with $\frac{1}{|S_j \cap U|}$ minimized, we have $c(x_i) \leq \frac{|\mathcal{C}^*|}{n-i+1}$.

H_n-Approximation Proof (3/3)

Need to prove: The approximation ratio of GREEDY-SET-COVER for any instance is $\frac{|\mathcal{C}|}{|\mathcal{C}^*|} \leq H_n$.

Lemma: For any $1 \leq i \leq n$, $c(x_i) \leq \frac{|\mathcal{C}^*|}{n-i+1}$.

Using this Lemma, we have

$$|\mathcal{C}| = \sum_{i=1}^{n} c(x_i) \leq \sum_{i=1}^{n} \frac{|\mathcal{C}^*|}{n-i+1} = H_n \cdot |\mathcal{C}^*|,$$

Recall: the Harmonic number $H(n) = \sum_{i=1}^{n} \frac{1}{i} = \ln n + \gamma$, where $\gamma \approx 0.577\cdots$

$$\Rightarrow \frac{|\mathcal{C}|}{|\mathcal{C}^*|} \leq H_n \leq \ln n + 1.$$

□

Thank you!
Questions?