14. Approximation, Vertex Cover, and TSP

CPSC 535

Kevin A. Wortman

CALIFORNIA STATE UNIVERSITY
FULLERTON

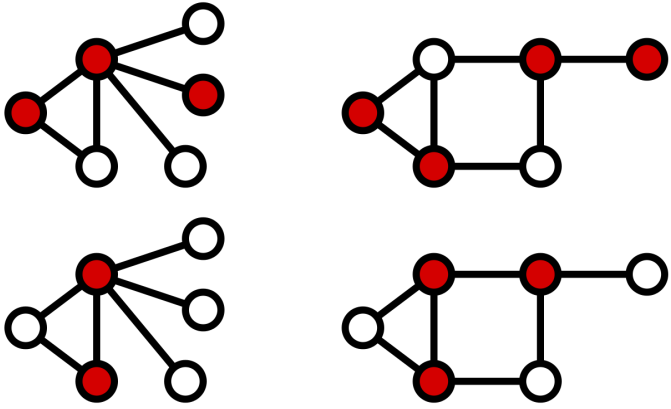
Recap: Vertex Cover Problem

vertex cover problem
input: undirected graph $G = (V, E)$
output: set of vertices $C \subseteq V$, of minimal size $|C|$, such that every edge in E is incident on at least one vertex in C

2-approximate vertex cover problem
input: undirected graph $G = (V, E)$
output: set of vertices $C \subseteq V$, such that every edge in E is incident on at least one vertex in C, and $|C| \leq 2|C^*|$ where C^* is a minimal vertex cover for G

See Wiki page: https://en.wikipedia.org/wiki/Vertex_cover

Recap: Vertex Cover Example

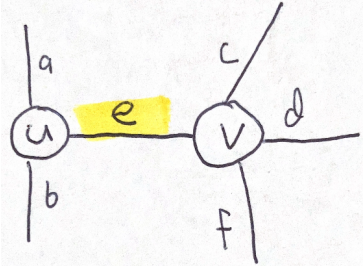


Images credit: Wikipedia user Miym, CC BY-SA 3.0, https://commons.wikimedia.org/wiki/File:Vertex-cover.svg,

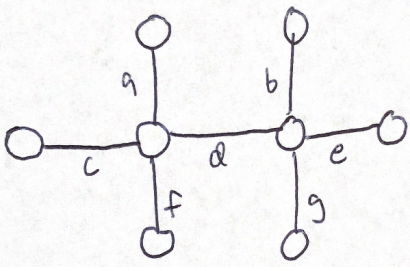
https://commons.wikimedia.org/wiki/File:Minimum-vertex-cover.svg

Recap: A Greedy Approximation Algorithm

- every edge $e = (u, v)$ needs both $u \in C$ and $v \in C$
- so grab an edge $e = (u, v)$ and include u and v in C
- every other edge touching u or v is now covered, so eliminate them
- continue until every edge is either grabbed or eliminated



Recap: Greedy Can Be Suboptimal



optimal edge choices: d

suboptimal edge choices: a, b (and others)

2-Approximate Vertex Cover Pseudocode

```
1: function APPROX-VERTEX-COVER(G = (V, E))
2:     C = ∅
3:     T = E                                   ▷ T = edges that may be taken
4:     while T ≠ ∅ do
5:         Let e = (u, v) be an arbitrary edge in T
6:         C = C ∪ {u, v}
7:         Remove from T any edge f that is incident on u or v
8:     end while
9:     return C
10: end function
```

Efficiency Analysis: $O(m+n)$ time (assuming efficient data structures for G, T, and C)

Recap: Performance Ratio for Minimization Problem

For **minimization** problem: if optimal quality is C^* and alg. produces quality C, by definition $C^* \leq C$ and define

$$\rho(n) = \frac{C}{C^*}$$

Recall 3-approx. vertex cover: # colors $\leq 3k^*$

Proof Strategy for 2-Approximate Vertex Cover

- **Want:** prove performance ratio 2
- **Need:** $\frac{C}{C^\star} \leq 2$ for all inputs
- **Problem:** optimal solution $C^\star$ is unknowable
- **Solution:**
  - identify a **bridge** variable b
  - prove that $b \leq C^\star$
  - prove that $C \leq 2b$
  - by substitution:

$$\frac{C}{C^\star} \leq \frac{C}{(b)} \leq \frac{(2b)}{b} = 2$$

Vertex Cover Performance Ratio

Lemma: APPROX-VERTEX-COVER is a 2-approximation algorithm.
Need: for any $G, |C| \leq 2|C^*|$
Proof sketch:

- **bridge variable:** let A be the set of edges chosen inside the **while** loop
- will bound $|C|$ and $|C^*|$ both in terms of $|A|$

Vertex Cover Performance Ratio (cont'd)

- **(1)** $|C^\star|$ vs. $|A|$
- $C^\star$ is a vertex cover, so for every edge $(u, v) \in A$, we must have $u \in C^\star$ and/or $v \in C^\star$
- the "Remove from T" step guarantees that, after (u, v) is chosen, no other edge incident on u or v will ever be chosen and added to A
- $\Rightarrow$ each vertex $x \in C^\star$ covers *at most one* edge in A
- $\Rightarrow |C^\star| \geq |A|$

Vertex Cover Performance Ratio (cont'd)

- **(2)** $|C|$ vs. $|A|$
- the $C = C \cup \{u, v\}$ step inserts 2 vertices into C
- due to the same "Remove from T" logic, neither u nor v was already in C
- $\Rightarrow |C| = 2|A|$ (note exact equality)
- **combining (1) and (2)**

$$\begin{aligned} |C| &= 2|A| \\ &\leq 2(|C^*|) \end{aligned}$$

- therefore $|C| \leq 2|C^*|$

Commentary on this Proof

- optimal set cover $C^\star$ is opaque
- we analysts cannot know which vertices will be in a $C^\star$
- the algorithm doesn't know what $C^\star$ is either
- all we do know is that, due to the definition of vertex cover, and the logic of our algorithm,

$$\#\text{ vertices in optimal cover} \geq \#\text{ iterations } \textbf{while} \text{ loop}$$

- and, due to algorithm logic,

$$\#\text{ vertices chosen for approx. cover} = 2 \times \#\text{ iterations } \textbf{while} \text{ loop}$$

General Approximation Ratio Proof Strategy

Minimization Problem

1. Identify **bridge variable** b
2. Prove $k^\star b \leq C^\star$
3. Prove $C \leq kb$
4. Show by substitution that

$$\frac{C}{C^\star} \leq \frac{(kb)}{(k^\star b)} = \frac{k}{k^\star}$$

5. Conclude approximation ratio is $\frac{k}{k^\star}$

Maximization Problem

1. Identify **bridge variable** b
2. Prove $k^\star b \leq C^\star$
3. Prove $C \leq kb$
4. Show by substitution that

$$\frac{C^\star}{C} \geq \frac{(k^\star b)}{(kb)} = \frac{k^\star}{k}$$

5. Conclude approximation ratio is $\frac{k^\star}{k}$

TSP

traveling salesperson problem (TSP)
input: a complete undirected graph $G = (V, E)$ where each edge has weight $w(e) \geq 0$
output: a sequence of vertices H forming a Hamiltonian cycle, minimizing total edge weight

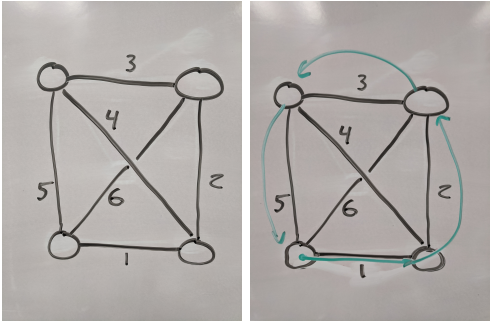
Recall:

- *Cycle:* path that starts and ends at same vertex
- *Hamiltonian:* visits each vertex exactly once
- every complete graph contains some Hamiltonian cycle

Bad news:

- TSP is *NP*-complete; if $P \neq NP$, no polynomial-time optimization algorithm
- TSP is also *APX*-complete; if $P \neq NP$, no PTAS

TSP



Triangle Inequality

Triangle inequality in general: for distance function d and sites a, b, c,

$$d(a, c) \leq d(a, b) + d(b, c)$$

$\Rightarrow$ direct path $a \to c$ always cheaper than two-step path $a \to b \to c$ (or tied)

Triangle inequality in a complete graph: for vertices x, y, z and edge weights w,

$$w(x, z) \leq w(x, y) + w(y, z)$$

$\Rightarrow$ same intuition; adding an intermediate step is never a shortcut
$\Rightarrow$ automatically holds for Euclidean graphs

TSP with Triangle Inequality (TSPTI)

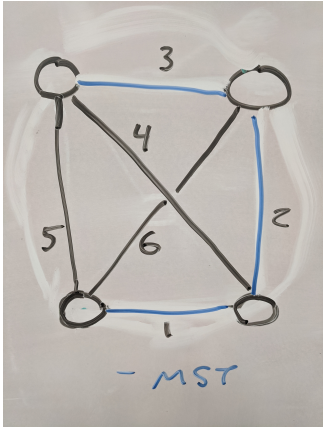
input: a complete undirected graph $G = (V, E)$ where each edge has weight $w(e) \geq 0$; and for any $x, y, z \in V$, $w(x, z) \leq w(x, y) + w(y, z)$
output: (same as conventional TSP)

- **renegotiating** TSP
- different problem; *NP*-completeness and *APX*-completeness proofs may not apply
- less-general problem
- probably still relevant to practical applications of TSP

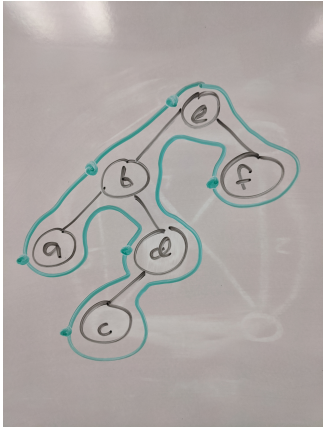
TSPTI Approximation Algorithm Idea

- need a structure that can lower-bound an optimal cycle $H^\star$ and upper-bound our approximate cycle H
- *minimum spanning tree* features
  - minimizes weight of chosen edges
  - connects all vertices
  - can be computed fast
- but an MST is not a Hamiltonian cycle; MST is acyclic, for one thing
- *Euler tour:* cycle around a tree; preorder, inorder, postorder
- build an MST; perform preorder traversal; treat that vertex order as Hamiltonian cycle

Review: Minimum Spanning Trees (MST)



Review: Preorder Traversal

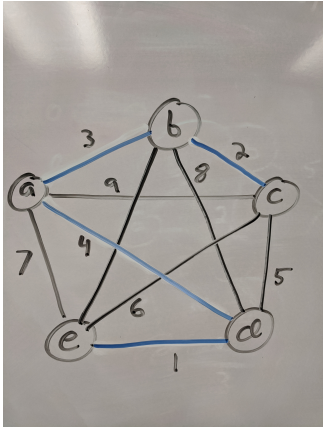


Approximate TSPTI Pseudocode

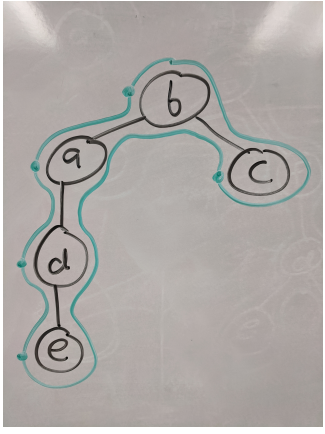
1: **function** APPROX-TSPTI($G = (V, E), w$)
2: $T = PRIM - MST(G, w)$
3: H = empty sequence of vertices
4: **for** vertex v in preorder traversal of tree T **do**
5: $H.ADDBACK(v)$
6: **end for**
7: $H.ADDBACK(H[0])$
8: **return** H
9: **end function**

Analysis: Prim's algorithm takes $O(m + n \log n)$ (w/ Fibonacci heap), traversal takes $O(m + n)$, total $O(m + n \log n)$ time

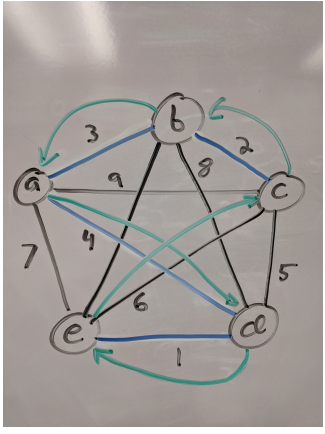
Approximate TSPTI: MST



Approximate TSPTI: Preorder Traversal



Approximate TSPTI: Solution



TSPTI Performance

Lemma: APPROX-TSPTI is a 2-approximation algorithm
Proof Sketch:

- let $H^\star$ be an optimal Hamiltonian cycle for G
- (1) every spanning tree is one edge short of a cycle; and weights are nonnegative; so the weight of our tree T obeys $w(T) \leq w(H^\star)$
- (2) a *full tour* W is the sequence of vertices in both a preorder and postorder tour, and has weight $w(W) = 2w(T)$
- (3) combining (1) and (2), $w(W) \leq 2w(H^\star)$
- (4) our H is like W with some vertices removed, so $w(H) \leq w(W)$
- combining (3) and (4),

$$w(H) \leq w(W) \leq 2w(H^\star)$$

Summary

Vertex cover:

- NP-complete
- exact exhaustive search alg. takes $\Theta(2^n m)$ time
- 2-approximate alg. takes $\Theta(n+m)$ time

TSP:

- NP-complete
- exact exhaustive search alg. takes $\Theta(n!)$ or $\Theta(2^m(n+m))$ time
- 2-approximate algorithm takes $O(m + n \log n)$ time