Approximation Algorithms

Textbook Reading

Algorithms: *Special Topics*:

Chapter 7

- 7.1 Traveling Salesman Problem
- 7.3 The Steiner Tree Problem

Approximation Algorithms

- Even though it would be ideal to find an optimal solution to a given optimization problem, in practice many such problems are NP-hard and no known polynomial time algorithm for solving the problem exists in the worst case.
- For these problems we are often satisfied with suitably good approximations to the optimal solution.
- There are various ways to measure the degree of approximation of a solution to the optimal solution.
- We restrict attention here to approximations that are within multiplicative factors of the optimal solution.

$\boldsymbol{\rho(n)}$-*approximation*

Let $C^*(n)$ denote the value of an optimal solution to a given optimization problem for an input of size *n* (we assume $C^*(n) >$ 0). If the problem is a maximization problem, then for a given real-valued positive function ρ(*n*), we say that a solution $C(n)$ is a $\boldsymbol{\rho(n)}$***-approximation*** to $C^*(n)$ if

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

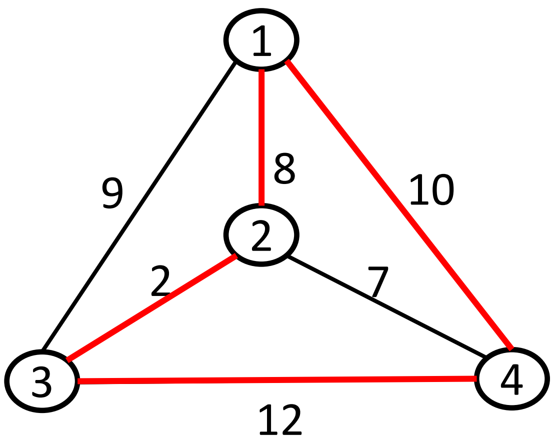
If the problem is a minimization problem, then $C(n)$ is a $\boldsymbol{\rho(n)}$ **-*approximation*** to $C^*(n)$ if

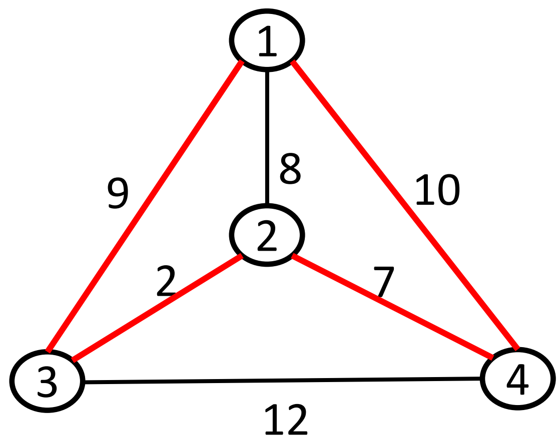
$$\frac{C(n)}{C^*(n)} \leq \rho(n)$$

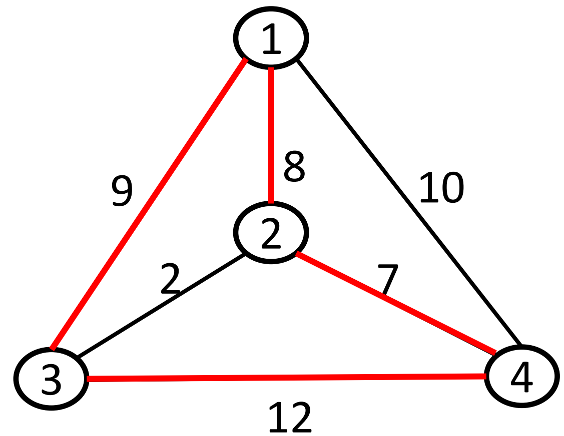
Traveling Salesman Problem

- The traveling salesman problem (TSP) is equivalent to finding a minimum weight Hamiltonian cycle in a weighted complete graph on *n* vertices.

Example TSP Tours







12341 8+2+12+10 = 32
14321 10+12+2+8 = 32

13241 9+2+7+10 = 28
14321 10+7+2+9 = 28

12431 8+7+12+9 = 36
13421 9+12+7+8 = 36

There are 6 Hamiltonian cycles with initial vertex 1.
There are 3 when paired with cycle traversed in reverse.
Minimum weight Hamiltonian cycle, i.e., minimum TSP tour, has weight 28.

Implementing Weighted Graph

- The weighted complete graph can be represented by weighted adjacency matrix (or cost matrix) where entry ij contains the weight (or cost) $\omega(ij)$ of edge ij
- Example

$$\begin{pmatrix}
0 & 19 & 14 & 16 & 17 & 13 & 18 & 20 \\
19 & 0 & 17 & 12 & 11 & 19 & 15 & 15 \\
14 & 17 & 0 & 18 & 16 & 12 & 16 & 16 \\
16 & 12 & 18 & 0 & 10 & 20 & 11 & 20 \\
17 & 11 & 16 & 10 & 0 & 16 & 10 & 14 \\
13 & 19 & 12 & 20 & 16 & 0 & 18 & 15 \\
18 & 15 & 16 & 11 & 10 & 18 & 0 & 17 \\
20 & 15 & 16 & 20 & 14 & 15 & 17 & 0
\end{pmatrix}$$

Triangle Inequality

We obtain a 2-approximation in the special case when the **triangle inequality** holds, i.e., for any three vertices a, b, c

$$\omega(ac) \leq \omega(ab) + \omega(bc).$$

2-approximation algorithm

1. Compute a minimum spanning tree (MST) T using Kruskal's or Prims algorithm.
2. Choosing any vertex v_1 as the root of T, perform preorder traversal of T to obtain preorder sequence v_1 , v_2 , ..., v_n.

The cycle $v_1 v_2 \dots v_n v_1$ is a Hamiltonian cycle C whose weight is at most twice the weight of a minimum-weight Hamiltonian cycle C^*, i.e., optimal traveling salesman tour.

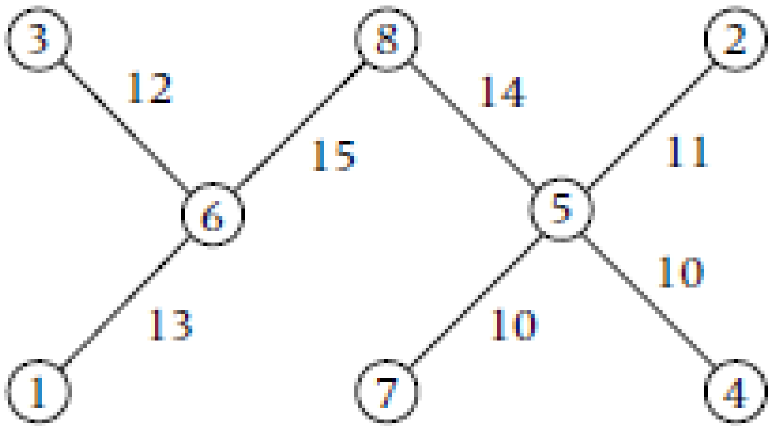
Action for an example

Consider weighted complete graph with weighted adjacency matrix given by

$$\begin{pmatrix} 0 & 19 & 14 & 16 & 17 & 13 & 18 & 20 \\ 19 & 0 & 17 & 12 & 11 & 19 & 15 & 15 \\ 14 & 17 & 0 & 18 & 16 & 12 & 16 & 16 \\ 16 & 12 & 18 & 0 & 10 & 20 & 11 & 20 \\ 17 & 11 & 16 & 10 & 0 & 16 & 10 & 14 \\ 13 & 19 & 12 & 20 & 16 & 0 & 18 & 15 \\ 18 & 15 & 16 & 11 & 10 & 18 & 0 & 17 \\ 20 & 15 & 16 & 20 & 14 & 15 & 17 & 0 \end{pmatrix}$$

Action cont'd

Compute minimum spanning tree T:

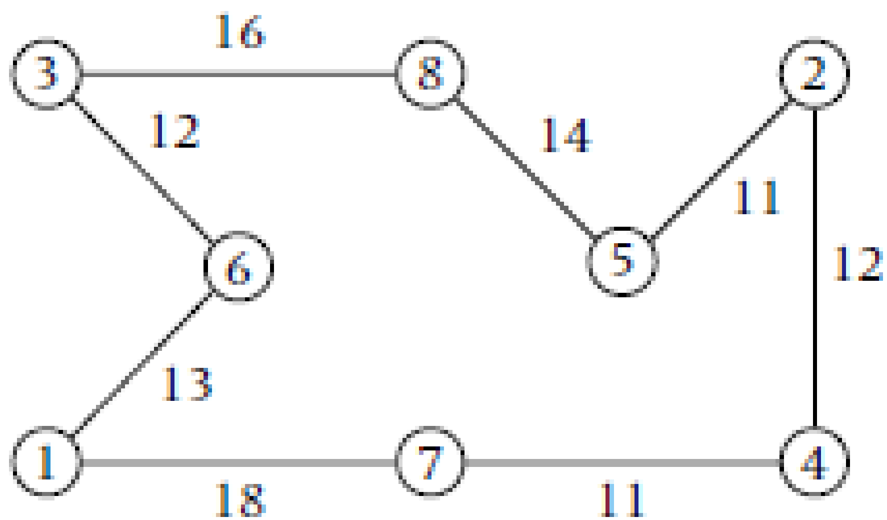


Preorder Traversal of T: 1 6 3 8 5 2 4 7

2-approximation of TSP tour

Hamilton cycle *C* determined by preorder traversal

1 6 3 8 5 2 4 7



Weight of *C* less than twice the weight of optimal tour *C**.

Correctness

- By the triangle inequality the weight of the edge vw in the Hamiltonian cycle C is no greater than the sum of the weights the edges traversed in the preorder traversal as it goes from v to w.
- Since an edge is traversed exactly twice (once in each "direction") during a preorder traversal, it follows that $\omega(C) \leq 2\omega(T)$.
- Since T has minimum weight over all spanning trees and a minimum-cost tour (minimum-weight Hamiltonian cycle) C^* must contains a spanning tree, it follows that $\omega(T) \leq \omega(C^*)$.
- Combining the two inequalities we have

$$\omega(C) \leq 2\omega(T) \leq 2\omega(C^*).$$

Q.E.D.

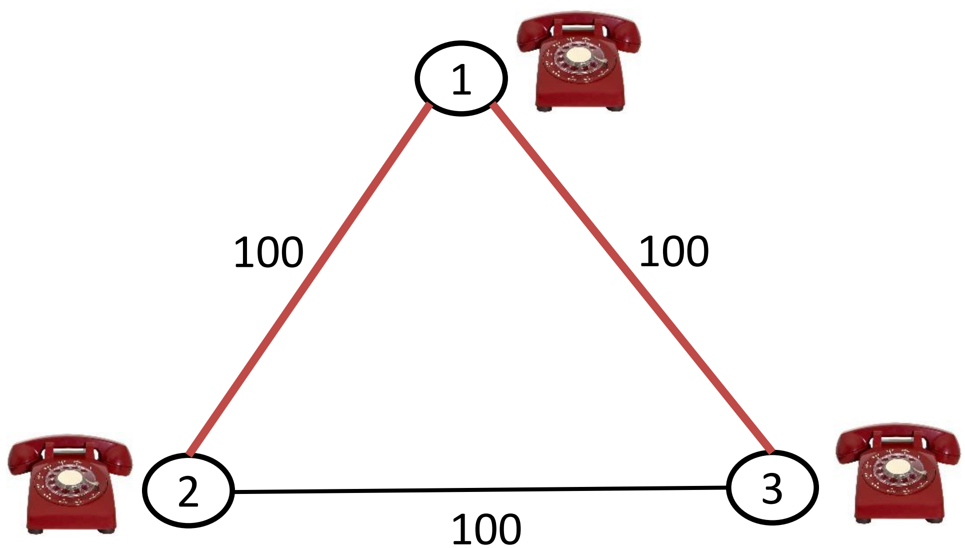
PSN

Show that the triangle inequality is necessary. In particular, show that the problem of finding a 2-approximation to a TSP tour (minimum-weight Hamiltonian cycle) is NP-hard.

Hint: Show that it can be used to find a Hamiltonian cycle in an (unweighted) graph.

Steiner Tree Problem Motivation with AT&T pricing model

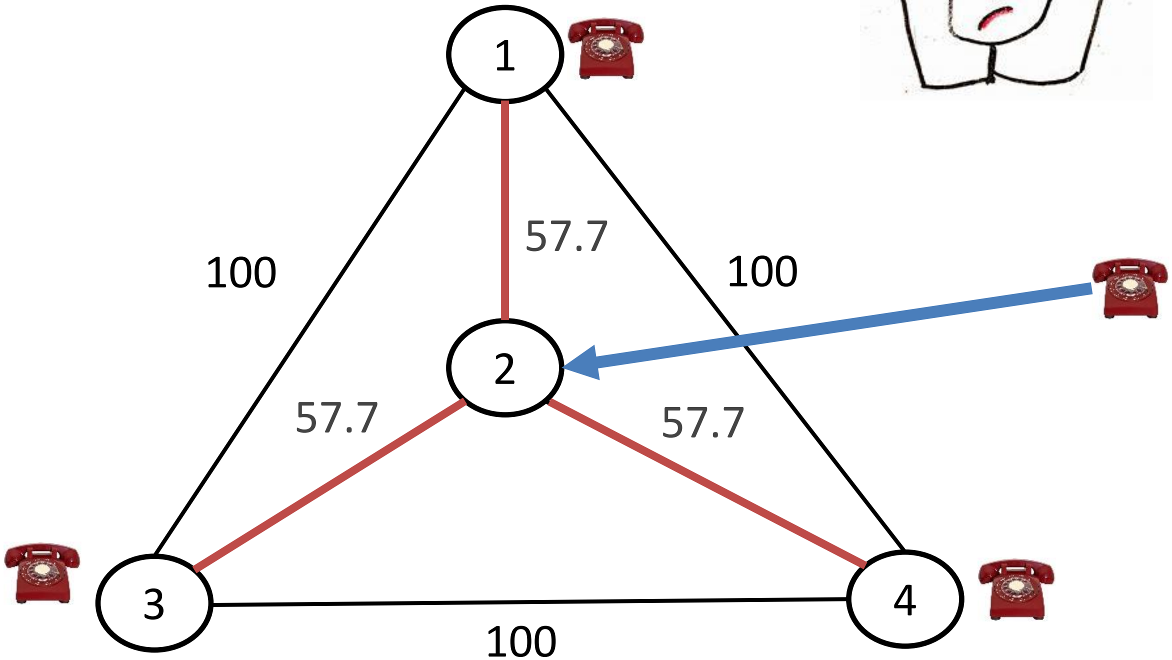
Charge is based on the minimum cost of lines to connect 3 phones, i.e., based on the weight of a minimum spanning tree in the graph whose vertex set corresponds to the phones and whose edges are weighted with the distance between phones.



In above example Cost would be $200

Cheaper to connect extra phone!

MST has smaller weight with extra phone!

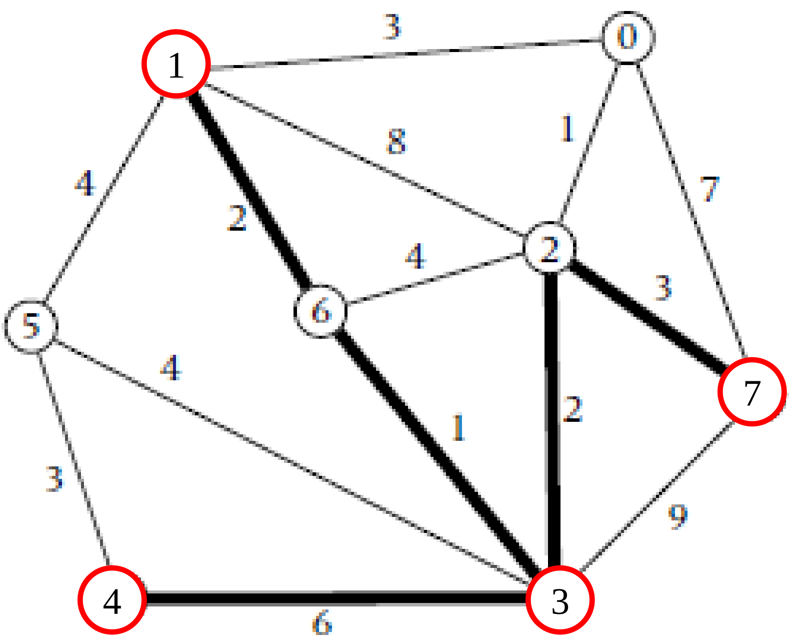


MST now has weight \$173.10 < \$200

Steiner Tree Definition

- Let $G = (V, E)$ be a graph on node (vertex) set V and edge set E representing a network, and suppose w is a positive real weighting of the edges.
- Given a subset S of k nodes in the graph G, a **Steiner tree** for S is a minimum weight subtree of G that contains all the nodes in S.
- Note that a Steiner tree for $S = V$ is a minimum spanning tree and a Steiner tree for $S = \{u,v\}$ is a shortest path joining u and v.

Steiner Tree in Example Graph



A Steiner tree for $S = \{1,3,4,7\}$

Steiner Tree Problem is NP-Hard

- Given *S* and a positive real number *k*, the problem of determining whether there exists a tree containing the nodes of *S* having minimum weight not great than *k* is NP-complete.
- Thus, the Steiner tree problem is NP-hard.

2-Approximation to Steiner Tree

- We now describe a 2-approximation algorithm for the Steiner tree problem.
- The algorithm we describe is actually slightly better than a 2-approximation, yielding a tree whose cost is at most $2 - 2/k$ times the cost of a Steiner tree.
- Our algorithm involves three stages.

Stage 1

- Compute the matrix *D* of distances between every pair of vertices *u*, *v* in *S*.
- The matrix *D* can be computed in time $O(kn^2)$ by applying Dijkstra's shortest path algorithm *k* times, once with each node in *S* as the root.
- *D* represents the weighted adjacency matrix of a weighted graph $H = (S,E_H)$ on node set *S* and edge set E_H, where E_H consists of all pairs of nodes $\{u,v\}$ from *S,* i.e., *H* is a complete graph on *S*, and where each edge $\{u,v\}$ is weighted with the distance $D(u,v)$.

Stage 2

- Compute a minimum spanning tree T_H in the graph *H.*
- This can be done in time O(k^2) using Prim's minimum spanning tree algorithm.

Stage 3

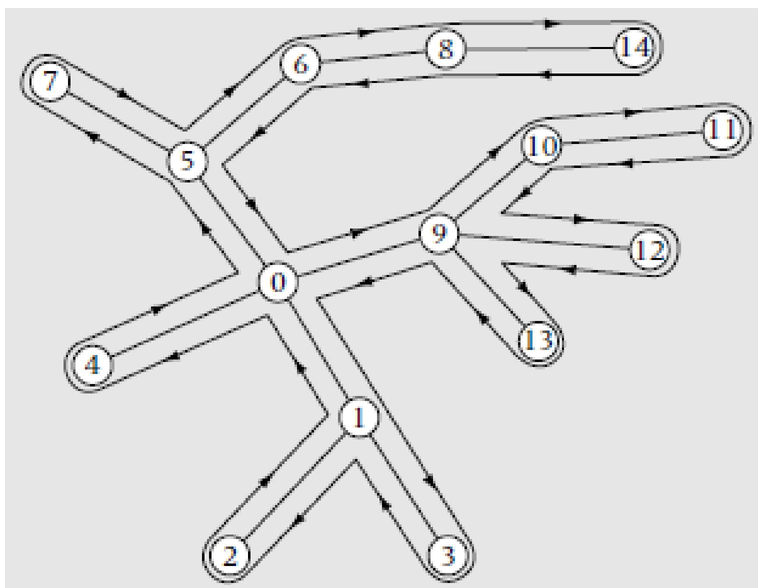
- Construct a tree T_G in G by joining a pair of vertices x and y from S with a shortest path from x to y, whenever xy is an edge of T_H , in the following manner. Let $x_1y_1, \ldots, x_{k\text{-}1}y_{k\text{-}1}$ denote the edges of T_H and let $P_1, \ldots, P_{k\text{-}1}$ be shortest paths from x_i to y_i, $i = 1, \ldots, k - 1$.
- We initialize T_G to be the path P_1, and successively add paths to T_G maintaining a forest at each stage.
- Suppose P_i contains at least two nodes that belong to the current T_G, and let a and b denote the first and last such nodes encountered, respectively, when traversing P_i from x_i to y_i. Then add the subpaths of P_i from x_i to a and from b to y_i to T_G.
- On the other hand, if P_i contains at most one node from T_G then add the entire path P_i to T_G .

2-approximation

Theorem. The cost of T_G generated by our algorithm is no greater than twice the cost of a Steiner Tree T^*.

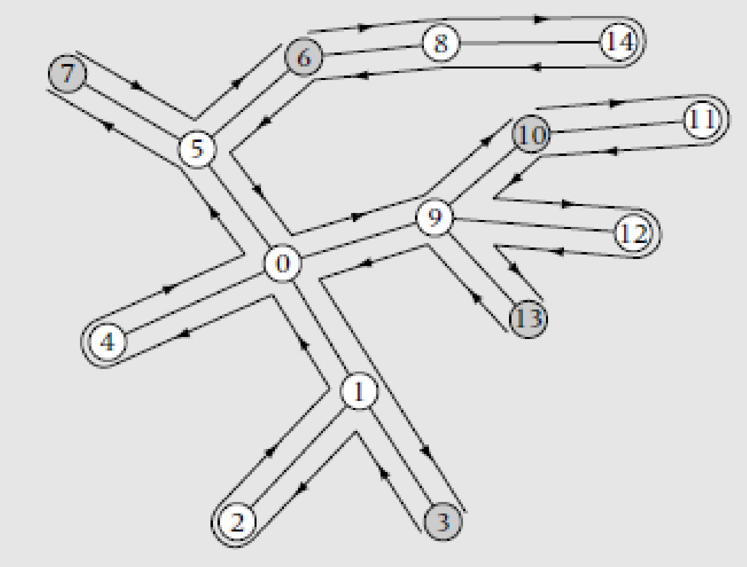
Proof

Consider the closed walk W around the tree T^* consisting of a sequence of nodes $v_0\ e_1\ v_1\ \ldots,\ v_{2n-2}\ v_{2n-1} = v_0$,
where $e_i = \{v_{i-1}, v_i\}$, $i = 1, \ldots, 2n-2$.
Note that every edge is traversed twice, once in each direction.



Proof cont'd

Let $s_1, s_2, \dots, s_k$ denote the nodes from S listed in the order that they are first encountered W. Decompose W decomposes into k subwalks $W_1, W_2, \dots, W_k$, where W_i is the subwalk joining s_i and $s_{i+1}, i = 1, \dots, k$, where $s_{k+1} = s_1$.



W_1: 3, 1, 2, 1, 0, 4, 0, 5, 7

W_2: 7, 5, 6

W_3: 6, 8, 14, 8, 6, 5, 0, 9, 10

W_4: 10, 11, 10, 9, 12, 9, 13

W_5: 13, 9, 0, 1, 3

Observations

Let C be the Hamiltonian cycle $s_1 s_2 \ldots s_k s_1$ in the complete graph H.

Observation 1. $\sum_{i=1}^{k} \omega(W_i) = 2\omega(T^*)$

Observation 2. $\omega(T_G) \leq D(T_H)$

Observation 3. $D(s_i s_{i+1}) \leq \omega(W_i)$

Observation 4. $D(C) \leq \sum_{i=1}^{k} \omega(W_i)$

Observation 5. $D(T_H) \leq D(C)$

PSN. Verify these observations.

Completing the Proof

$\omega(T_G) \leq D(T_H)$ (by Observation 2)

$< D(C)$ (by Observation 5)

$\leq \sum_{i=1}^{k} \omega(W_i)$ (by Observation 4)

$= 2\omega(T^*)$ (by Observation 1)

Q.E.D

Slightly stronger result

It can be shown that our algorithm achieves a $(2 - 2/k)$-approximation.

Approximation Joke

What do you call a snake that
is approximately 3.14 feet long?

A πthon