Module 13
Approximation Algorithms

S. Lakshmivarahan
School of Computer Science
University of Oklahoma
USA-73019
Varahan@ou.edu

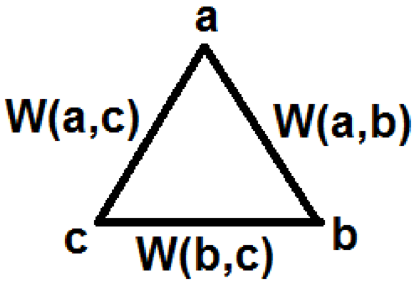
Approximation Algorithms

- It is now well established that the Travelling Salesman Problem (TSP) is NP-hard
- Given this challenge, there have been attempts to solve this problem approximately using ideas that only take polynomial time
- These are called polynomial time approximations

Approximation Algorithms

- We now describe a polynomial time approximate algorithm to find a tour in the special case of TSP: namely, the Euclidean version of the problem
- Let G=(V, E, W) be a weighted graph, with the weights satisfying a special condition: If nodes {a,b,c} form a triangle, then

$$W(a,c) \leq W(a,b) + W(b,c) \rightarrow 1$$

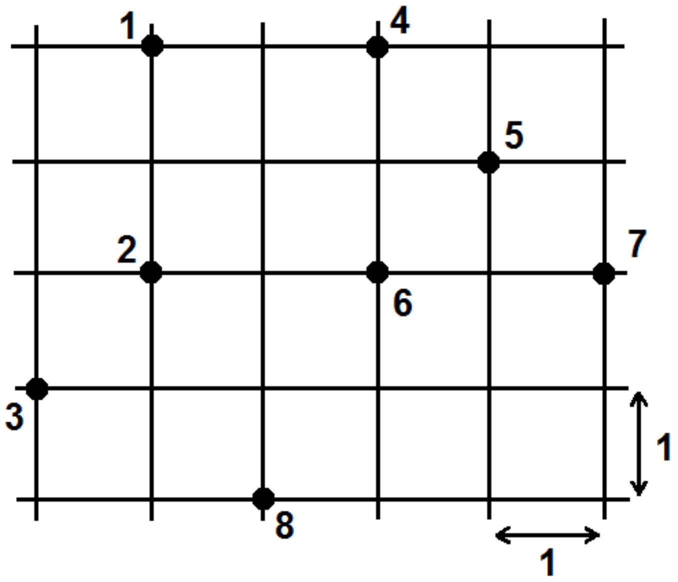


- Called the <u>triangle inequality</u> in plane geometry

Approximation Algorithms

- In the following, we derive a polynomial time approximation for the version of the TSP satisfying ①
- Consider a set of 8 cities labelled 1 through 8, located at the intersection of the grid of roads where the grid spacing is taken as unity in both the directions

Approximation Algorithms



- Figure 1: Location of 8 cities in a map with unit grid

Approximation Algorithms

- Distances between cities can be measured in at least one of two ways.

a) <u>Euclidean Distance</u>

$$d(i,j) = \ [(x_1(i) + x_1(j))^2 + (x_2(i) + x_2(j))^2]^{1/2} \ \rightarrow \ ②$$

where the coordinates of city i are $(x_1(i), x_2(i))^T$. This is the standard distance used in geometry

Approximation Algorithms

b) Manhattan Distance

$$d(i,j) = \sum_{k=1}^{2} |x_k(i) - x_k(j)| \rightarrow \text{③}$$

Is the sum of the absolute values of the differences in the first and second coordinates

- Note: In the following, we could use either distance. In our illustrations, we use the Euclidean distance ②.

Distance Matrix

1. Computation of weight/distance matrix

- Find the distance between the cities d(i,j)
- Set the weight matrix $w = [w_{ij}]$ where

$$w_{ij} = d(i, j) = w_{ji}$$

Distance Matrix

City

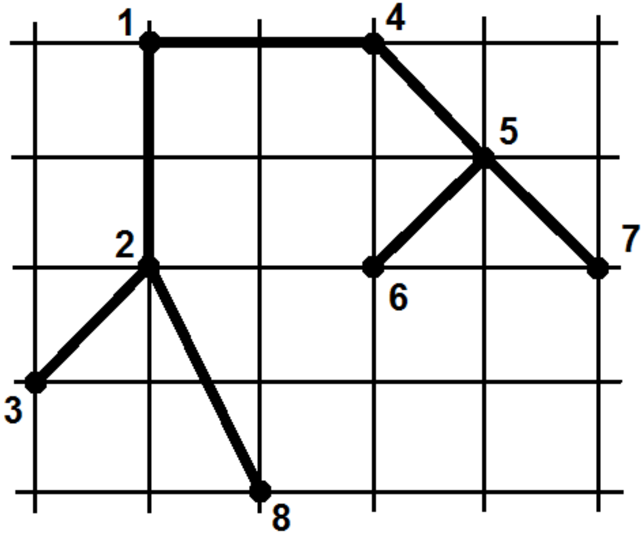
City	1	2	3	4	5	6	7	8
1	-	2	$\sqrt{10}$	2	$\sqrt{10}$	$\sqrt{8}$	$\sqrt{20}$	$\sqrt{17}$
2	2	-	$\sqrt{2}$	$\sqrt{8}$	$\sqrt{10}$	2	4	$\sqrt{5}$
3	$\sqrt{10}$	$\sqrt{2}$	-	$\sqrt{18}$	$\sqrt{20}$	$\sqrt{10}$	$\sqrt{26}$	$\sqrt{5}$
4	2	$\sqrt{8}$	$\sqrt{18}$	-	$\sqrt{2}$	2	$\sqrt{8}$	$\sqrt{17}$
5	$\sqrt{10}$	$\sqrt{10}$	$\sqrt{20}$	$\sqrt{2}$	-	$\sqrt{2}$	$\sqrt{2}$	$\sqrt{13}$
6	$\sqrt{8}$	2	$\sqrt{10}$	2	$\sqrt{2}$	-	2	$\sqrt{5}$
7	$\sqrt{20}$	4	$\sqrt{26}$	$\sqrt{8}$	$\sqrt{2}$	2	-	$\sqrt{13}$
8	$\sqrt{17}$	$\sqrt{5}$	$\sqrt{5}$	$\sqrt{17}$	$\sqrt{13}$	$\sqrt{5}$	$\sqrt{13}$	-

④

Minimum Spanning Tree

2. MST: Given G=(V, E, W), first find the minimum spanning tree (MST) using Kruskal's or Prim's algorithm described in the module on graph algorithms.

Minimum Spanning Tree



- Figure 2: The MST for the graph in Figure 1 with weight matrix in

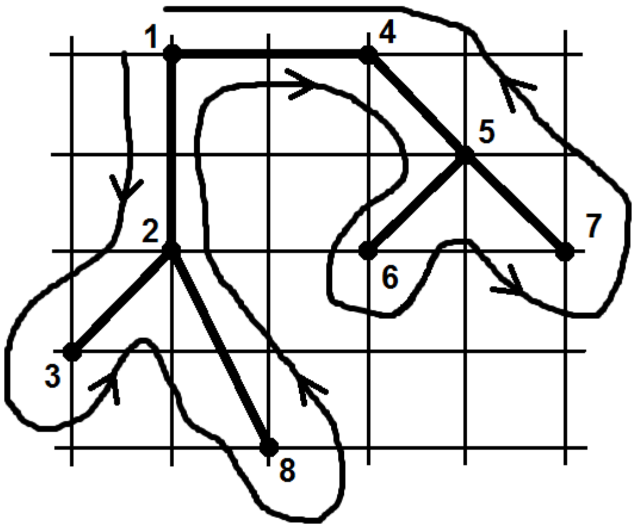
Walk

3. <u>Walk</u>: Considering node 1 as the root, do a preorder traversal of this MST to obtain a walk T

- The walk T induced by this traversal is

W= 1, 2, 3, 2, 8, 2, 1, 4, 5, 6, 5, 7, 5, 4, 1

Walk



- Figure 3: Preorder traversal superimposed on MST

Prune

4. <u>Prune the walk to get an approximate tour</u>

- In this walk, each edge is traversed twice in opposite directions
- Consider the triangle formed by {2,3,8}. In the walk, after visiting 2,3 to visit 8 we go to node 2 again.

Prune

- By Euclidean geometry, we know that

$$d(3,8) < d(3,2) + d(2,8)$$

- Hence, it is more meaningful to go to 8 from 3 directly rather than going via 2

Prune

- We can apply this strategy to each and every triangle that is a part of the walk.
- This is done by pruning the walk, where we keep only the first occurrence of each node label and deleting subsequent occurences

Prune

- Consider W reproduced below:

  W= 1, 2, 3, ~~2~~, 8, ~~2~~, ~~1~~, 4, 5, 6, ~~5~~, 7, ~~5~~, ~~4~~, ~~1~~

- The above said pruning leaves us with an approximate tour

  W= 1, 2, 3, 8, 4, 5, 6, 7

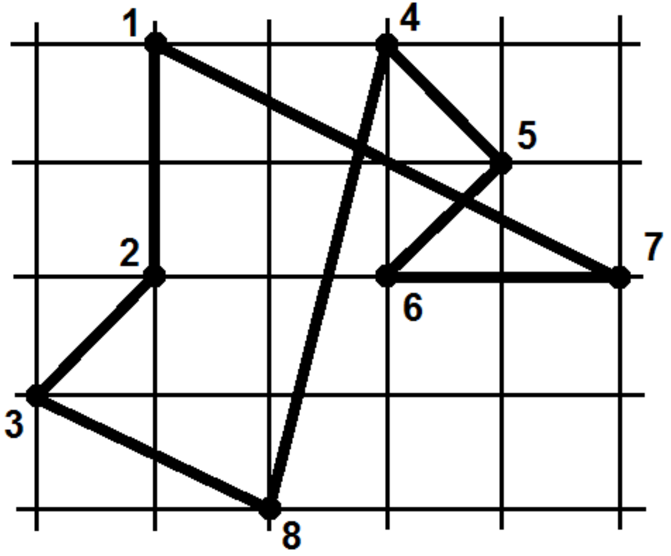
  which is shown in Figure 4

- The cost/distance of this approximate tour is

  $C(H) =$ sum of the distances of each edge in this tour H

  $= 19.074$

Prune

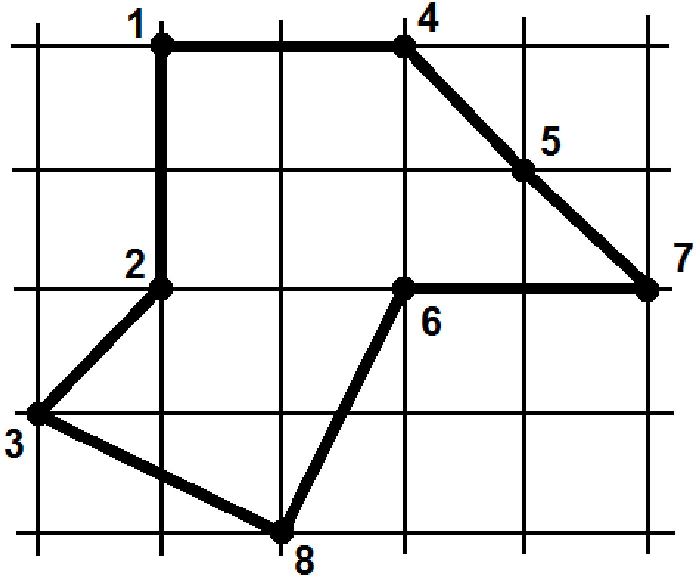


- Figure 4: Approximate tour H={1, 2, 3, 8, 4, 5, 6, 7}

Optimal Tour

- This problem being a simple problem with only 8 nodes, one can exhaustively enumerate all the towns and find the optimal tour, which is shown in Figure 5.
- It can be verified that the cost of this optimal tour is 14.715

Optimal Tour



- Figure 5: Optimal tour H* with cost C(H*) = 14.715

Polynomial Time Approximations

- Each of the steps involved, including finding the weights of $\frac{n(n-1)}{2}$ edges, find the MST, preorder traversal, and pruning, involve only polynomial time. Since the sum of the polynomial is a polynomial, the approximate town H in Step 4 takes only polynomial time.

Further Analysis

- We now examine the relation between C(H), the cost of the approximate tour, and that of the optimal tour C(H*).
- Let H* be an optimal tour, and H be an approximate tour, where

  $C(H^*) \leq C(H)$

- If we delete an edge from the optimal tour H*, we get a spanning tree T* with cost C(T)

  $C(T^*) \leq C(H^*) \rightarrow$ 5

Further Analysis

- If T is the minimum spanning tree for the graph G = (V,E,W), then

  $C(T) \leq C(T^*) \rightarrow$ ⑥

- Combing, we can see that C(T*), the cost of the MST, is a lower bound on the cost of the optimal tour H*, namely

  $C(T) \leq C(H^*) \rightarrow$ ⑦

Further Analysis

- The cost $C(W)$ of the walk W in step 4 is twice the cost of the MST T since we traverse each edge exactly twice. That is,

  $$C(W) = 2\,C(T) \rightarrow ⑧$$

- From ⑦, it then folows that

  $$C(W) \leq 2\,C(H^*) \rightarrow ⑨$$

Further Analysis

- The approximate tour H is obtained by using the triangle inequality, and hence

  $C(H) \leq C(W) \rightarrow$ ⑩

- Combining ⑨ and ⑩, we see that

  $C(H) \leq 2\, C(H^*) \rightarrow$ ⑪

- That is, the cost of the approximate tour is no more than twice the cost of the optimal tour

Further Analysis

- Note: Deriving polynomial approximation and finding the associated performance guarantees, without actually knowing what the optional cost, is one of the marvelous achievements of the approximation theory
- Note: The example used above is taken from chapter 35 of "Introduction to Algorithms" by T.H. Cormen et. al listed in Module 1