Math 567 (Spring 2025)

Heuristics and Approximation Algorithms for the TSP

From the chapter by D.S. Johnson and *C.H. Papadimitriou*

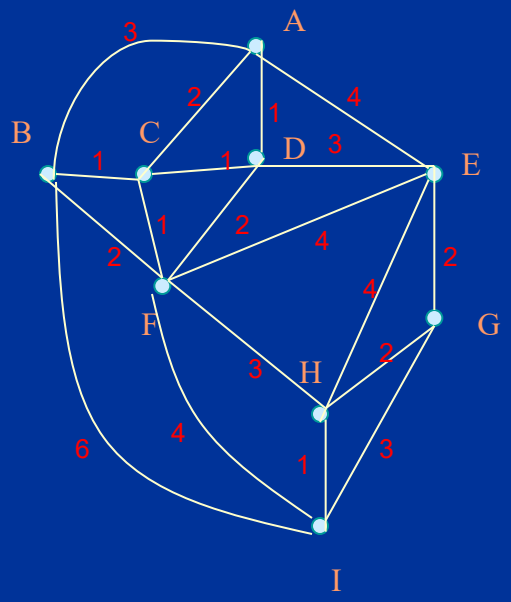
in the book "The Traveling Salesman Problem"

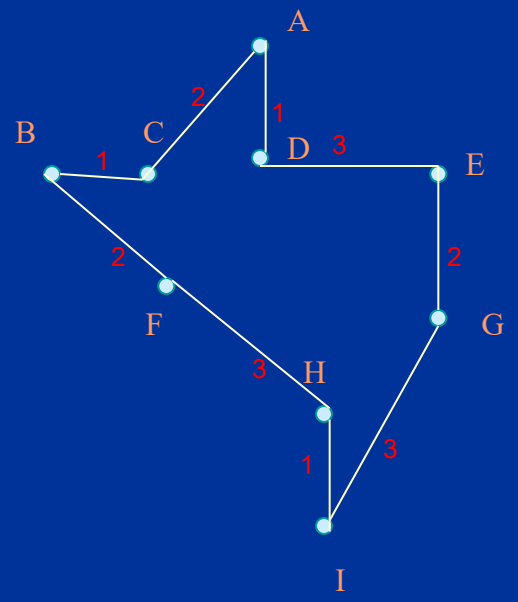
The Traveling Salesman Problem (TSP)

Given : A finite set of points (cities) V and the costs (distances) C_{ij} between each pair of points $i,j \in V$

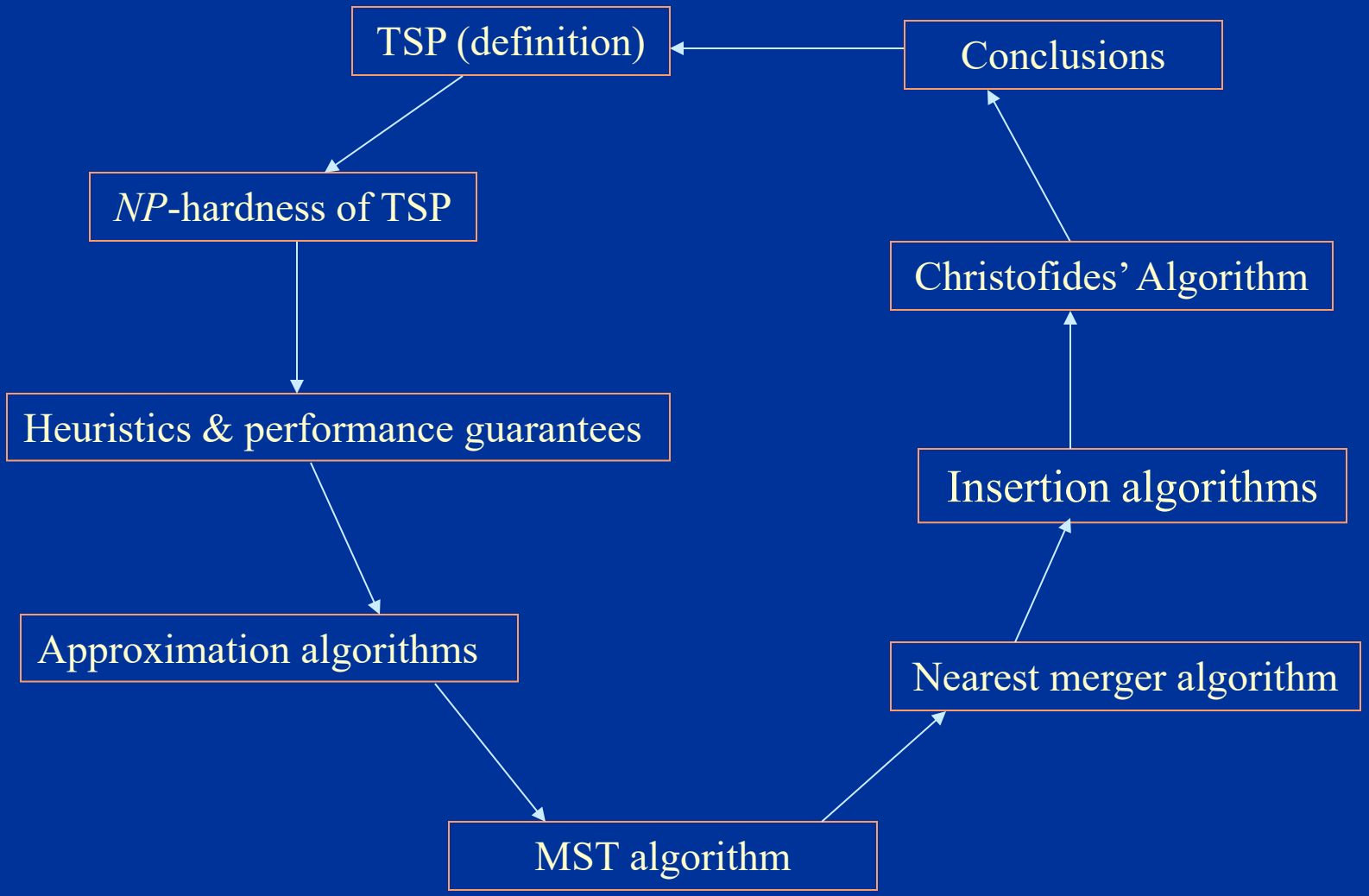
A *tour* : A *circuit* which passes exactly once through each point in V

TSP : Find a *tour* of *minimum cost (distance)*





The tour which we're going to take



Two classes of optimization problems : *P and NP-hard*

P set of problems which could be solved in polynomial time ("*easy to solve*")

NP-hard set of problems not known to be solvable in polynomial time ("*hard to solve*")

TSP is *NP* -hard

Heuristics Algorithms not guaranteed to find the optimal solution, but find a solution "reasonably close" to the optimal solution in reasonable amount of time

Performance Guarantees: provable bounds on how far the constructed tour can be from the optimal tour, *in the worst case*

Every tour is at least as good as (or better than) the bound

Approximation Algorithms for TSP

A is an ε-approximation algo for the TSP if, for each instance I of the problem, and for an error bound $\varepsilon > 0$, *A* gives a tour S of length $l(S)$ such that

$$l(S) \leq (1+\varepsilon)\, opt(I)$$

where opt(I) - length of the minimal TSP tour

Polynomial ε-approximation algorithm: ε-approximation algorithm that runs in polynomial time

Result: TSP has no polynomial ε-approximation scheme for any $\varepsilon > 0$

Assumptions

=> costs(distances) are symmetric $\{ C_{ij} = C_{ji} \}$

=> costs(distances) satisfy the *triangle inequality*

$C_{ik} \leq C_{ij} + C_{jk}$ for any $i, j,$ and k

Euclidean setting: shortest distance between two points (cities) is a straight line (direct route)

Which of the solution methods for the TSP produce polynomial ε-approximation algorithms, and for what ε ?

Minimum Spanning Tree algorithm

Spanning tree : Set of *n-1* edges which join the *n* vertices (cities) into a single connected component

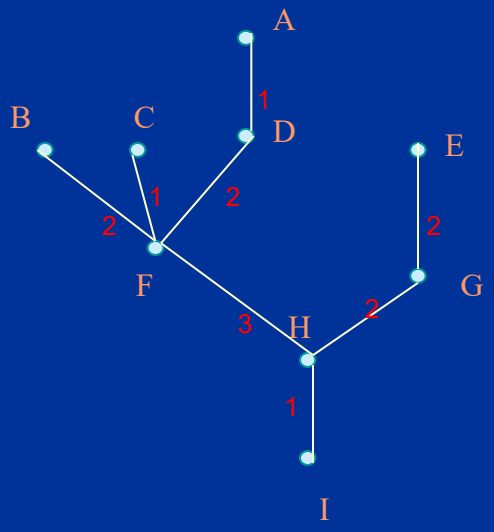
Minimum spanning tree : A spanning tree with the minimum cost

Deletion of a single edge from a traveling salesman tour gives a spanning tree

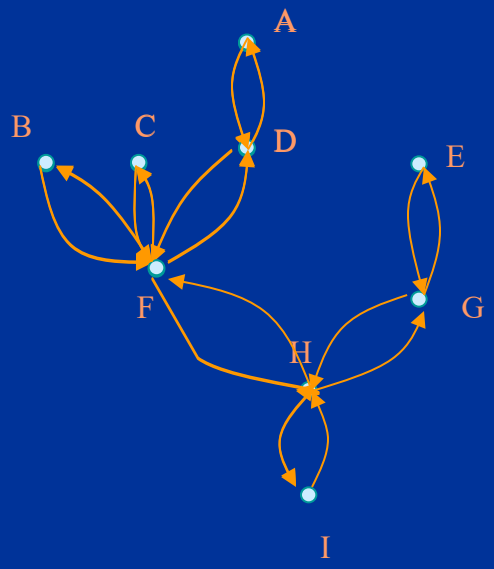
opt(TSP) $\geq$ MST (lower bound)

Upper Bound on the Optimal Traveling Salesman tour

Depth-first traversal of the minimum spanning tree



minimum spanning tree



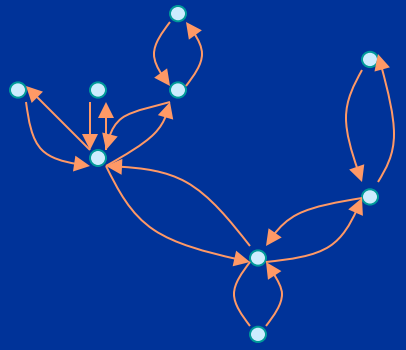
depth first traversal tour

This visits all cities

Length of tour = 2(MST) ≤ 2 opt(TSP)

Not a Traveling salesman tour as some of the cities might be visited more than once!

Convert this tour to a traveling salesman tour using *shortcuts*

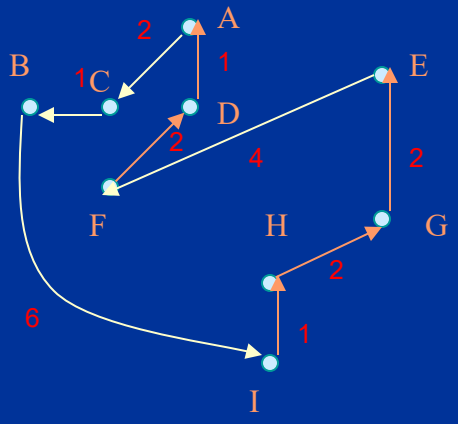


Instead of going back to the predecessor node, jump to the next as yet unvisited node

Skip already visited nodes

By *triangle inequality,*

MST tour $\leq$ depth first traversal tour



MST tour

MST tour $\leq$ 2 opt(TSP)

MST algorithm is 1-approximate for the TSP

Other 1-approximate algorithms

Nearest merger algorithm

Start with n partial tours (each consisting of a single city)

Merge tours successively until a single tour containing all the cities is obtained

If there are more than one partial tour, the tours T and T' to be merged are chosen such that $\min\{C_{ij} : i \in T, j \in T'\}$ is as small as possible

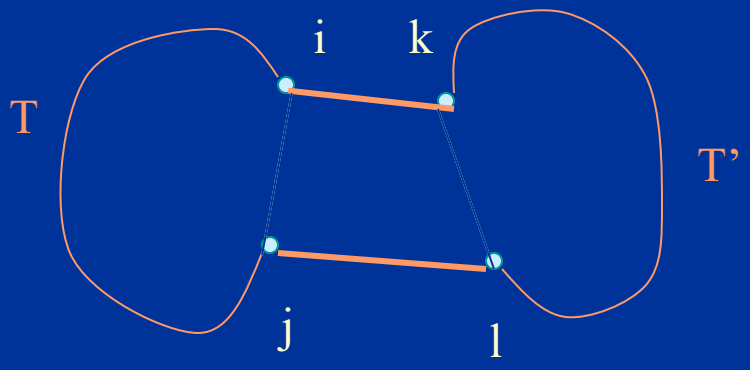
Rule for merging two partial tours T and T'

(i,j): edge in T (k,l): edge in T'

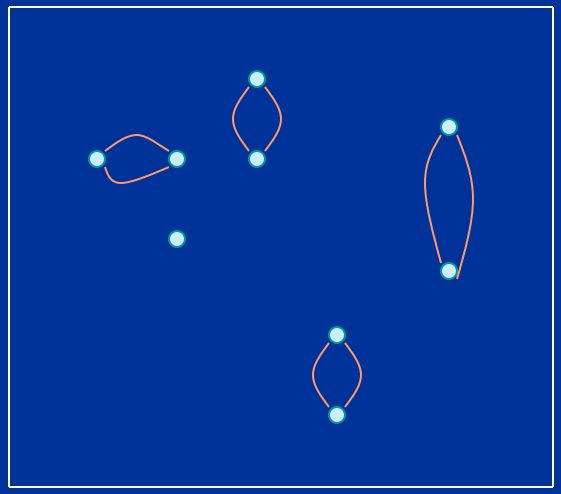
$\min \{C_{ik} + C_{jl} - C_{ij} - C_{kl}\}$

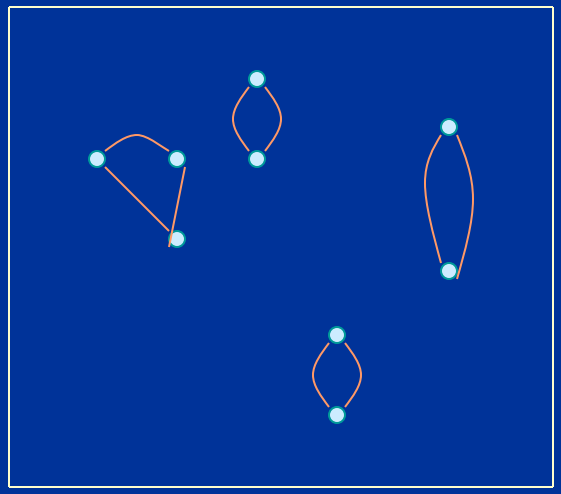
add (i,k) and (j,l)

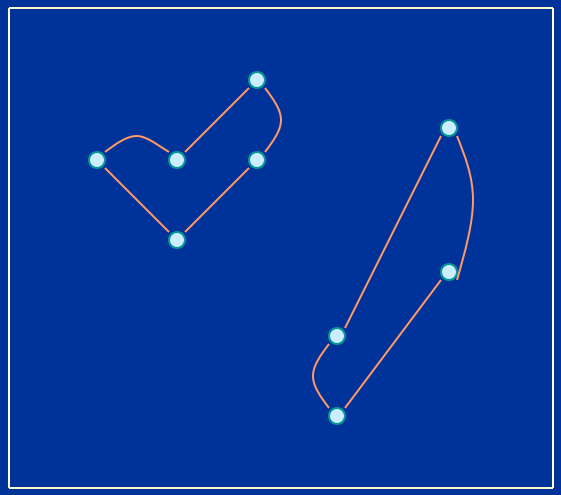
delete (i,j) and (k,l)

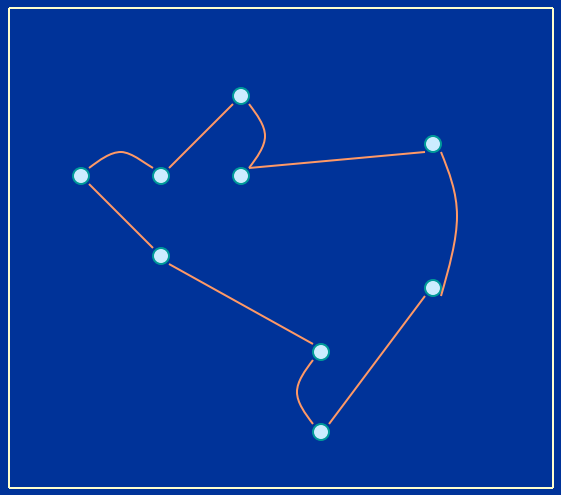


Nearest merger algorithm in action









Length of nearest merger tour = opt(TSP) = 18

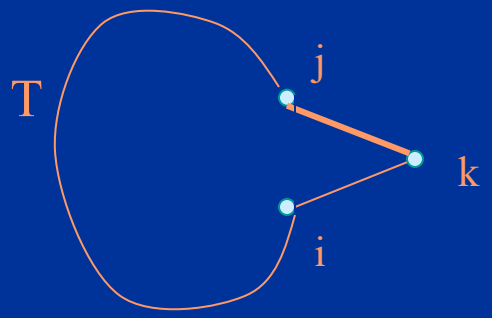
Insertion Algorithms

step 1) start with a single city (arbitrarily chosen)

step 2) If all cities are not included in the current partial tour T, select a city $k \notin T$ to be inserted into the partial tour

step 3) Delete (i,j), and insert (i,k) and (k,j)

Cost of inserting k into the tour = $C_{ik} + C_{kj} - C_{ij}$

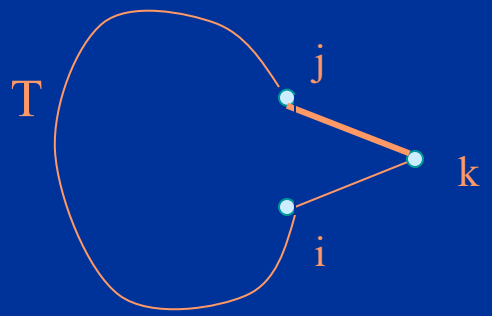


1) Nearest Addition algorithm

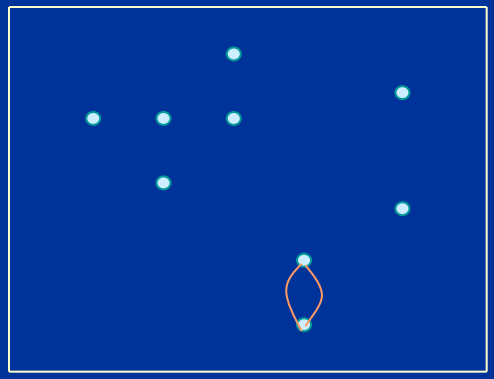
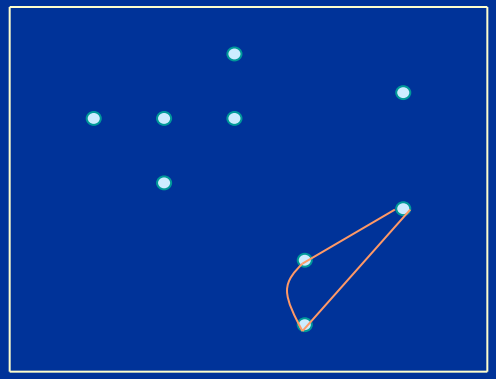
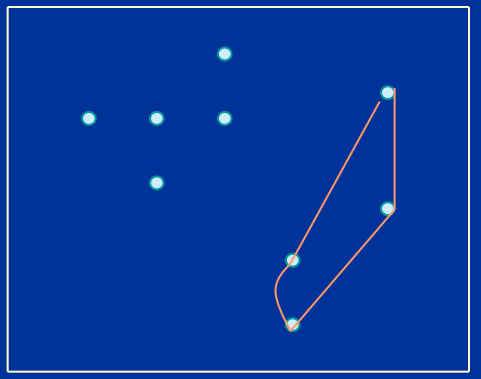
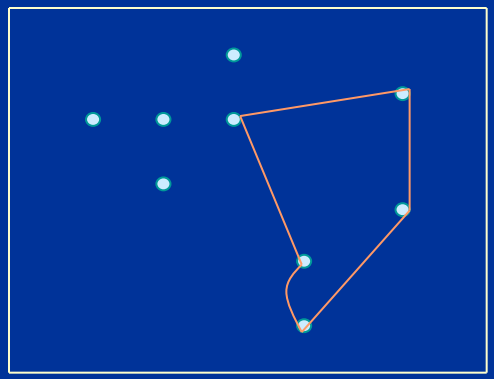
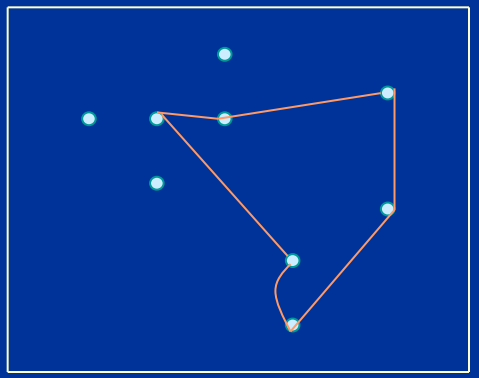
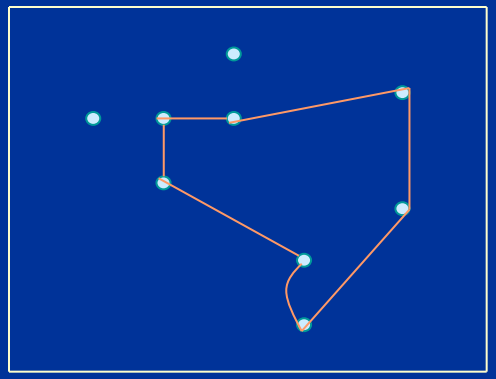
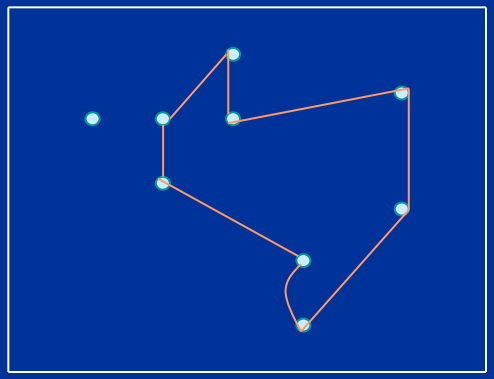
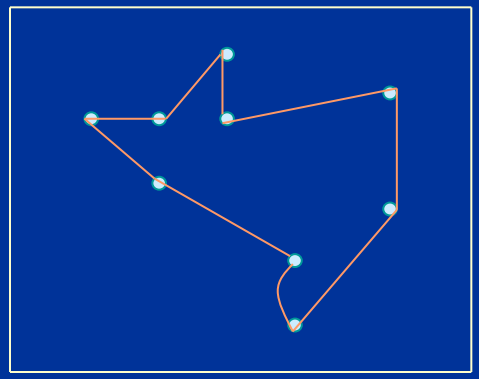
step 1) start with a single city (arbitrarily chosen)

step 2) If all cities are not included in the current partial tour T, find cities j and k, $j \in T$, $k \notin T$, such that C_{kj} is minimized

step 3) Delete (i,j), where i is the neighbor (one of the two nodes immediately near j in the tour) and insert (i,k) and (k,j)



Nearest addition algorithm in action

Length of nearest addition tour = opt(TSP) = 18

2) Nearest insertion algorithm

Chooses *k* (city to be inserted) as in the nearest addition algorithm.

But *k* is inserted between *i'* and *j'* such that $C_{i'k} + C_{kj'} - C_{i'j'}$ is minimized

3) Cheapest insertion algorithm

The new city k to be inserted (in step 2 of the nearest addition algorithm) is chosen such that $C_{ik} + C_{kj} - C_{ij}$ is minimized and inserted accordingly

The proofs for the upper bound for the nearest merger and insertion algorithms are variants of the proof for the MST algorithm

All the above algorithms are 1-approximate for the TSP

Christofides' Algorithm

Eulerian graph : Contains a tour which passes through every edge exactly once (Eulerian tour)

All vertices have even degree in an Eulerian graph

Idea : start with the MST, find an Eulerian graph containing the MST, convert the Eulerian tour to a TSP tour using shortcuts

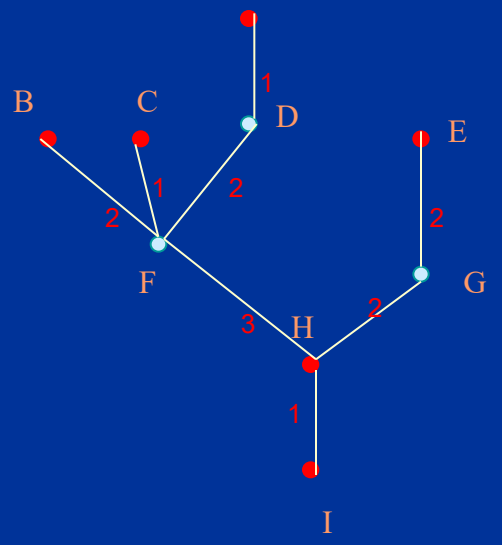
Step 1) Find a minimum spanning tree (T) for the given set of vertices(cities)

Step 2) Take the vertices with odd degree alone

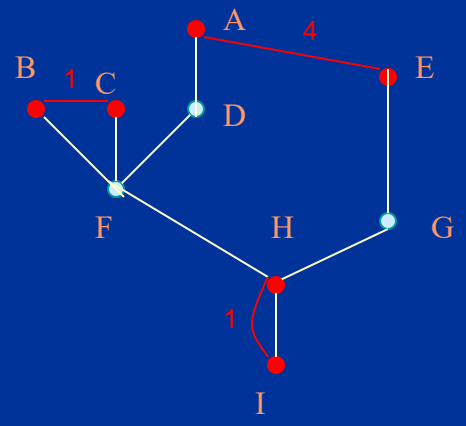
Step 3) Find a minimum weighted matching (M) for these vertices

Step 4) Add M to T. This gives an Eulerian graph that has minimum length among those that contain T

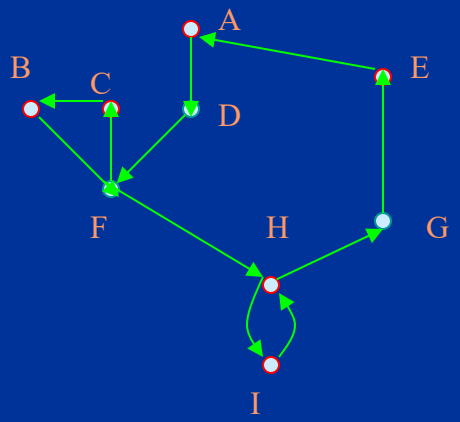
Step 5) Convert the Eulerian tour of the above graph into a traveling salesman tour using *shortcuts*



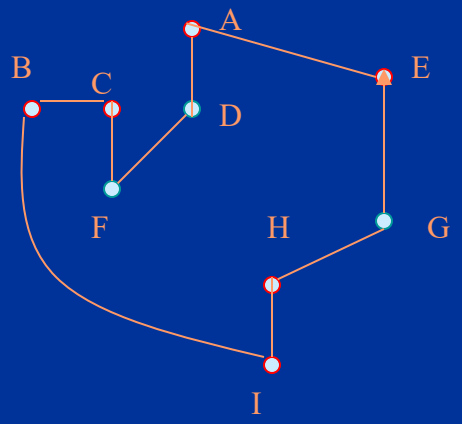
MST (T)



Minimum matching (M)

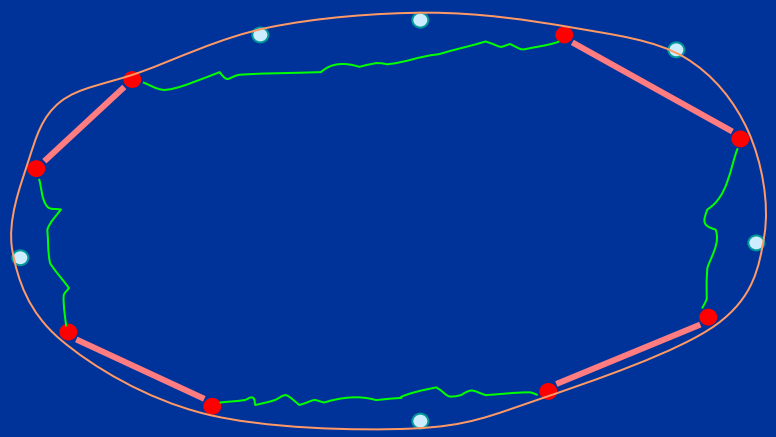


Eulerian Tour



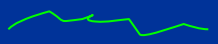
Eulerian Traveling salesman tour

Upper Bound provided by Christofides' algorithm



Opt. tour

M



M'

Odd degree vertex

The TSP tour determines two matchings M and M'

by triangle inequality

length(M) + length(M') ≤ Opt(TSP)

min weighted matching ≤ (1/2)opt(TSP)

min(M,M') ≤ (1/2) opt(TSP)

length(MST) ≤ opt(TSP)

Eulerian traveling salesman tour ≤ (3/2) opt(TSP)

Christofides' algorithm is (1/2)-approximate for the TSP

Conclusions

TSP is NP- hard

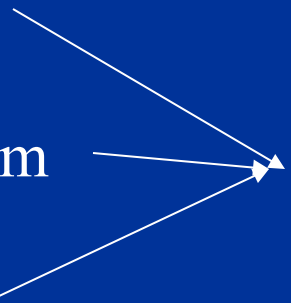
No ε-approximation schemes for the general TSP

With costs(distances) being symmetric and satisfying the triangle inequality

MST algorithm(s)

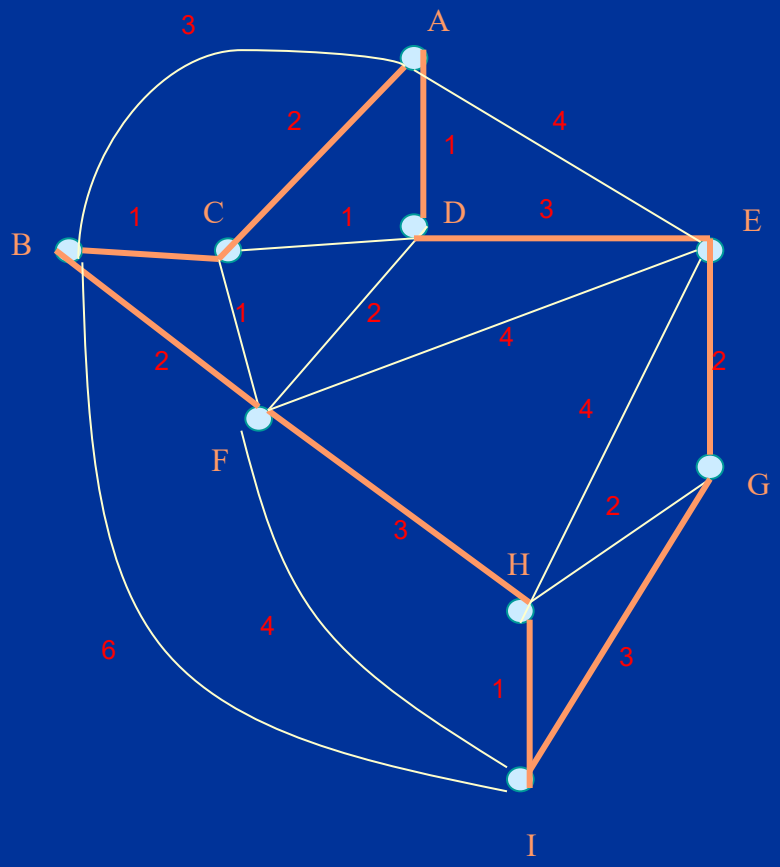
Nearest merger algorithm

Insertion algorithms

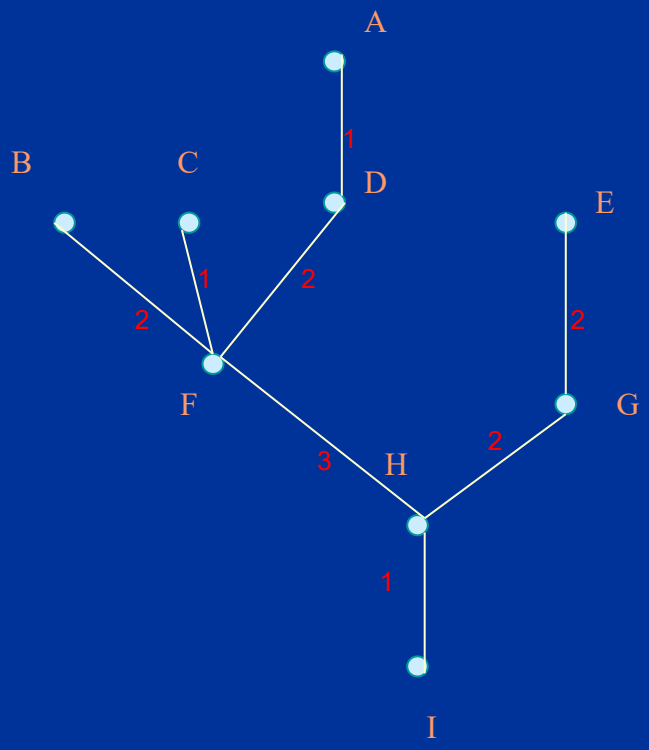


2-optimal (upper bound)

Christofides' algorithm: (3/2)-optimal (best bound)



Optimal tour



Minimum spanning tree