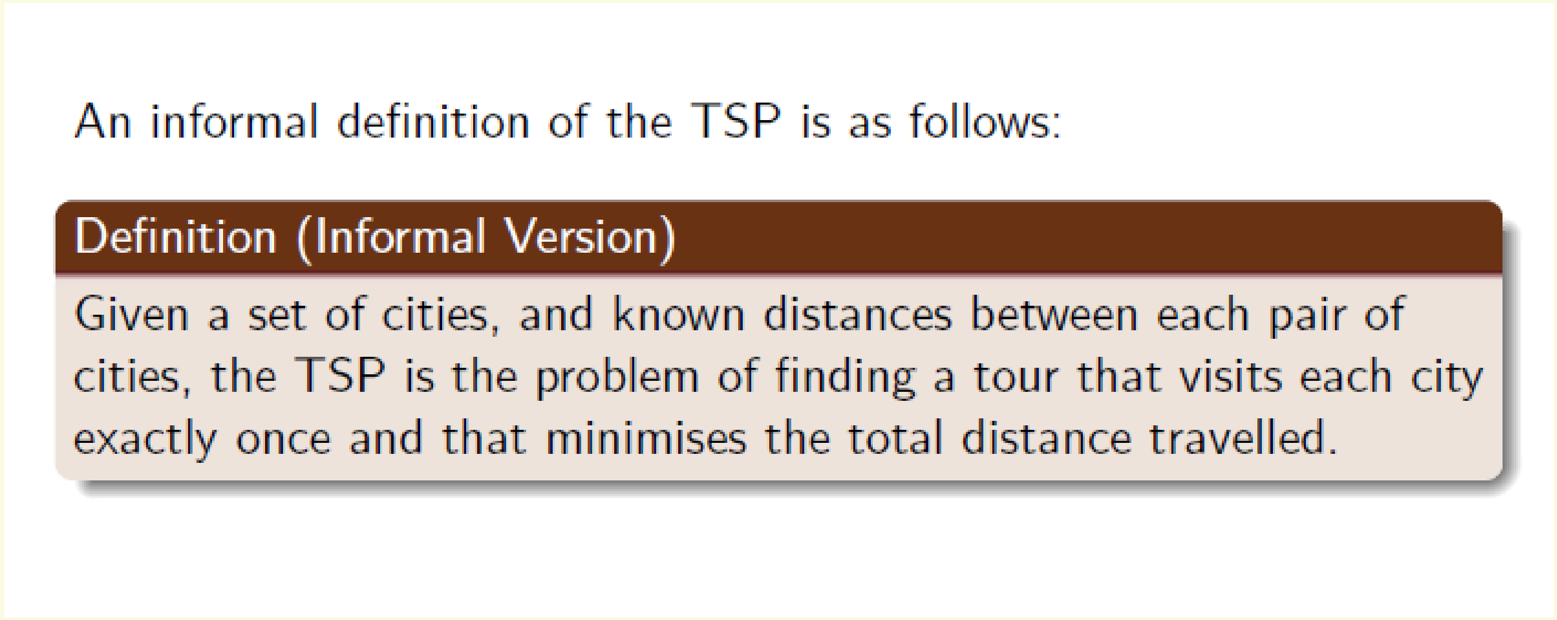
Traveling Salesman Problem

The TSP involves finding the minimum traveling cost for visiting a fixed set of customers.

The vehicle must visit each customer exactly once and return to its point of origin also called depot.

The objective function is the total cost of the tour.

Traveling Salesman Problem

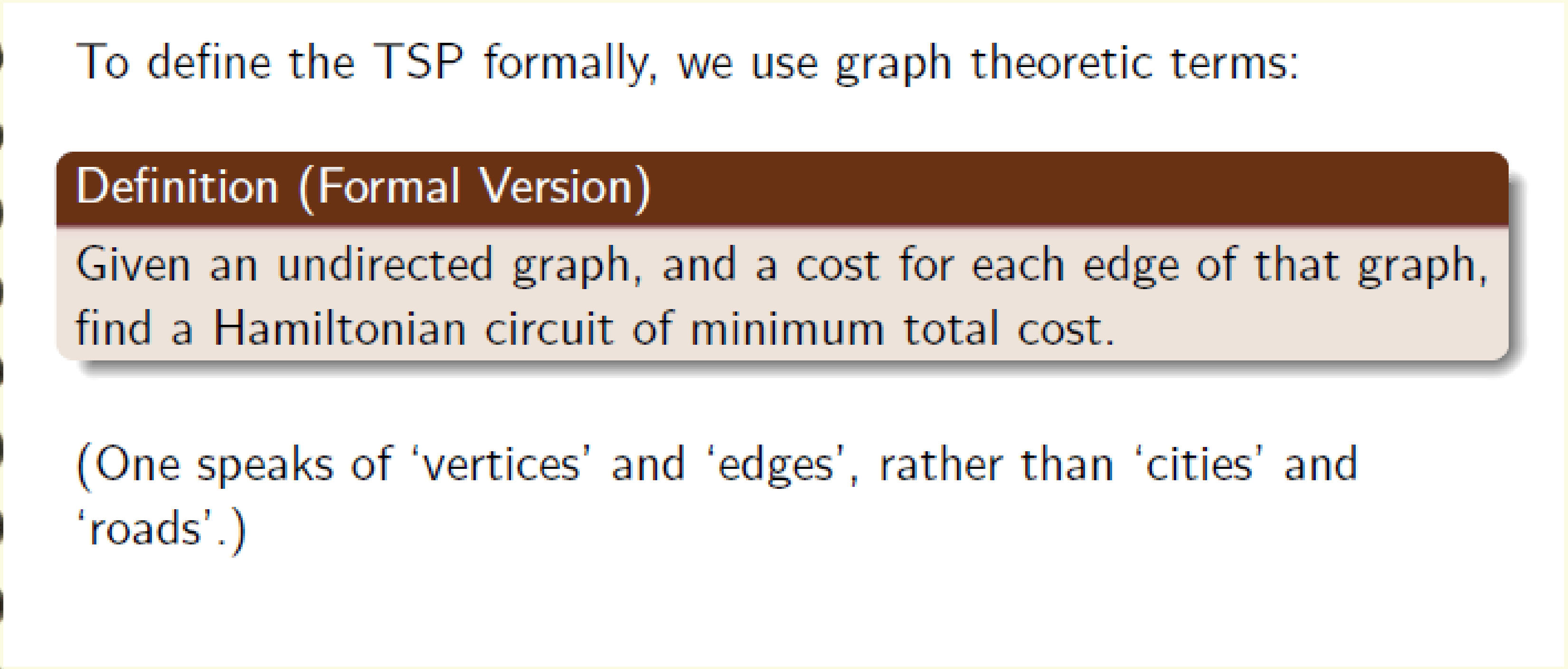


An informal definition of the TSP is as follows:

Definition (Informal Version)

Given a set of cities, and known distances between each pair of cities, the TSP is the problem of finding a tour that visits each city exactly once and that minimises the total distance travelled.

Traveling Salesman Problem



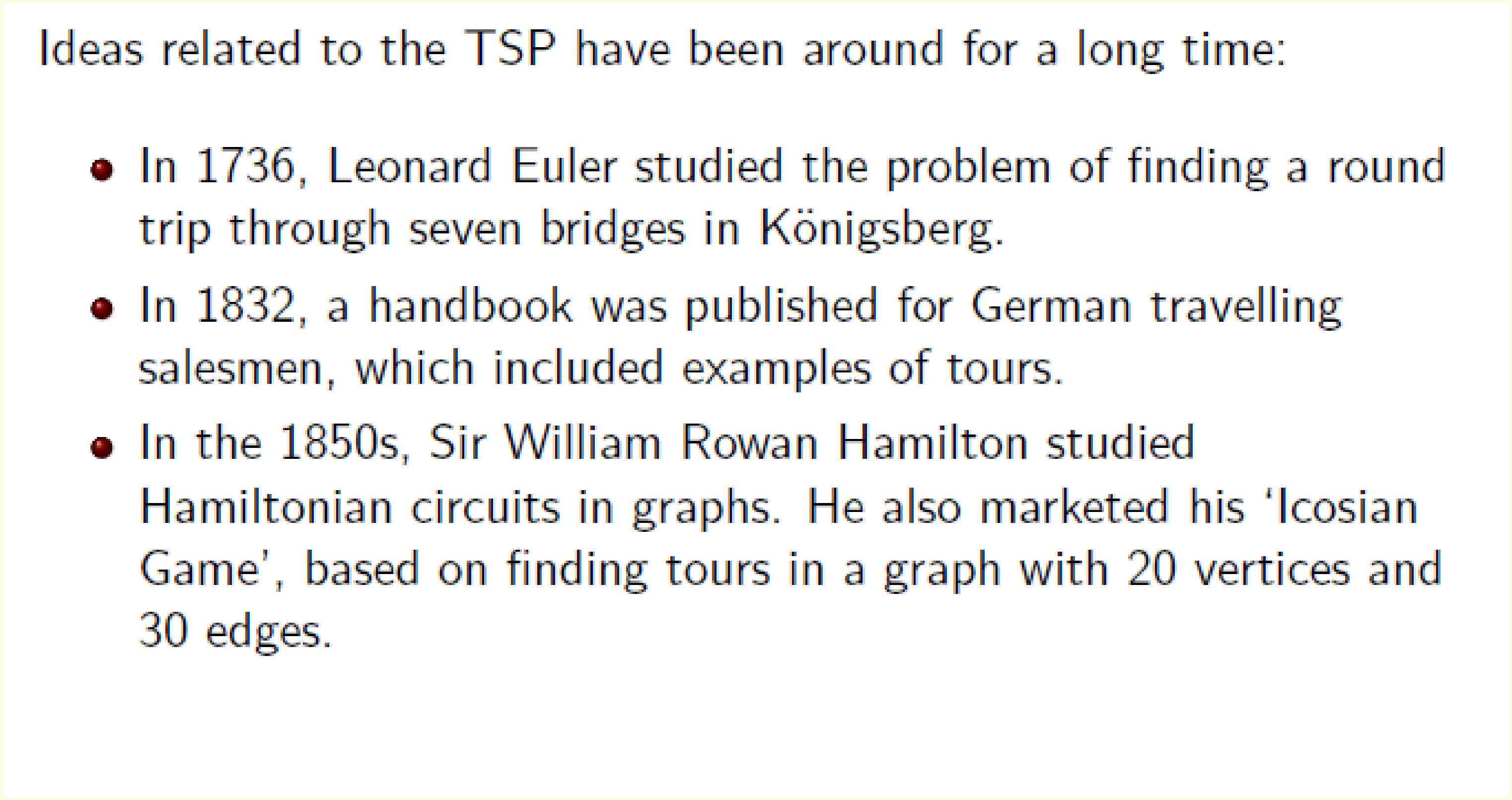
To define the TSP formally, we use graph theoretic terms:

Definition (Formal Version)

Given an undirected graph, and a cost for each edge of that graph, find a Hamiltonian circuit of minimum total cost.

(One speaks of ‘vertices’ and ‘edges’, rather than ‘cities’ and ‘roads’.)

Traveling Salesman Problem



Ideas related to the TSP have been around for a long time:

- In 1736, Leonard Euler studied the problem of finding a round trip through seven bridges in Königsberg.
- In 1832, a handbook was published for German travelling salesmen, which included examples of tours.
- In the 1850s, Sir William Rowan Hamilton studied Hamiltonian circuits in graphs. He also marketed his 'Icosian Game', based on finding tours in a graph with 20 vertices and 30 edges.

Traveling Salesman Problem

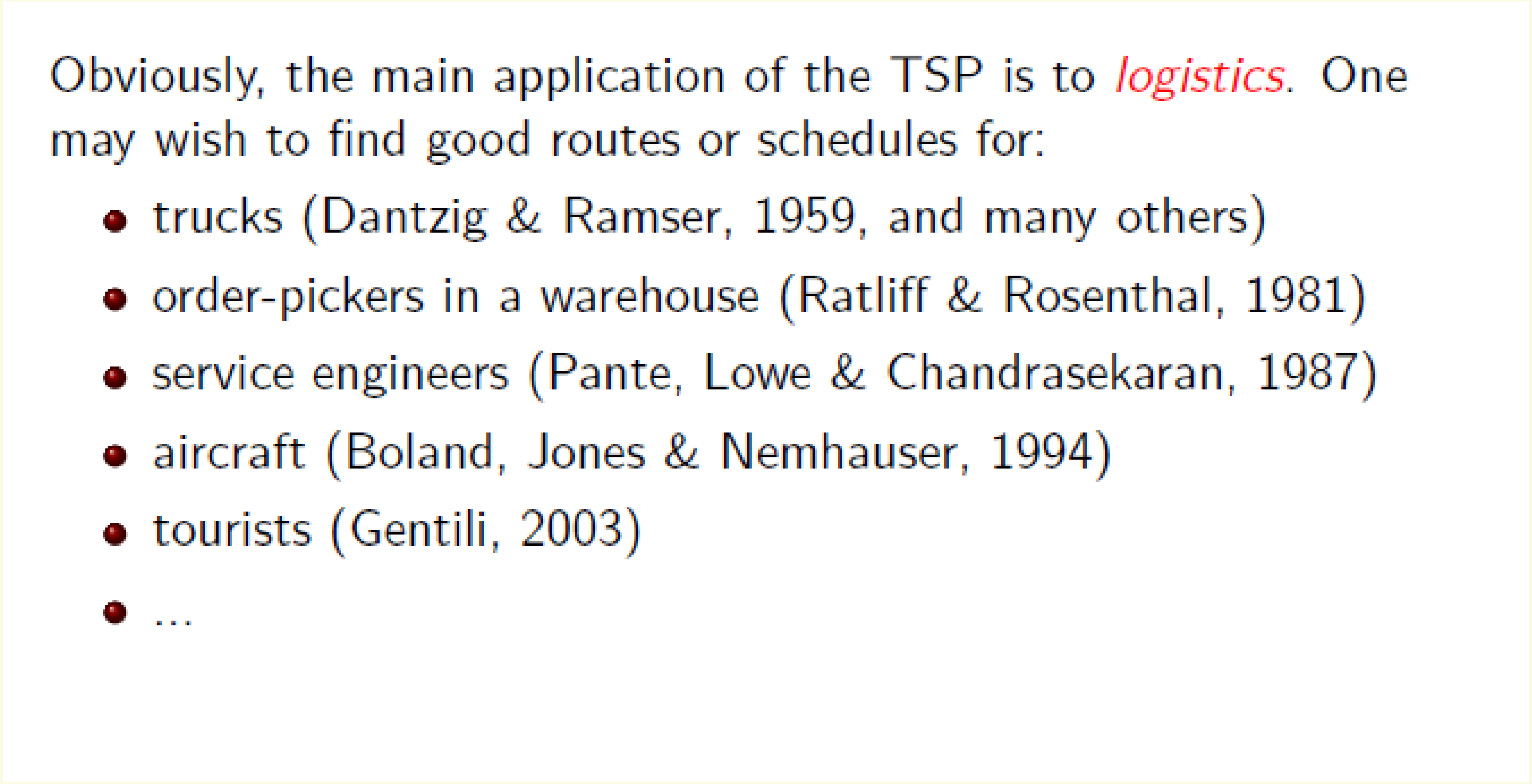
Moving to the 20th century...

- In the early 1930s, Karl Menger discussed the problem with colleagues in Vienna and Harvard.
- In the late 1930s, the problem reappeared at Princeton University. Hassler Whitney called it the TSP.
- In the mid-1940s, the TSP was studied by several statisticians.
- In the late 1940s and early 1950s, it was studied intensively by researchers at the RAND corporation.

More later...

Traveling Salesman Problem

Applications of the TSP



Obviously, the main application of the TSP is to *logistics*. One may wish to find good routes or schedules for:

- trucks (Dantzig & Ramser, 1959, and many others)
- order-pickers in a warehouse (Ratliff & Rosenthal, 1981)
- service engineers (Pante, Lowe & Chandrasekaran, 1987)
- aircraft (Boland, Jones & Nemhauser, 1994)
- tourists (Gentili, 2003)
- ...

Traveling Salesman Problem

Applications of the TSP

There are however some *less obvious* applications:

- scheduling jobs on machines
- controlling satellites, telescopes, microscopes, lasers...
- computing DNA sequences
- designing telecommunications networks
- designing and testing VLSI circuits
- x-ray crystallography
- clustering data arrays
- ...

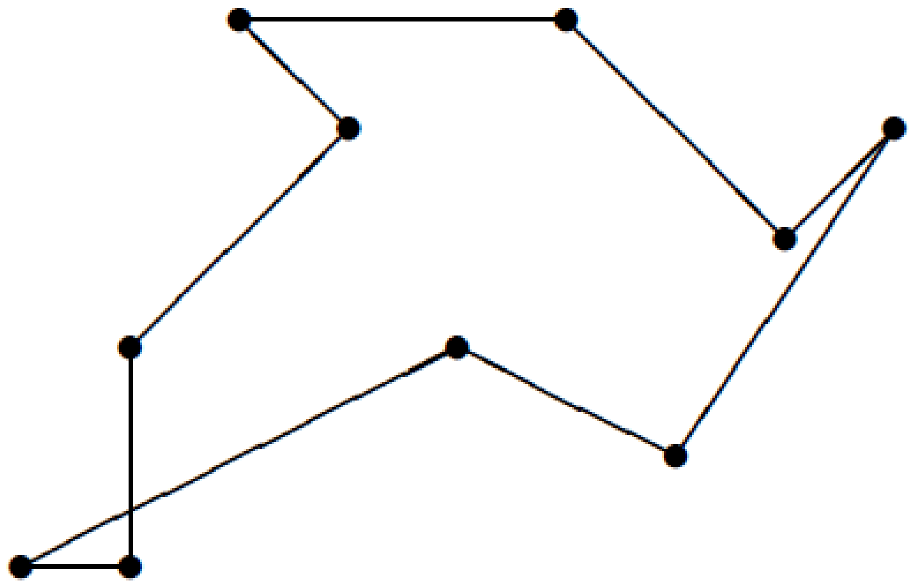
Traveling Salesman Problem

The Hardness of the TSP

The TSP is a lot more complicated than it may appear!

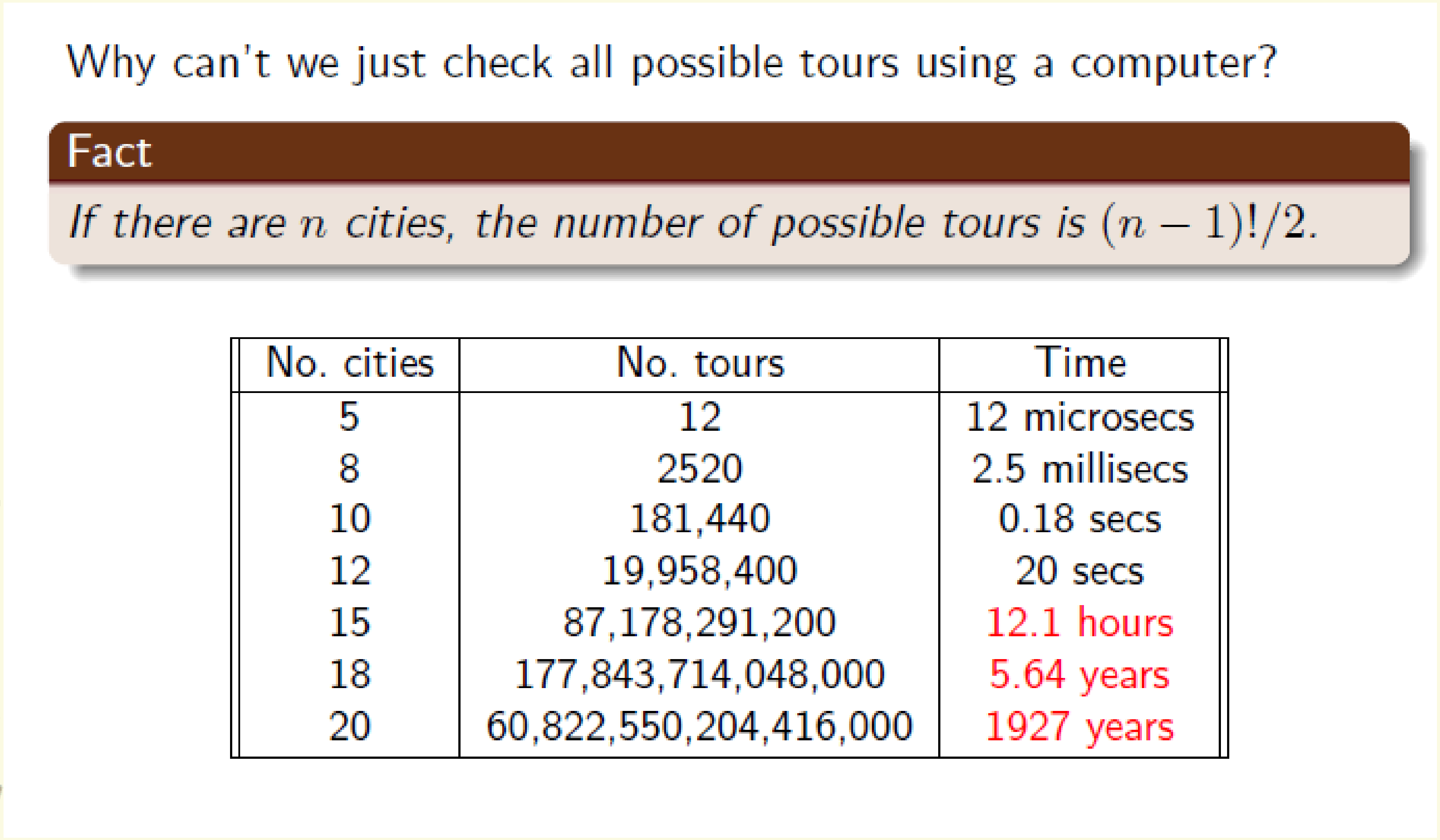
Fact

If you just start at an arbitrary city and keep going to the city nearest to it, you can get a bad solution.



Traveling Salesman Problem

The Hardness of the TSP



Why can't we just check all possible tours using a computer?

Fact

If there are n cities, the number of possible tours is $(n-1)!/2$.

No. cities	No. tours	Time
5	12	12 microsecs
8	2520	2.5 millisecs
10	181,440	0.18 secs
12	19,958,400	20 secs
15	87,178,291,200	12.1 hours
18	177,843,714,048,000	5.64 years
20	60,822,550,204,416,000	1927 years

Traveling Salesman Problem

In the 1950s and 1960s, great progress was made on various discrete optimisation problems. Efficient methods were found for:

- the transportation problem (Charnes & Cooper, 1954)
- the assignment problem (Kuhn, 1955)
- the maximum flow problem (Ford & Fulkerson, 1956)
- the minimum spanning tree problem (Prim, 1956)
- the matching problem (Edmonds, 1963).

But *nobody could solve the TSP!*

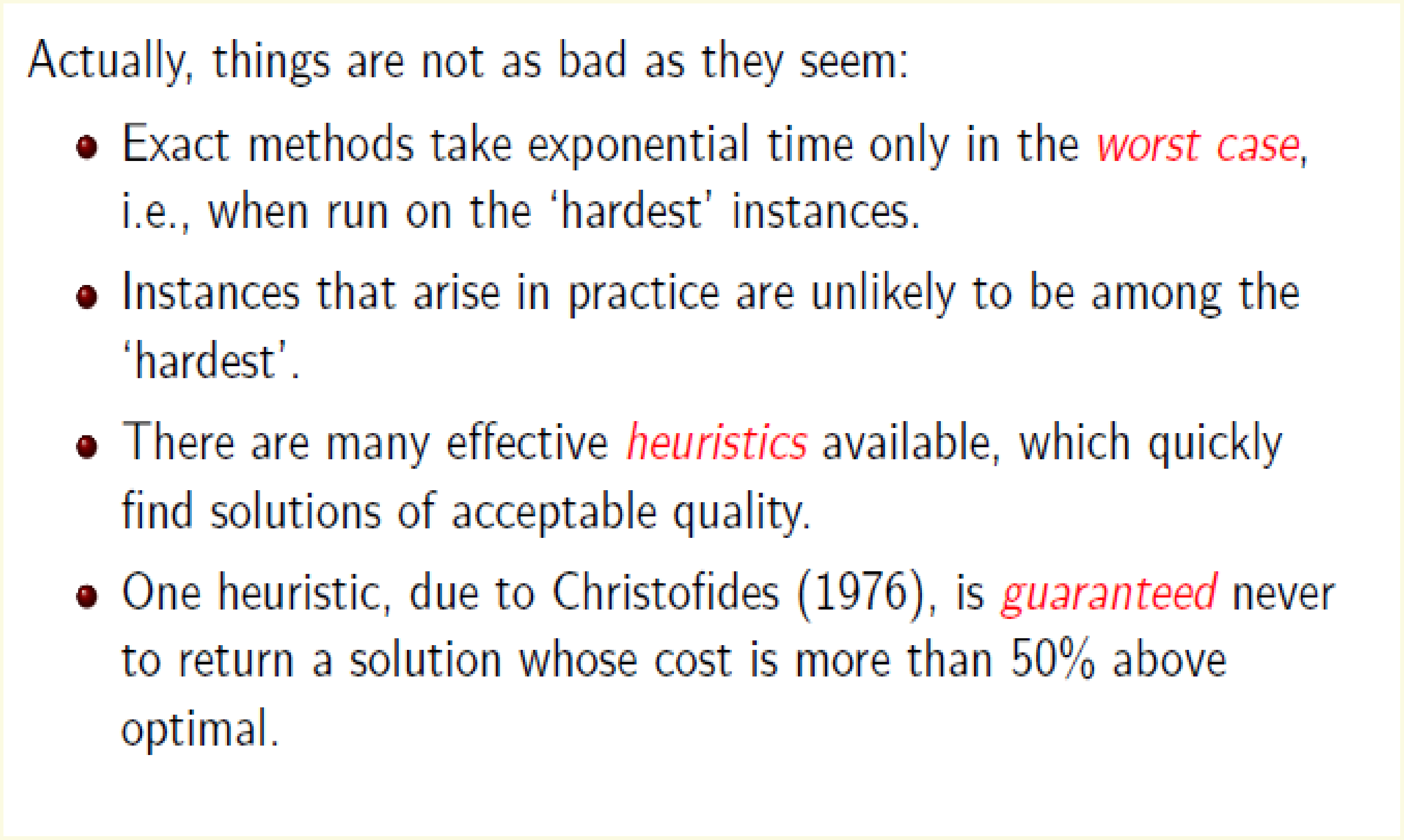
Traveling Salesman Problem

In 1972, Karp proved that the TSP is an '$\mathcal{NP}$-hard' problem.

- All known *exact* methods for $\mathcal{NP}$-hard problems run in *exponential* time.
- Nobody knows how to solve $\mathcal{NP}$-hard problems in *polynomial* time.
- Whether this can be done is a famous open question in theoretical computer science.

(Many other important problems in Operational Research and related fields are also $\mathcal{NP}$-hard.)

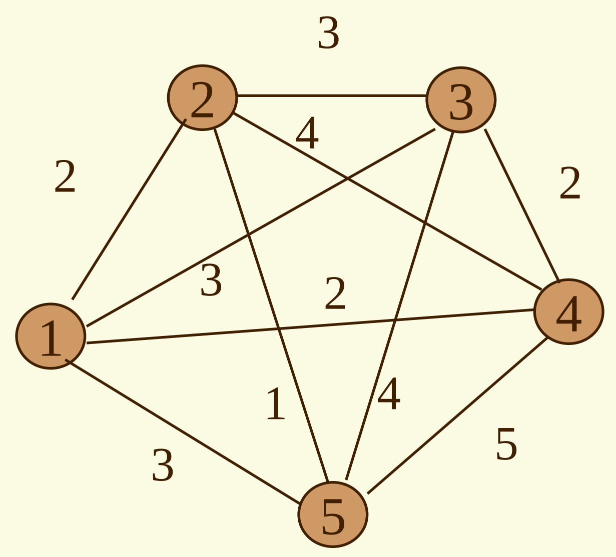
Traveling Salesman Problem



Actually, things are not as bad as they seem:

- Exact methods take exponential time only in the *worst case*, i.e., when run on the 'hardest' instances.
- Instances that arise in practice are unlikely to be among the 'hardest'.
- There are many effective *heuristics* available, which quickly find solutions of acceptable quality.
- One heuristic, due to Christofides (1976), is *guaranteed* never to return a solution whose cost is more than 50% above optimal.

Traveling Salesman Problem

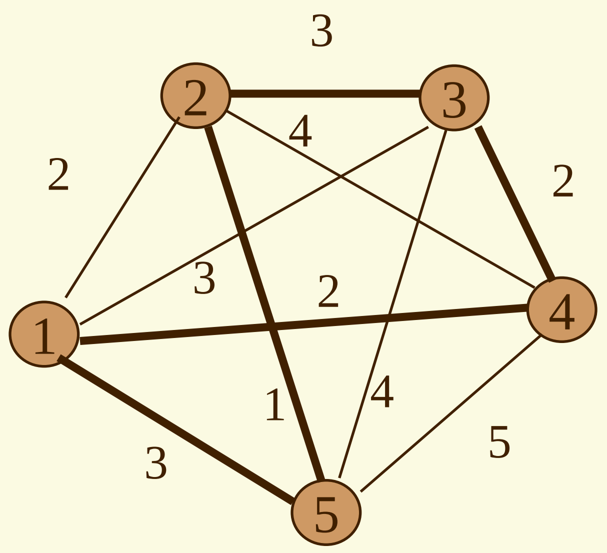


The total number of solutions is (n-1)! /2 if the distances are symmetric.

For example, if there are 50 customers to visit, the total number of solutions is $49!/2=3.04\text{x}10^{62}$.

Traveling Salesman Problem

Solution



If the depot is located at node 1, then the optimal tour is 1-5-2-3-4-1 with total cost equal to 11.

Traveling Salesman Problem

Mathematical Programming

Inputs: n = number of customers including the depot

c_{ij} = cost of traveling from customer i to j

Decision variables:

$$x_{ij} = \begin{cases} 1 & \text{if the vehicle travels from customer } i \text{ to } j. \\ 0 & \text{otherwise} \end{cases}$$

Traveling Salesman Problem

$$\min \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij}$$

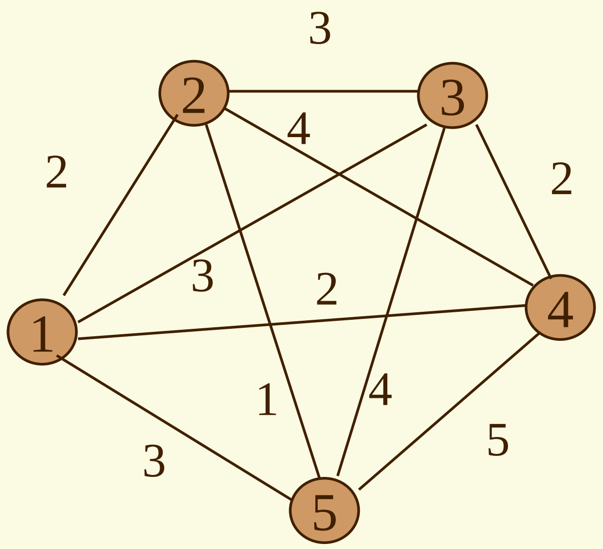
$$\text{s.t.} \quad \sum_{i=1}^{n} x_{ij} = 1 \quad \text{for all } j \qquad (1)$$

$$\sum_{j=1}^{n} x_{ij} = 1 \quad \text{for all } i \qquad (2)$$

$$x_{ij} \in \{0, 1\} \quad \text{for all } i, j$$

Constraints (1) and (2) ensure that each customer is visited exactly once.

Traveling Salesman Problem



minimize $2x_{12}+3x_{13}+2x_{14}+3x_{15}+3x_{23}+4x_{24}+\ldots+5x_{45}$

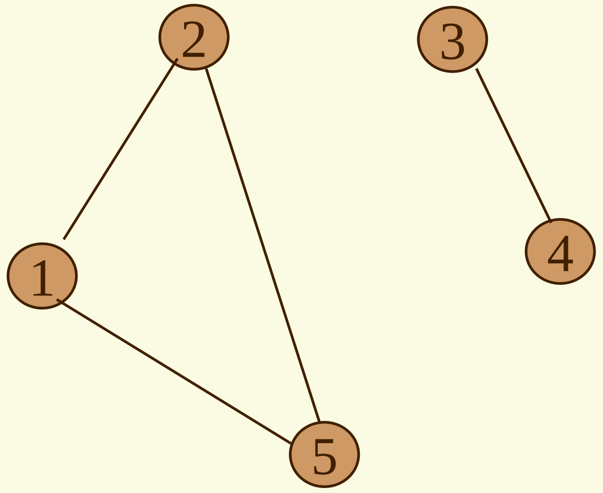
s.t. $x_{12}+x_{13}+x_{14}+x_{15} = 1$ for node 1

$x_{21}+x_{31}+x_{41}+x_{51} = 1$ for node 1

...

...

Traveling Salesman Problem



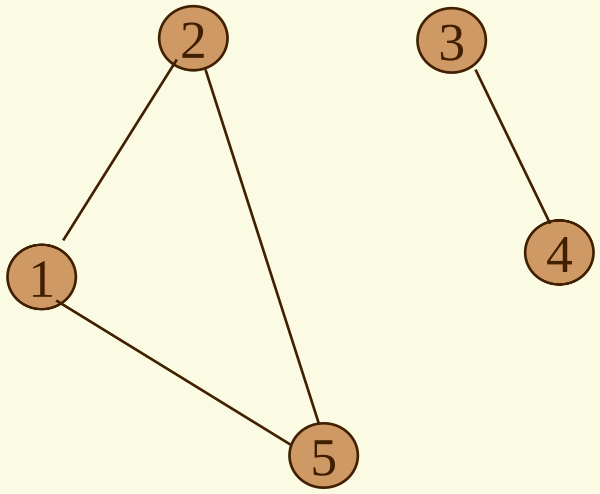
Is this solution feasible to our formulation?

We need additional constraints, so-called subtour elimination constraints.

Traveling Salesman Problem

$$\sum_{i \in S} \sum_{j \in \bar{S}} x_{ij} \geq 1 \quad \text{for every subset } S$$

$$\sum_{i \in S} \sum_{j \in S} x_{ij} \leq |S| - 1 \quad \text{for every subset } S$$

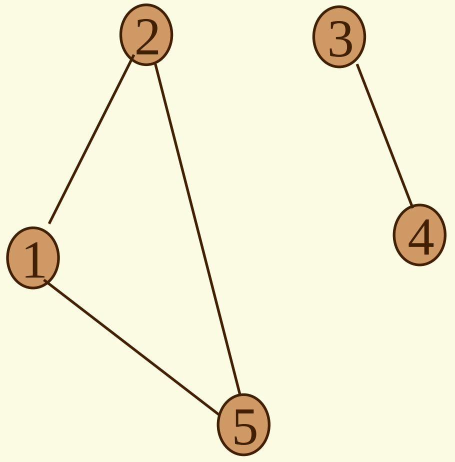


Traveling Salesman Problem

Subtour elimination constraints for the example:

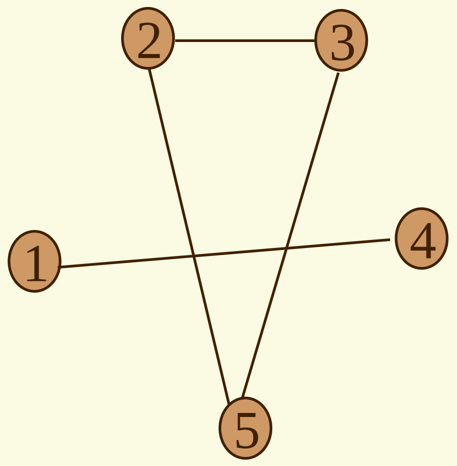
$$S = \{1,2,5\}, \bar{S} = \{3,4\}$$

$$x_{13} + x_{14} + x_{23} + x_{24} + x_{53} + x_{54} \geq 1$$



$$S = \{1,4\}, \bar{S} = \{2,3,5\}$$

$$x_{12} + x_{13} + x_{15} + x_{42} + x_{43} + x_{45} \geq 1$$



Traveling Salesman Problem

Heuristics for the TSP

1. Construction Heuristics

build a feasible solution by adding a node to the partial tour one at a time and stopping when a feasible solution is found.

Many construction heuristics are also called greedy heuristics because they seek to maximize the improvement at each step.

Traveling Salesman Problem

Nearest Neighbor

Step 1. Start with any node as the beginning of the path.

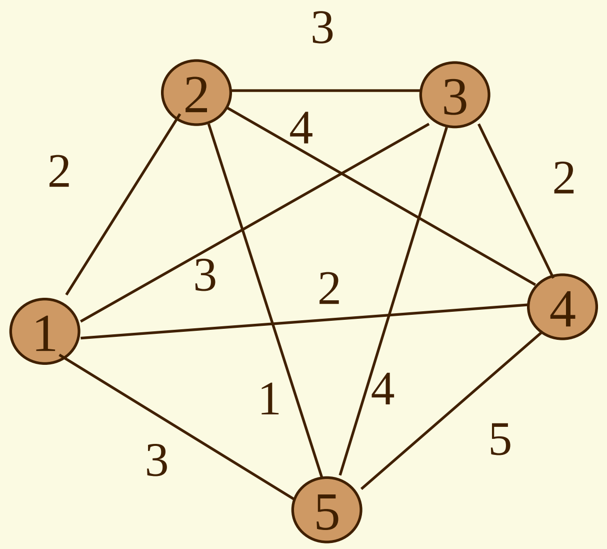
Step 2. Find the node closest to the last node added to the path.

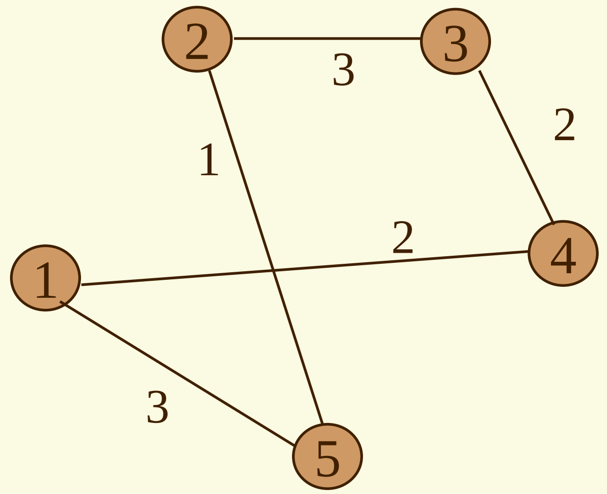
Step 3. Repeat Step 2 until all nodes are contained in the path. Then, join the first and last nodes.

Traveling Salesman Problem

Start with node 2.

Nodes added	Tour
2	2
5	2-5
1	2-5-1
4	2-5-1-4
3	2-5-1-4-3
	2-5-1-4-3-2





Traveling Salesman Problem

Nearest Insertion

Step 1. Start with a node *i* only.

Step 2. Find node *k* such that c_{ik} is minimal and form subtour *i*–*k*–*i*.

Step 3. *Selection step*. Given a subtour, find node *k* not in the subtour closest to any node in the subtour.

Step 4. *Insertion step*. Find the arc (*i*,*j*) in the subtour which minimizes $c_{ik}+c_{kj}-c_{ij}$. Insert *k* between *i* and *j*.

Step 5. Go to step 3 unless we have a Hamiltonian cycle.

Traveling Salesman Problem

Farthest Insertion

Step 1. Start with a node i only.

Step 2. Find node *k* such that c_{ik} is maximal and form subtour *i*–*k*–*i*.

Step 3. *Selection step*. Given a subtour, find node *k* not in the subtour farthest from any node in the subtour.

Step 4. *Insertion step*. Find the arc (*i*,*j*) in the subtour which minimizes $c_{ik}+c_{kj}-c_{ij}$. Insert *k* between *i* and *j*.

Step 5. Go to step 3 unless we have a Hamiltonian cycle.

Traveling Salesman Problem

Arbitrary Insertion

Step 1. Start with a node i only.

Step 2. Find node *k* such that c_{ik} is minimal and form subtour *i*–*k*–*i*.

Step 3. *Selection step*. Arbitrarily select node *k* not in the subtour to enter the subtour.

Step 4. *Insertion step*. Find the arc (*i*,*j*) in the subtour which minimizes $c_{ik}+c_{kj}-c_{ij}$. Insert *k* between *i* and *j*.

Step 5. Go to step 3 unless we have a Hamiltonian cycle.

Traveling Salesman Problem

Cheapest Insertion

Step 1. Start with a node i only.

Step 2. Find node *k* such that c_{ik} is minimal and form subtour *i*–*k*–*i*.

Step 3. *Insertion step*. Find arc (*i*,*j*) in the subtour and node *k* not, such that $c_{ik}+c_{kj}-c_{ij}$ is minimal and, then, insert *k* between *i* and *j*.

Step 4. Go to step 3 unless we have a Hamiltonian cycle.

Traveling Salesman Problem

Clarke and Wright Savings Heuristic

1. Select any node as the central depot which is denoted as node 0. Form subtours *i*-0-*i* for i=1,2,...,*n*. (Each customer is visited by a separate vehicle)

2. Compute savings $s_{ij}=c_{0i} + c_{0j} - c_{ij}$ for all *i*,*j*

3. Identify the node pair (*i*,*j*) that gives the highest saving s_{ij}

4. Form a new subtour by connecting (*i*,*j*) and deleting arcs (*i*,0) and (0, *j*) if the following conditions are satisfied

   a) both node *i* and node *j* have to be directly accessible from node 0

   b) node *i* and node *j* are not in the same tour.

Traveling Salesman Problem

Clarke and Wright Savings Heuristic

5. Set s_{ij}=-∞, which means that this node pair is processed.

   Go to Step 3, unless we have a tour.

Traveling Salesman Problem

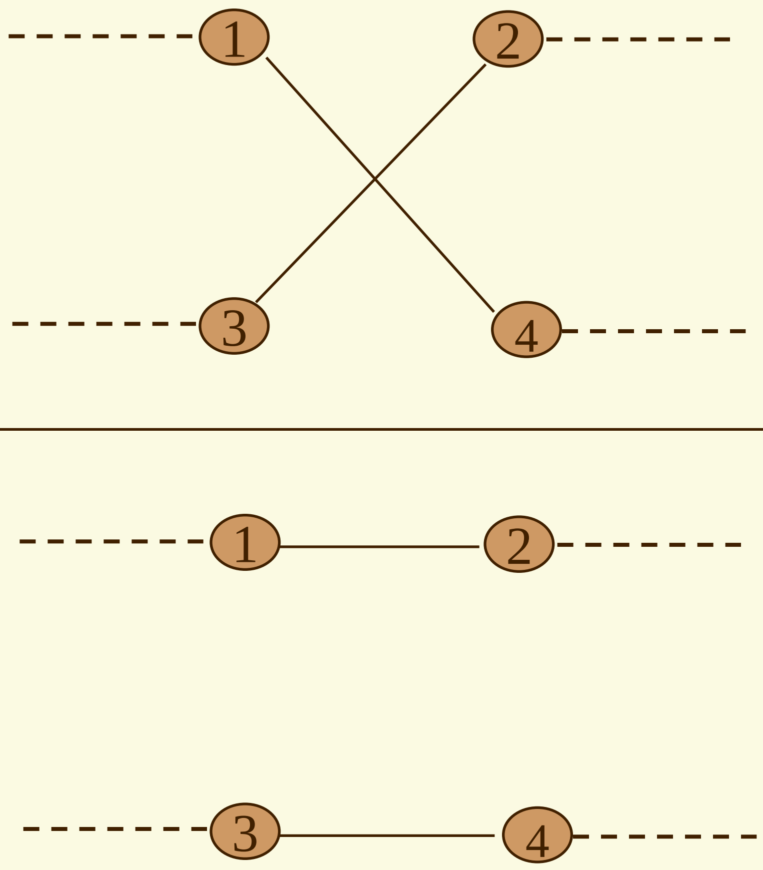
Heuristics for the TSP

2. Improvement Heuristics

begin with a feasible solution and successively improve it by a sequence of exchanges while maintaining a feasible solution throughout the process.

Traveling Salesman Problem

Heuristics for the TSP



Traveling Salesman Problem

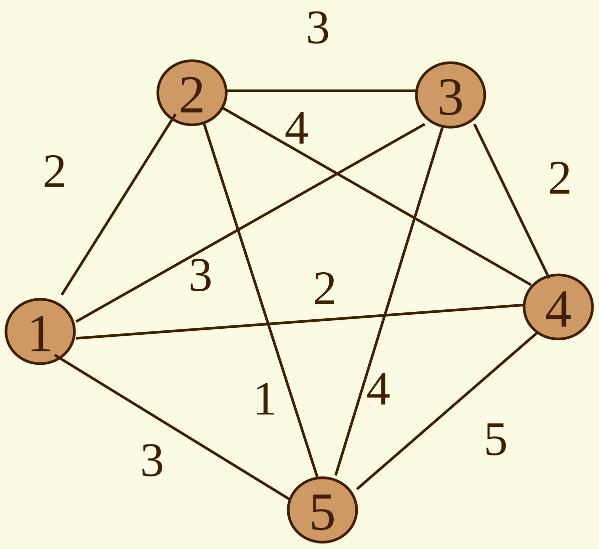
Two exchange heuristics (2-opt)

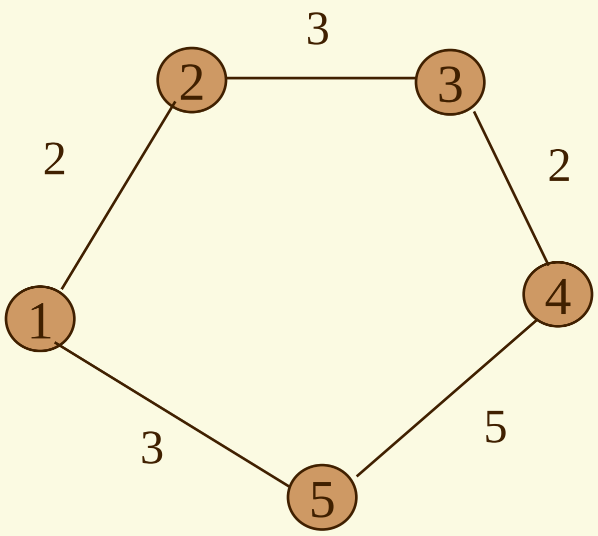
Starting with any tour, we consider the effect of removing any two arcs in the tour and replacing them with the unique set of two arcs that form a different tour.

If we find a tour with a lower cost than the current tour, then we use it as the new tour.

When no possible exchange can produce a tour that is better than the current tour, we stop.

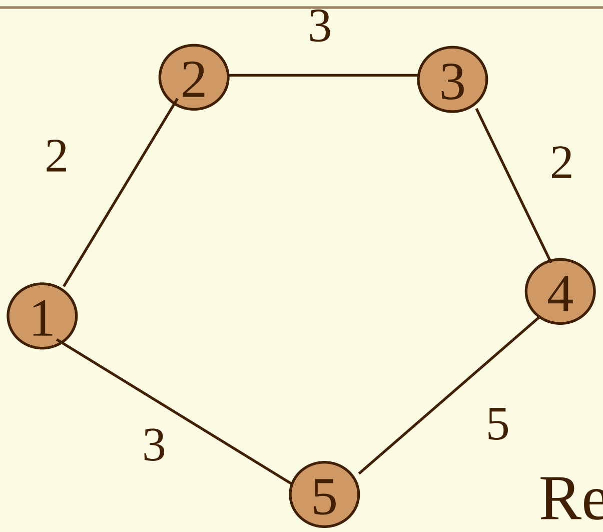
Traveling Salesman Problem



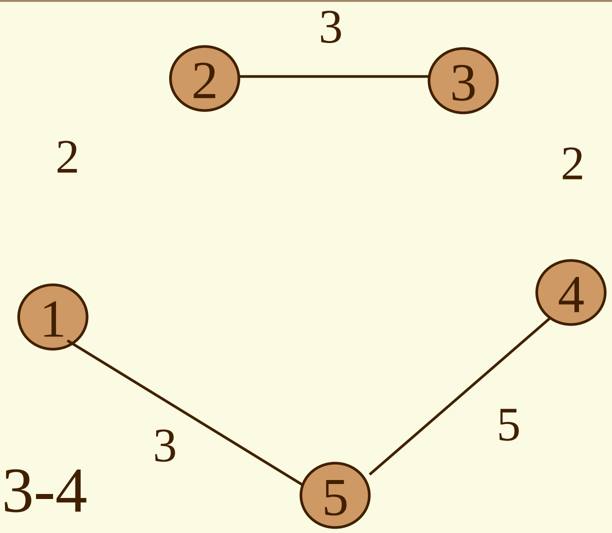


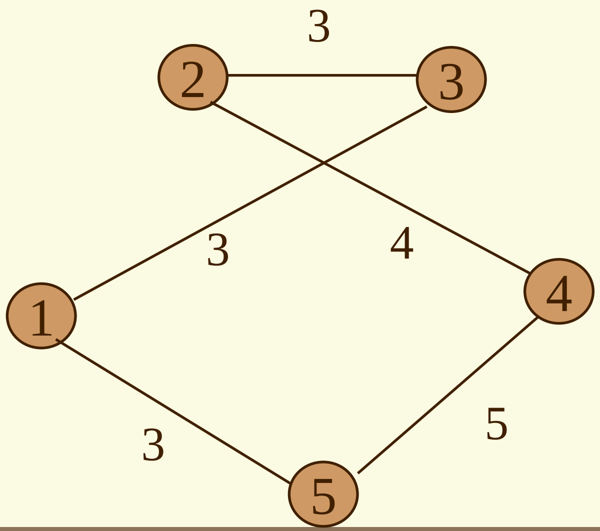
Suppose the initial tour is given as 1-2-3-4-5 with cost 15.

Traveling Salesman Problem



Remove 1-2 and 3-4

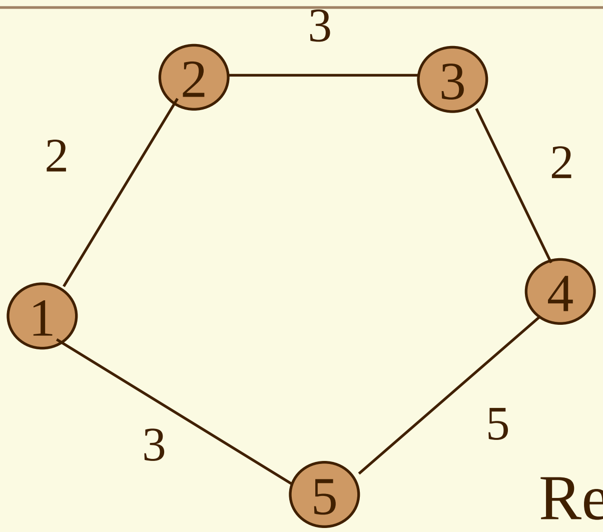




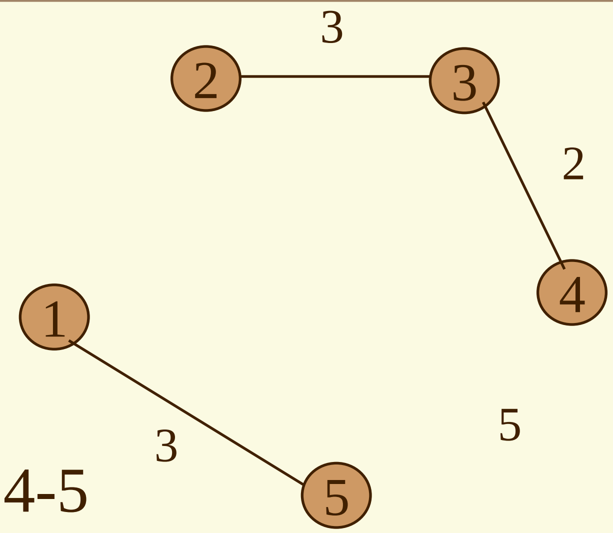
Add 1-3 and 2-4

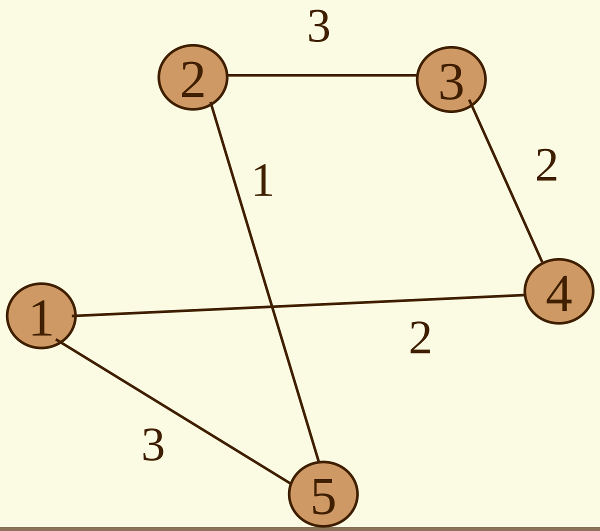
cost=18

Traveling Salesman Problem



Remove 1-2 and 4-5





Add 1-4 and 2-5

cost=11

Traveling Salesman Problem

There are *n* (*n*-1)/2 - n possible ways of removing and adding arcs. In this case, *n*=5, and we have 5 combinations.

The best tour obtained from the process at the first iteration is 2-3-4-1-5-2 with cost 11.

We use this solution as the starting tour and apply the process again.

Vehicle Routing Problem

The vehicle routing problem involves finding a set of trips, one for each vehicle, to deliver known quantities of goods to a set of customers.

The objective is to minimize the travel costs of all trips combined.

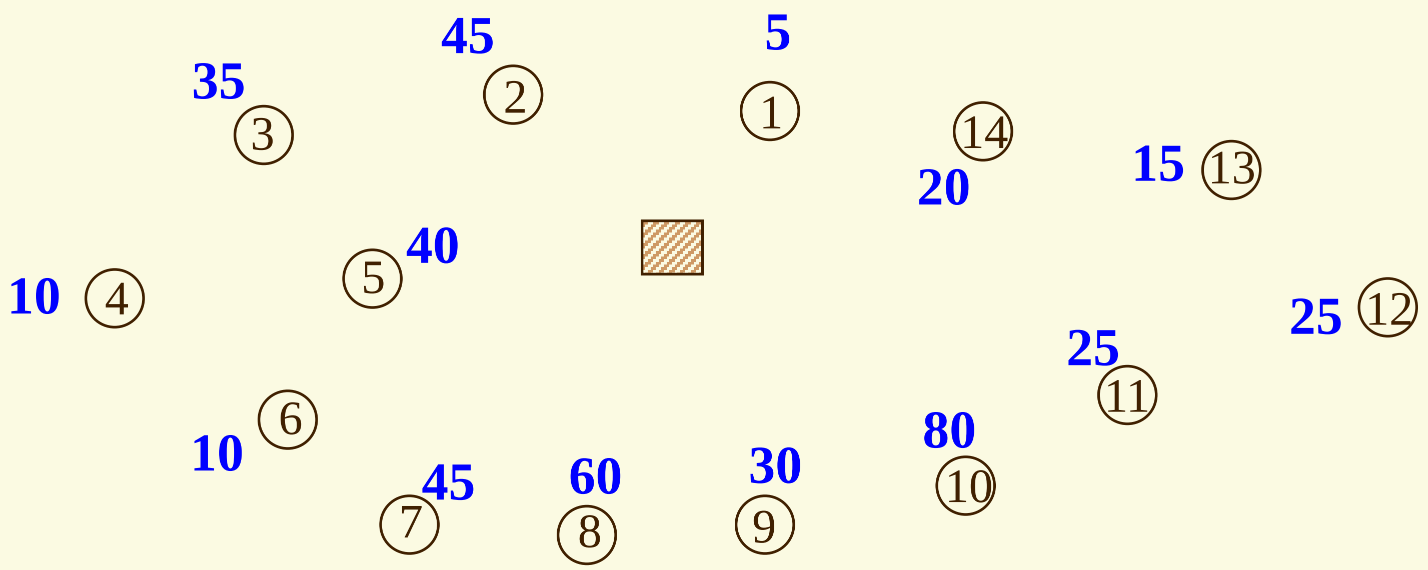
There may be upper bounds on the total load of each vehicle and the total duration of its trip.

The most basic Vehicle Routing Problem (VRP) is the single-depot capacitated VRP.

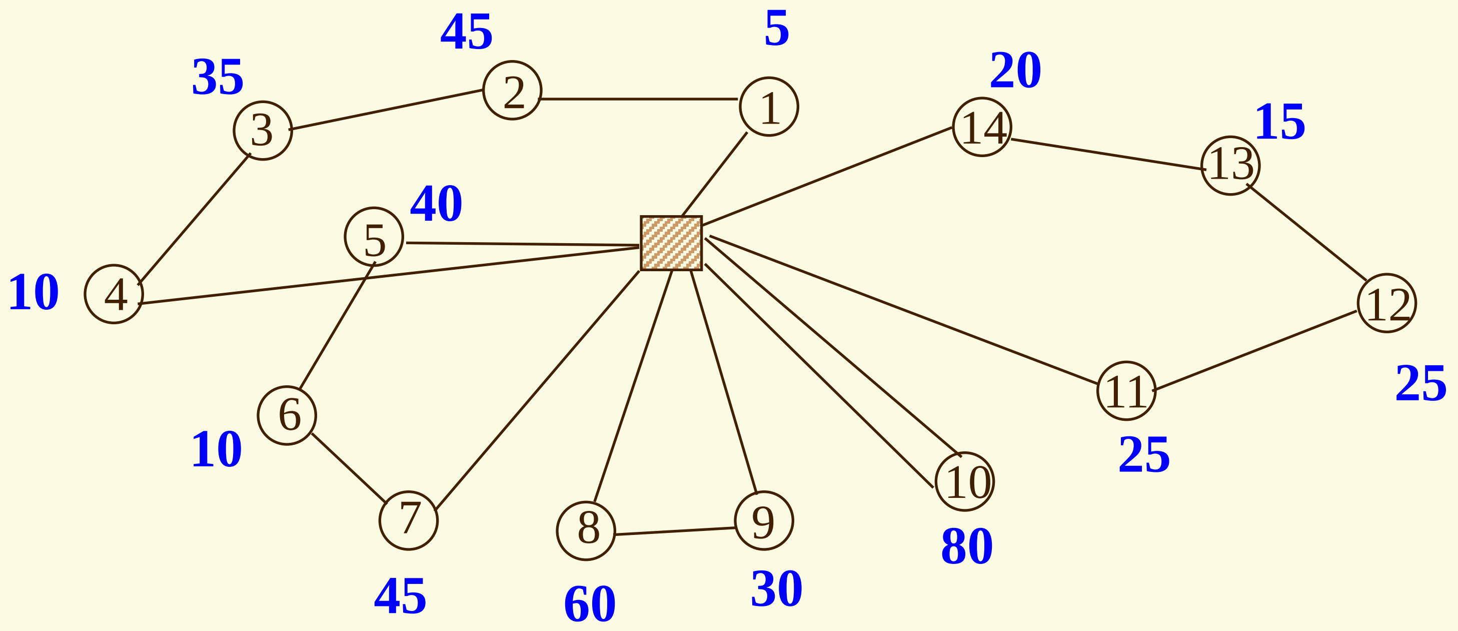
Vehicle Routing Problem

Inputs:		
	n	= number of customers
	K	= number of vehicles available
	c_{ij}	= cost of traveling from customer i to j
	d_i	= demand of customer i
	Q_k	= capacity of vehicle k

n=14, K=5, Q_k=100



Vehicle Routing Problem



Each customer must be visited by a vehicle

The demand assigned to vehicle *k* must not exceed its capacity

The tour for vehicle *k* begins at the depot, visits all its customers and returns to the depot.

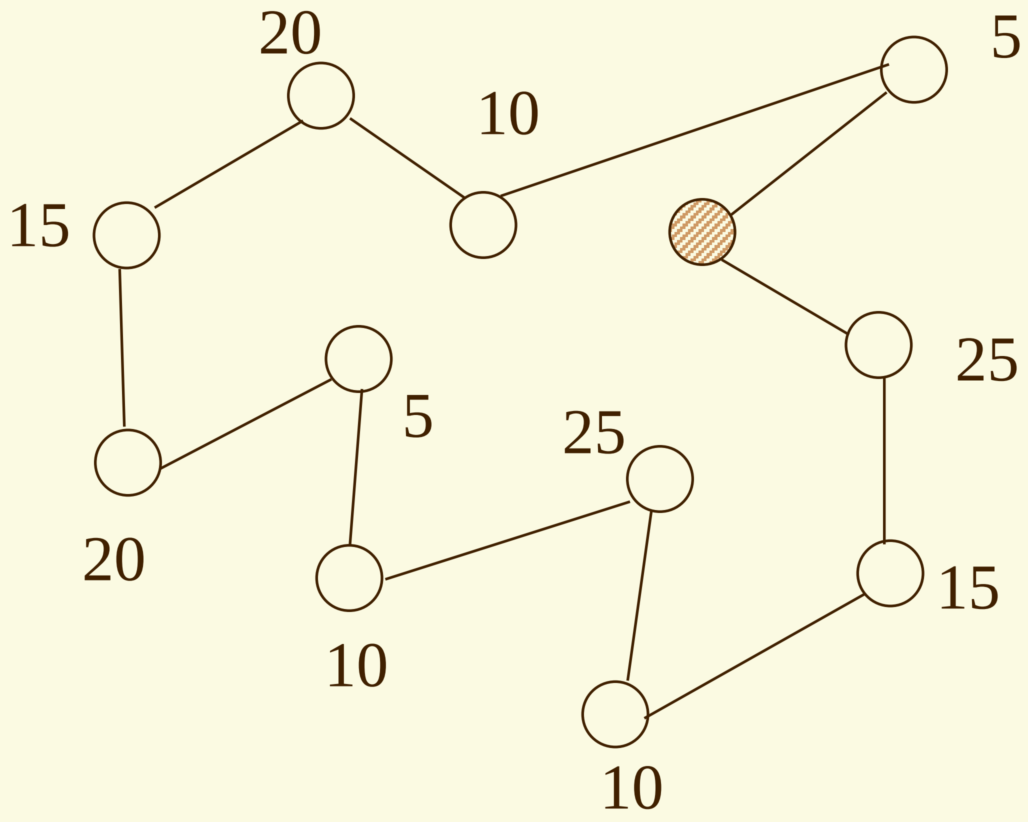
Vehicle Routing Problem

Iterated Tour Partitioning Heuristic

1. Find the traveling salesman tour through all customers and the depot.

2. Starting at the depot and following the tour in any arbitrary orientation, partition the path into disjoint segments such that the total demand in each segment does not exceed the vehicle capacity *Q*.

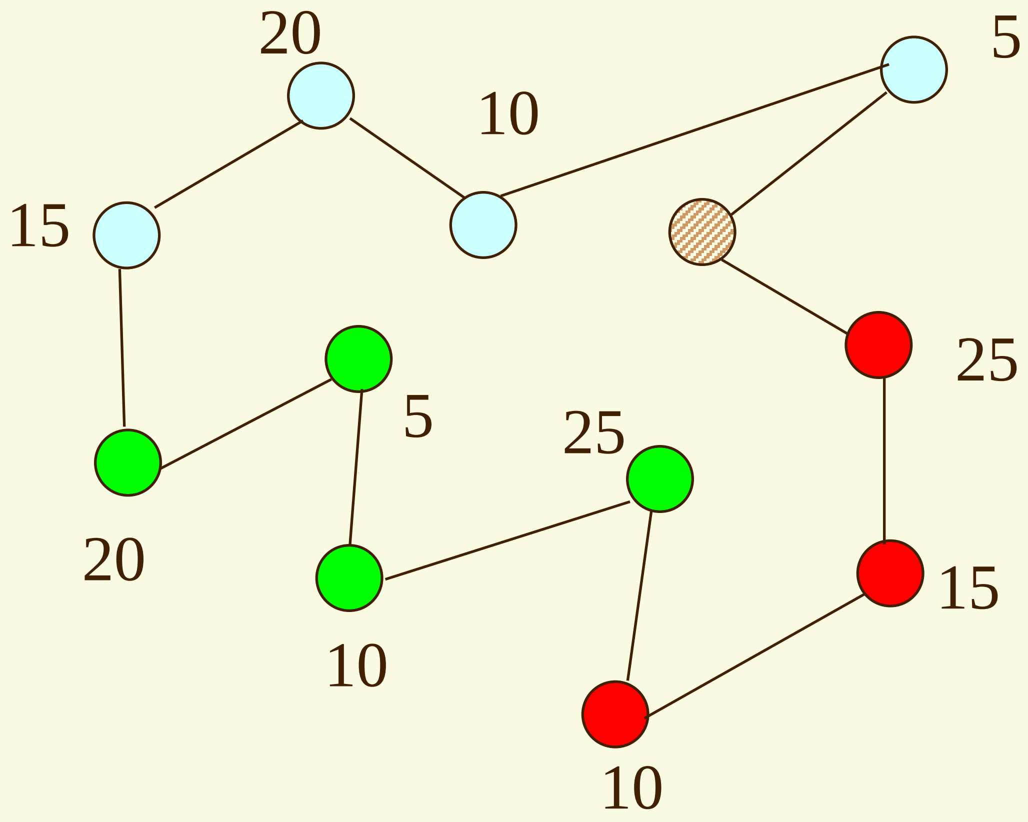
Vehicle Routing Problem



Q=60

The TSP tour

Vehicle Routing Problem



Q=60

Vehicle Routing Problem

Sweep Heuristic (Gillett and Miller)

1. Calculate the polar coordinates of all customers where the center is the depot and an arbitrary customer is chosen to be at angle 0. Reorder the customers so that

$$0 = \theta_1 \leq \theta_2 \leq \ldots \leq \theta_n$$

2. Starting from the unrouted customer *i* with smallest angle θ_i construct a new cluster by sweeping consecutive customers i+1, i+2 ... until the capacity constraint will not allow the next customer to be added.

3. Continue Step 2 until all customers are included in a cluster.

4. For each cluster constructed, solve the TSP on the subset of customers and the depot.

Vehicle Routing Problem

Clarke and Wright Savings Heuristic

1. Form subtours *i*-0-*i* for i=1,2,...,*n*. (Each customer is visited by a separate vehicle)

2. Compute savings $s_{ij}=c_{0i} + c_{0j} - c_{ij}$ for all *i*,*j*

3. Identify the node pair (*i*,*j*) that gives the highest saving s_{ij}

4. Form a new subtour by connecting (*i*,*j*) and deleting arcs (*i*,0) and (0, *j*) if the following conditions are satisfied

   a) both node *i* and node *j* have to be directly accessible from node 0

   b) node *i* and node *j* are not in the same tour.

   c) forming the new subtour does not violate any of the constraints associated with the vehicles

Vehicle Routing Problem

Clarke and Wright Savings Heuristic

5. Set $s_{ij}=-\infty$, which means that this node pair is processed.

Go to Step 3, unless all node pairs with $s_{ij} \geq 0$ are processed.

Vehicle Routing Problem

A chain of convenience stores has seven locations in one city. Each week goods must be delivered from a central warehouse to the stores. Items are packaged in standard-sized containers. The following table gives he number of containers that must be delivered for one week and the one-way travel times in minute between each pair of customers. Each delivery vehicle has a capacity for 80 containers. The company would like to make all deliveries within an 8-hour shift on a single day of the week.

Vehicle Routing Problem

i / *j*	0	1	2	3	4	5	6	7	Req
0	-	20	57	51	50	10	15	90	
1	20	-	51	10	55	25	30	53	46
2	57	51	-	50	20	30	10	47	55
3	51	10	50	-	50	11	60	38	33
4	50	55	20	50	-	50	60	10	30
5	10	25	30	11	50	-	20	90	24
6	15	30	10	60	60	20	-	12	75
7	90	53	47	38	10	90	12	-	30

When each customer is serviced individually from the depot, the total travel time becomes $2\left(\sum_{i=1}^{7} t_{0j}\right) = 586 \text{ minutes}$

Vehicle Routing Problem

Savings

i / *j*	1	2	3	4	5	6	7
1	-						
2	26	-					
3	61	58	-				
4	15	87	51	-			
5	5	37	50	10	-		
6	5	62	6	5	5	-	
7	57	100	103	**130**	10	93	-

$$s_{47} = t_{04} + t_{07} - t_{47} = 50 + 90 - 10 = 130$$

Because the total demand is 60, we may combine these routes.

Vehicle Routing Problem

The next largest savings is s_{37}=103. If we try combine customer 3 with customer 7 (who is already combined with customer 4), the total demand on the route would be 93, which exceeds the vehicle capacity.

We therefore move on to the next-largest savings, s_{27}=100. Adding customer 2 also violates the capacity constraint.

So does the next-largest savings, s_{67}=93.

Vehicle Routing Problem

i / j	1	2	3	4	5	6	7	Req
1	-							46
2	26	-						55
3	**61**	58	-					33
4	15	**87**	51	-				30
5	5	37	50	10	-			24
6	5	**62**	6	5	5	-		75
7	57	**100**	**103**	**130**	10	93	-	30

Route:	0-4-7-0	0-3-1-0	0-2-5-0	0-6-0
Time:	150	81	97	30
Capacity	60	79	79	75