Introduction to Artificial Intelligence

COMP307 Planning and Scheduling 4: Routing

Outline

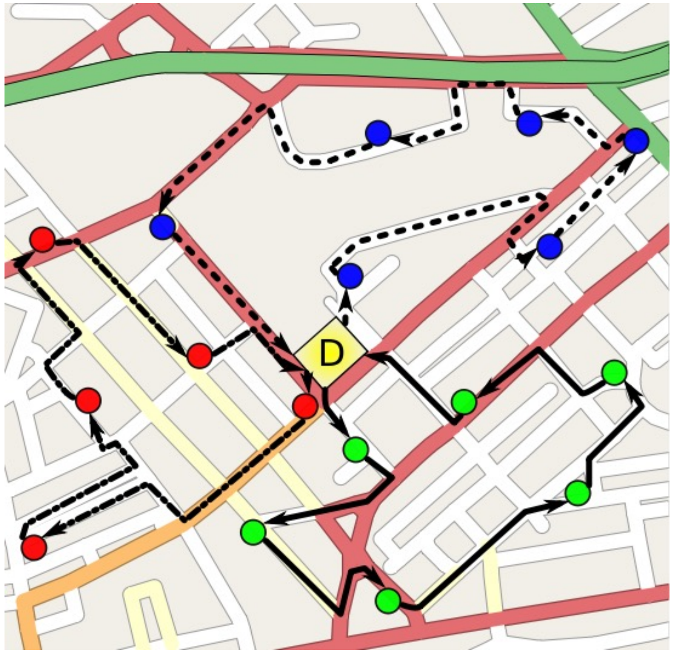
- Why Routing?
- Vehicle Routing Problem
- Problem and Solution Representation
- Some Heuristics

Why Routing?

- A lot of online shopping
- Delivery/Logistics

Why Routing

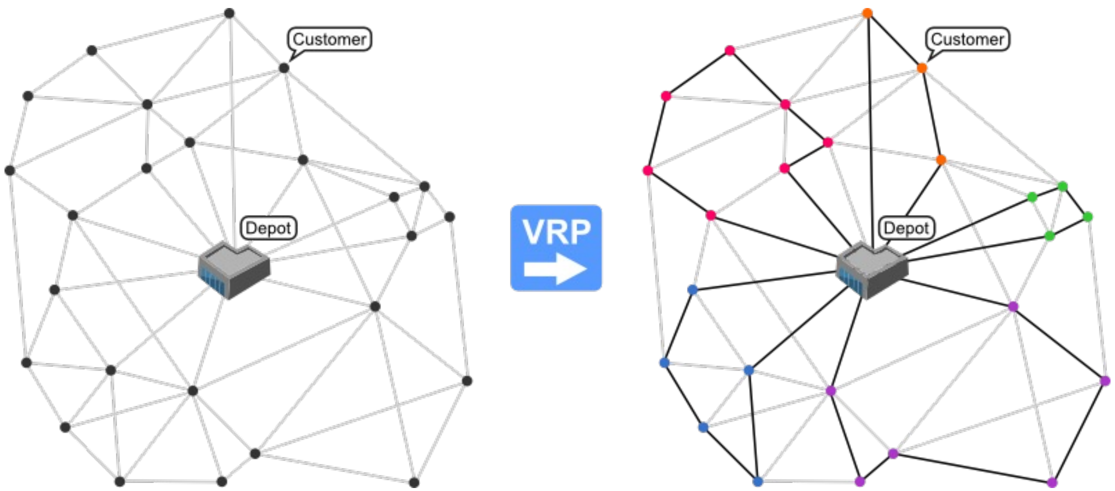
- Delivery
  - Online shopping
  - Food & Milk
  - Newspapers & Post
  - …

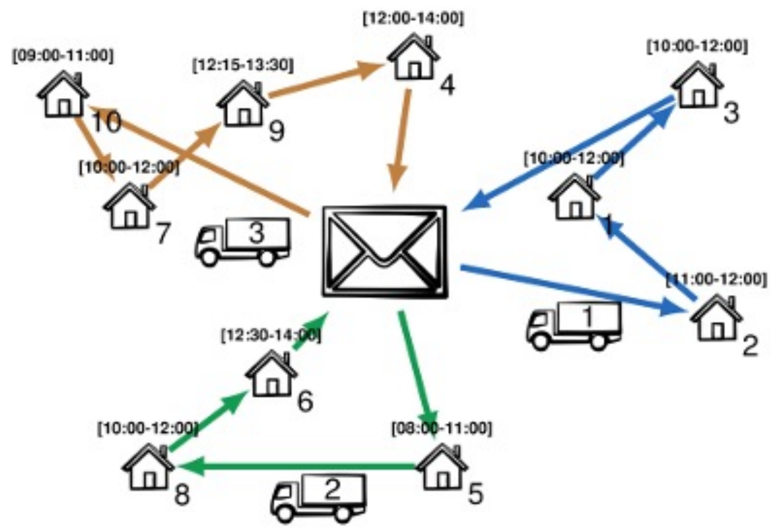


- Logistics & Transportation
  - Waste Collection
  - …
- Dynamic Analysis and Replanning Tool by US forces in 1990s.
  - 50,000 vehicles, cargo, people...
  - This single application more than paid back DARPA's 30-year investment in AI.
- Wellington Free Ambulance

Vehicle Routing

- A company delivering products to customers with its vehicles
  - All the products and vehicles are located at a depot
  - Each customer has a location and a demand (the amount of products requested)
  - The vehicles have limited capacity
  - **Objective**: Find the shortest routes to serve the costumers

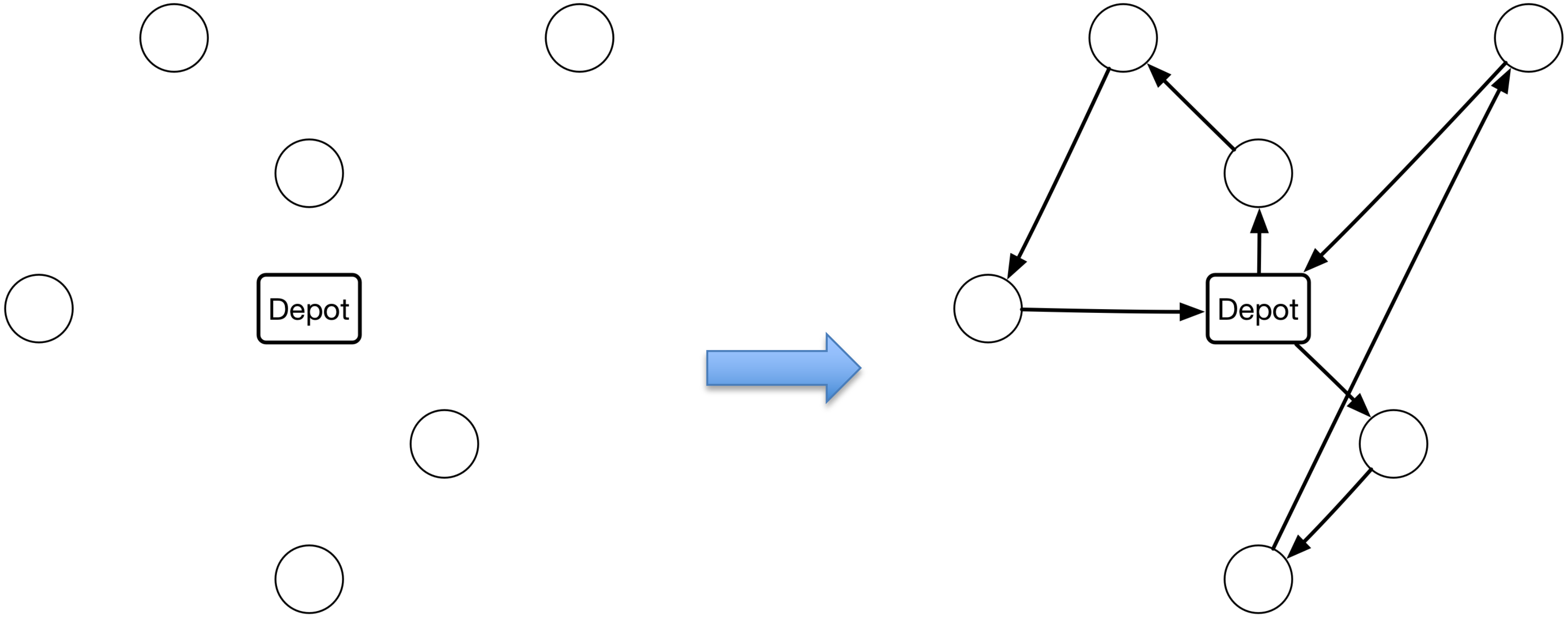




Vehicle Routing

- Problem:
  - A graph with the node set and the edge set
  - A special depot node
  - Each edge has a cost (length, travel time, …)
  - Each node except depot has a demand (customer demand)
  - Vehicles with limited capacity
- Find a **solution**:
  - A set of routes, each for a vehicle
  - Each node is visited exactly once
  - Each route starts and ends at the depot (cycle)
  - The total demand of the nodes in each route does not exceed capacity
- **Objective**: minimize the total cost of the routes

Vehicle Routing: An Example



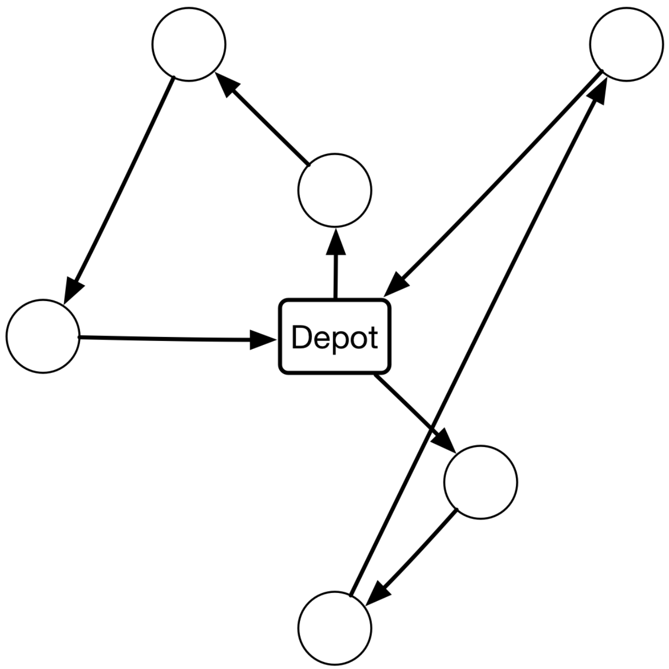
Node demand = 1
Capacity = 3

Problem/Graph

Node demand = 1
Capacity = 3

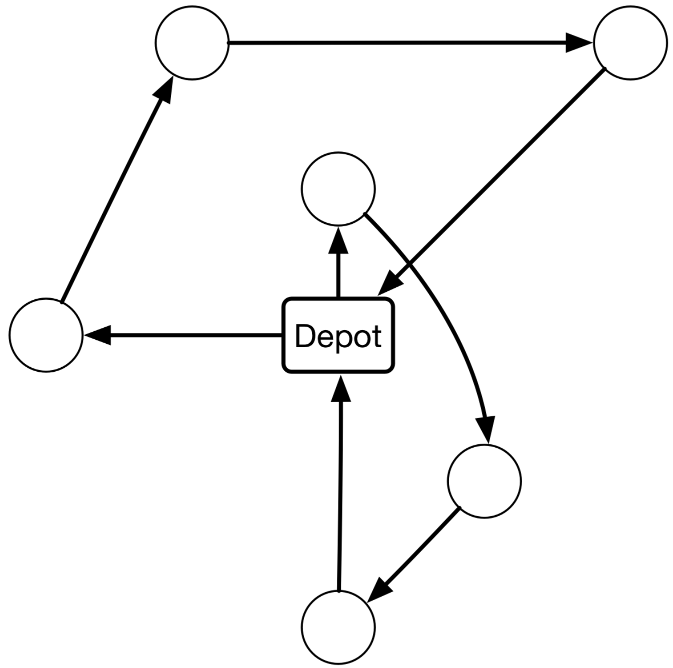
Solution

Vehicle Routing: An Example



Node demand = 1
Capacity = 3

Solution 1



Node demand = 1
Capacity = 3

Solution 2

- Which one is better/shorter?

Vehicle Routing is Hard

- Too many solutions
  - 10 nodes (excluding depot), 1 vehicle (TSP)
  - 3.6 million solutions
- **NP-hard**
  - Cannot guarantee to find the optimal solution in reasonable time

- How to Solve Vehicle Routing
  - Heuristics: Search for a reasonably good solution in a given (short) time
  - This lecture:
    - Nearest Neighbor heuristic
    - Savings heuristic

A Typical Problem Description

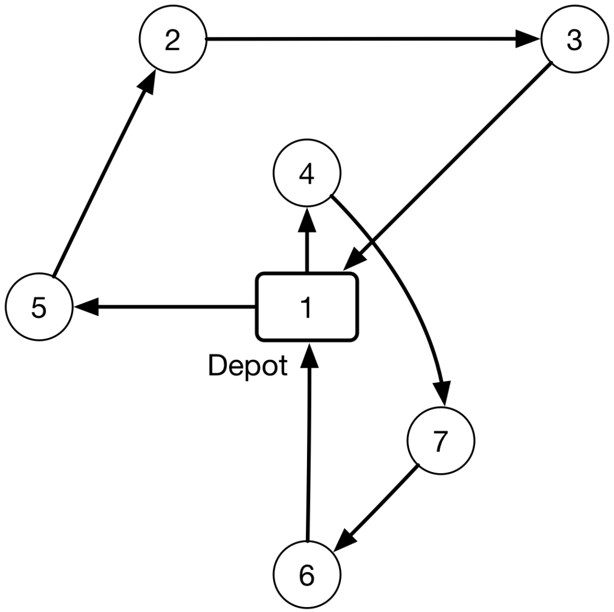
Node	X-coord	Y-coord	Demand
1 (depot)	82	76	0
2	96	44	19
3	59	5	21
4	49	8	6
…	…	…	…

- Depot has index of 1 (or 0), and demand of 0
- Unique edge between each pair of nodes
  - Edge length = Euclidean distance

$$L(i,j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$$

Solution Representation

- A set of routes, each route is a sequence of nodes
  - Start and end at the depot
  - (**1**, 3, 5, 2, **1**) …



R1	1	5	2	3	1
R2	1	4	7	6	1

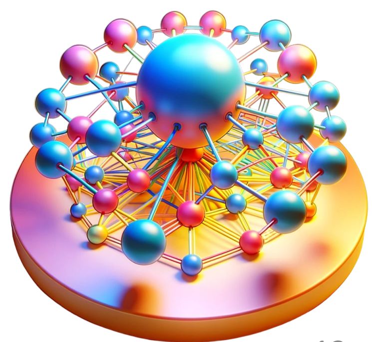
$$Cost(R_1) = L(1,5) + L(5,2) + L(2,3) + L(3,1)$$

$$Cost(R_2) = L(1,4) + L(4,7) + L(7,6) + L(6,1)$$

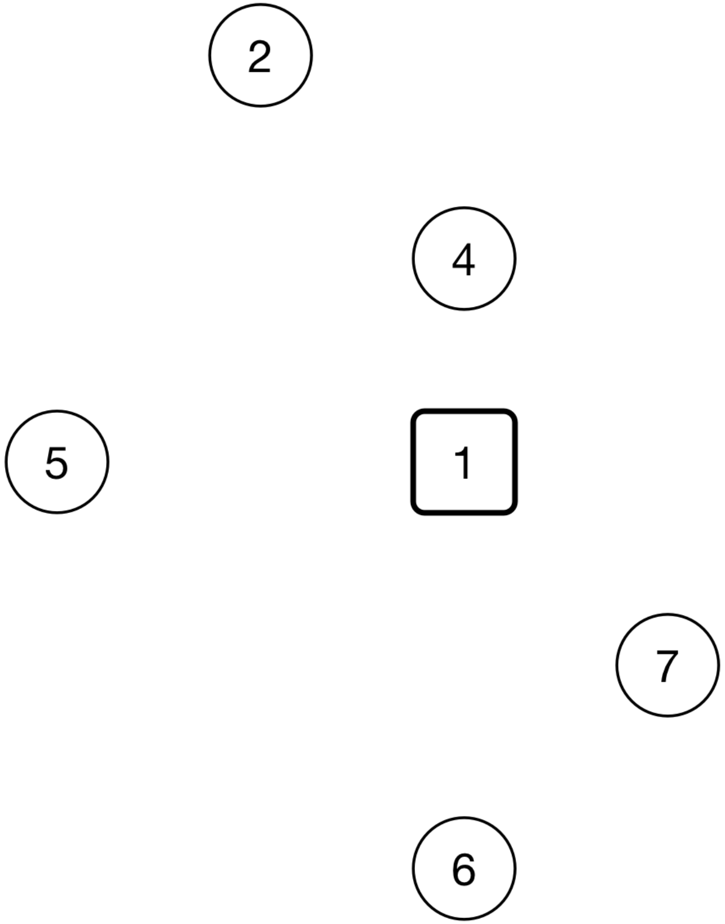
$$Cost(S) = Cost(R_1) + Cost(R_2)$$

Nearest Neighbor Heuristic

1. Initialize a solution: a route starting from the depot
2. Append the nearest feasible node to the end of the current route
   – Unvisited
   – After inserting the node, the total demand of the route does not exceed the capacity
3. If no feasible node is found for the current route, then close the current route (return to the depot) and create a new route starting from the depot
4. Repeat 2. and 3. until all nodes are visited



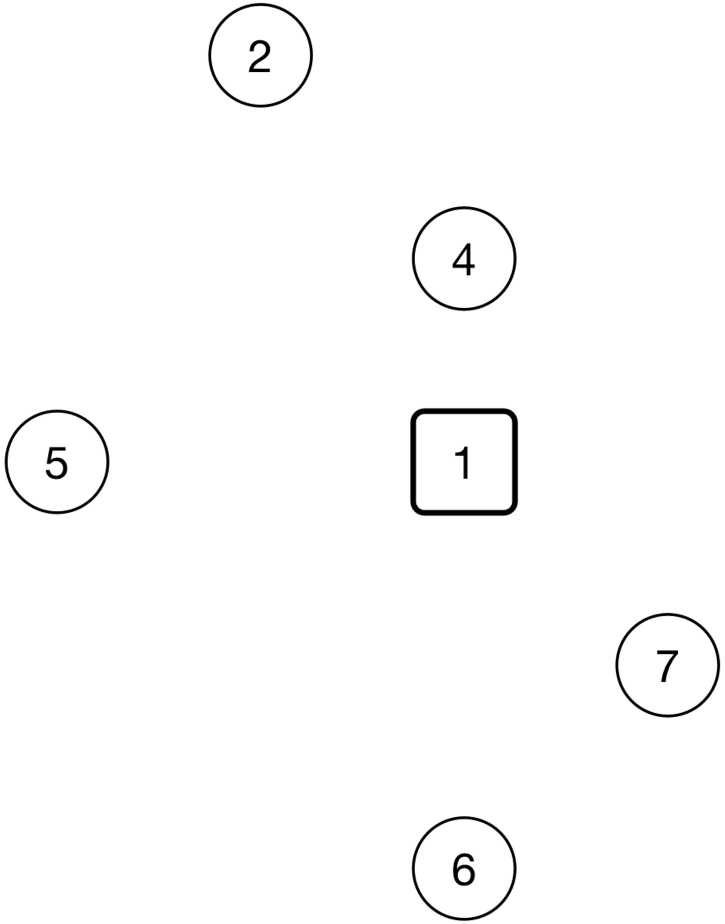
Nearest Neighbour Heuristic



	1	2	3	4	5	6	7
1	0	2.2	2.8	1	2	2	1.4
2	2.2	0	3	1.4	2.2	4.1	3.6
3	2.8	3	0	2.2	4.5	4.5	3.2
4	1	1.4	2.2	0	2.2	3	2.2
5	2	2.2	4.5	2.2	0	2.8	3.2
6	2	4.1	4.5	3	2.8	0	1.4
7	1.4	3.6	3.2	2.2	3.2	1.4	0

Capacity = 3

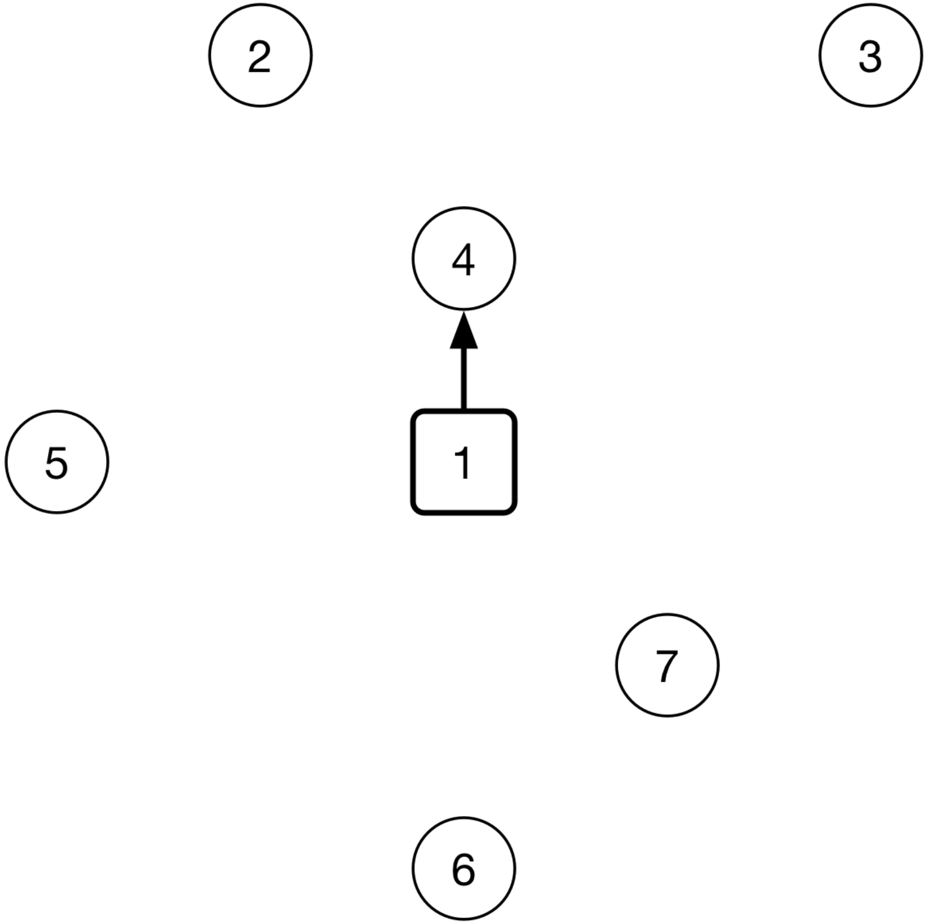
Nearest Neighbour Heuristic



	1	2	3	4	5	6	7
1	0	2.2	2.8	1	2	2	1.4
2	2.2	0	3	1.4	2.2	4.1	3.6
3	2.8	3	0	2.2	4.5	4.5	3.2
4	1	1.4	2.2	0	2.2	3	2.2
5	2	2.2	4.5	2.2	0	2.8	3.2
6	2	4.1	4.5	3	2.8	0	1.4
7	1.4	3.6	3.2	2.2	3.2	1.4	0

Capacity = 3

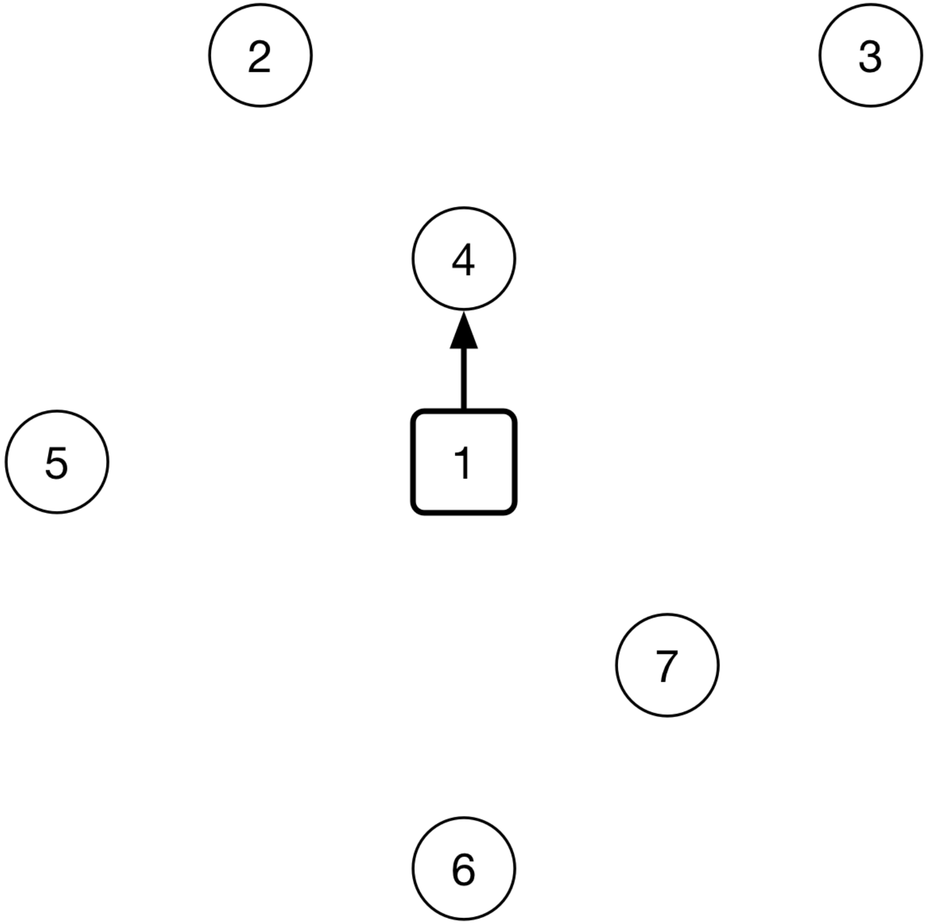
Nearest Neighbour Heuristic



	1	2	3	4	5	6	7
1	0	2.2	2.8	**1**	2	2	1.4
2	2.2	0	3	~~1.4~~	2.2	4.1	3.6
3	2.8	3	0	~~2.2~~	4.5	4.5	3.2
4	1	1.4	2.2	~~0~~	2.2	3	2.2
5	2	2.2	4.5	~~2.2~~	0	2.8	3.2
6	2	4.1	4.5	~~3~~	2.8	0	1.4
7	1.4	3.6	3.2	~~2.2~~	3.2	1.4	0

Capacity = 3

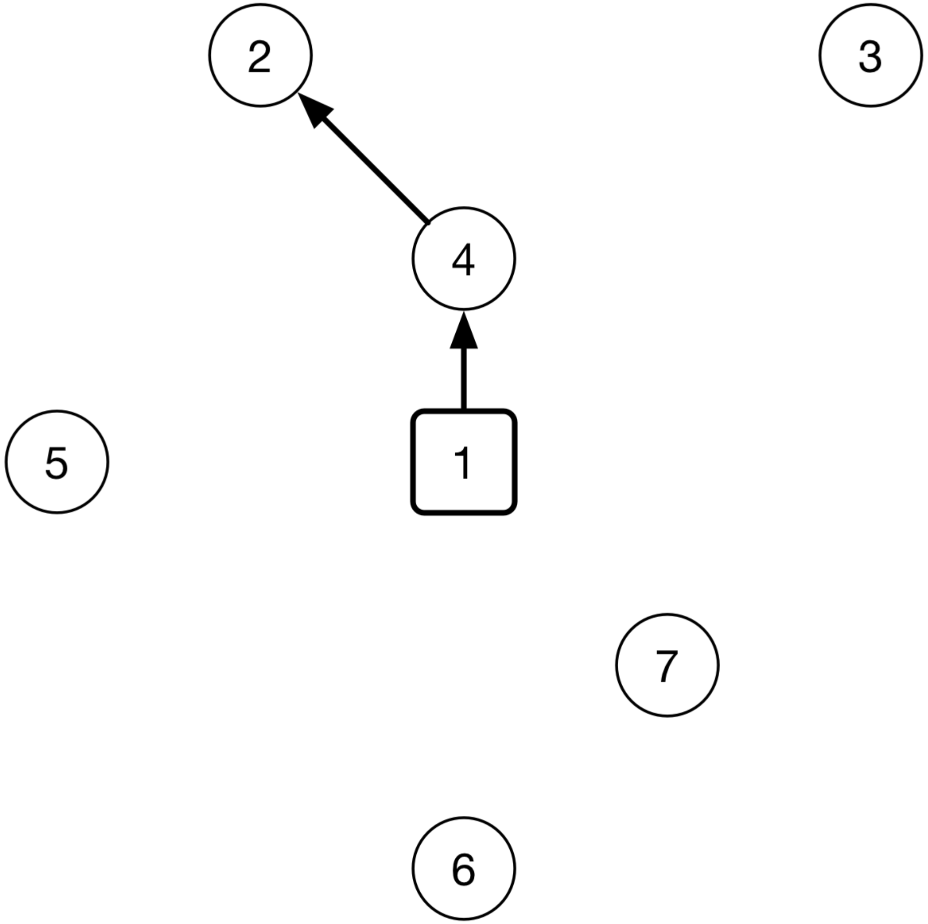
Nearest Neighbour Heuristic



	1	2	3	4	5	6	7
1	0	2.2	2.8	**1**	2	2	1.4
2	2.2	0	3	1.4	2.2	4.1	3.6
3	2.8	3	0	2.2	4.5	4.5	3.2
4	1	1.4	2.2	0	2.2	3	2.2
5	2	2.2	4.5	2.2	0	2.8	3.2
6	2	4.1	4.5	3	2.8	0	1.4
7	1.4	3.6	3.2	2.2	3.2	1.4	0

Capacity = 3

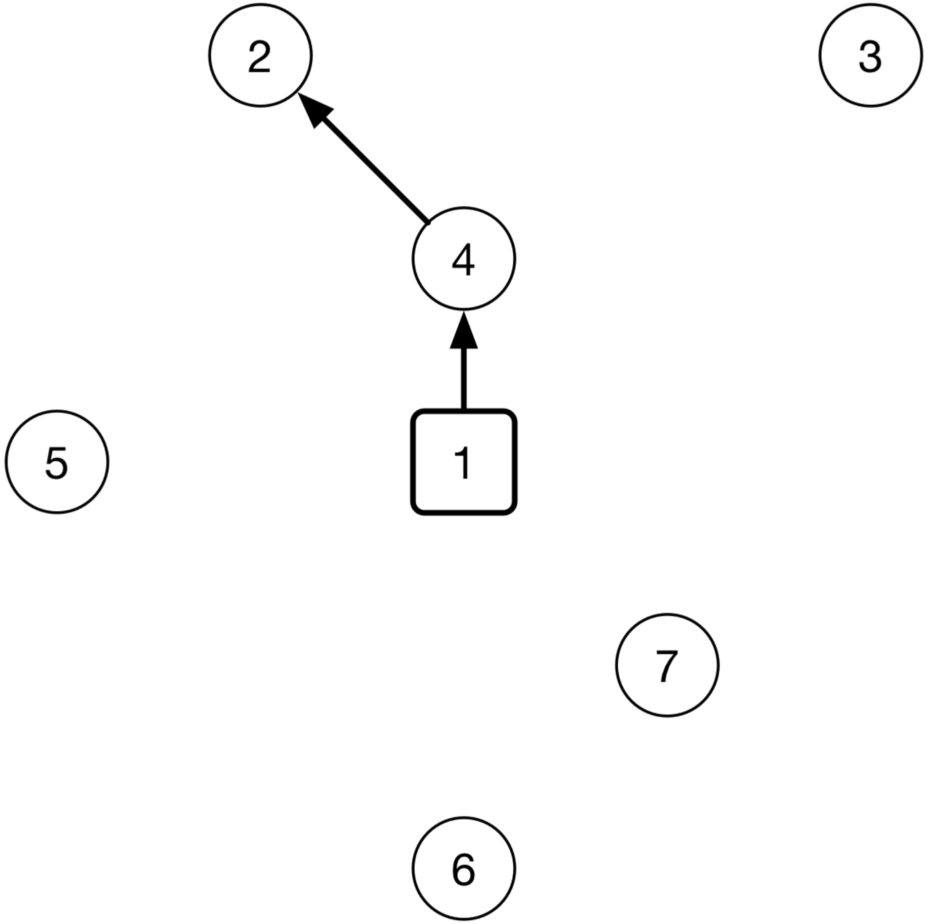
Nearest Neighbour Heuristic



	1	2	3	4	5	6	7
1	0	~~2.2~~	2.8	**1**	2	2	1.4
2	2.2	~~0~~	3	~~1.4~~	2.2	4.1	3.6
3	2.8	~~3~~	0	~~2.2~~	4.5	4.5	3.2
4	1	**1.4**	2.2	~~0~~	2.2	3	2.2
5	2	~~2.2~~	4.5	~~2.2~~	0	2.8	3.2
6	2	~~4.1~~	4.5	~~3~~	2.8	0	1.4
7	1.4	~~3.6~~	3.2	~~2.2~~	3.2	1.4	0

Capacity = 3

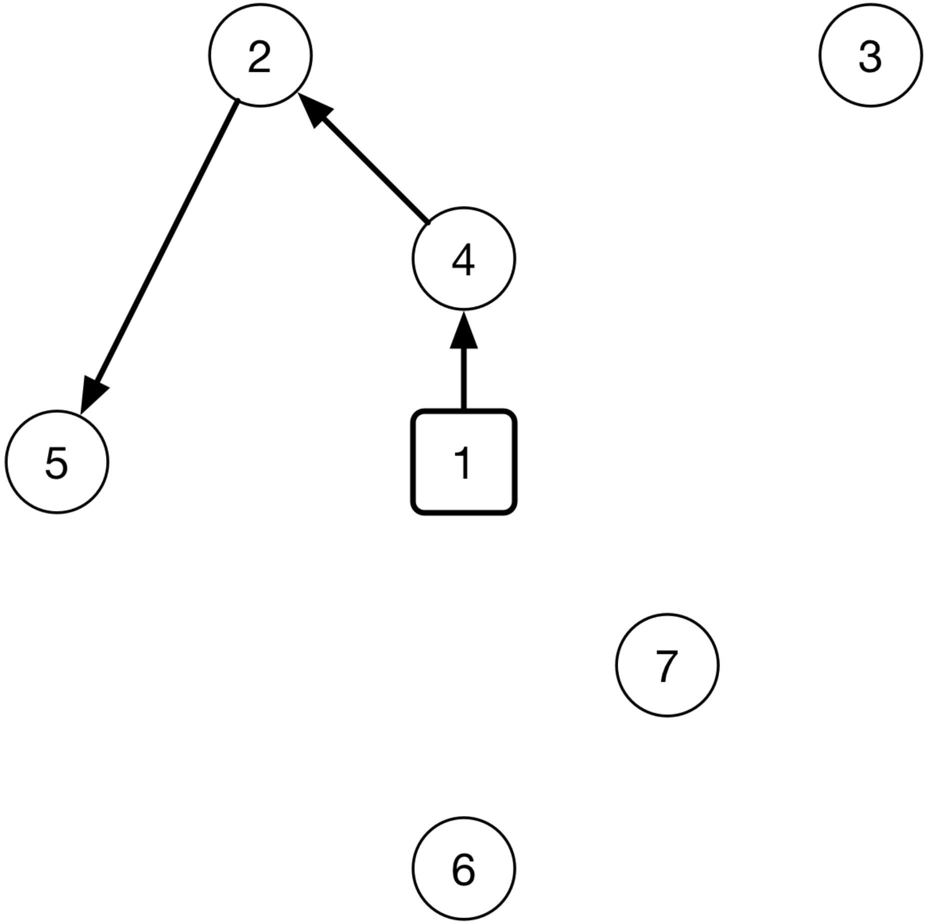
Nearest Neighbour Heuristic



	1	2	3	4	5	6	7
1	0	2.2	2.8	**1**	2	2	1.4
2	2.2	0	3	1.4	2.2	4.1	3.6
3	2.8	3	0	2.2	4.5	4.5	3.2
4	1	**1.4**	2.2	0	2.2	3	2.2
5	2	2.2	4.5	2.2	0	2.8	3.2
6	2	4.1	4.5	3	2.8	0	1.4
7	1.4	3.6	3.2	2.2	3.2	1.4	0

Capacity = 3

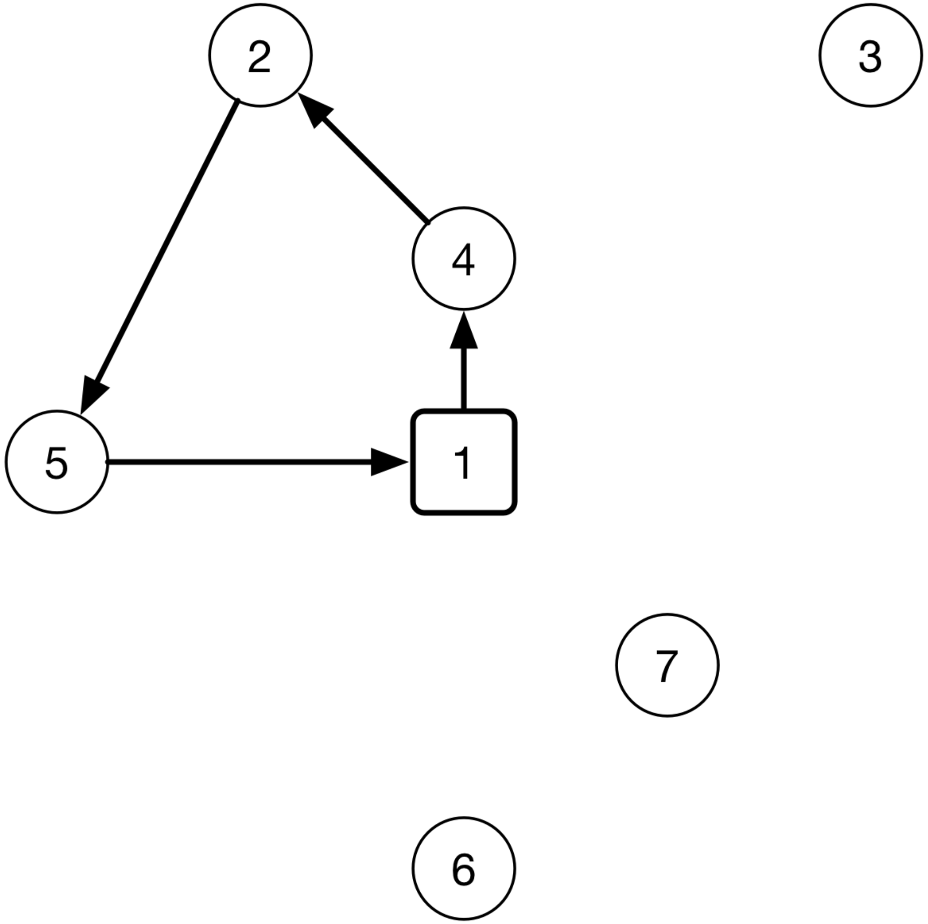
Nearest Neighbour Heuristic



	1	2	3	4	5	6	7
1	0	2.2	2.8	**1**	2	2	1.4
2	2.2	0	3	1.4	**2.2**	4.1	3.6
3	2.8	3	0	2.2	4.5	4.5	3.2
4	1	**1.4**	2.2	0	2.2	3	2.2
5	2	2.2	4.5	2.2	0	2.8	3.2
6	2	4.1	4.5	3	2.8	0	1.4
7	1.4	3.6	3.2	2.2	3.2	1.4	0

Capacity = 3

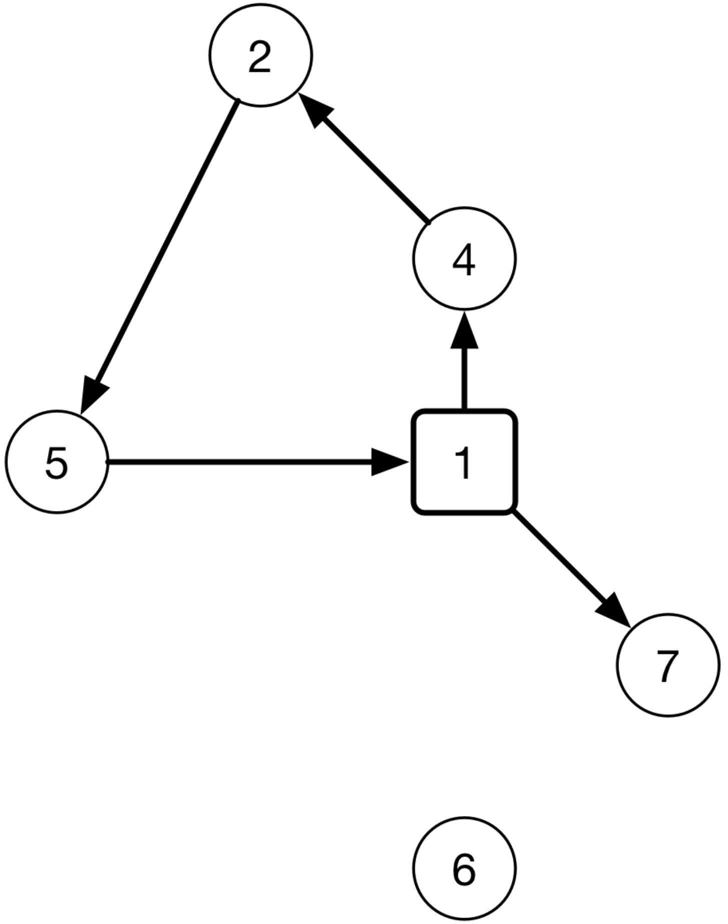
Nearest Neighbour Heuristic



	1	2	3	4	5	6	7
1	0	2.2	2.8	**1**	2	2	1.4
2	2.2	0	3	1.4	**2.2**	4.1	3.6
3	2.8	3	0	2.2	4.5	4.5	3.2
4	1	**1.4**	2.2	0	2.2	3	2.2
5	**2**	2.2	4.5	2.2	0	2.8	3.2
6	2	4.1	4.5	3	2.8	0	1.4
7	1.4	3.6	3.2	2.2	3.2	1.4	0

Capacity = 3

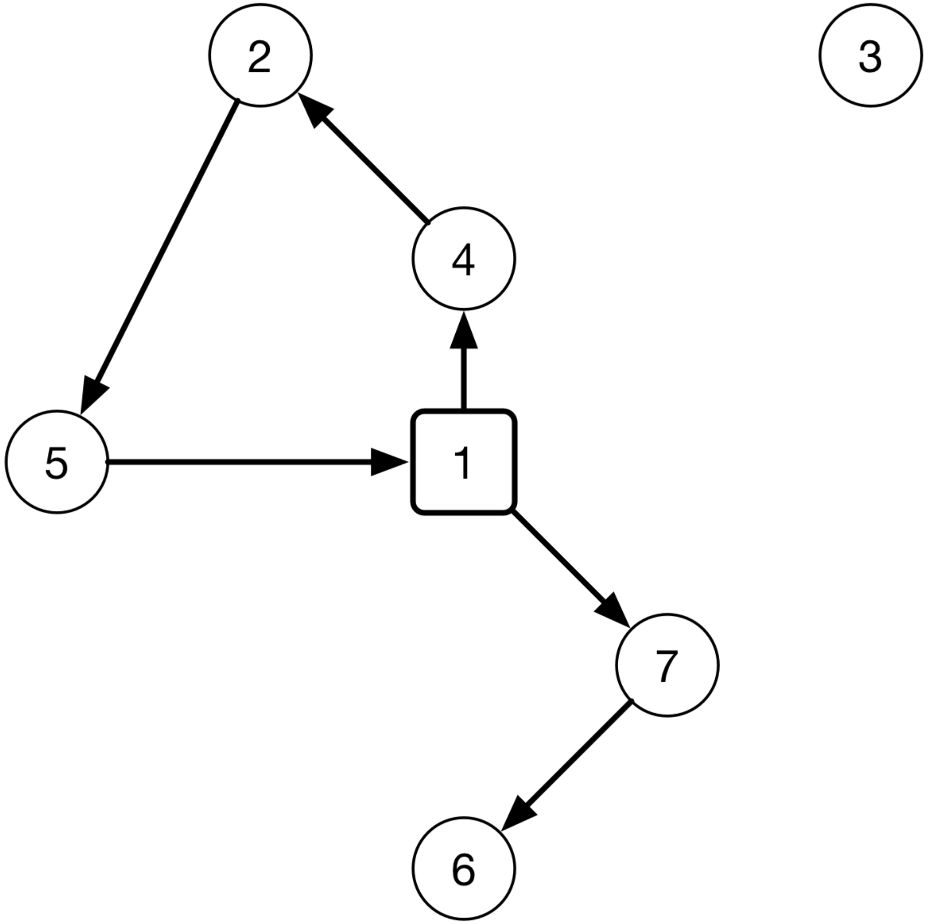
Nearest Neighbour Heuristic



	1	2	3	4	5	6	7
1	0	2.2	2.8	**1**	2	2	**1.4**
2	2.2	0	3	1.4	**2.2**	4.1	3.6
3	2.8	3	0	2.2	4.5	4.5	3.2
4	1	**1.4**	2.2	0	2.2	3	2.2
5	**2**	2.2	4.5	2.2	0	2.8	3.2
6	2	4.1	4.5	3	2.8	0	1.4
7	1.4	3.6	3.2	2.2	3.2	1.4	0

Capacity = 3

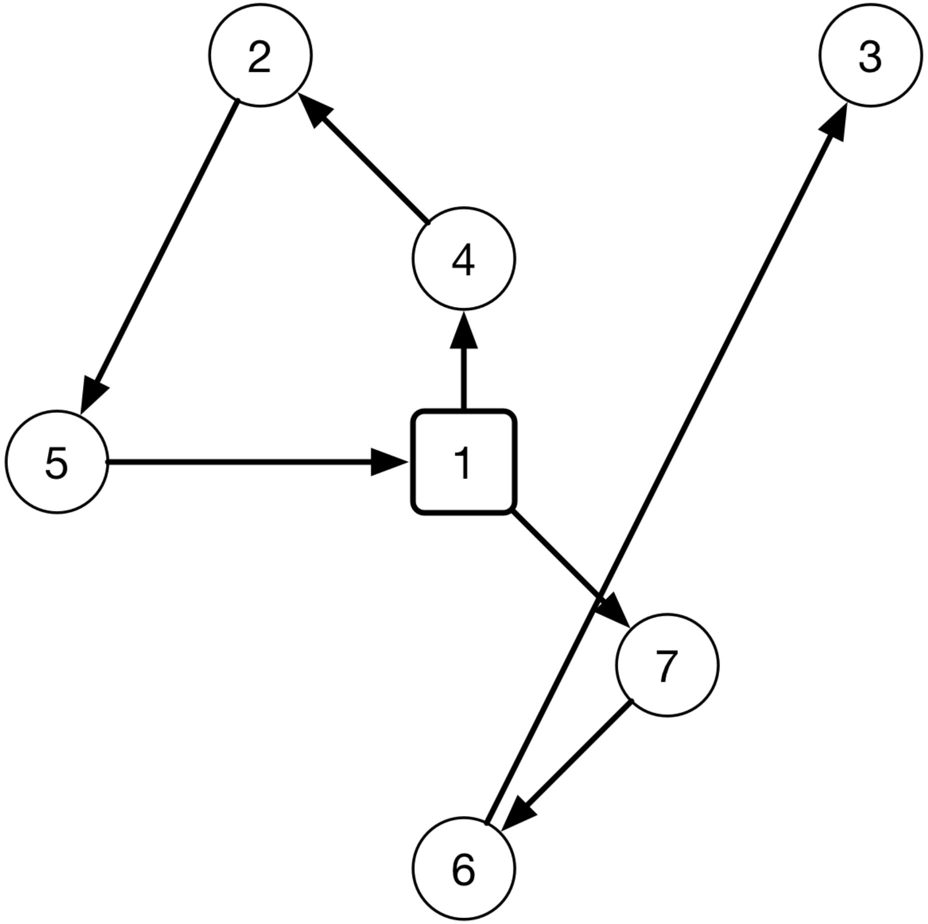
Nearest Neighbour Heuristic



	1	2	3	4	5	6	7
1	0	2.2	2.8	**1**	2	2	**1.4**
2	2.2	0	3	1.4	**2.2**	4.1	3.6
3	2.8	3	0	2.2	4.5	4.5	3.2
4	1	**1.4**	2.2	0	2.2	3	2.2
5	**2**	2.2	4.5	2.2	0	2.8	3.2
6	2	4.1	4.5	3	2.8	0	1.4
7	1.4	3.6	3.2	2.2	3.2	**1.4**	0

Capacity = 3

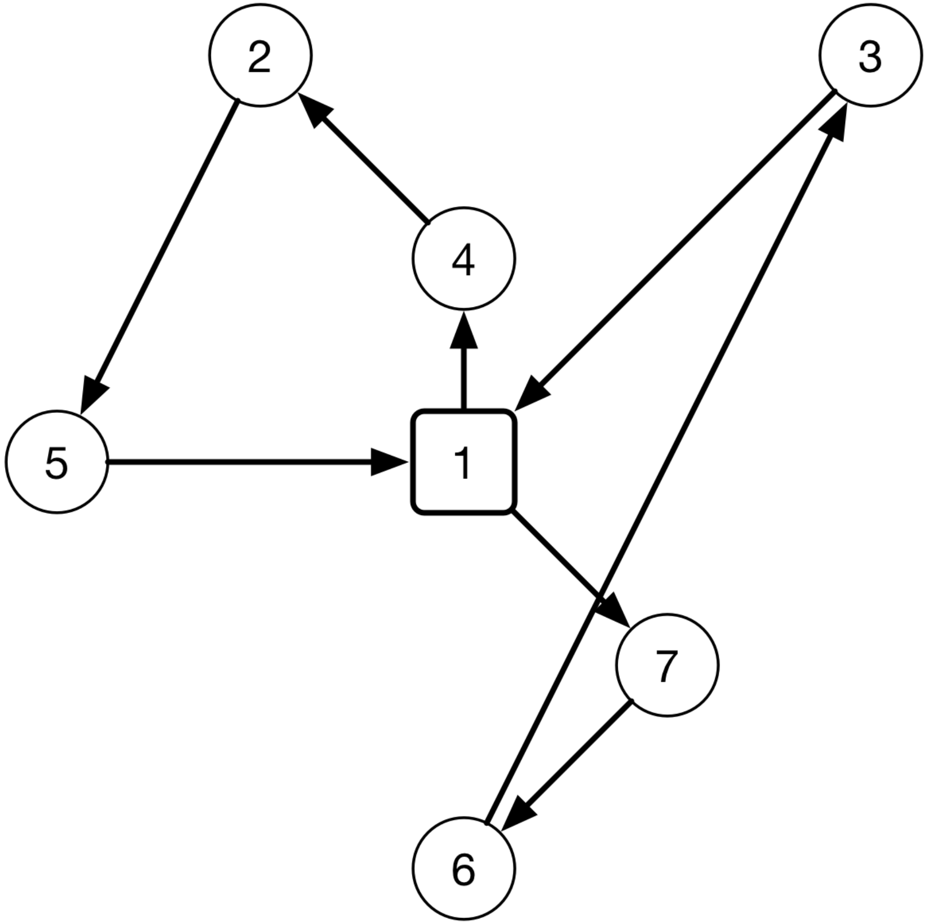
Nearest Neighbour Heuristic



	1	2	3	4	5	6	7
1	0	2.2	2.8	**1**	2	2	**1.4**
2	2.2	0	3	1.4	**2.2**	4.1	3.6
3	2.8	3	0	2.2	4.5	4.5	3.2
4	1	**1.4**	2.2	0	2.2	3	2.2
5	**2**	2.2	4.5	2.2	0	2.8	3.2
6	2	4.1	**4.5**	3	2.8	0	1.4
7	1.4	3.6	3.2	2.2	3.2	**1.4**	0

Capacity = 3

Nearest Neighbour Heuristic

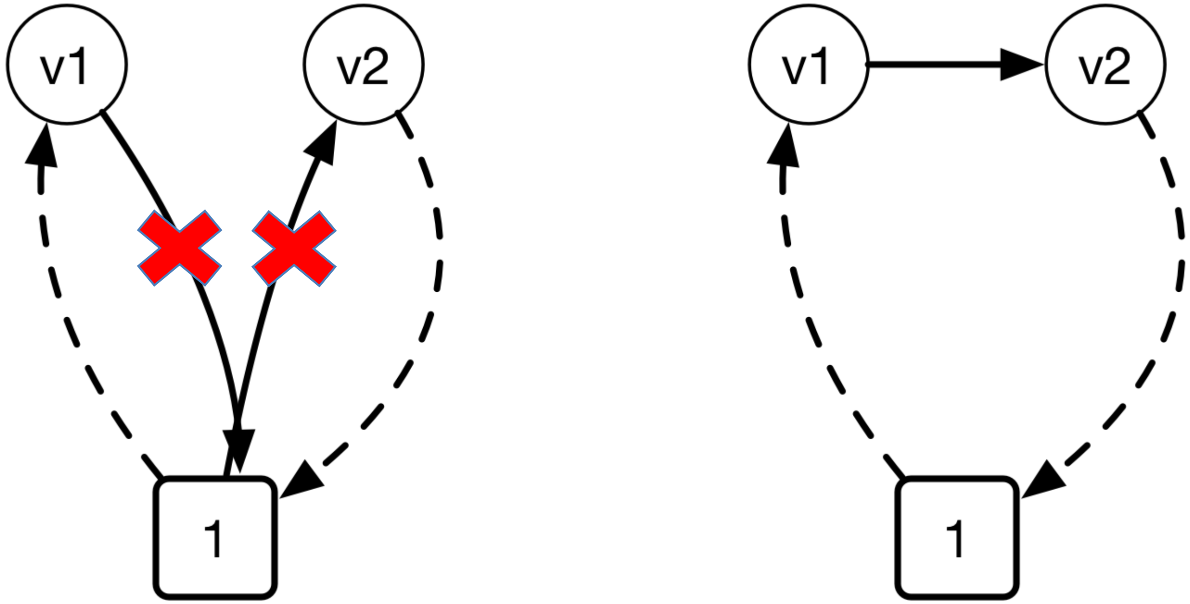


	1	2	3	4	5	6	7
1	0	2.2	2.8	**1**	2	2	**1.4**
2	2.2	0	3	1.4	**2.2**	4.1	3.6
3	**2.8**	3	0	2.2	4.5	4.5	3.2
4	1	**1.4**	2.2	0	2.2	2	2.2
5	**2**	2.2	4.5	2.2	0	2.8	3.2
6	2	4.1	**4.5**	2	2.8	0	1.4
7	1.4	3.6	3.2	2.2	3.2	**1.4**	0

Capacity = 3

Savings Heuristic

- Start with smallest cycles (depot -> node -> depot), and keep merging routes with the largest saving in cost



- Merge two routes

$$saving = L(v_1, 1) + L(1, v_2) - L(v_1, v_2)$$

Savings Heuristic

1. Initialize routes $(1, v_i, 1)$ for each node v_i except the depot
2. Compute and store the savings for each possible merge

$$saving(v_i, v_j) = L(v_i, 1) + L(1, v_j) - L(v_i, v_j)$$

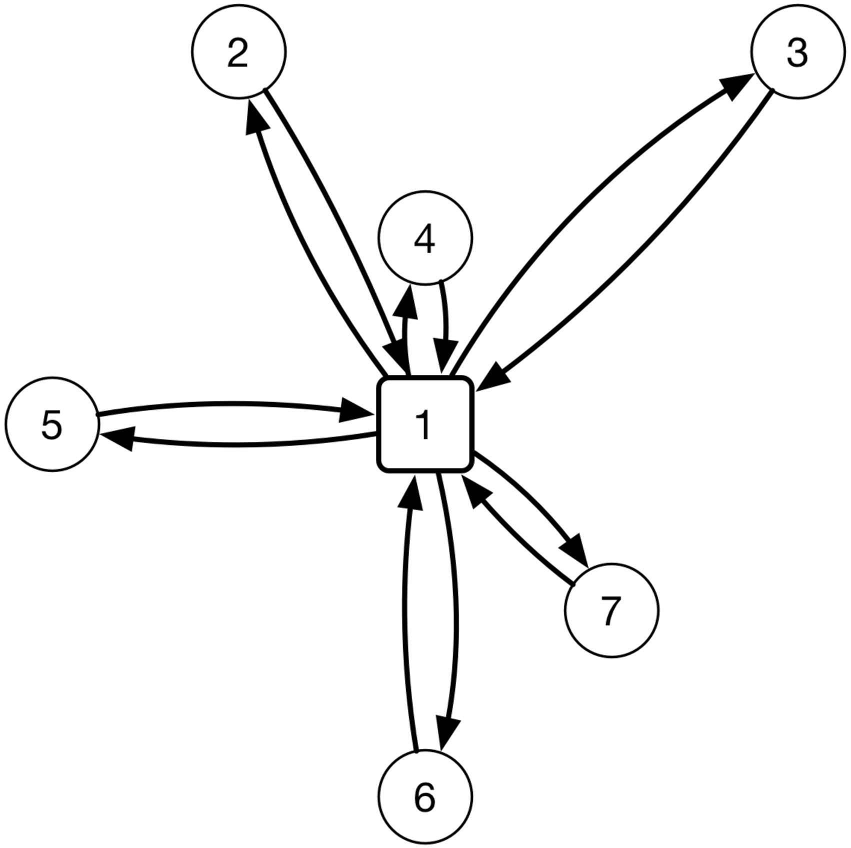
3. Check all the possible/feasible route merges
   – Merge $route_1$ and $route_2$: merge the last non-depot node of $route_1$ and the first non-depot node of $route_2$

$(1, 3, 5, 2, 1)$ $(1, 4, 7, 6, 1)$

$(1, 3, 5, 2, 4, 7, 6, 1)$

4. Select the merge with the largest saving and merge the routes
5. **Repeat** 3 and 4 until no more merge can be done

Savings Heuristic

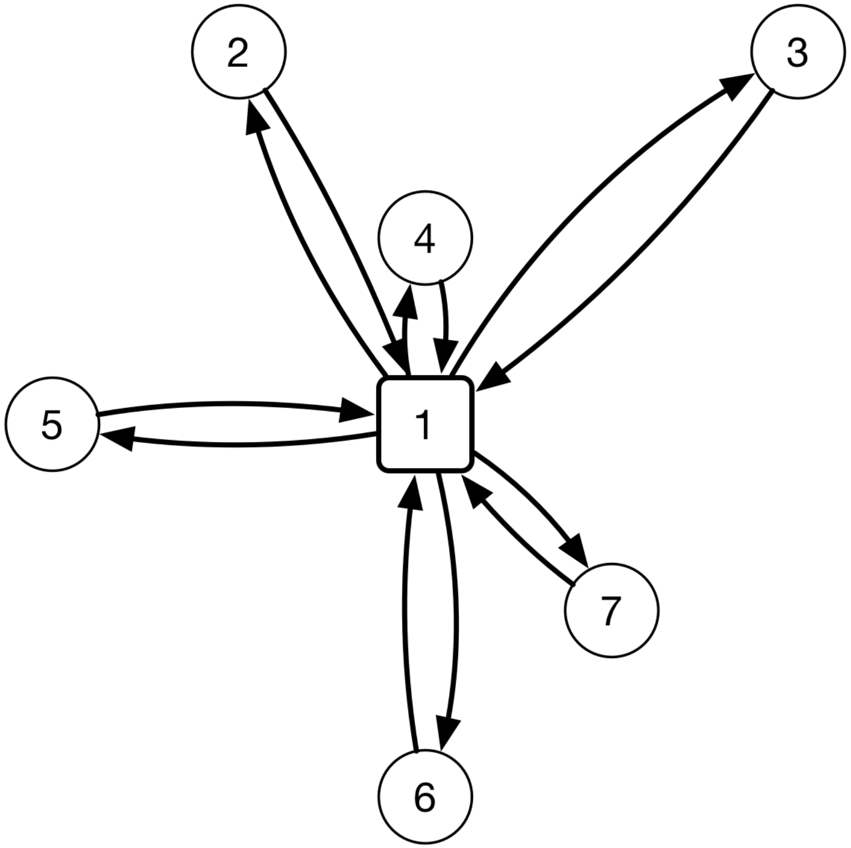


Feasible merges

Merge	Sav
(2,3)	2
(2,5)	2
(6,7)	2
(2,4)	1.8
(3,4)	1.6
…	…
(4,7)	0.2
…	…

Merge	Sav
(2,3)	2
(2,5)	2
(6,7)	2
(2,4)	1.8
(3,4)	1.6
…	…
(4,7)	0.2
…	…

Savings Heuristic

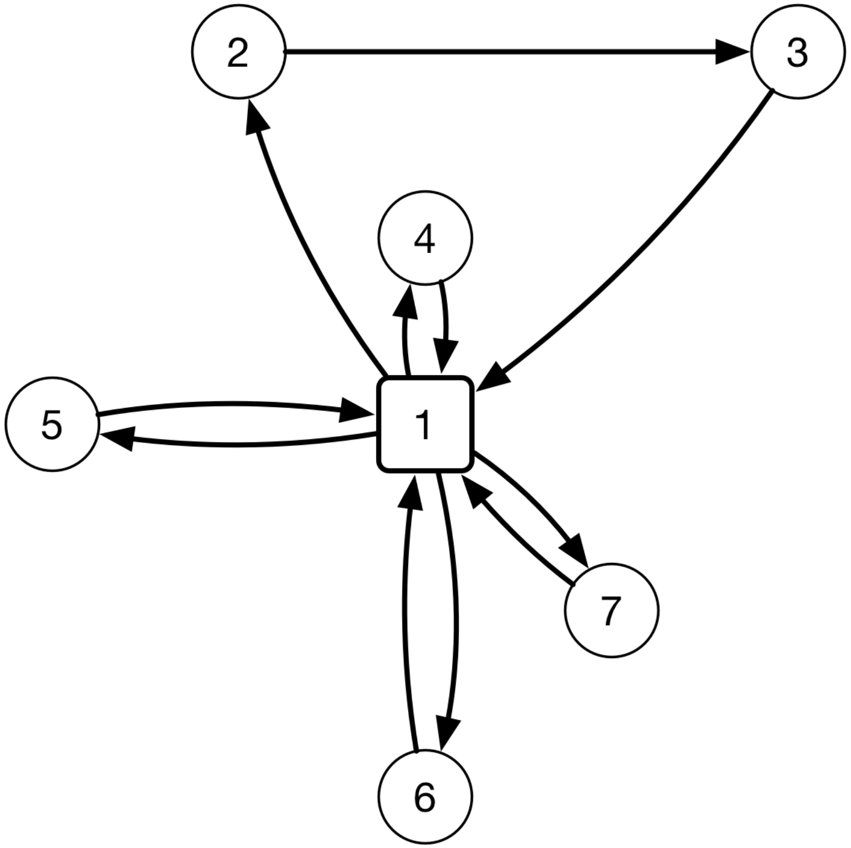


Feasible merges

Merge	Sav
(2,3)	2
(2,5)	2
(6,7)	2
(2,4)	1.8
(3,4)	1.6
…	…
(4,7)	0.2
…	…

Merge	Sav
(2,3)	2
(2,5)	2
(6,7)	2
(2,4)	1.8
(3,4)	1.6
…	…
(4,7)	0.2
…	…

Savings Heuristic

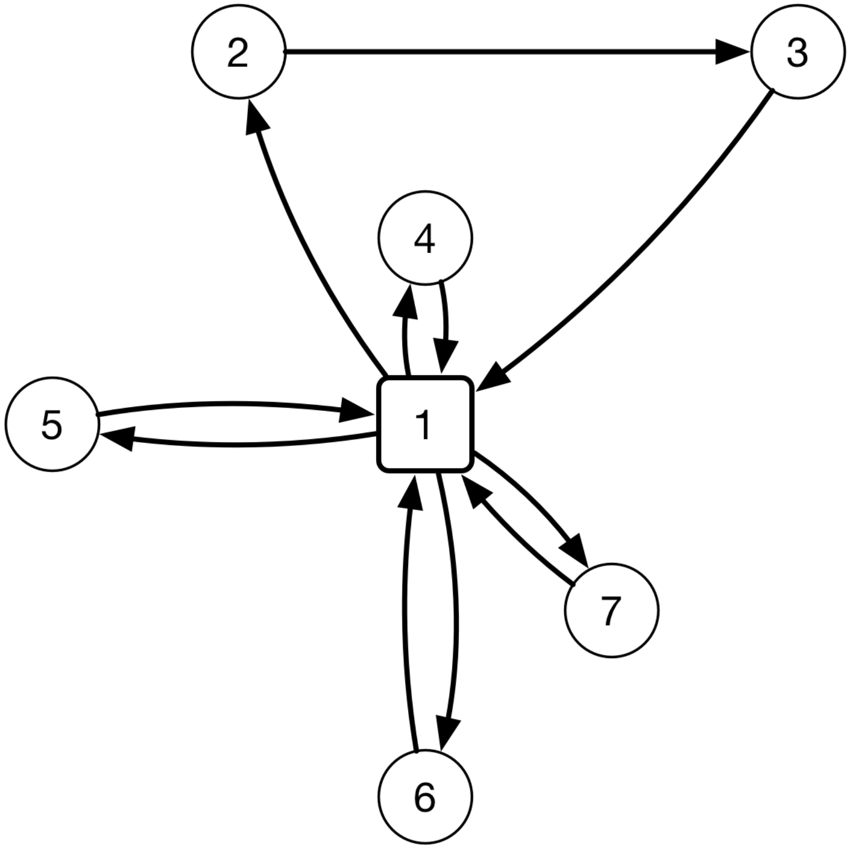


Feasible merges

Merge	Sav
~~(2,3)~~	~~2~~
(2,5)	2
(6,7)	2
~~(2,4)~~	~~1.8~~
(3,4)	1.6
…	…
(4,7)	0.2
…	…

Merge	Sav
(2,3)	2
(2,5)	2
(6,7)	2
(2,4)	1.8
(3,4)	1.6
…	…
(4,7)	0.2
…	…

Savings Heuristic

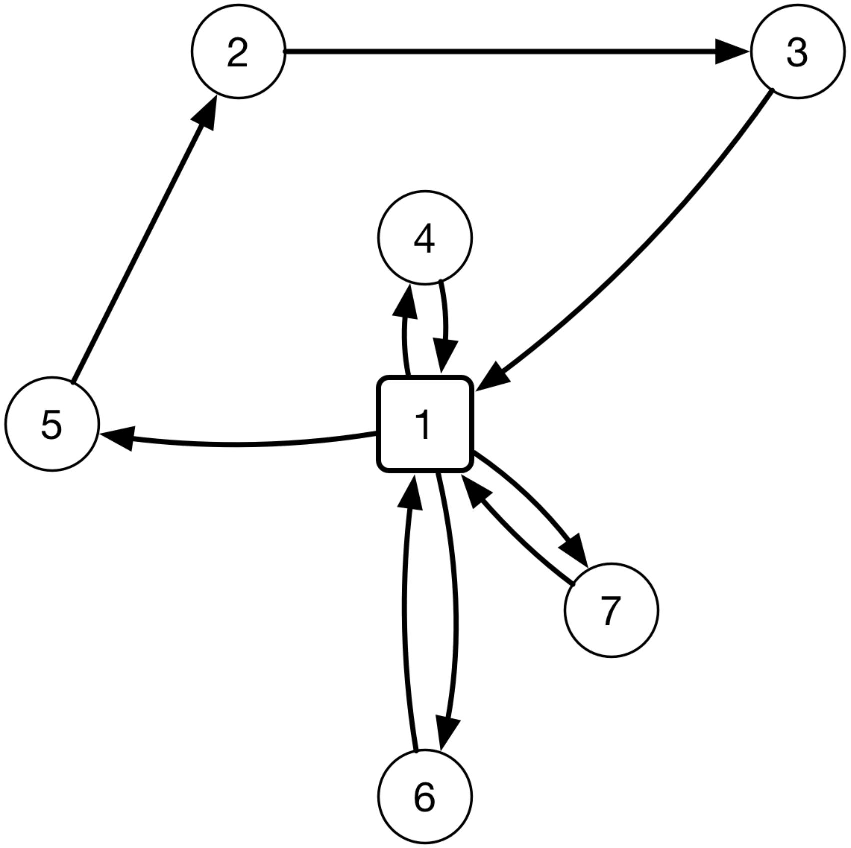


Feasible merges

Merge	Sav
~~(2,3)~~	~~2~~
(2,5)	2
(6,7)	2
~~(2,4)~~	~~1.8~~
(3,4)	1.6
...	...
(4,7)	0.2
...	...

Merge	Sav
(2,3)	2
(2,5)	2
(6,7)	2
(2,4)	1.8
(3,4)	1.6
...	...
(4,7)	0.2
...	...

Savings Heuristic

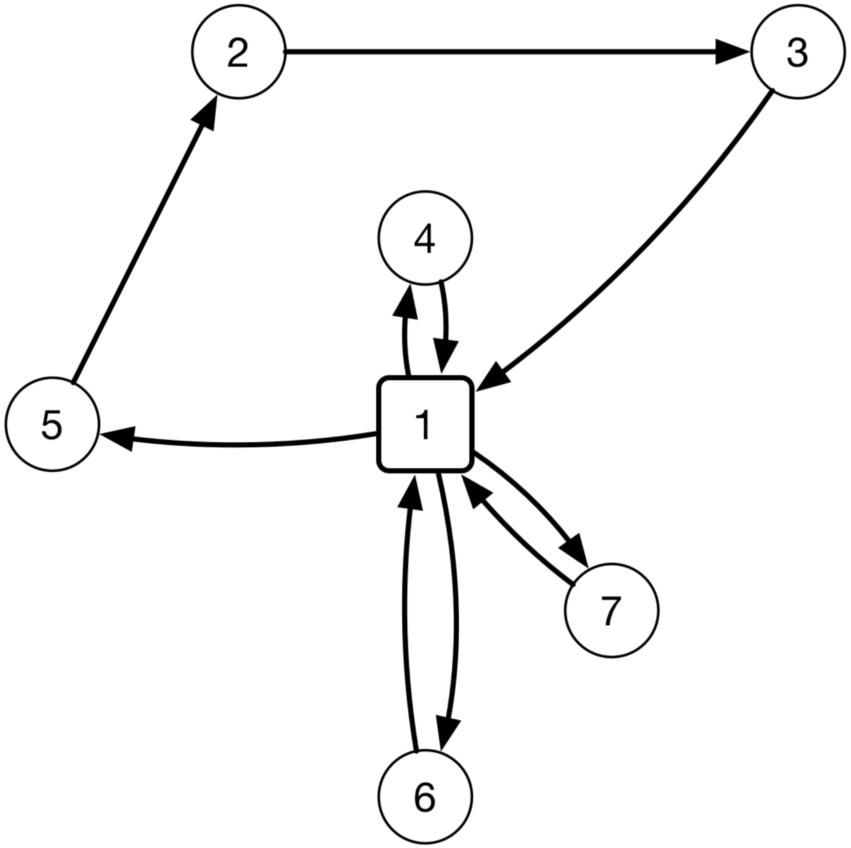


Feasible merges

Merge	Sav
~~(2,3)~~	~~2~~
~~(2,5)~~	~~2~~
(6,7)	2
~~(2,4)~~	~~1.8~~
~~(3,4)~~	~~1.6~~
…	…
~~(4,7)~~	~~0.2~~
…	…

Merge	Sav
(2,3)	2
(2,5)	2
(6,7)	2
(2,4)	1.8
(3,4)	1.6
…	…
(4,7)	0.2
…	…

Savings Heuristic

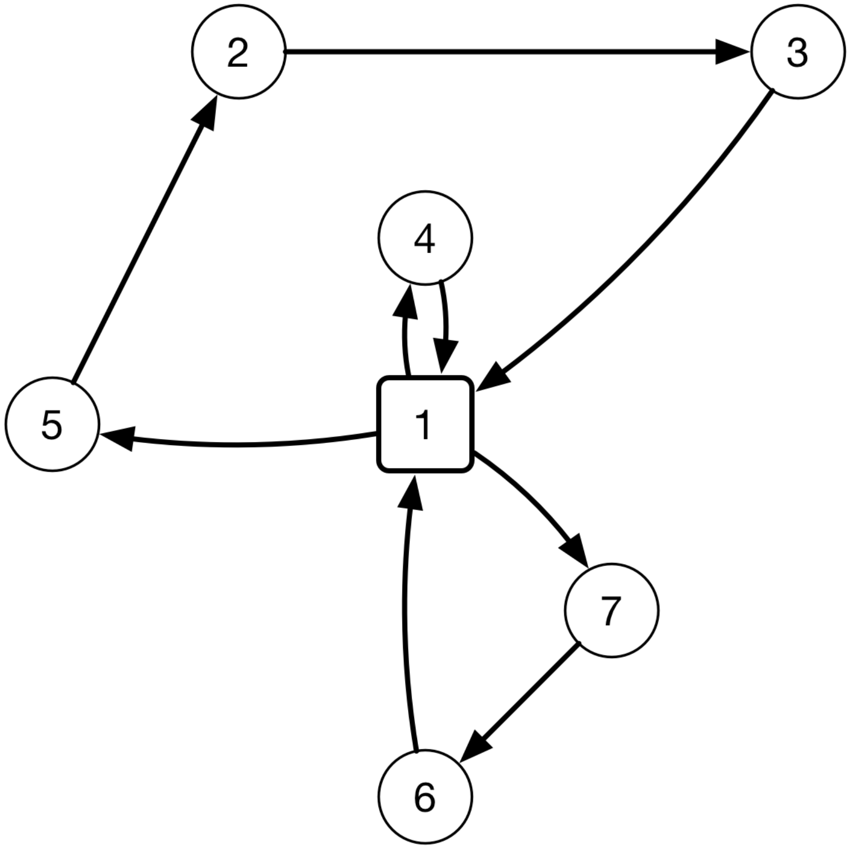


Feasible merges

Merge	Sav
~~(2,3)~~	~~2~~
~~(2,5)~~	~~2~~
(6,7)	2
~~(2,4)~~	~~1.8~~
~~(3,4)~~	~~1.6~~
…	…
~~(4,7)~~	~~0.2~~
…	…

Merge	Sav
(2,3)	2
(2,5)	2
(6,7)	2
(2,4)	1.8
(3,4)	1.6
…	…
(4,7)	0.2
…	…

Savings Heuristic

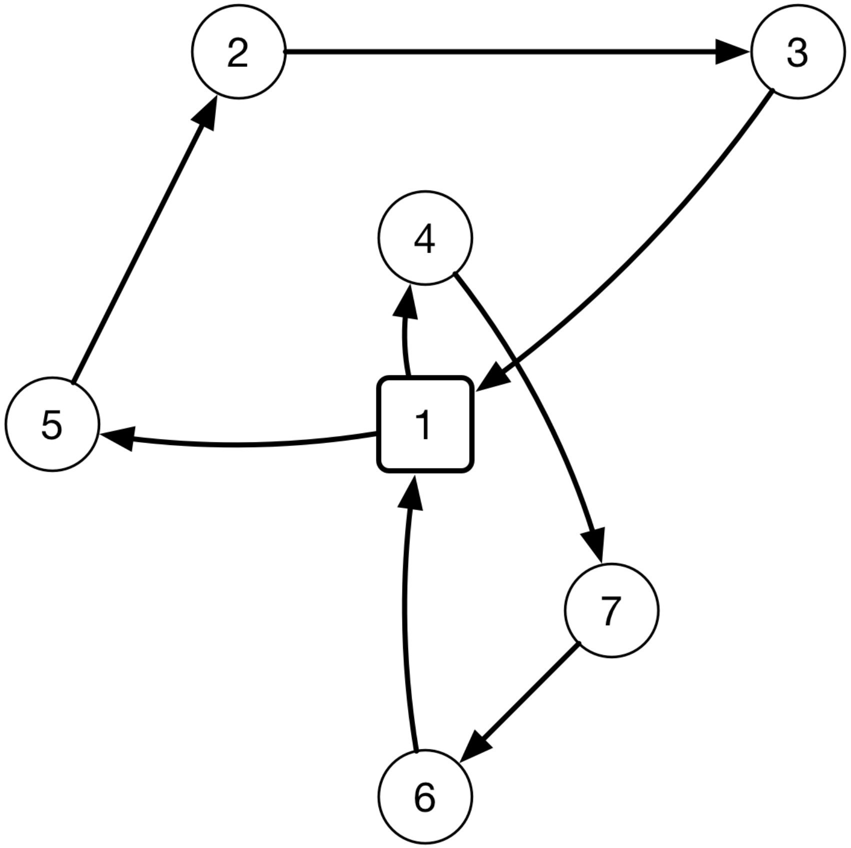


Feasible merges

Merge	Sav
~~(2,3)~~	~~2~~
~~(2,5)~~	~~2~~
~~(6,7)~~	~~2~~
~~(2,4)~~	~~1.8~~
~~(3,4)~~	~~1.6~~
…	…
~~(4,7)~~	~~0.2~~
…	…

Merge	Sav
(2,3)	2
(2,5)	2
(6,7)	2
(2,4)	1.8
(3,4)	1.6
…	…
(4,7)	0.2
…	…

Savings Heuristic

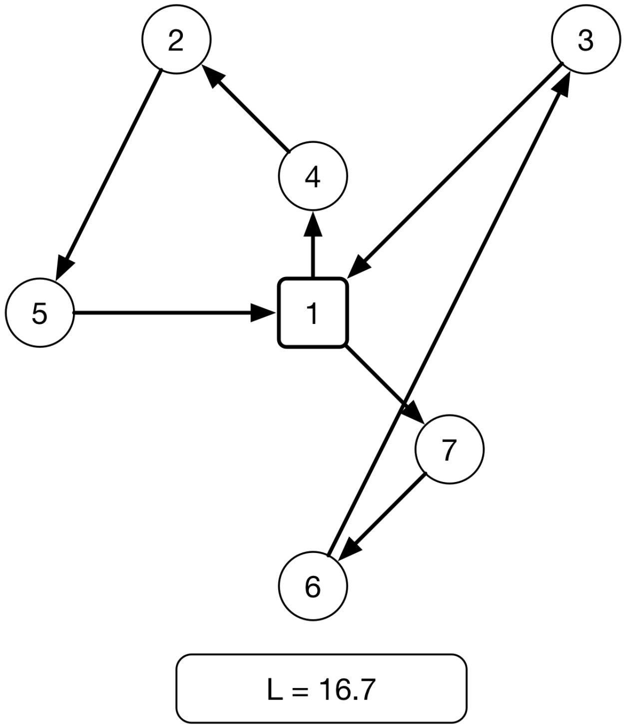


Feasible merges

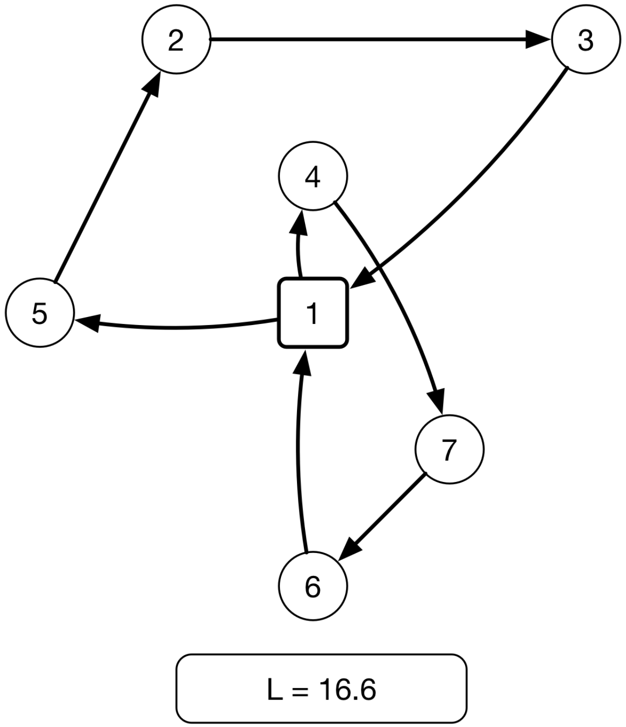
Merge	Sav
~~(2,3)~~	~~2~~
~~(2,5)~~	~~2~~
~~(6,7)~~	~~2~~
~~(2,4)~~	~~1.8~~
~~(3,4)~~	~~1.6~~
…	…
~~(4,7)~~	~~0.2~~
…	…

Merge	Sav
(2,3)	2
(2,5)	2
(6,7)	2
(2,4)	1.8
(3,4)	1.6
…	…
(4,7)	0.2
…	…

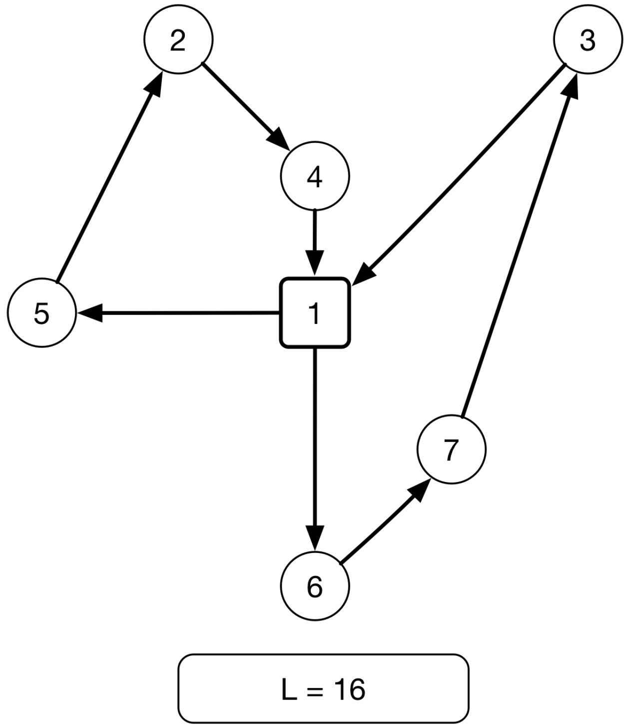
Compare Solutions



Nearest Neighbor



Savings



Optimum

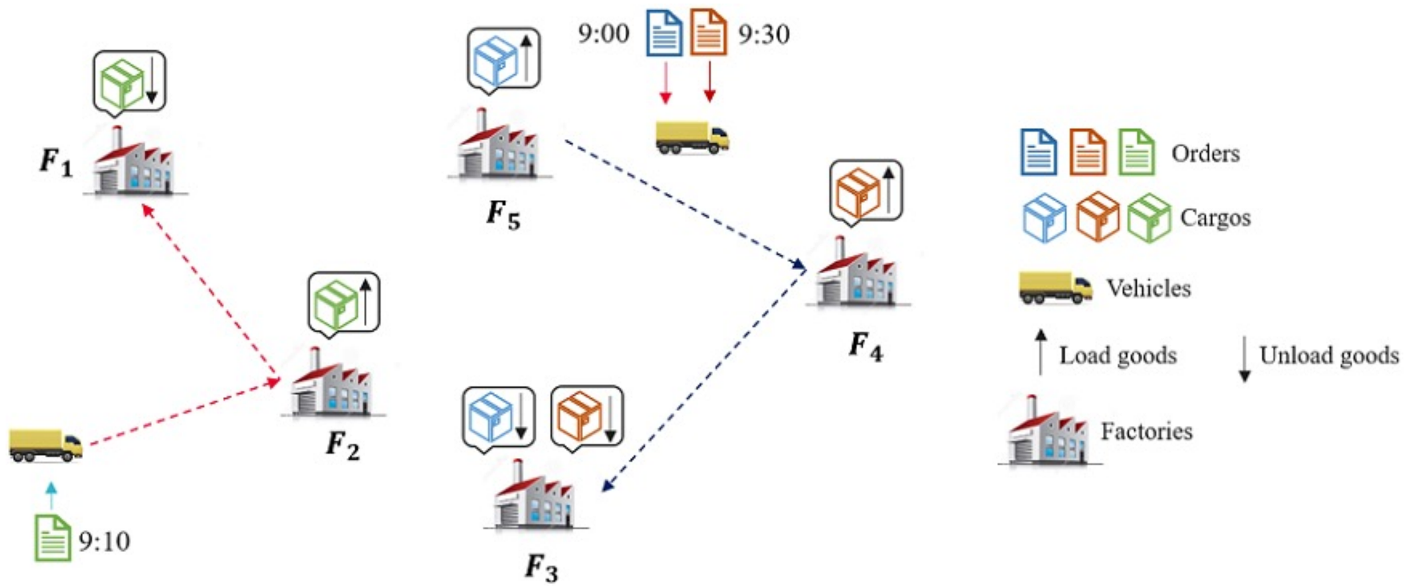
- Further improvement: simulated annealing, tabu search, genetic algorithms, …
- GP to learn heuristics
- Reinforcement learning

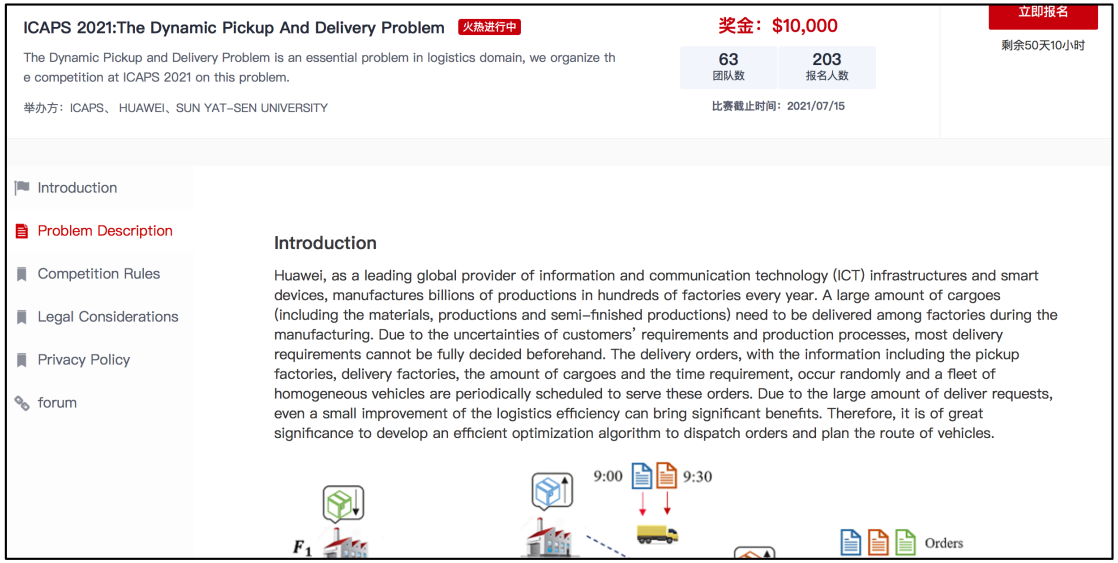
Other Routing Problems

- Arc Routing: serving edges rather than nodes

Other Routing Problems

- Pickup and Delivery: pickup nodes and delivery nodes




ICAPS 2021:The Dynamic Pickup And Delivery Problem 火热进行中

奖金：$10,000

立即报名

剩余50天10小时

The Dynamic Pickup and Delivery Problem is an essential problem in logistics domain, we organize th e competition at ICAPS 2021 on this problem.

63 团队数

203 报名人数

举办方：ICAPS、 HUAWEI、SUN YAT-SEN UNIVERSITY

比赛截止时间：2021/07/15

- Introduction
- Problem Description
- Competition Rules
- Legal Considerations
- Privacy Policy
- forum

Introduction

Huawei, as a leading global provider of information and communication technology (ICT) infrastructures and smart devices, manufactures billions of productions in hundreds of factories every year. A large amount of cargoes (including the materials, productions and semi-finished productions) need to be delivered among factories during the manufacturing. Due to the uncertainties of customers' requirements and production processes, most delivery requirements cannot be fully decided beforehand. The delivery orders, with the information including the pickup factories, delivery factories, the amount of cargoes and the time requirement, occur randomly and a fleet of homogeneous vehicles are periodically scheduled to serve these orders. Due to the large amount of deliver requests, even a small improvement of the logistics efficiency can bring significant benefits. Therefore, it is of great significance to develop an efficient optimization algorithm to dispatch orders and plan the route of vehicles.