Transportation Logistics

Part V: An introduction to VRP

Example 1

A furniture retailer is planning to supply a given number of customers with the furniture they ordered, using the company owned vehicle fleet. Each vehicle is of limited capacity. Which customers should be put on which vehicle's tour (starting and ending at the warehouse) and in which order, such that the total distance traveled is minimized?

Example 2

On a given winter day, the road maintenance authority has to make sure that all major roads are plowed. How shall the available snow plowing vehicles by employed and which road segment shall be plowed by which vehicle such that the total distance traveled is minimal?

Source: Domschke (1997) Logistik: Rundreisen und Touren, Chapter 5.

Example 1

A furniture retailer is planning to supply a given number of customers with the furniture they ordered, using the company owned vehicle fleet. Each vehicle is of limited capacity. Which customers should be put on which vehicle's tour (starting and ending at the warehouse) and in which order, such that the total distance traveled is minimized?

Example 2

On a given winter day, the road maintenance authority has to make sure that all major roads are plowed. How shall the available snow plowing vehicles by employed and which road segment shall be plowed by which vehicle such that the total distance traveled is minimal?

$\rightarrow$ **node routing problem**

Source: Domschke (1997) Logistik: Rundreisen und Touren, Chapter 5.

Example 1

A furniture retailer is planning to supply a given number of customers with the furniture they ordered, using the company owned vehicle fleet. Each vehicle is of limited capacity. Which customers should be put on which vehicle's tour (starting and ending at the warehouse) and in which order, such that the total distance traveled is minimized?

→ **node routing problem**

Example 2

On a given winter day, the road maintenance authority has to make sure that all major roads are plowed. How shall the available snow plowing vehicles by employed and which road segment shall be plowed by which vehicle such that the total distance traveled is minimal?

→ **arc-routing problem**

Source: Domschke (1997) Logistik: Rundreisen und Touren, Chapter 5.

Example 1

A furniture retailer is planning to supply a given number of customers with the furniture they ordered, using the company owned vehicle fleet. Each vehicle is of limited capacity. Which customers should be put on which vehicle's tour (starting and ending at the warehouse) and in which order, such that the total distance traveled is minimized?

→ **node routing problem**

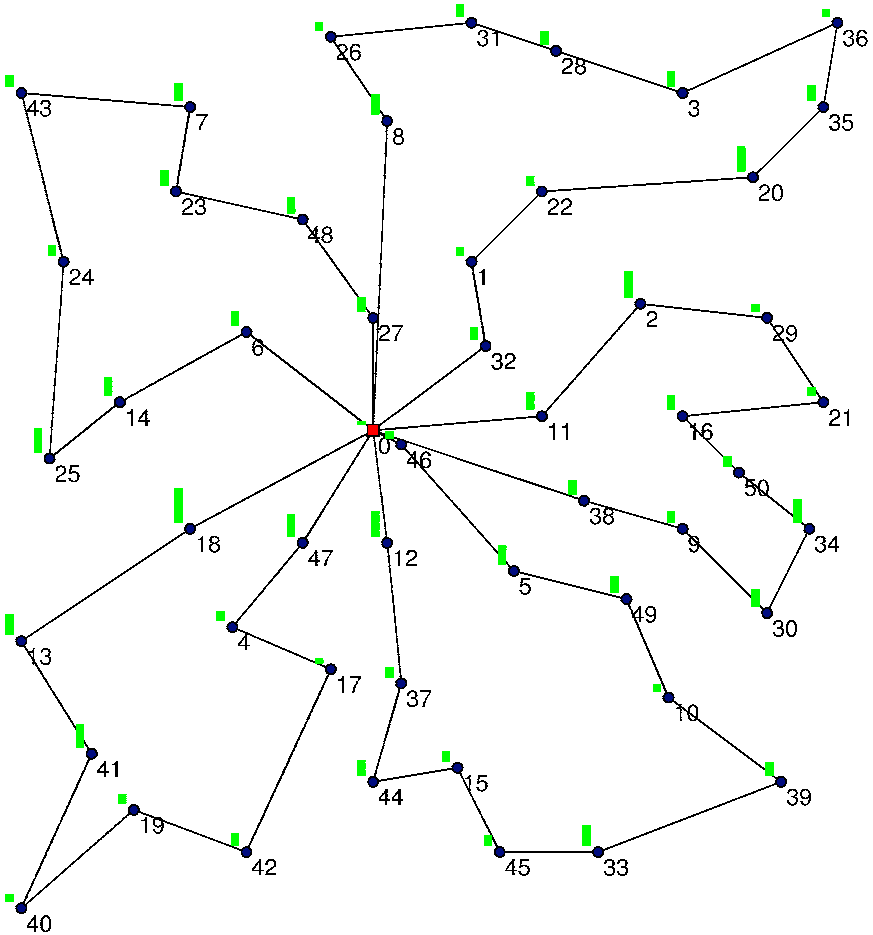
Example 2

On a given winter day, the road maintenance authority has to make sure that all major roads are plowed. How shall the available snow plowing vehicles by employed and which road segment shall be plowed by which vehicle such that the total distance traveled is minimal?

→ **arc-routing problem**

Source: Domschke (1997) Logistik: Rundreisen und Touren, Chapter 5.

The 'basic problem': capacitated VRP (CVRP)



constraints: capacity of the vehicles

Classical extensions

Classical extensions

- multiple depots (MD)

Classical extensions

- multiple depots (MD)
- heterogeneous fleet (HF)

Classical extensions

- multiple depots (MD)
- heterogeneous fleet (HF)
- time windows (TW)

Classical extensions

- multiple depots (MD)
- heterogeneous fleet (HF)
- time windows (TW)
- soft time windows (STW)

Classical extensions

- multiple depots (MD)
- heterogeneous fleet (HF)
- time windows (TW)
- soft time windows (STW)
- multiple time windows (MTW)

Classical extensions

- multiple depots (MD)
- heterogeneous fleet (HF)
- time windows (TW)
- soft time windows (STW)
- multiple time windows (MTW)
- split deliveries (SD)

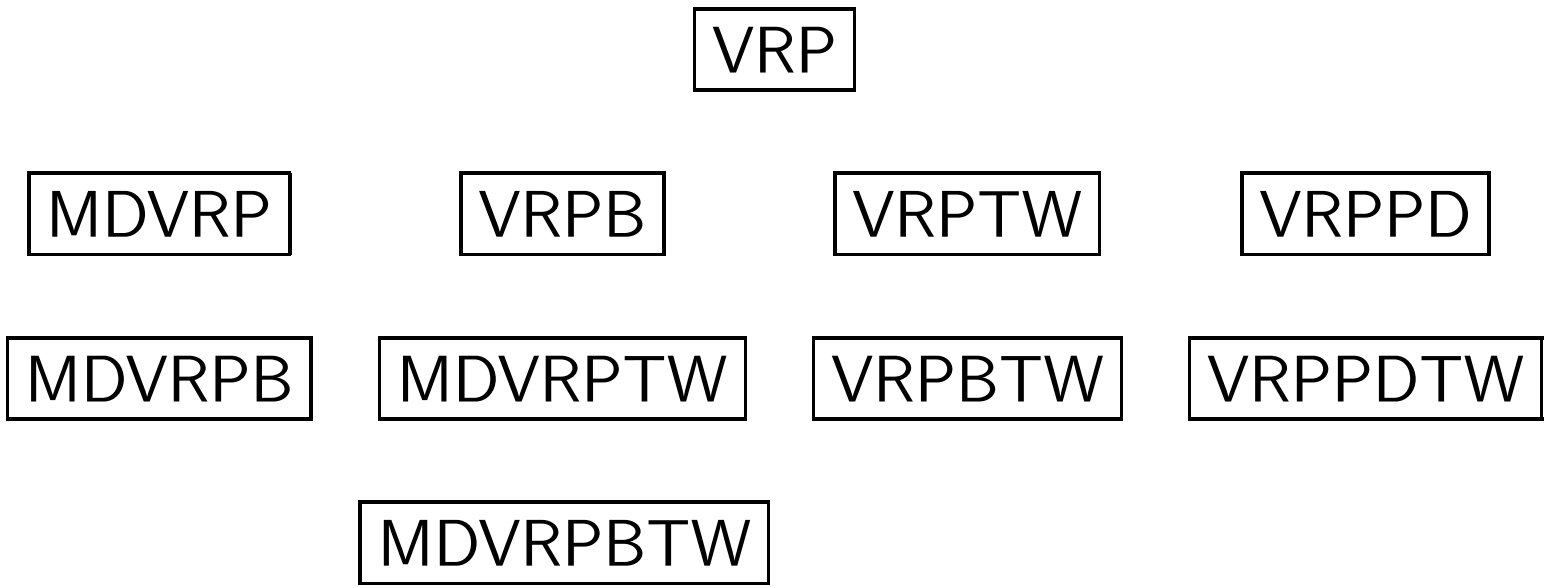
Classical extensions

- multiple depots (MD)
- heterogeneous fleet (HF)
- time windows (TW)
- soft time windows (STW)
- multiple time windows (MTW)
- split deliveries (SD)
- backhauls (B)

Classical extensions

- multiple depots (MD)
- heterogeneous fleet (HF)
- time windows (TW)
- soft time windows (STW)
- multiple time windows (MTW)
- split deliveries (SD)
- backhauls (B)
- pickup and delivery (PD)

Classical extensions



The capacitated vehicle routing problem

Notation

$$x_{ij} = \begin{cases} 1, \text{ if arc } (ij) \text{ is part of the solution,} \\ 0, \text{ otherwise.} \end{cases}$$

c_{ij} = the costs to traverse arc (i, j)

d_i = demand of customer i

C = vehicle capacity

A...set of arcs,
V...set of vertices
n...number of customers
0...depot
S...subest of customers

$r(S) = \lceil \sum_{i \in S} d_i / C \rceil$...the min. number of vehicles needed to serve the nodes in S.

(CVRP is formulated on a complete directed graph)

The capacitated vehicle routing problem

CVRP 1 - capacity cut constraints (CCCs)

$$\sum_{i \in V} \sum_{j \in V} c_{ij} x_{ij} \rightarrow \min \tag{1}$$

$$\sum_{i \in V \setminus \{j\}} x_{ij} = 1 \qquad \forall j \in V \setminus \{0\}, \tag{2}$$

$$\sum_{j \in V \setminus \{i\}} x_{ij} = 1 \qquad \forall i \in V \setminus \{0\}, \tag{3}$$

$$\sum_{i \in S} \sum_{j \notin S} x_{ij} \geq r(S) \qquad \forall S \subseteq V \setminus \{0\}, S \neq \emptyset, \tag{4}$$

$$x_{ij} \in \{0, 1\} \qquad \forall i, j \in V. \tag{5}$$

set of CCC: cardinality grows exponentially with n

The capacitated vehicle routing problem

CVRP 2 - generalized subtour elmination constraints (GSECs)

$$\sum_{i \in V} \sum_{j \in V} c_{ij} x_{ij} \rightarrow \min \tag{6}$$

$$\sum_{i \in V \setminus \{j\}} x_{ij} = 1 \qquad \forall j \in V \setminus \{0\}, \tag{7}$$

$$\sum_{j \in V \setminus \{i\}} x_{ij} = 1 \qquad \forall i \in V \setminus \{0\}, \tag{8}$$

$$\sum_{i \in S} \sum_{j \in S} x_{ij} \leq |S| - r(S) \qquad \forall S \subseteq V \setminus \{0\}, S \neq \emptyset, \tag{9}$$

$$x_{ij} \in \{0, 1\} \qquad \forall i, j \in V. \tag{10}$$

set of GSECs: cardinality grows exponentially with n

The capacitated vehicle routing problem

CVRP 3 - adapted Miller-Tucker-Zemlin (MTZ) subtour elimination constraints

$$\sum_{i \in V} \sum_{j \in V} c_{ij} x_{ij} \rightarrow \min \tag{11}$$

$$\sum_{i \in V \setminus \{j\}} x_{ij} = 1 \qquad \forall j \in V \setminus \{0\}, \tag{12}$$

$$\sum_{j \in V \setminus \{i\}} x_{ij} = 1 \qquad \forall i \in V \setminus \{0\}, \tag{13}$$

$$u_i - u_j + C x_{ij} \leq C - d_j \qquad \forall i, j \in V \setminus \{0\}, i \neq j, \text{ such that } d_i + d_j \leq C \tag{14}$$

$$x_{ij} \in \{0, 1\} \qquad \forall i, j \in V, \tag{15}$$

$$d_i \leq u_i \leq C \qquad \forall i \in V \setminus \{0\} \tag{16}$$

$u_i \ \forall i \in V \setminus \{0\}$ gives the load of the vehicle after visiting cust. i

set of MTZ subtour elim: polynomial cardinality

Given vehicle fleet m

$$\sum_{j \in V} x_{0j} \leq m \tag{17}$$

$$\sum_{i \in V} x_{i0} \leq m \tag{18}$$

$$\sum_{j \in V} x_{0j} = m \tag{19}$$

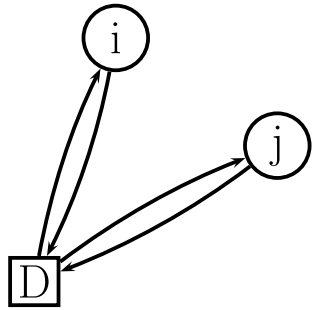
$$\sum_{i \in V} x_{i0} = m \tag{20}$$

Constructive heuristics

- Clarke and Wright savings algorithm
- Sequential and parallel insertion heuristics
- Cluster first route second heuristics
- Petal algorithms
- Route first cluster second heuristics

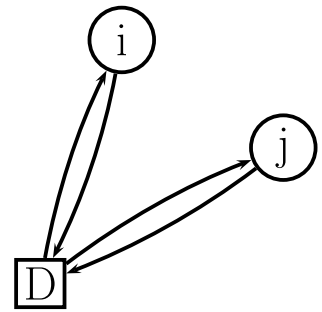
Clarke and Wright savings algorithm

★ Merging $(0, \ldots, i, 0)$ and $(0, j, \ldots, 0)$ into $(0, \ldots, i, j, \ldots, 0)$ results in a distance saving of $s_{ij} = c_{i0} + c_{0j} - c_{ij}$.



Clarke and Wright savings algorithm

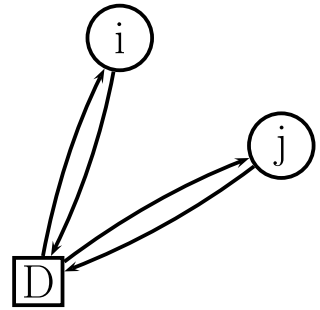
★ Merging $(0, \ldots, i, 0)$ and $(0, j, \ldots, 0)$ into $(0, \ldots, i, j, \ldots, 0)$ results in a distance saving of $s_{ij} = c_{i0} + c_{0j} - c_{ij}$.



- **Initialization** Computation of savings: for each i and j $(i \neq j)$ compute the savings $s_{ij} = c_{i0} + c_{0j} - c_{ij}$; create a single customer route for each customer i: $(0, i, 0)$; put savings into a list in decreasing order.

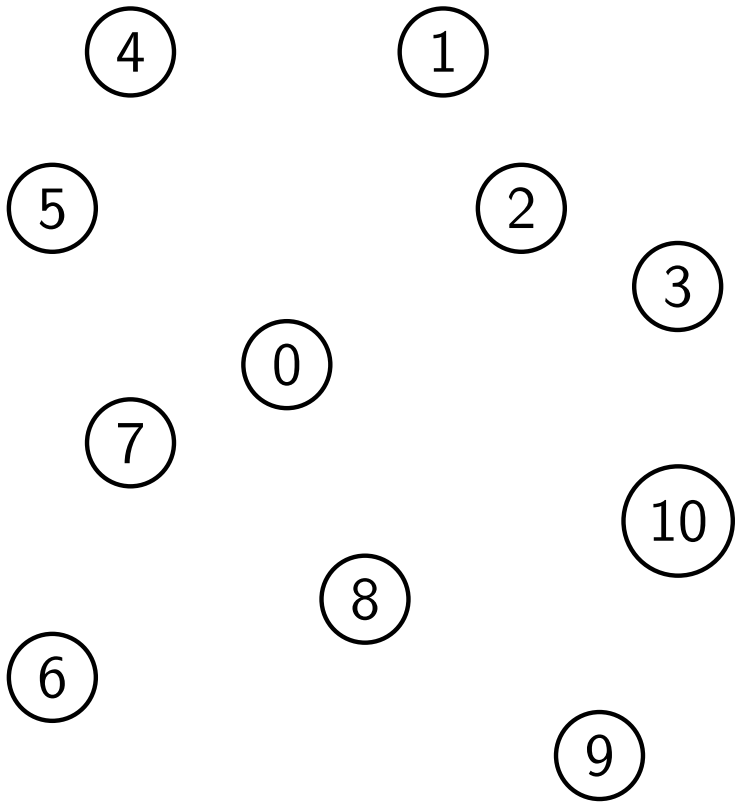
Clarke and Wright savings algorithm

★ Merging $(0, \ldots, i, 0)$ and $(0, j, \ldots, 0)$ into $(0, \ldots, i, j, \ldots, 0)$ results in a distance saving of $s_{ij} = c_{i0} + c_{0j} - c_{ij}$.



- **Initialization** Computation of savings: for each i and j $(i \neq j)$ compute the savings $s_{ij} = c_{i0} + c_{0j} - c_{ij}$; create a single customer route for each customer i: $(0, i, 0)$; put savings into a list in decreasing order.
- **Step 1** Best feasible merge: start with the first entry of the savings list. For each saving s_{ij}, determine if there exist two routes, one with arc or edge $(0, j)$ and one with arc $(i, 0)$, that can be merged in a feasible way. If yes combine the two: delete $(0, j)$ and $(i, 0)$, introduce (i, j).

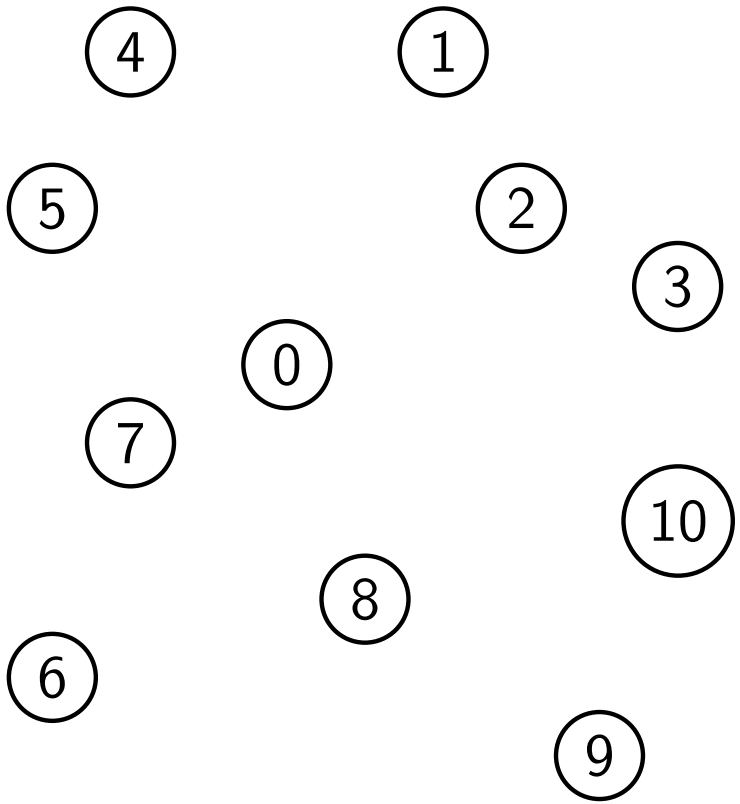
Savings - Example - Data



c_{ij}	0	1	2	3	4	5	6	7	8	9	10
0	0.0	4.5	3.6	5.1	4.5	3.6	5.0	2.2	3.2	6.4	5.4
1	4.5	0.0	2.2	4.2	4.0	5.4	9.4	6.4	7.1	9.2	6.7
2	3.6	2.2	0.0	2.2	5.4	6.0	8.5	5.8	5.4	7.1	4.5
3	5.1	4.2	2.2	0.0	7.6	8.1	9.4	7.3	5.7	6.1	3.0
4	4.5	4.0	5.4	7.6	0.0	2.2	8.1	5.0	7.6	10.8	9.2
5	3.6	5.4	6.0	8.1	2.2	0.0	6.0	3.2	6.4	9.9	8.9
6	5.0	9.4	8.5	9.4	8.1	6.0	0.0	3.2	4.1	7.1	8.2
7	2.2	6.4	5.8	7.3	5.0	3.2	3.2	0.0	3.6	7.2	7.1
8	3.2	7.1	5.4	5.7	7.6	6.4	4.1	3.6	0.0	3.6	4.1
9	6.4	9.2	7.1	6.1	10.8	9.9	7.1	7.2	3.6	0.0	3.2
10	5.4	6.7	4.5	3.0	9.2	8.9	8.2	7.1	4.1	3.2	0.0
d_i	0	2	1	1	2	1	1	2	1	1	2

$C = 4$

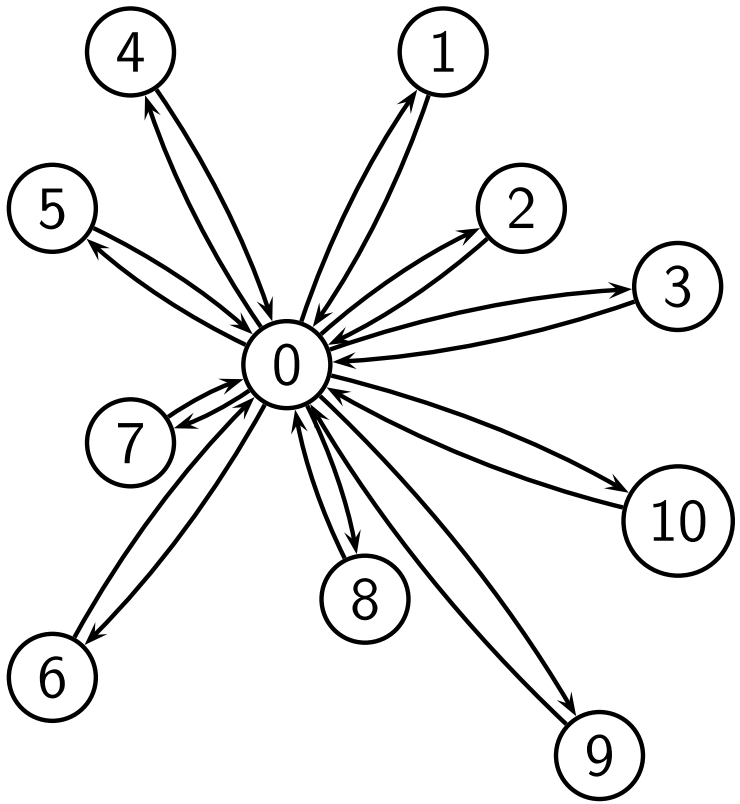
Savings - Example



s_{ij}	1	2	3	4	5	6	7	8	9	10
1	×	5.8	5.3	4.9	2.7	0.0	0.3	0.6	1.7	3.1
2	5.8	×	6.5	2.7	1.2	0.1	0.0	1.4	2.9	4.5
3	5.3	6.5	×	2.0	0.6	0.7	0.1	2.6	5.4	7.5
4	4.9	2.7	2.0	×	5.8	1.4	1.7	0.0	0.1	0.6
5	2.7	1.2	0.6	5.8	×	2.6	2.7	0.4	0.1	0.0
6	0.0	0.1	0.7	1.4	2.6	×	4.1	4.0	4.3	2.1
7	0.3	0.0	0.1	1.7	2.7	4.1	×	1.8	1.4	0.6
8	0.6	1.4	2.6	0.0	0.4	4.0	1.8	×	6.0	4.4
9	1.7	2.9	5.4	0.1	0.1	4.3	1.4	6.0	×	8.6
10	3.1	4.5	7.5	0.6	0.0	2.1	0.6	4.4	8.6	×
d_i	2	1	1	2	1	1	2	1	1	2

$C = 4$

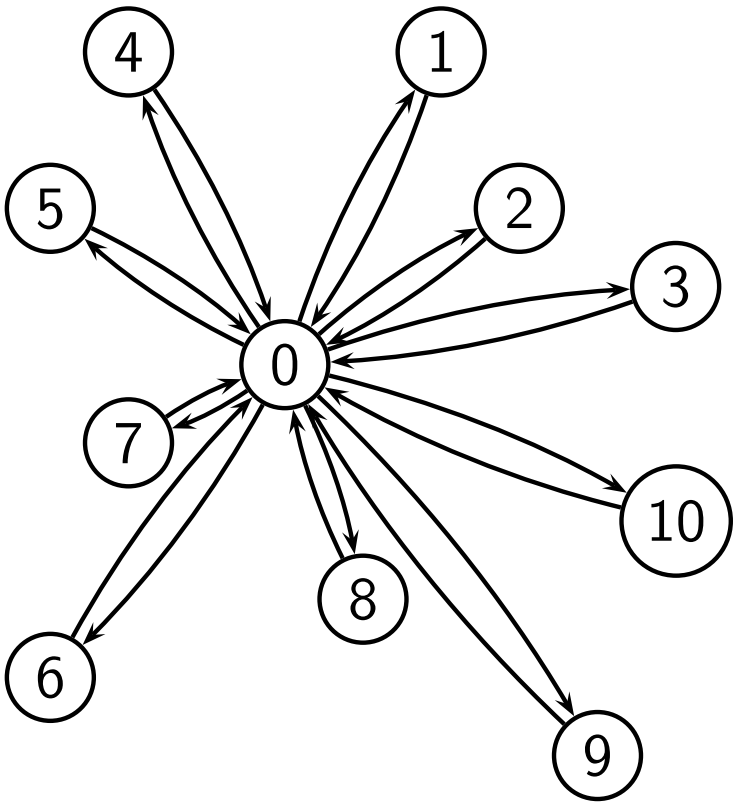
Savings - Example



s_{ij}	1	2	3	4	5	6	7	8	9	10
1	×	5.8	5.3	4.9	2.7	0.0	0.3	0.6	1.7	3.1
2	5.8	×	6.5	2.7	1.2	0.1	0.0	1.4	2.9	4.5
3	5.3	6.5	×	2.0	0.6	0.7	0.1	2.6	5.4	7.5
4	4.9	2.7	2.0	×	5.8	1.4	1.7	0.0	0.1	0.6
5	2.7	1.2	0.6	5.8	×	2.6	2.7	0.4	0.1	0.0
6	0.0	0.1	0.7	1.4	2.6	×	4.1	4.0	4.3	2.1
7	0.3	0.0	0.1	1.7	2.7	4.1	×	1.8	1.4	0.6
8	0.6	1.4	2.6	0.0	0.4	4.0	1.8	×	6.0	4.4
9	1.7	2.9	5.4	0.1	0.1	4.3	1.4	6.0	×	8.6
10	3.1	4.5	7.5	0.6	0.0	2.1	0.6	4.4	8.6	×
d_i	2	1	1	2	1	1	2	1	1	2

$C = 4$

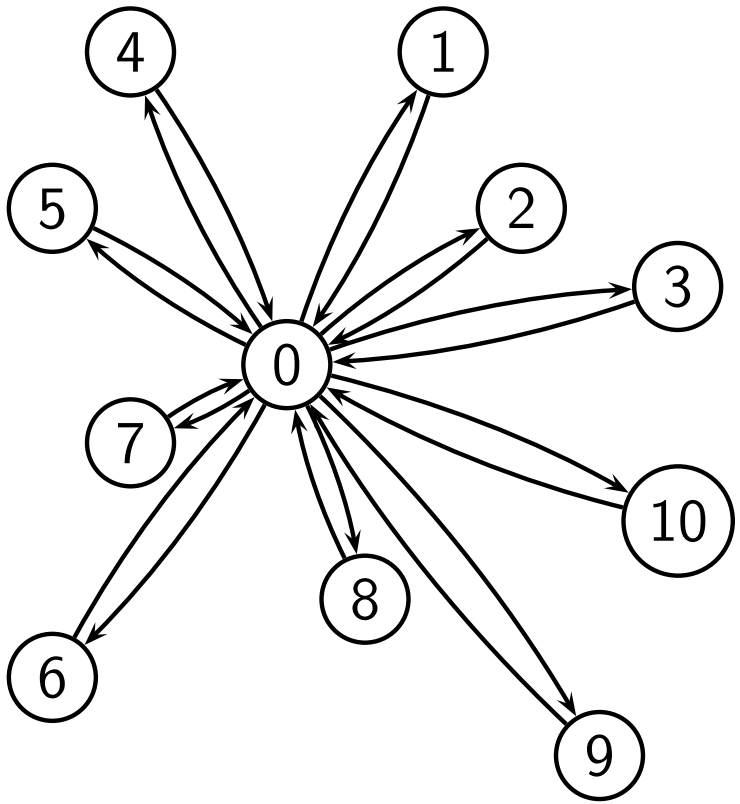
Savings - Example



s_{ij}	1	2	3	4	5	6	7	8	9	10
1	×	5.8	5.3	4.9	2.7	0.0	0.3	0.6	1.7	3.1
2	5.8	×	6.5	2.7	1.2	0.1	0.0	1.4	2.9	4.5
3	5.3	6.5	×	2.0	0.6	0.7	0.1	2.6	5.4	7.5
4	4.9	2.7	2.0	×	5.8	1.4	1.7	0.0	0.1	0.6
5	2.7	1.2	0.6	5.8	×	2.6	2.7	0.4	0.1	0.0
6	0.0	0.1	0.7	1.4	2.6	×	4.1	4.0	4.3	2.1
7	0.3	0.0	0.1	1.7	2.7	4.1	×	1.8	1.4	0.6
8	0.6	1.4	2.6	0.0	0.4	4.0	1.8	×	6.0	4.4
9	1.7	2.9	5.4	0.1	0.1	4.3	1.4	6.0	×	8.6
10	3.1	4.5	7.5	0.6	0.0	2.1	0.6	4.4	8.6	×
d_i	2	1	1	2	1	1	2	1	1	2

$C = 4$

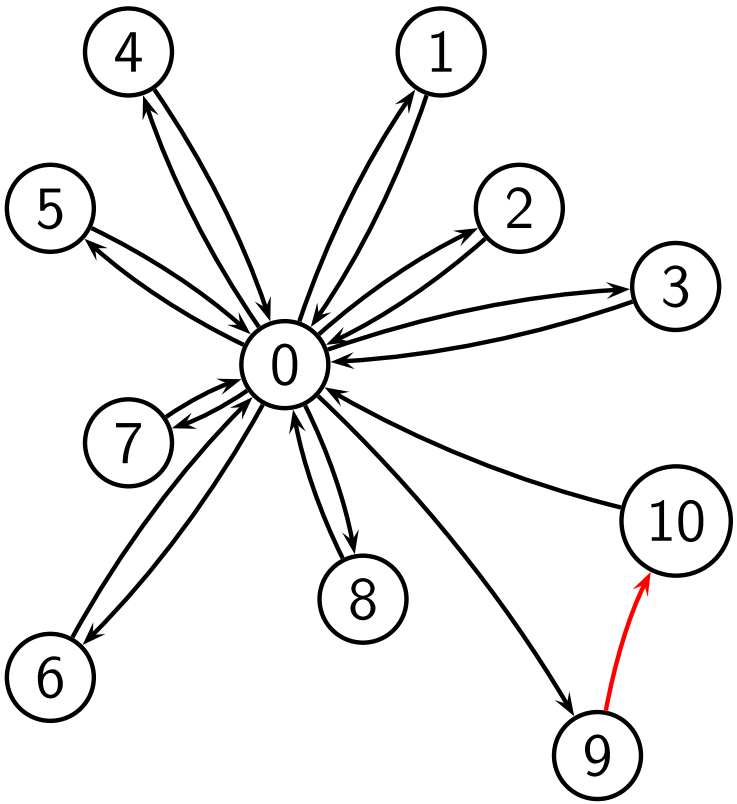
Savings - Example



s_{ij}	1	2	3	4	5	6	7	8	9	10
1	×	5.8	5.3	4.9	2.7	0.0	0.3	0.6	1.7	3.1
2	5.8	×	6.5	2.7	1.2	0.1	0.0	1.4	2.9	4.5
3	5.3	6.5	×	2.0	0.6	0.7	0.1	2.6	5.4	7.5
4	4.9	2.7	2.0	×	5.8	1.4	1.7	0.0	0.1	0.6
5	2.7	1.2	0.6	5.8	×	2.6	2.7	0.4	0.1	0.0
6	0.0	0.1	0.7	1.4	2.6	×	4.1	4.0	4.3	2.1
7	0.3	0.0	0.1	1.7	2.7	4.1	×	1.8	1.4	0.6
8	0.6	1.4	2.6	0.0	0.4	4.0	1.8	×	6.0	4.4
9	1.7	2.9	5.4	0.1	0.1	4.3	1.4	6.0	×	8.6
10	3.1	4.5	7.5	0.6	0.0	2.1	0.6	4.4	8.6	×
d_i	2	1	1	2	1	1	2	1	1	2

$C = 4$

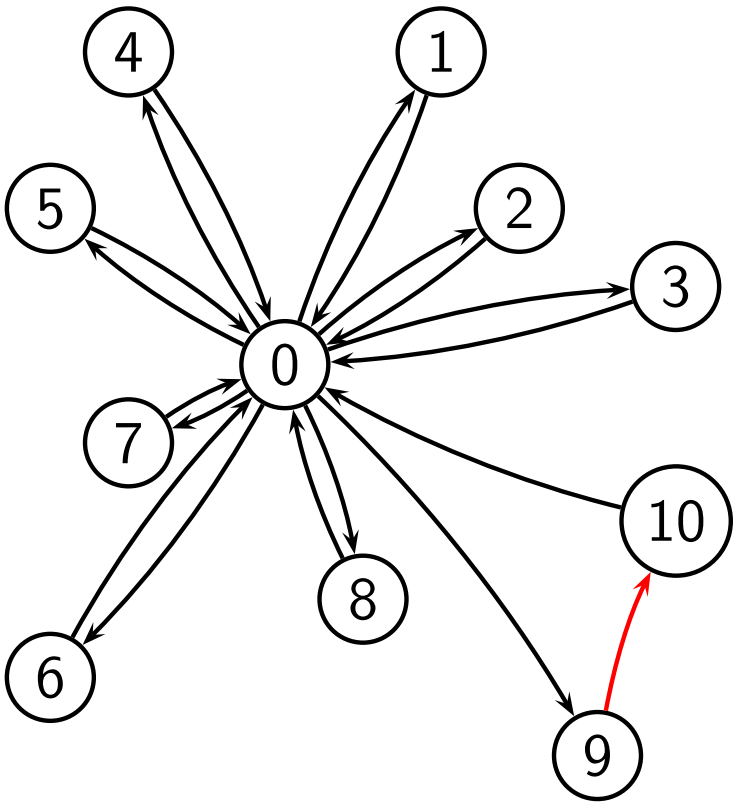
Savings - Example



s_{ij}	1	2	3	4	5	6	7	8	9	10
1	×	5.8	5.3	4.9	2.7	0.0	0.3	0.6	1.7	3.1
2	5.8	×	6.5	2.7	1.2	0.1	0.0	1.4	2.9	4.5
3	5.3	6.5	×	2.0	0.6	0.7	0.1	2.6	5.4	7.5
4	4.9	2.7	2.0	×	5.8	1.4	1.7	0.0	0.1	0.6
5	2.7	1.2	0.6	5.8	×	2.6	2.7	0.4	0.1	0.0
6	0.0	0.1	0.7	1.4	2.6	×	4.1	4.0	4.3	2.1
7	0.3	0.0	0.1	1.7	2.7	4.1	×	1.8	1.4	0.6
8	0.6	1.4	2.6	0.0	0.4	4.0	1.8	×	6.0	4.4
9	1.7	2.9	5.4	0.1	0.1	4.3	1.4	6.0	×	8.6
10	3.1	4.5	7.5	0.6	0.0	2.1	0.6	4.4	8.6	×
d_i	2	1	1	2	1	1	2	1	1	2

$C = 4$

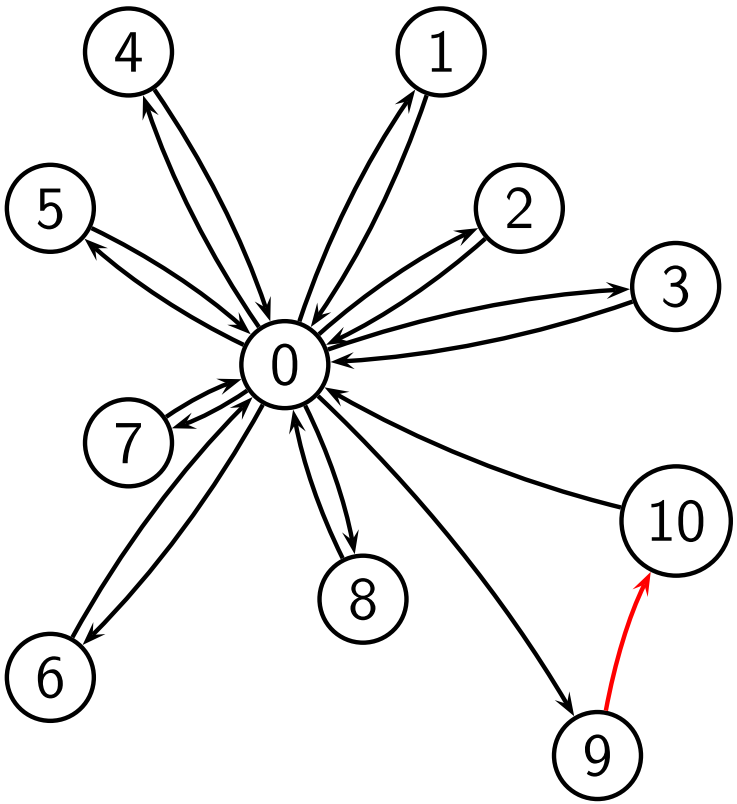
Savings - Example



s_{ij}	1	2	3	4	5	6	7	8	9	10
1	×	5.8	5.3	4.9	2.7	0.0	0.3	0.6	1.7	3.1
2	5.8	×	6.5	2.7	1.2	0.1	0.0	1.4	2.9	4.5
3	5.3	6.5	×	2.0	0.6	0.7	0.1	2.6	5.4	7.5
4	4.9	2.7	2.0	×	5.8	1.4	1.7	0.0	0.1	0.6
5	2.7	1.2	0.6	5.8	×	2.6	2.7	0.4	0.1	0.0
6	0.0	0.1	0.7	1.4	2.6	×	4.1	4.0	4.3	2.1
7	0.3	0.0	0.1	1.7	2.7	4.1	×	1.8	1.4	0.6
8	0.6	1.4	2.6	0.0	0.4	4.0	1.8	×	6.0	4.4
9	1.7	2.9	5.4	0.1	0.1	4.3	1.4	6.0	×	8.6
10	3.1	4.5	7.5	0.6	0.0	2.1	0.6	4.4	8.6	×
d_i	2	1	1	2	1	1	2	1	1	2

$C = 4$

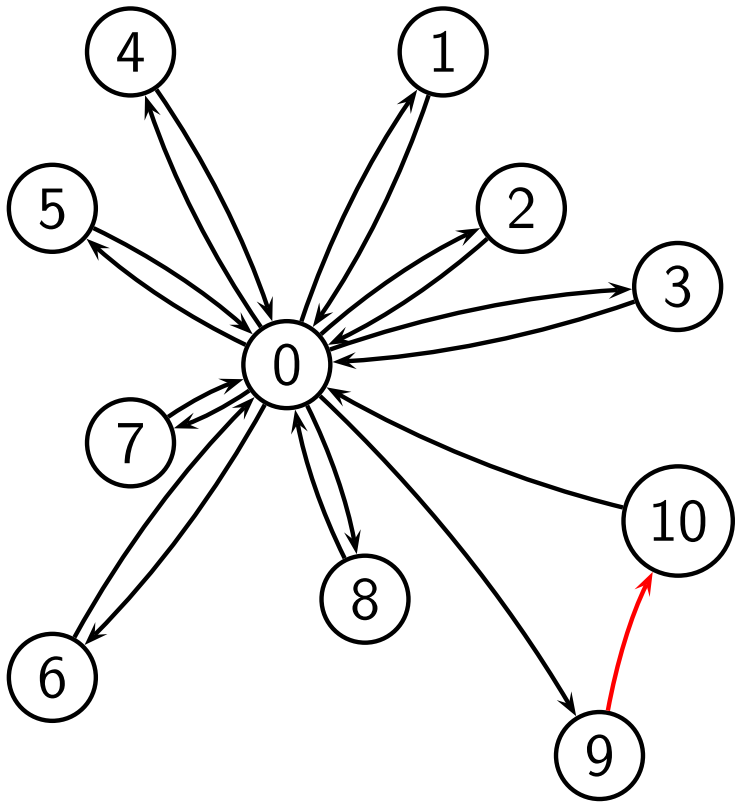
Savings - Example



s_{ij}	1	2	3	4	5	6	7	8	9	10
1	×	5.8	5.3	4.9	2.7	0.0	0.3	0.6	1.7	3.1
2	5.8	×	6.5	2.7	1.2	0.1	0.0	1.4	2.9	4.5
3	5.3	6.5	×	2.0	0.6	0.7	0.1	2.6	5.4	7.5
4	4.9	2.7	2.0	×	5.8	1.4	1.7	0.0	0.1	0.6
5	2.7	1.2	0.6	5.8	×	2.6	2.7	0.4	0.1	0.0
6	0.0	0.1	0.7	1.4	2.6	×	4.1	4.0	4.3	2.1
7	0.3	0.0	0.1	1.7	2.7	4.1	×	1.8	1.4	0.6
8	0.6	1.4	2.6	0.0	0.4	4.0	1.8	×	6.0	4.4
9	1.7	2.9	5.4	0.1	0.1	4.3	1.4	6.0	×	8.6
10	3.1	4.5	7.5	0.6	0.0	2.1	0.6	4.4	8.6	×
d_i	2	1	1	2	1	1	2	1	1	2

$C = 4$

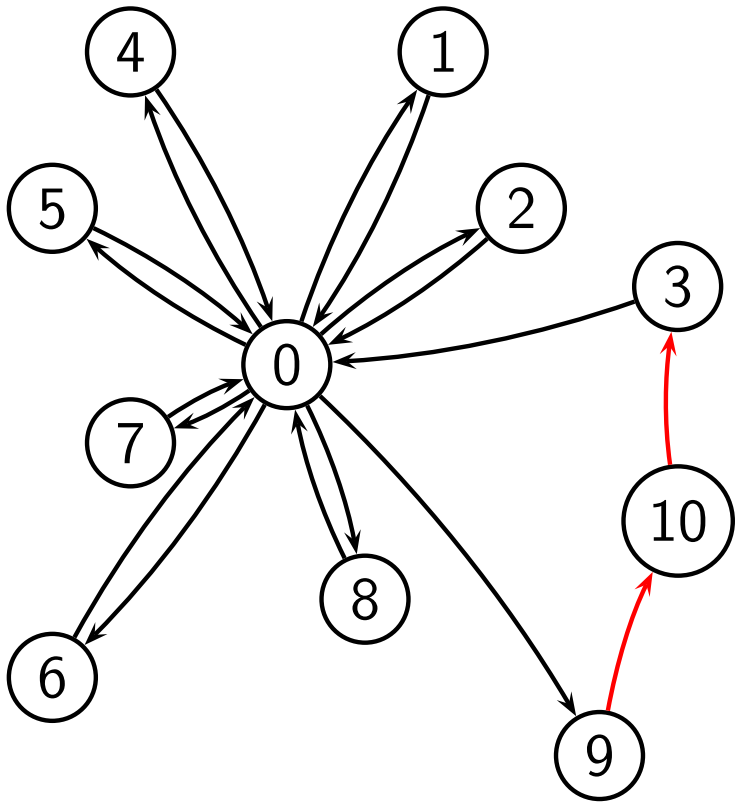
Savings - Example



s_{ij}	1	2	3	4	5	6	7	8	9	10
1	×	5.8	5.3	4.9	2.7	0.0	0.3	0.6	1.7	3.1
2	5.8	×	6.5	2.7	1.2	0.1	0.0	1.4	2.9	4.5
3	5.3	6.5	×	2.0	0.6	0.7	0.1	2.6	5.4	7.5
4	4.9	2.7	2.0	×	5.8	1.4	1.7	0.0	0.1	0.6
5	2.7	1.2	0.6	5.8	×	2.6	2.7	0.4	0.1	0.0
6	0.0	0.1	0.7	1.4	2.6	×	4.1	4.0	4.3	2.1
7	0.3	0.0	0.1	1.7	2.7	4.1	×	1.8	1.4	0.6
8	0.6	1.4	2.6	0.0	0.4	4.0	1.8	×	6.0	4.4
9	1.7	2.9	5.4	0.1	0.1	4.3	1.4	6.0	×	8.6
10	3.1	4.5	7.5	0.6	0.0	2.1	0.6	4.4	8.6	×
d_i	2	1	1	2	1	1	2	1	1	2

$C = 4$

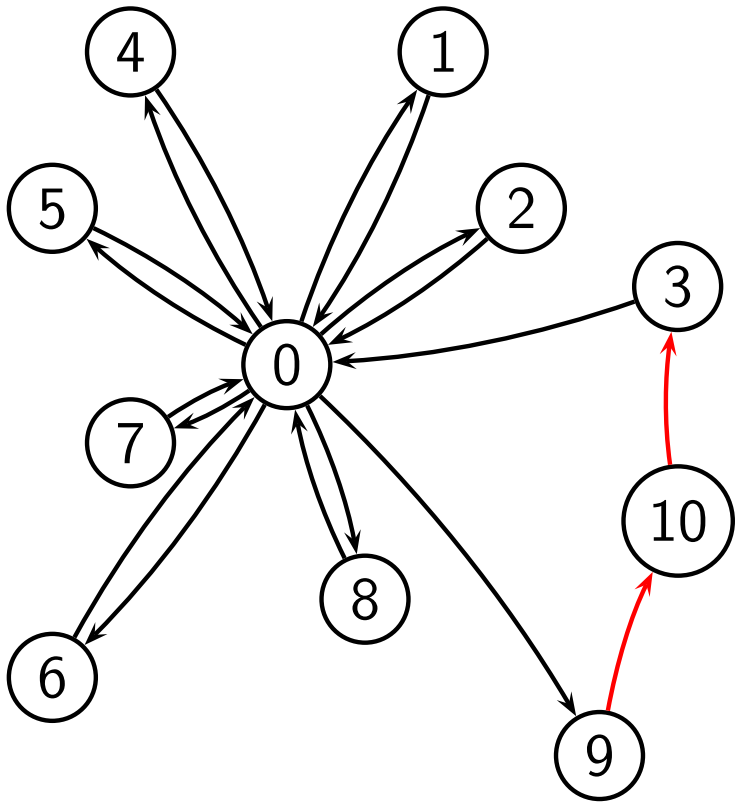
Savings - Example



s_{ij}	1	2	3	4	5	6	7	8	9	10
1	×	5.8	5.3	4.9	2.7	0.0	0.3	0.6	1.7	3.1
2	5.8	×	6.5	2.7	1.2	0.1	0.0	1.4	2.9	4.5
3	5.3	6.5	×	2.0	0.6	0.7	0.1	2.6	5.4	7.5
4	4.9	2.7	2.0	×	5.8	1.4	1.7	0.0	0.1	0.6
5	2.7	1.2	0.6	5.8	×	2.6	2.7	0.4	0.1	0.0
6	0.0	0.1	0.7	1.4	2.6	×	4.1	4.0	4.3	2.1
7	0.3	0.0	0.1	1.7	2.7	4.1	×	1.8	1.4	0.6
8	0.6	1.4	2.6	0.0	0.4	4.0	1.8	×	6.0	4.4
9	1.7	2.9	5.4	0.1	0.1	4.3	1.4	6.0	×	8.6
10	3.1	4.5	7.5	0.6	0.0	2.1	0.6	4.4	8.6	×
d_i	2	1	1	2	1	1	2	1	1	2

$C = 4$

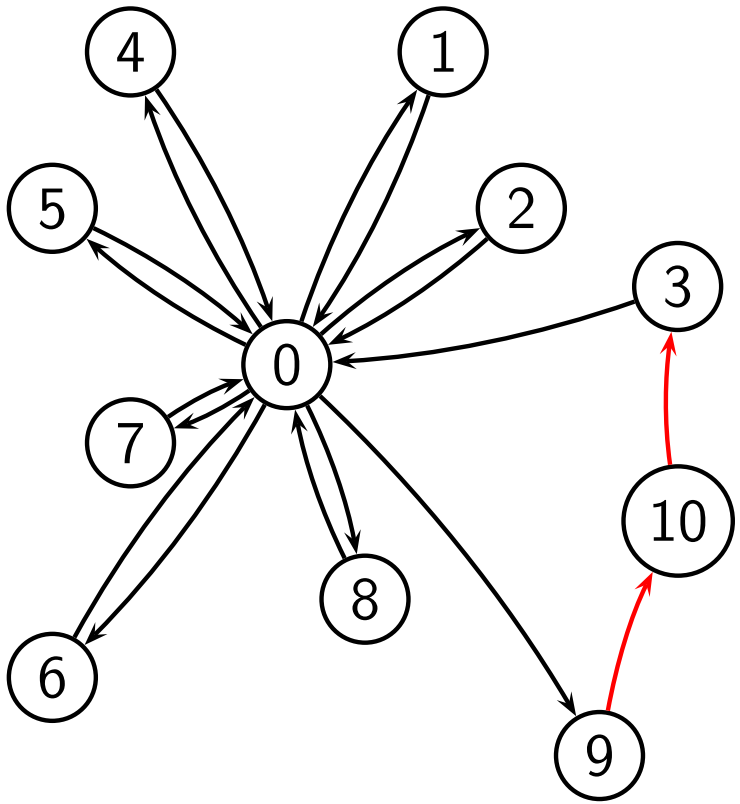
Savings - Example



s_{ij}	1	2	3	4	5	6	7	8	9	10
1	×	5.8	5.3	4.9	2.7	0.0	0.3	0.6	1.7	3.1
2	5.8	×	6.5	2.7	1.2	0.1	0.0	1.4	2.9	4.5
3	5.3	6.5	×	2.0	0.6	0.7	0.1	2.6	5.4	7.5
4	4.9	2.7	2.0	×	5.8	1.4	1.7	0.0	0.1	0.6
5	2.7	1.2	0.6	5.8	×	2.6	2.7	0.4	0.1	0.0
6	0.0	0.1	0.7	1.4	2.6	×	4.1	4.0	4.3	2.1
7	0.3	0.0	0.1	1.7	2.7	4.1	×	1.8	1.4	0.6
8	0.6	1.4	2.6	0.0	0.4	4.0	1.8	×	6.0	4.4
9	1.7	2.9	5.4	0.1	0.1	4.3	1.4	6.0	×	8.6
10	3.1	4.5	7.5	0.6	0.0	2.1	0.6	4.4	8.6	×
d_i	2	1	1	2	1	1	2	1	1	2

$C = 4$

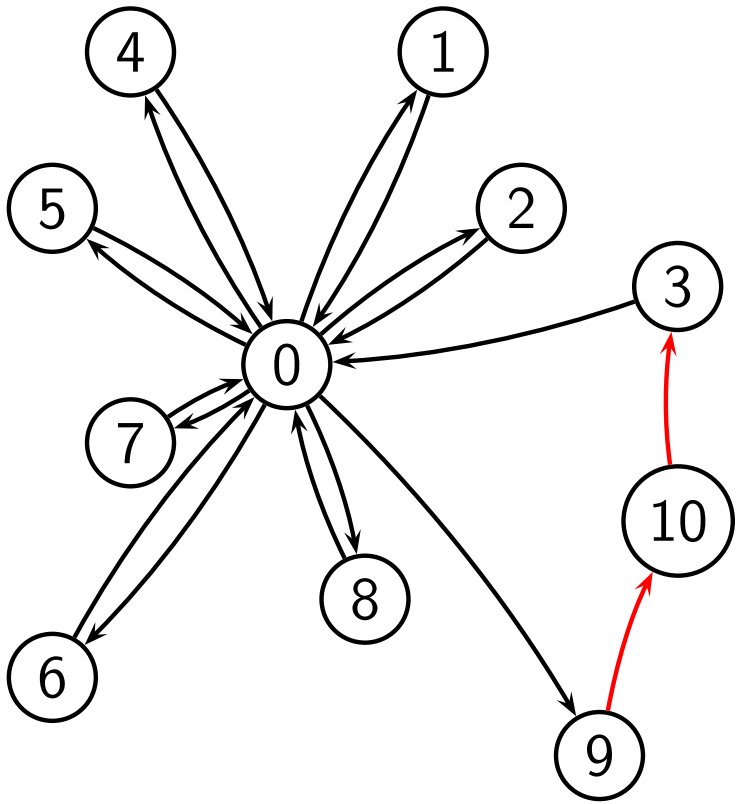
Savings - Example



s_{ij}	1	2	3	4	5	6	7	8	9	10
1	×	5.8	5.3	4.9	2.7	0.0	0.3	0.6	1.7	3.1
2	5.8	×	6.5	2.7	1.2	0.1	0.0	1.4	2.9	4.5
3	5.3	6.5	×	2.0	0.6	0.7	0.1	2.6	5.4	7.5
4	4.9	2.7	2.0	×	5.8	1.4	1.7	0.0	0.1	0.6
5	2.7	1.2	0.6	5.8	×	2.6	2.7	0.4	0.1	0.0
6	0.0	0.1	0.7	1.4	2.6	×	4.1	4.0	4.3	2.1
7	0.3	0.0	0.1	1.7	2.7	4.1	×	1.8	1.4	0.6
8	0.6	1.4	2.6	0.0	0.4	4.0	1.8	×	6.0	4.4
9	1.7	2.9	5.4	0.1	0.1	4.3	1.4	6.0	×	8.6
10	3.1	4.5	7.5	0.6	0.0	2.1	0.6	4.4	8.6	×
d_i	2	1	1	2	1	1	2	1	1	2

$C = 4$

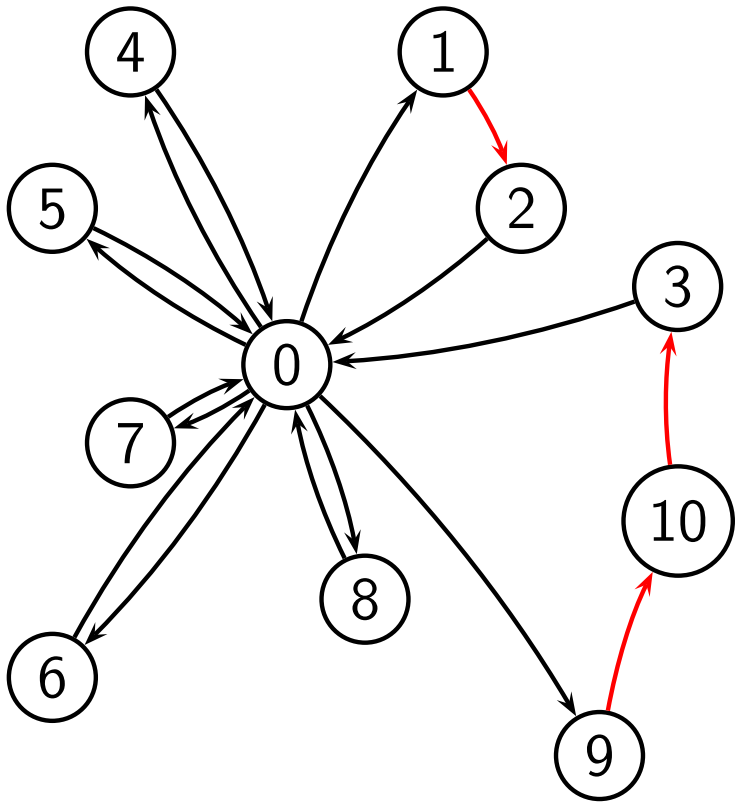
Savings - Example



s_{ij}	1	2	3	4	5	6	7	8	9	10
1	×	5.8	5.3	4.9	2.7	0.0	0.3	0.6	1.7	3.1
2	5.8	×	6.5	2.7	1.2	0.1	0.0	1.4	2.9	4.5
3	5.3	6.5	×	2.0	0.6	0.7	0.1	2.6	5.4	7.5
4	4.9	2.7	2.0	×	5.8	1.4	1.7	0.0	0.1	0.6
5	2.7	1.2	0.6	5.8	×	2.6	2.7	0.4	0.1	0.0
6	0.0	0.1	0.7	1.4	2.6	×	4.1	4.0	4.3	2.1
7	0.3	0.0	0.1	1.7	2.7	4.1	×	1.8	1.4	0.6
8	0.6	1.4	2.6	0.0	0.4	4.0	1.8	×	6.0	4.4
9	1.7	2.9	5.4	0.1	0.1	4.3	1.4	6.0	×	8.6
10	3.1	4.5	7.5	0.6	0.0	2.1	0.6	4.4	8.6	×
d_i	2	1	1	2	1	1	2	1	1	2

$C = 4$

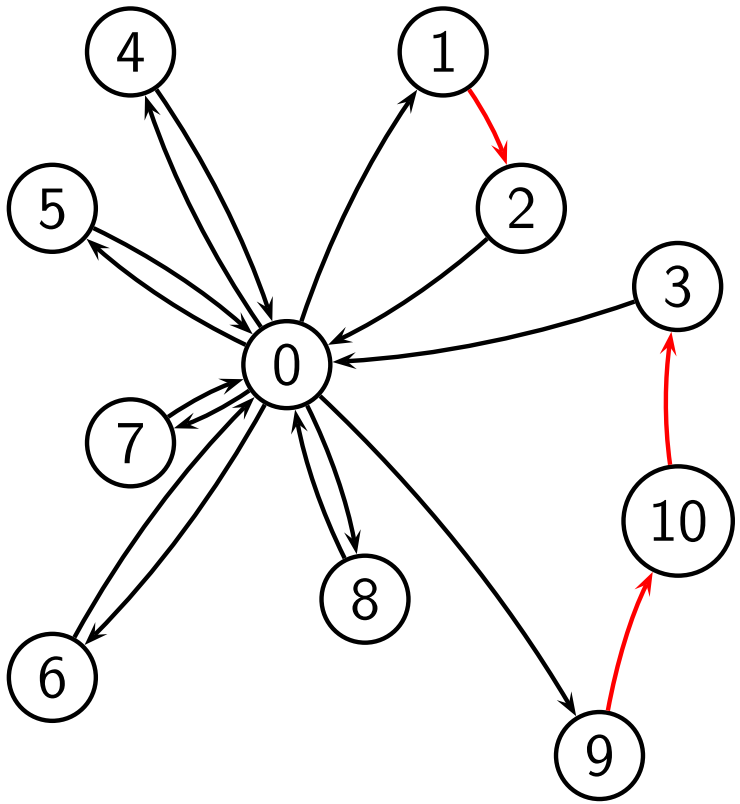
Savings - Example



s_{ij}	1	2	3	4	5	6	7	8	9	10
1	×	5.8	5.3	4.9	2.7	0.0	0.3	0.6	1.7	3.1
2	5.8	×	6.5	2.7	1.2	0.1	0.0	1.4	2.9	4.5
3	5.3	6.5	×	2.0	0.6	0.7	0.1	2.6	5.4	7.5
4	4.9	2.7	2.0	×	5.8	1.4	1.7	0.0	0.1	0.6
5	2.7	1.2	0.6	5.8	×	2.6	2.7	0.4	0.1	0.0
6	0.0	0.1	0.7	1.4	2.6	×	4.1	4.0	4.3	2.1
7	0.3	0.0	0.1	1.7	2.7	4.1	×	1.8	1.4	0.6
8	0.6	1.4	2.6	0.0	0.4	4.0	1.8	×	6.0	4.4
9	1.7	2.9	5.4	0.1	0.1	4.3	1.4	6.0	×	8.6
10	3.1	4.5	7.5	0.6	0.0	2.1	0.6	4.4	8.6	×
d_i	2	1	1	2	1	1	2	1	1	2

$C = 4$

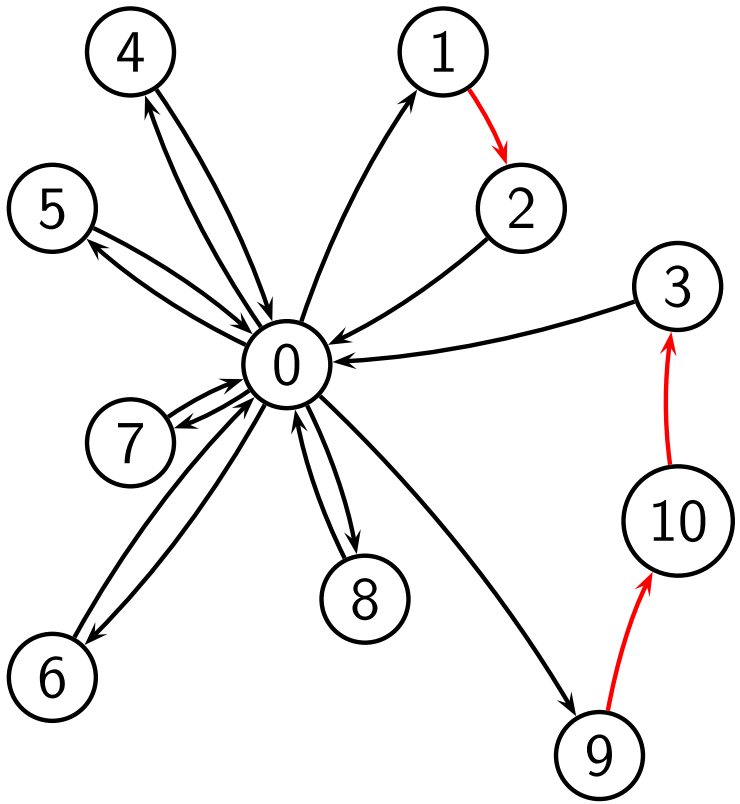
Savings - Example



s_{ij}	1	2	3	4	5	6	7	8	9	10
1	×	5.8	5.3	4.9	2.7	0.0	0.3	0.6	1.7	3.1
2	5.8	×	6.5	2.7	1.2	0.1	0.0	1.4	2.9	4.5
3	5.3	6.5	×	2.0	0.6	0.7	0.1	2.6	5.4	7.5
4	4.9	2.7	2.0	×	5.8	1.4	1.7	0.0	0.1	0.6
5	2.7	1.2	0.6	5.8	×	2.6	2.7	0.4	0.1	0.0
6	0.0	0.1	0.7	1.4	2.6	×	4.1	4.0	4.3	2.1
7	0.3	0.0	0.1	1.7	2.7	4.1	×	1.8	1.4	0.6
8	0.6	1.4	2.6	0.0	0.4	4.0	1.8	×	6.0	4.4
9	1.7	2.9	5.4	0.1	0.1	4.3	1.4	6.0	×	8.6
10	3.1	4.5	7.5	0.6	0.0	2.1	0.6	4.4	8.6	×
d_i	2	1	1	2	1	1	2	1	1	2

$C = 4$

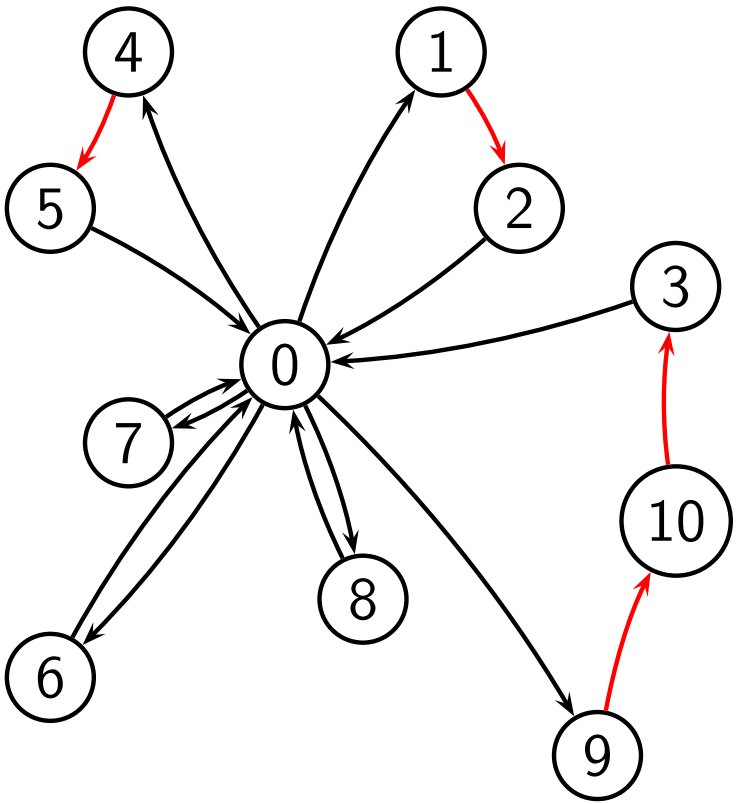
Savings - Example



s_{ij}	1	2	3	4	5	6	7	8	9	10
1	×	5.8	5.3	4.9	2.7	0.0	0.3	0.6	1.7	3.1
2	5.8	×	6.5	2.7	1.2	0.1	0.0	1.4	2.9	4.5
3	5.3	6.5	×	2.0	0.6	0.7	0.1	2.6	5.4	7.5
4	4.9	2.7	2.0	×	5.8	1.4	1.7	0.0	0.1	0.6
5	2.7	1.2	0.6	5.8	×	2.6	2.7	0.4	0.1	0.0
6	0.0	0.1	0.7	1.4	2.6	×	4.1	4.0	4.3	2.1
7	0.3	0.0	0.1	1.7	2.7	4.1	×	1.8	1.4	0.6
8	0.6	1.4	2.6	0.0	0.4	4.0	1.8	×	6.0	4.4
9	1.7	2.9	5.4	0.1	0.1	4.3	1.4	6.0	×	8.6
10	3.1	4.5	7.5	0.6	0.0	2.1	0.6	4.4	8.6	×
d_i	2	1	1	2	1	1	2	1	1	2

$C = 4$

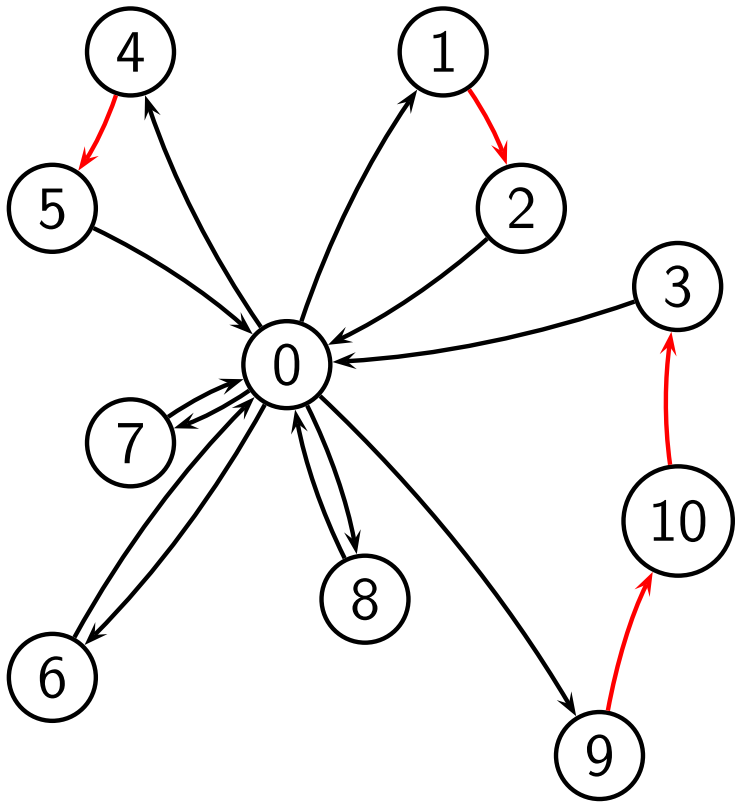
Savings - Example



s_{ij}	1	2	3	4	5	6	7	8	9	10
1	×	5.8	5.3	4.9	2.7	0.0	0.3	0.6	1.7	3.1
2	5.8	×	6.5	2.7	1.2	0.1	0.0	1.4	2.9	4.5
3	5.3	6.5	×	2.0	0.6	0.7	0.1	2.6	5.4	7.5
4	4.9	2.7	2.0	×	5.8	1.4	1.7	0.0	0.1	0.6
5	2.7	1.2	0.6	5.8	×	2.6	2.7	0.4	0.1	0.0
6	0.0	0.1	0.7	1.4	2.6	×	4.1	4.0	4.3	2.1
7	0.3	0.0	0.1	1.7	2.7	4.1	×	1.8	1.4	0.6
8	0.6	1.4	2.6	0.0	0.4	4.0	1.8	×	6.0	4.4
9	1.7	2.9	5.4	0.1	0.1	4.3	1.4	6.0	×	8.6
10	3.1	4.5	7.5	0.6	0.0	2.1	0.6	4.4	8.6	×
d_i	2	1	1	2	1	1	2	1	1	2

$C = 4$

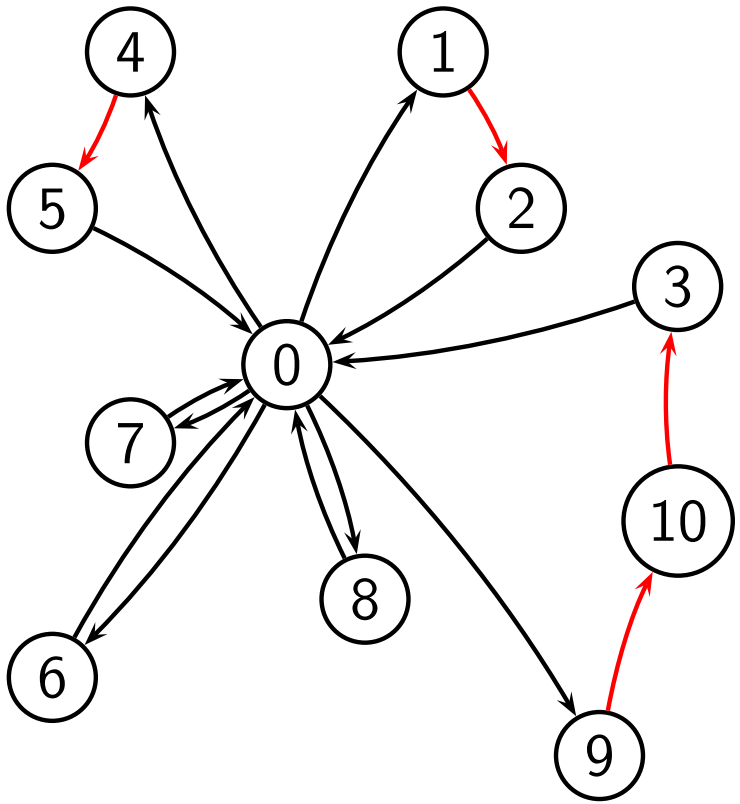
Savings - Example



s_{ij}	1	2	3	4	5	6	7	8	9	10
1	x	5.8	5.3	4.9	2.7	0.0	0.3	0.6	1.7	3.1
2	5.8	x	6.5	2.7	1.2	0.1	0.0	1.4	2.9	4.5
3	5.3	6.5	x	2.0	0.6	0.7	0.1	2.6	5.4	7.5
4	4.9	2.7	2.0	x	5.8	1.4	1.7	0.0	0.1	0.6
5	2.7	1.2	0.6	5.8	x	2.6	2.7	0.4	0.1	0.0
6	0.0	0.1	0.7	1.4	2.6	x	4.1	4.0	4.3	2.1
7	0.3	0.0	0.1	1.7	2.7	4.1	x	1.8	1.4	0.6
8	0.6	1.4	2.6	0.0	0.4	4.0	1.8	x	6.0	4.4
9	1.7	2.9	5.4	0.1	0.1	4.3	1.4	6.0	x	8.6
10	3.1	4.5	7.5	0.6	0.0	2.1	0.6	4.4	8.6	x
d_i	2	1	1	2	1	1	2	1	1	2

$C = 4$

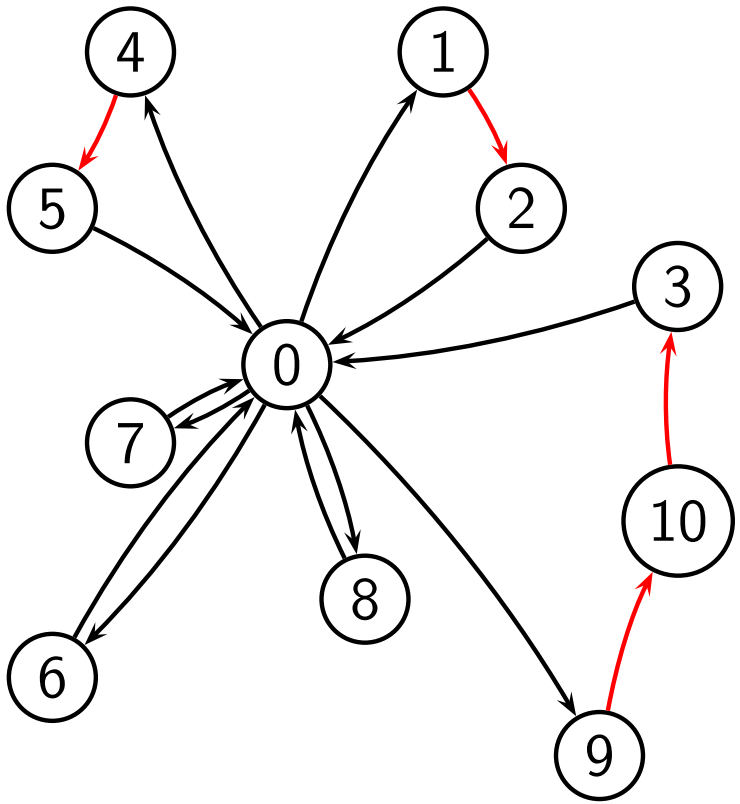
Savings - Example



s_{ij}	1	2	3	4	5	6	7	8	9	10
1	x	5.8	5.3	4.9	2.7	0.0	0.3	0.6	1.7	3.1
2	5.8	x	6.5	2.7	1.2	0.1	0.0	1.4	2.9	4.5
3	5.3	6.5	x	2.0	0.6	0.7	0.1	2.6	5.4	7.5
4	4.9	2.7	2.0	x	5.8	1.4	1.7	0.0	0.1	0.6
5	2.7	1.2	0.6	5.8	x	2.6	2.7	0.4	0.1	0.0
6	0.0	0.1	0.7	1.4	2.6	x	4.1	4.0	4.3	2.1
7	0.3	0.0	0.1	1.7	2.7	4.1	x	1.8	1.4	0.6
8	0.6	1.4	2.6	0.0	0.4	4.0	1.8	x	6.0	4.4
9	1.7	2.9	5.4	0.1	0.1	4.3	1.4	6.0	x	8.6
10	3.1	4.5	7.5	0.6	0.0	2.1	0.6	4.4	8.6	x
d_i	2	1	1	2	1	1	2	1	1	2

$C = 4$

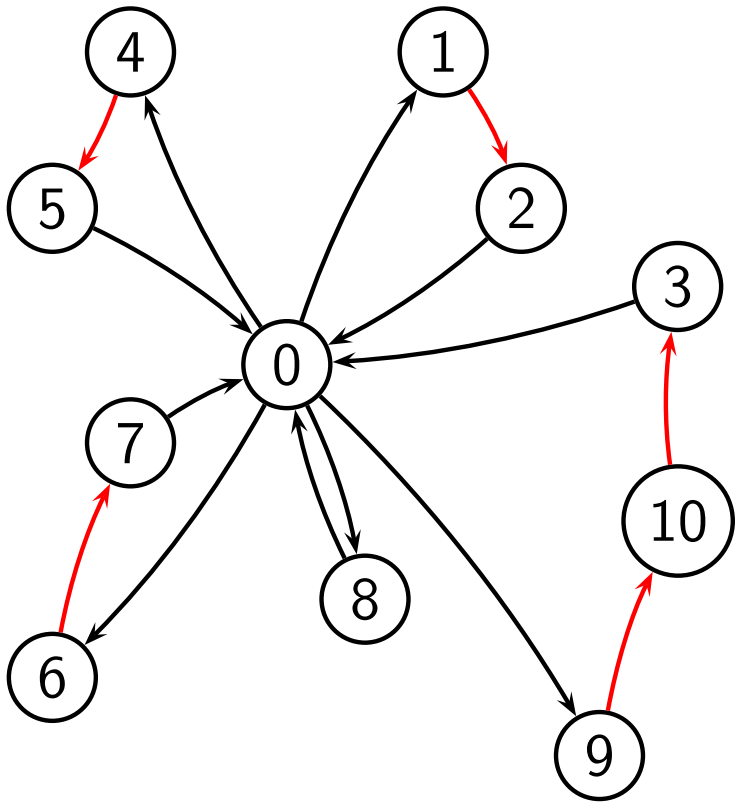
Savings - Example



s_{ij}	1	2	3	4	5	6	7	8	9	10
1	x	5.8	5.3	4.9	2.7	0.0	0.3	0.6	1.7	3.1
2	5.8	x	6.5	2.7	1.2	0.1	0.0	1.4	2.9	4.5
3	5.3	6.5	x	2.0	0.6	0.7	0.1	2.6	5.4	7.5
4	4.9	2.7	2.0	x	5.8	1.4	1.7	0.0	0.1	0.6
5	2.7	1.2	0.6	5.8	x	2.6	2.7	0.4	0.1	0.0
6	0.0	0.1	0.7	1.4	2.6	x	4.1	4.0	4.3	2.1
7	0.3	0.0	0.1	1.7	2.7	4.1	x	1.8	1.4	0.6
8	0.6	1.4	2.6	0.0	0.4	4.0	1.8	x	6.0	4.4
9	1.7	2.9	5.4	0.1	0.1	4.3	1.4	6.0	x	8.6
10	3.1	4.5	7.5	0.6	0.0	2.1	0.6	4.4	8.6	x
d_i	2	1	1	2	1	1	2	1	1	2

$C = 4$

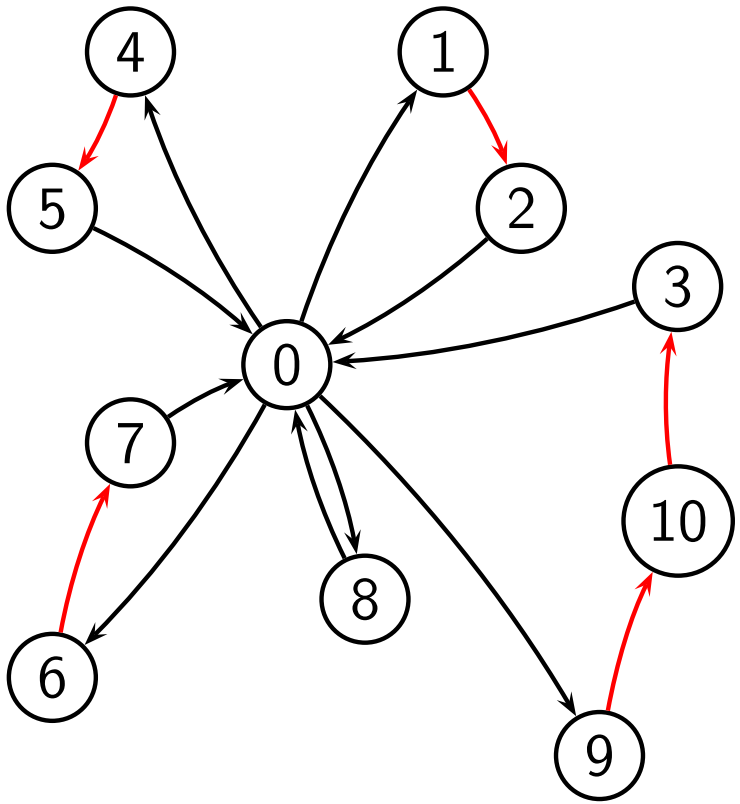
Savings - Example



s_{ij}	1	2	3	4	5	6	7	8	9	10
1	x	5.8	5.3	4.9	2.7	0.0	0.3	0.6	1.7	3.1
2	5.8	x	6.5	2.7	1.2	0.1	0.0	1.4	2.9	4.5
3	5.3	6.5	x	2.0	0.6	0.7	0.1	2.6	5.4	7.5
4	4.9	2.7	2.0	x	5.8	1.4	1.7	0.0	0.1	0.6
5	2.7	1.2	0.6	5.8	x	2.6	2.7	0.4	0.1	0.0
6	0.0	0.1	0.7	1.4	2.6	x	4.1	4.0	4.3	2.1
7	0.3	0.0	0.1	1.7	2.7	4.1	x	1.8	1.4	0.6
8	0.6	1.4	2.6	0.0	0.4	4.0	1.8	x	6.0	4.4
9	1.7	2.9	5.4	0.1	0.1	4.3	1.4	6.0	x	8.6
10	3.1	4.5	7.5	0.6	0.0	2.1	0.6	4.4	8.6	x
d_i	2	1	1	2	1	1	2	1	1	2

$C = 4$

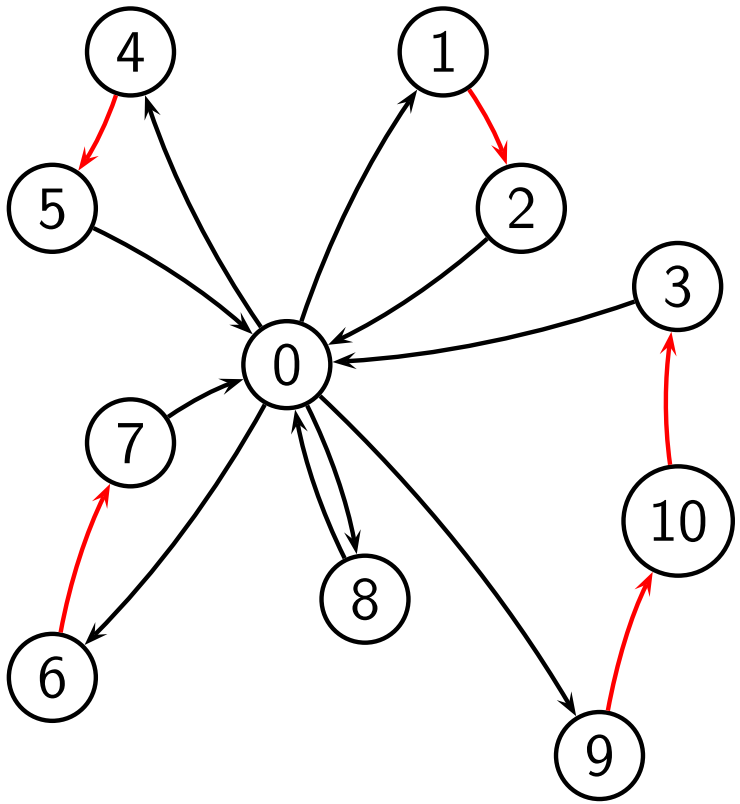
Savings - Example



s_{ij}	1	2	3	4	5	6	7	8	9	10
1	x	5.8	5.3	4.9	2.7	0.0	0.3	0.6	1.7	3.1
2	5.8	x	6.5	2.7	1.2	0.1	0.0	1.4	2.9	4.5
3	5.3	6.5	x	2.0	0.6	0.7	0.1	2.6	5.4	7.5
4	4.9	2.7	2.0	x	5.8	1.4	1.7	0.0	0.1	0.6
5	2.7	1.2	0.6	5.8	x	2.6	2.7	0.4	0.1	0.0
6	0.0	0.1	0.7	1.4	2.6	x	4.1	4.0	4.3	2.1
7	0.3	0.0	0.1	1.7	2.7	4.1	x	1.8	1.4	0.6
8	0.6	1.4	2.6	0.0	0.4	4.0	1.8	x	6.0	4.4
9	1.7	2.9	5.4	0.1	0.1	4.3	1.4	6.0	x	8.6
10	3.1	4.5	7.5	0.6	0.0	2.1	0.6	4.4	8.6	x
d_i	2	1	1	2	1	1	2	1	1	2

$C = 4$

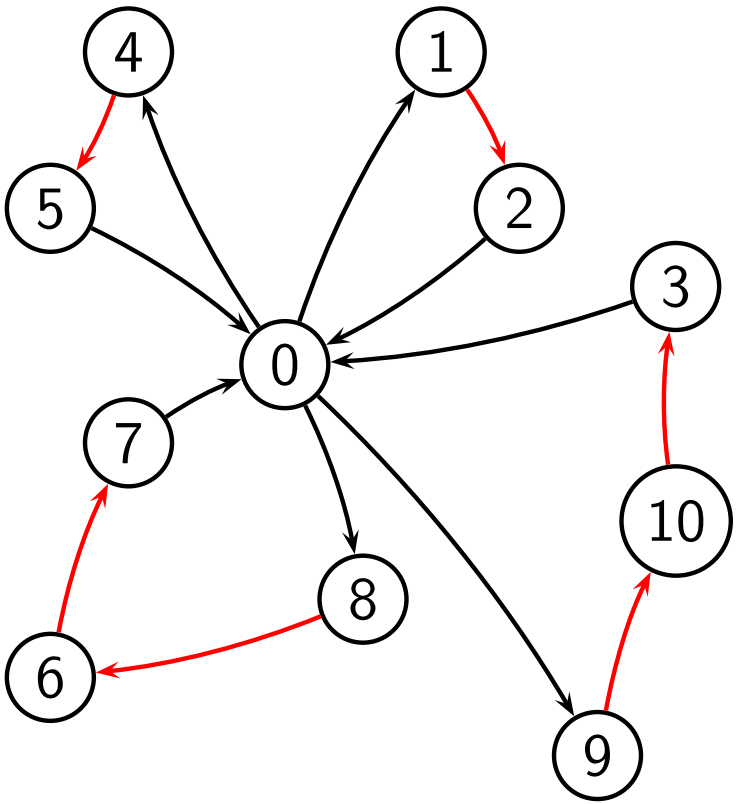
Savings - Example



s_{ij}	1	2	3	4	5	6	7	8	9	10
1	×	5.8	5.3	4.9	2.7	0.0	0.3	0.6	1.7	3.1
2	5.8	×	6.5	2.7	1.2	0.1	0.0	1.4	2.9	4.5
3	5.3	6.5	×	2.0	0.6	0.7	0.1	2.6	5.4	7.5
4	4.9	2.7	2.0	×	5.8	1.4	1.7	0.0	0.1	0.6
5	2.7	1.2	0.6	5.8	×	2.6	2.7	0.4	0.1	0.0
6	0.0	0.1	0.7	1.4	2.6	×	4.1	4.0	4.3	2.1
7	0.3	0.0	0.1	1.7	2.7	4.1	×	1.8	1.4	0.6
8	0.6	1.4	2.6	0.0	0.4	4.0	1.8	×	6.0	4.4
9	1.7	2.9	5.4	0.1	0.1	4.3	1.4	6.0	×	8.6
10	3.1	4.5	7.5	0.6	0.0	2.1	0.6	4.4	8.6	×
d_i	2	1	1	2	1	1	2	1	1	2

$C = 4$

Savings - Example



s_{ij}	1	2	3	4	5	6	7	8	9	10
1	x	5.8	5.3	4.9	2.7	0.0	0.3	0.6	1.7	3.1
2	5.8	x	6.5	2.7	1.2	0.1	0.0	1.4	2.9	4.5
3	5.3	6.5	x	2.0	0.6	0.7	0.1	2.6	5.4	7.5
4	4.9	2.7	2.0	x	5.8	1.4	1.7	0.0	0.1	0.6
5	2.7	1.2	0.6	5.8	x	2.6	2.7	0.4	0.1	0.0
6	0.0	0.1	0.7	1.4	2.6	x	4.1	4.0	4.3	2.1
7	0.3	0.0	0.1	1.7	2.7	4.1	x	1.8	1.4	0.6
8	0.6	1.4	2.6	0.0	0.4	4.0	1.8	x	6.0	4.4
9	1.7	2.9	5.4	0.1	0.1	4.3	1.4	6.0	x	8.6
10	3.1	4.5	7.5	0.6	0.0	2.1	0.6	4.4	8.6	x
d_i	2	1	1	2	1	1	2	1	1	2

$C = 4$

A sequential insertion algorithm

- **Step 1** If all vertices belong to a route $\rightarrow$ STOP. Otherwise, initialize a new tour with an unrouted vertex k. New current tour $(0, k, 0)$. Choose k according to a certain criterion, e.g. random, closests to 0, farthest from 0 ...
- **Step 2** Do cheapest feasible insertion into the current tour until no additional vertex can be feasibly inserted. In this case, go to step 1.

A sequential insertion algorithm

- **Step 1** If all vertices belong to a route $\rightarrow$ STOP. Otherwise, initialize a new tour with an unrouted vertex k. New current tour $(0, k, 0)$. Choose k according to a certain criterion, e.g. random, closests to 0, farthest from 0 ...
- **Step 2** Do cheapest feasible insertion into the current tour until no additional vertex can be feasibly inserted. In this case, go to step 1.

★ One route is filled after the other.

A parallel insertion algorithm

- **Step 1** Initialize the minimum number of routes $\lceil \sum_i d_i / C \rceil$ with seed customers (chosen randomly, closest to 0, farthest from 0 ...)
- **Step 2** Start with customer 1. Sequentially, insert one customer after the other at its best insertion position. If a customer cannot be inserted in a feasible way in any of the routes, add an additional route.

A parallel insertion algorithm

- **Step 1** Initialize the minimum number of routes $\lceil \sum_i d_i / C \rceil$ with seed customers (chosen randomly, closest to 0, farthest from 0 ...)
- **Step 2** Start with customer 1. Sequentially, insert one customer after the other at its best insertion position. If a customer cannot be inserted in a feasible way in any of the routes, add an additional route.

★ Several routes are considered and filled simultaneously.

Cluster first route second heuristics

- Sweep algorithm
- Fisher and Jaikumar Algorithm

Sweep algorithm

★ Only applicable to planar instances (polar coordinates necessary: angle and ray length).

Sweep algorithm

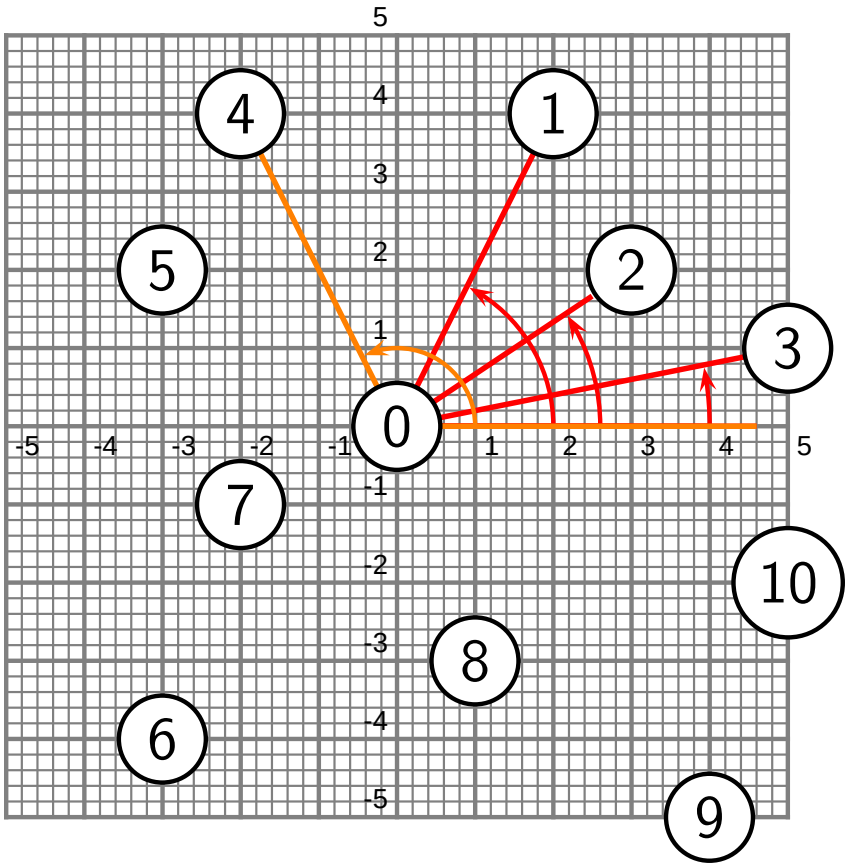
★ Only applicable to planar instances (polar coordinates necessary: angle and ray length).

Clusters are formed by rotating a ray centered at the depot:

- **Clustering** Start with the unassigned vertex with the smallest anlge. Assign vertices to vehicle k as long as the capacity (or the route length) is not exceeded. If there are still unrouted vertices, initialize a new vehicle.
- **Routing** Solve a TSP for each cluster (exact or heuristic methods, e.g. nearest neighbor).

Sweep algorithm - Example

Clustering

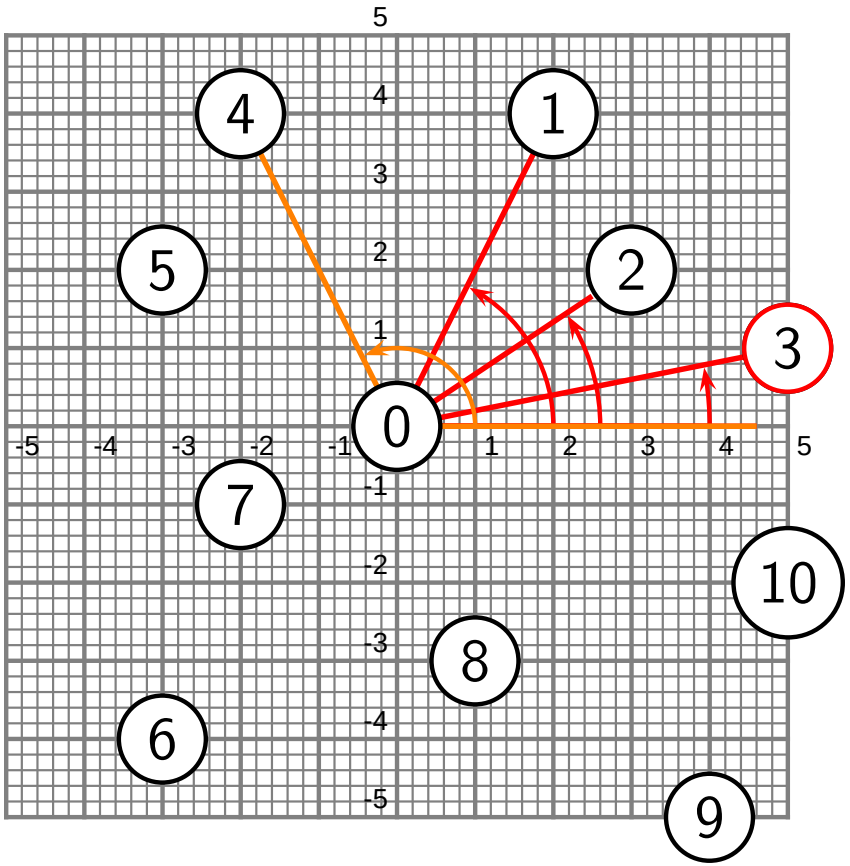


i	x	y	d_i	angle
0	0	0	0	0.00
1	2	4	2	63.4
2	3	2	1	33.7
3	5	1	1	11.3
4	-2	4	2	116.6
5	-3	2	1	146.3
6	-3	-4	1	233.1
7	-2	-1	2	206.6
8	1	-3	1	288.4
9	4	-5	1	308.7
10	5	-2	2	338.2

$C = 4$

Sweep algorithm - Example

Clustering



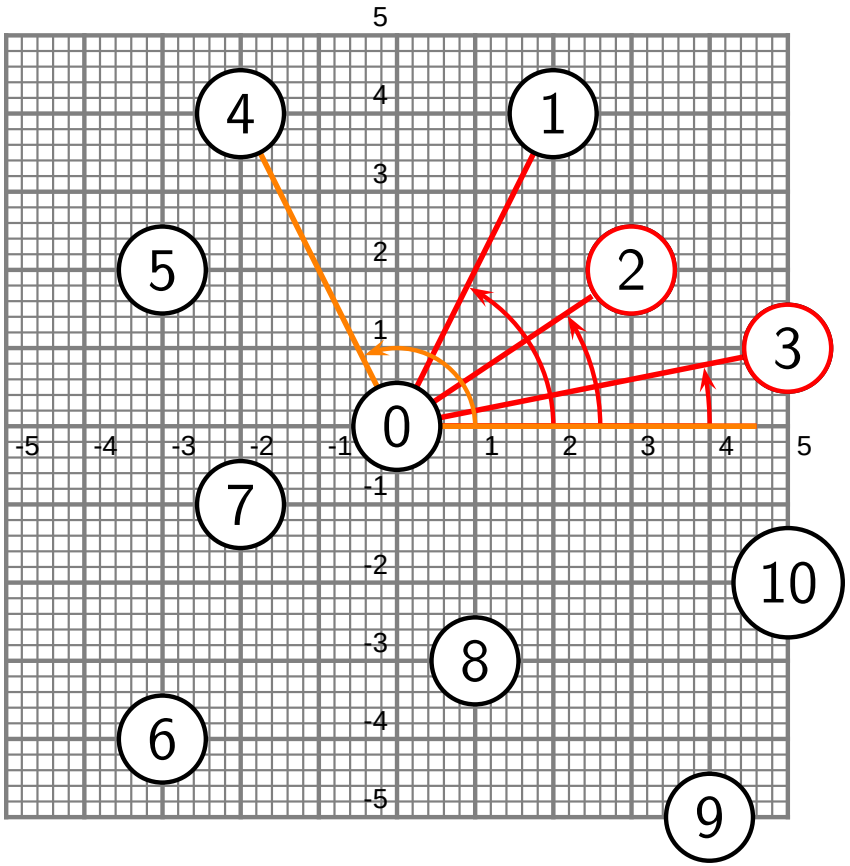
i	x	y	d_i	angle
0	0	0	0	0.00
1	2	4	2	63.4
2	3	2	1	33.7
3	5	1	1	11.3
4	-2	4	2	116.6
5	-3	2	1	146.3
6	-3	-4	1	233.1
7	-2	-1	2	206.6
8	1	-3	1	288.4
9	4	-5	1	308.7
10	5	-2	2	338.2

$C = 4$

3,

Sweep algorithm - Example

Clustering



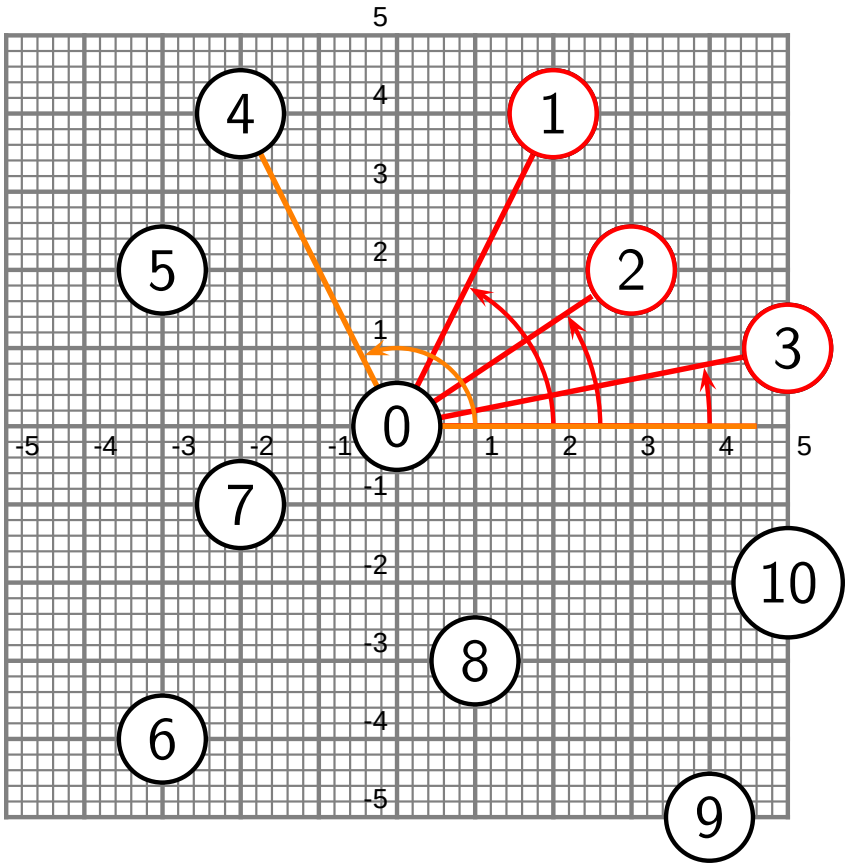
i	x	y	d_i	angle
0	0	0	0	0.00
1	2	4	2	63.4
2	3	2	1	33.7
3	5	1	1	11.3
4	-2	4	2	116.6
5	-3	2	1	146.3
6	-3	-4	1	233.1
7	-2	-1	2	206.6
8	1	-3	1	288.4
9	4	-5	1	308.7
10	5	-2	2	338.2

$C = 4$

3, 2,

Sweep algorithm - Example

Clustering



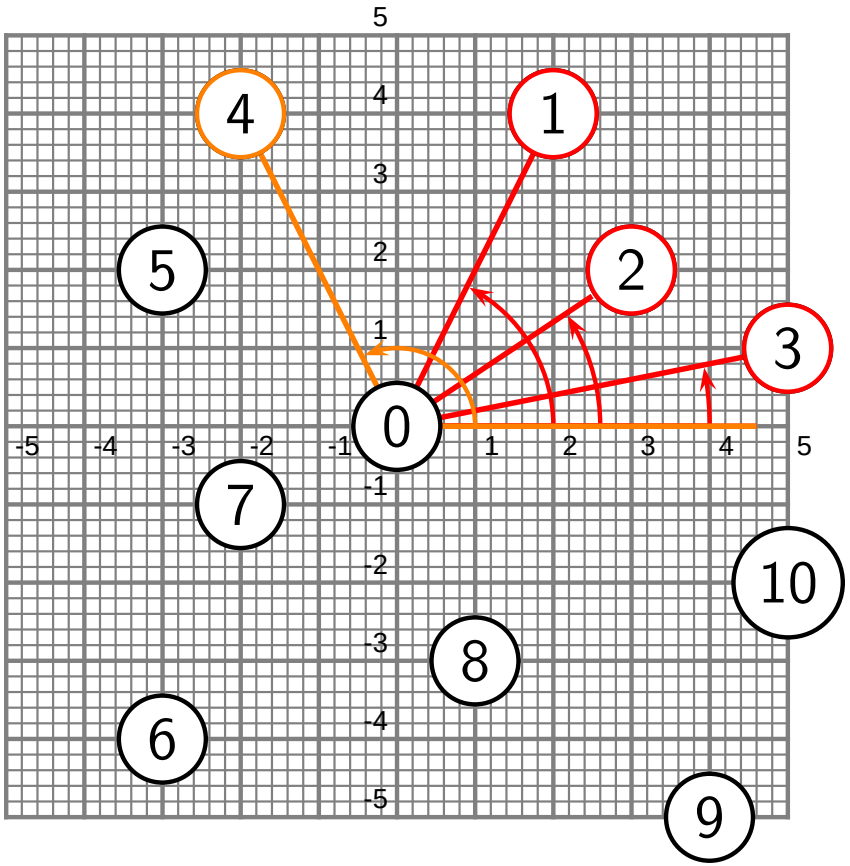
i	x	y	d_i	angle
0	0	0	0	0.00
1	2	4	2	63.4
2	3	2	1	33.7
3	5	1	1	11.3
4	-2	4	2	116.6
5	-3	2	1	146.3
6	-3	-4	1	233.1
7	-2	-1	2	206.6
8	1	-3	1	288.4
9	4	-5	1	308.7
10	5	-2	2	338.2

$C = 4$

3, 2, 1

Sweep algorithm - Example

Clustering



i	x	y	d_i	angle
0	0	0	0	0.00
1	2	4	2	63.4
2	3	2	1	33.7
3	5	1	1	11.3
4	-2	4	2	116.6
5	-3	2	1	146.3
6	-3	-4	1	233.1
7	-2	-1	2	206.6
8	1	-3	1	288.4
9	4	-5	1	308.7
10	5	-2	2	338.2

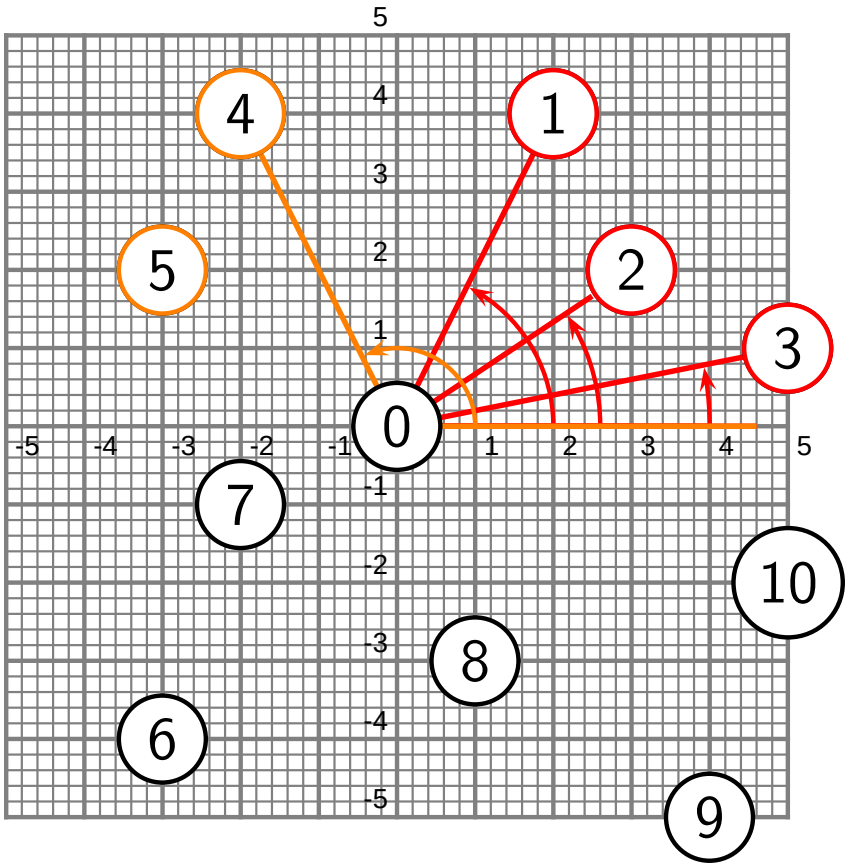
$C = 4$

3, 2, 1

4,

Sweep algorithm - Example

Clustering



i	x	y	d_i	angle
0	0	0	0	0.00
1	2	4	2	63.4
2	3	2	1	33.7
3	5	1	1	11.3
4	-2	4	2	116.6
5	-3	2	1	146.3
6	-3	-4	1	233.1
7	-2	-1	2	206.6
8	1	-3	1	288.4
9	4	-5	1	308.7
10	5	-2	2	338.2

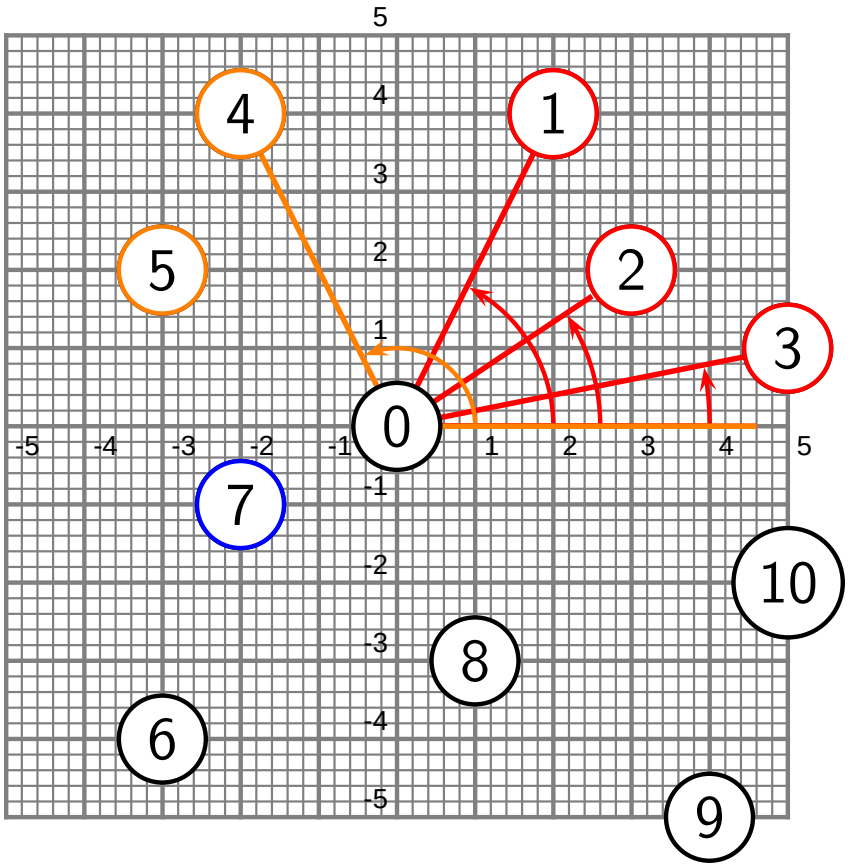
$C = 4$

3, 2, 1

4, 5

Sweep algorithm - Example

Clustering



i	x	y	d_i	angle
0	0	0	0	0.00
1	2	4	2	63.4
2	3	2	1	33.7
3	5	1	1	11.3
4	-2	4	2	116.6
5	-3	2	1	146.3
6	-3	-4	1	233.1
7	-2	-1	2	206.6
8	1	-3	1	288.4
9	4	-5	1	308.7
10	5	-2	2	338.2

$C = 4$

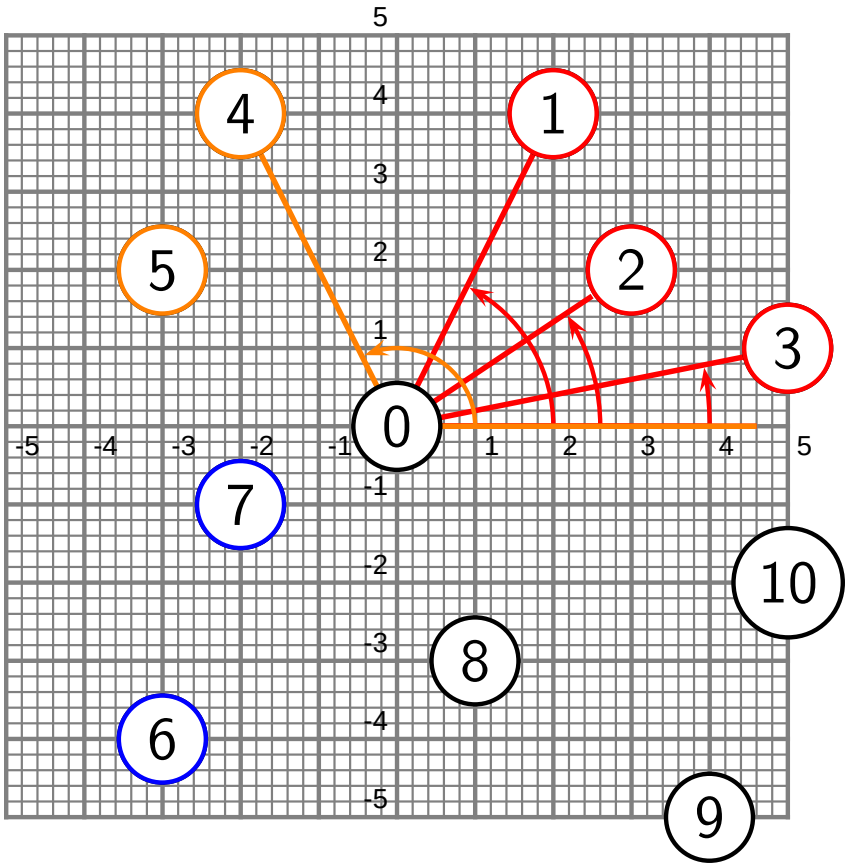
3, 2, 1

4, 5

7,

Sweep algorithm - Example

Clustering



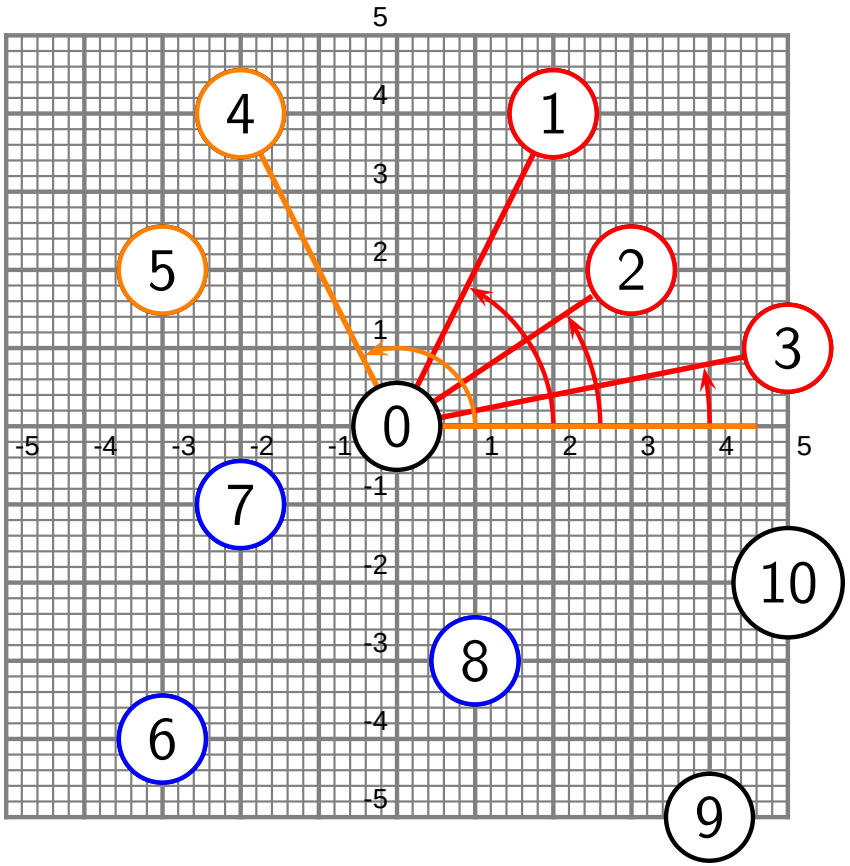
i	x	y	d_i	angle
0	0	0	0	0.00
1	2	4	2	63.4
2	3	2	1	33.7
3	5	1	1	11.3
4	-2	4	2	116.6
5	-3	2	1	146.3
6	-3	-4	1	233.1
7	-2	-1	2	206.6
8	1	-3	1	288.4
9	4	-5	1	308.7
10	5	-2	2	338.2

$C = 4$

3, 2, 1
4, 5
7, 6,

Sweep algorithm - Example

Clustering



i	x	y	d_i	angle
0	0	0	0	0.00
1	2	4	2	63.4
2	3	2	1	33.7
3	5	1	1	11.3
4	-2	4	2	116.6
5	-3	2	1	146.3
6	-3	-4	1	233.1
7	-2	-1	2	206.6
8	1	-3	1	288.4
9	4	-5	1	308.7
10	5	-2	2	338.2

$C = 4$

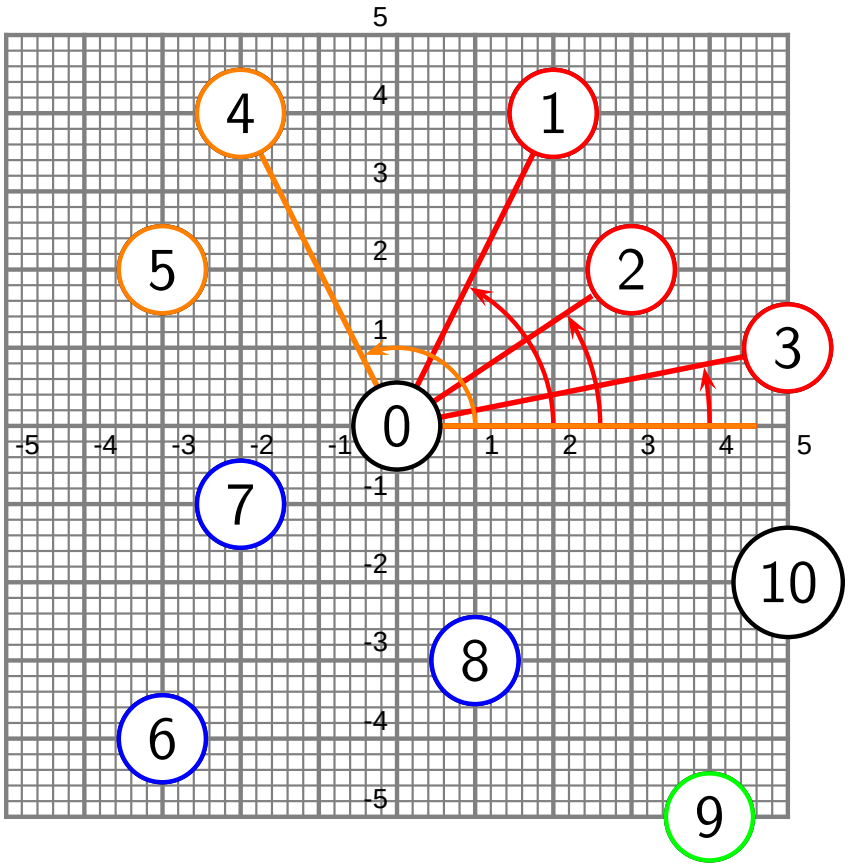
3, 2, 1

4, 5

7, 6, 8

Sweep algorithm - Example

Clustering



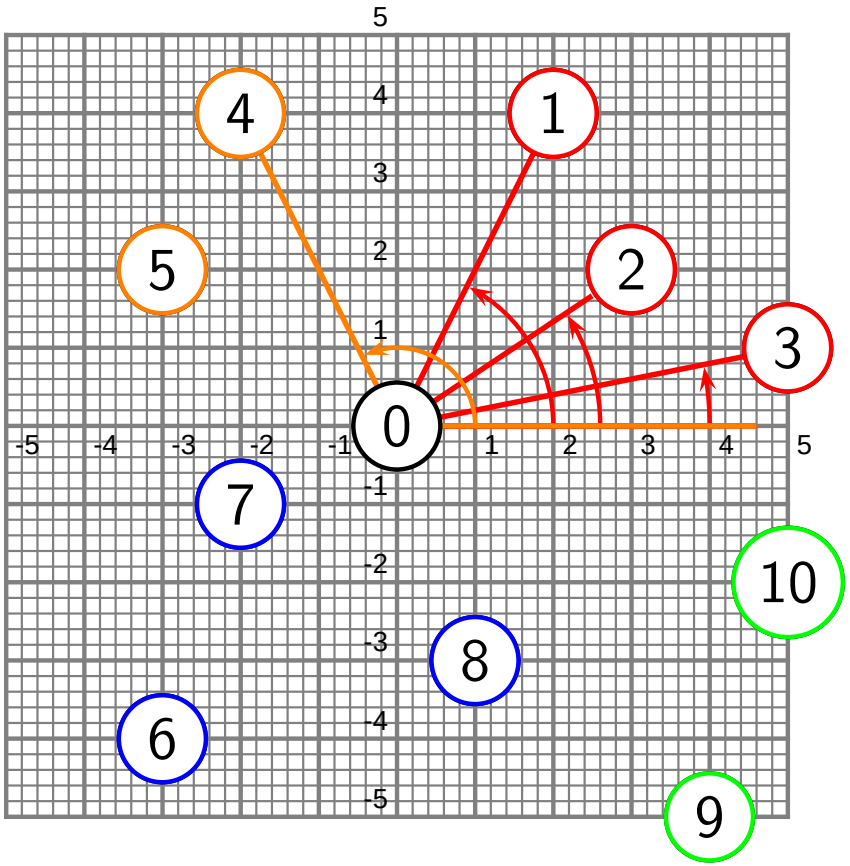
i	x	y	d_i	angle
0	0	0	0	0.00
1	2	4	2	63.4
2	3	2	1	33.7
3	5	1	1	11.3
4	-2	4	2	116.6
5	-3	2	1	146.3
6	-3	-4	1	233.1
7	-2	-1	2	206.6
8	1	-3	1	288.4
9	4	-5	1	308.7
10	5	-2	2	338.2

$C = 4$

3, 2, 1
4, 5
7, 6, 8
9,

Sweep algorithm - Example

Clustering



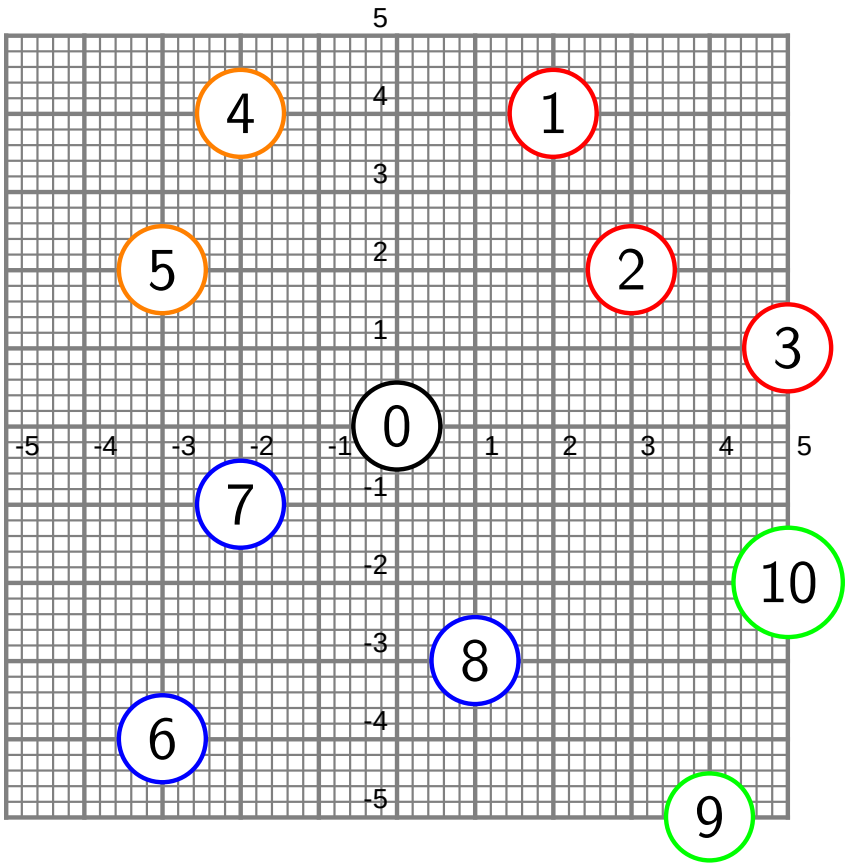
i	x	y	d_i	angle
0	0	0	0	0.00
1	2	4	2	63.4
2	3	2	1	33.7
3	5	1	1	11.3
4	-2	4	2	116.6
5	-3	2	1	146.3
6	-3	-4	1	233.1
7	-2	-1	2	206.6
8	1	-3	1	288.4
9	4	-5	1	308.7
10	5	-2	2	338.2

$C = 4$

3, 2, 1
4, 5
7, 6, 8
9, 10

Sweep algorithm - Example

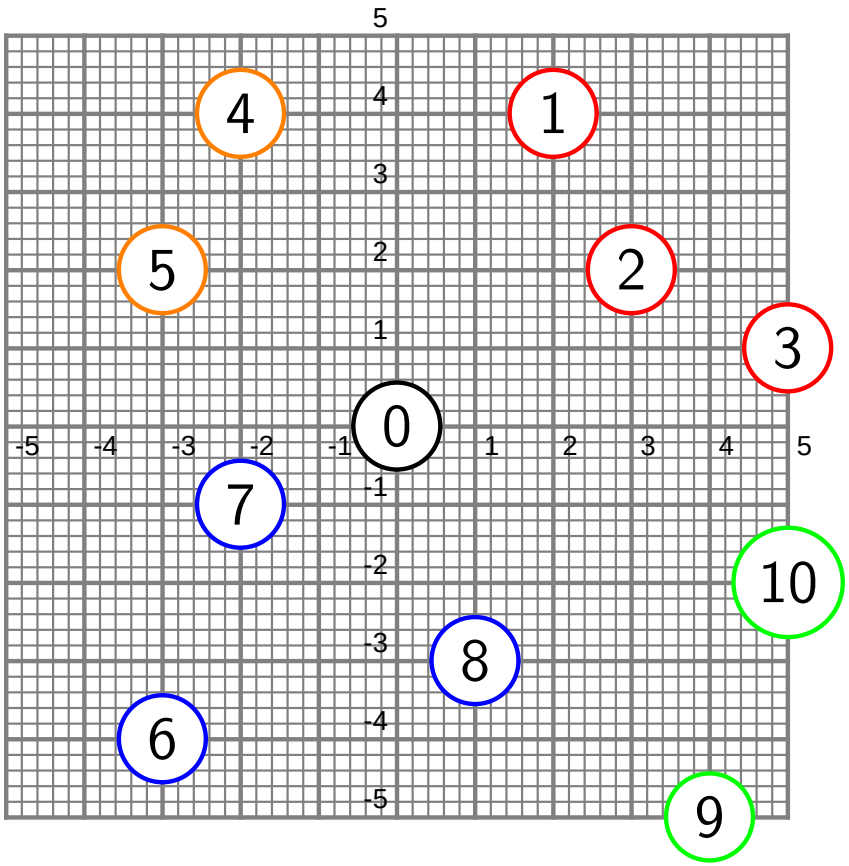
Routing (Nearest neighbor)



c_{ij}	0	1	2	3	4	5	6	7	8	9	10
0	0.0	4.5	3.6	5.1	4.5	3.6	5.0	2.2	3.2	6.4	5.4
1	4.5	0.0	2.2	4.2	4.0	5.4	9.4	6.4	7.1	9.2	6.7
2	3.6	2.2	0.0	2.2	5.4	6.0	8.5	5.8	5.4	7.1	4.5
3	5.1	4.2	2.2	0.0	7.6	8.1	9.4	7.3	5.7	6.1	3.0
4	4.5	4.0	5.4	7.6	0.0	2.2	8.1	5.0	7.6	10.8	9.2
5	3.6	5.4	6.0	8.1	2.2	0.0	6.0	3.2	6.4	9.9	8.9
6	5.0	9.4	8.5	9.4	8.1	6.0	0.0	3.2	4.1	7.1	8.2
7	2.2	6.4	5.8	7.3	5.0	3.2	3.2	0.0	3.6	7.2	7.1
8	3.2	7.1	5.4	5.7	7.6	6.4	4.1	3.6	0.0	3.6	4.1
9	6.4	9.2	7.1	6.1	10.8	9.9	7.1	7.2	3.6	0.0	3.2
10	5.4	6.7	4.5	3.0	9.2	8.9	8.2	7.1	4.1	3.2	0.0

Sweep algorithm - Example

Routing (Nearest neighbor)

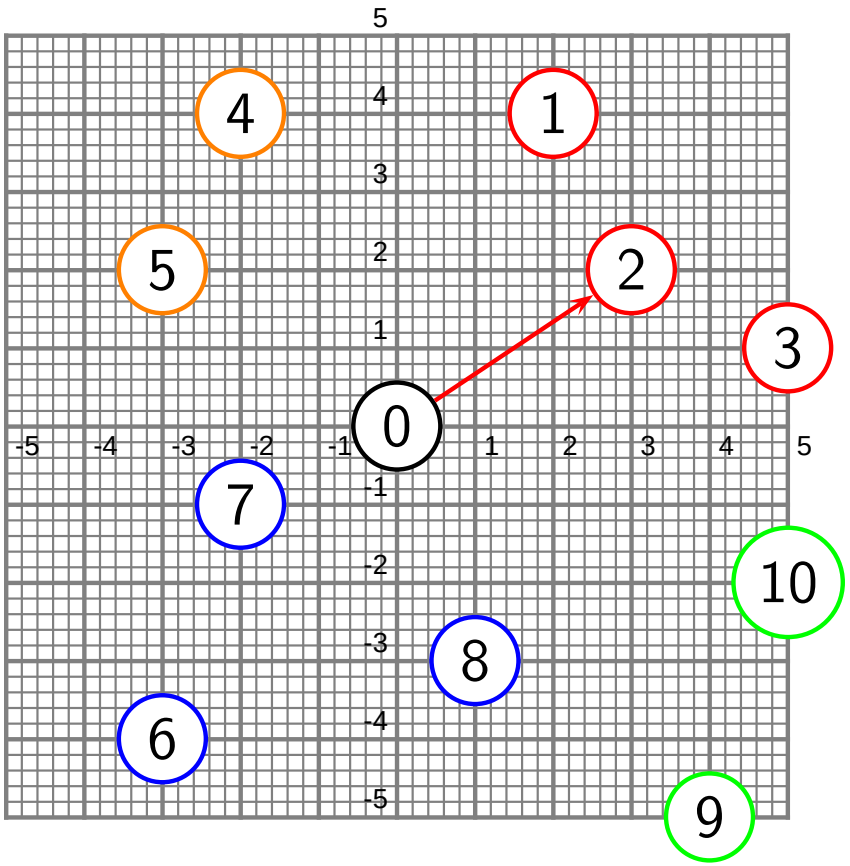


c_{ij}	0	1	2	3	4	5	6	7	8	9	10
0	0.0	4.5	3.6	5.1	4.5	3.6	5.0	2.2	3.2	6.4	5.4
1	4.5	0.0	2.2	4.2	4.0	5.4	9.4	6.4	7.1	9.2	6.7
2	3.6	2.2	0.0	2.2	5.4	6.0	8.5	5.8	5.4	7.1	4.5
3	5.1	4.2	2.2	0.0	7.6	8.1	9.4	7.3	5.7	6.1	3.0
4	4.5	4.0	5.4	7.6	0.0	2.2	8.1	5.0	7.6	10.8	9.2
5	3.6	5.4	6.0	8.1	2.2	0.0	6.0	3.2	6.4	9.9	8.9
6	5.0	9.4	8.5	9.4	8.1	6.0	0.0	3.2	4.1	7.1	8.2
7	2.2	6.4	5.8	7.3	5.0	3.2	3.2	0.0	3.6	7.2	7.1
8	3.2	7.1	5.4	5.7	7.6	6.4	4.1	3.6	0.0	3.6	4.1
9	6.4	9.2	7.1	6.1	10.8	9.9	7.1	7.2	3.6	0.0	3.2
10	5.4	6.7	4.5	3.0	9.2	8.9	8.2	7.1	4.1	3.2	0.0

0

Sweep algorithm - Example

Routing (Nearest neighbor)

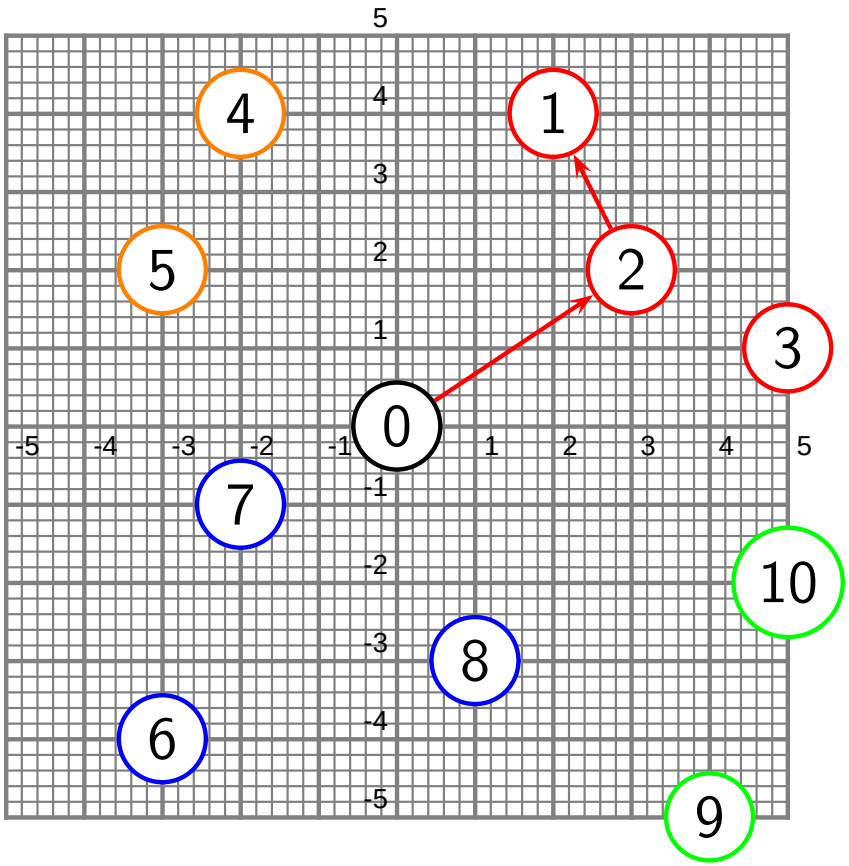


c_{ij}	0	1	2	3	4	5	6	7	8	9	10
0	0.0	4.5	3.6	5.1	4.5	3.6	5.0	2.2	3.2	6.4	5.4
1	4.5	0.0	2.2	4.2	4.0	5.4	9.4	6.4	7.1	9.2	6.7
2	3.6	2.2	0.0	2.2	5.4	6.0	8.5	5.8	5.4	7.1	4.5
3	5.1	4.2	2.2	0.0	7.6	8.1	9.4	7.3	5.7	6.1	3.0
4	4.5	4.0	5.4	7.6	0.0	2.2	8.1	5.0	7.6	10.8	9.2
5	3.6	5.4	6.0	8.1	2.2	0.0	6.0	3.2	6.4	9.9	8.9
6	5.0	9.4	8.5	9.4	8.1	6.0	0.0	3.2	4.1	7.1	8.2
7	2.2	6.4	5.8	7.3	5.0	3.2	3.2	0.0	3.6	7.2	7.1
8	3.2	7.1	5.4	5.7	7.6	6.4	4.1	3.6	0.0	3.6	4.1
9	6.4	9.2	7.1	6.1	10.8	9.9	7.1	7.2	3.6	0.0	3.2
10	5.4	6.7	4.5	3.0	9.2	8.9	8.2	7.1	4.1	3.2	0.0

0-2

Sweep algorithm - Example

Routing (Nearest neighbor)

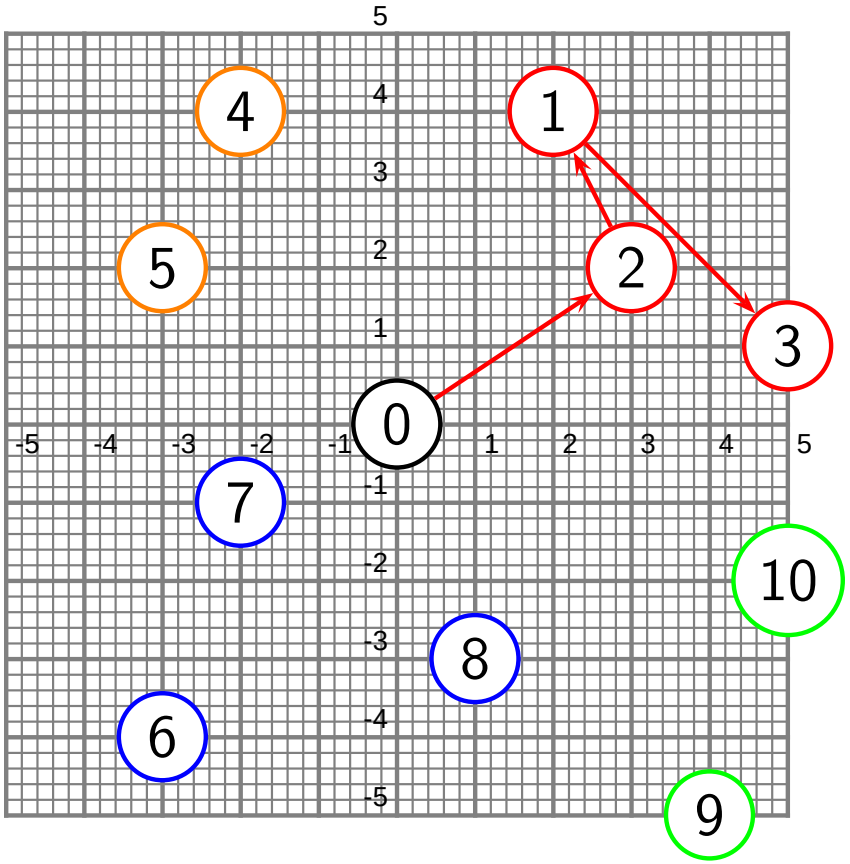


c_{ij}	0	1	2	3	4	5	6	7	8	9	10
0	0.0	4.5	3.6	5.1	4.5	3.6	5.0	2.2	3.2	6.4	5.4
1	4.5	0.0	2.2	4.2	4.0	5.4	9.4	6.4	7.1	9.2	6.7
2	3.6	2.2	0.0	2.2	5.4	6.0	8.5	5.8	5.4	7.1	4.5
3	5.1	4.2	2.2	0.0	7.6	8.1	9.4	7.3	5.7	6.1	3.0
4	4.5	4.0	5.4	7.6	0.0	2.2	8.1	5.0	7.6	10.8	9.2
5	3.6	5.4	6.0	8.1	2.2	0.0	6.0	3.2	6.4	9.9	8.9
6	5.0	9.4	8.5	9.4	8.1	6.0	0.0	3.2	4.1	7.1	8.2
7	2.2	6.4	5.8	7.3	5.0	3.2	3.2	0.0	3.6	7.2	7.1
8	3.2	7.1	5.4	5.7	7.6	6.4	4.1	3.6	0.0	3.6	4.1
9	6.4	9.2	7.1	6.1	10.8	9.9	7.1	7.2	3.6	0.0	3.2
10	5.4	6.7	4.5	3.0	9.2	8.9	8.2	7.1	4.1	3.2	0.0

0-2-1

Sweep algorithm - Example

Routing (Nearest neighbor)

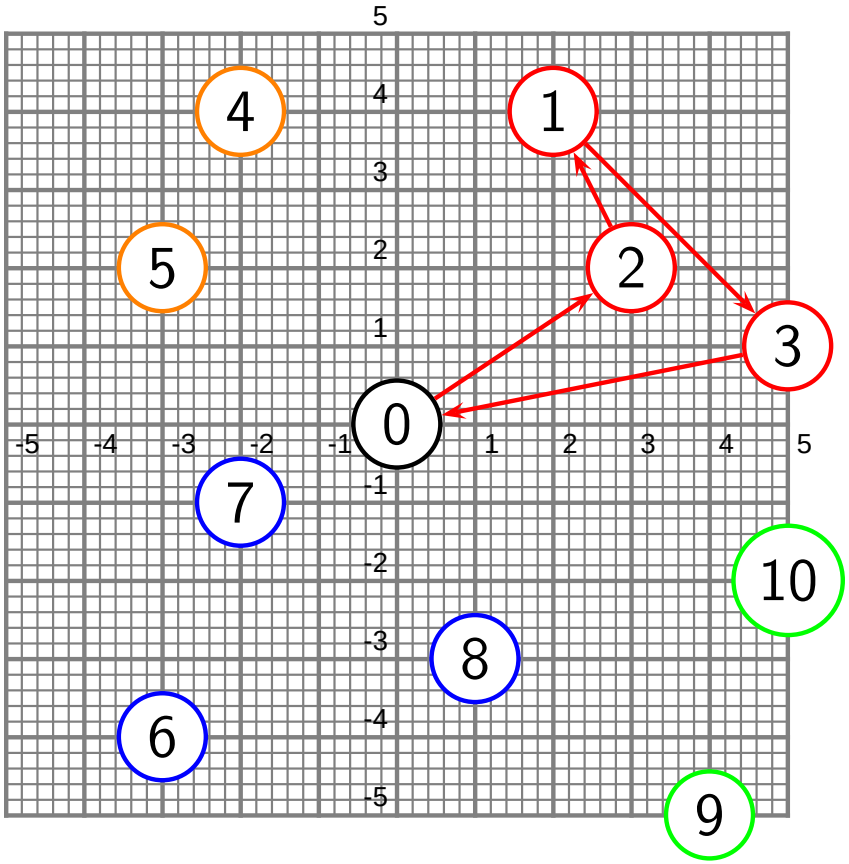


c_{ij}	0	1	2	3	4	5	6	7	8	9	10
0	0.0	4.5	3.6	5.1	4.5	3.6	5.0	2.2	3.2	6.4	5.4
1	4.5	0.0	2.2	4.2	4.0	5.4	9.4	6.4	7.1	9.2	6.7
2	3.6	2.2	0.0	2.2	5.4	6.0	8.5	5.8	5.4	7.1	4.5
3	5.1	4.2	2.2	0.0	7.6	8.1	9.4	7.3	5.7	6.1	3.0
4	4.5	4.0	5.4	7.6	0.0	2.2	8.1	5.0	7.6	10.8	9.2
5	3.6	5.4	6.0	8.1	2.2	0.0	6.0	3.2	6.4	9.9	8.9
6	5.0	9.4	8.5	9.4	8.1	6.0	0.0	3.2	4.1	7.1	8.2
7	2.2	6.4	5.8	7.3	5.0	3.2	3.2	0.0	3.6	7.2	7.1
8	3.2	7.1	5.4	5.7	7.6	6.4	4.1	3.6	0.0	3.6	4.1
9	6.4	9.2	7.1	6.1	10.8	9.9	7.1	7.2	3.6	0.0	3.2
10	5.4	6.7	4.5	3.0	9.2	8.9	8.2	7.1	4.1	3.2	0.0

0-2-1-3

Sweep algorithm - Example

Routing (Nearest neighbor)

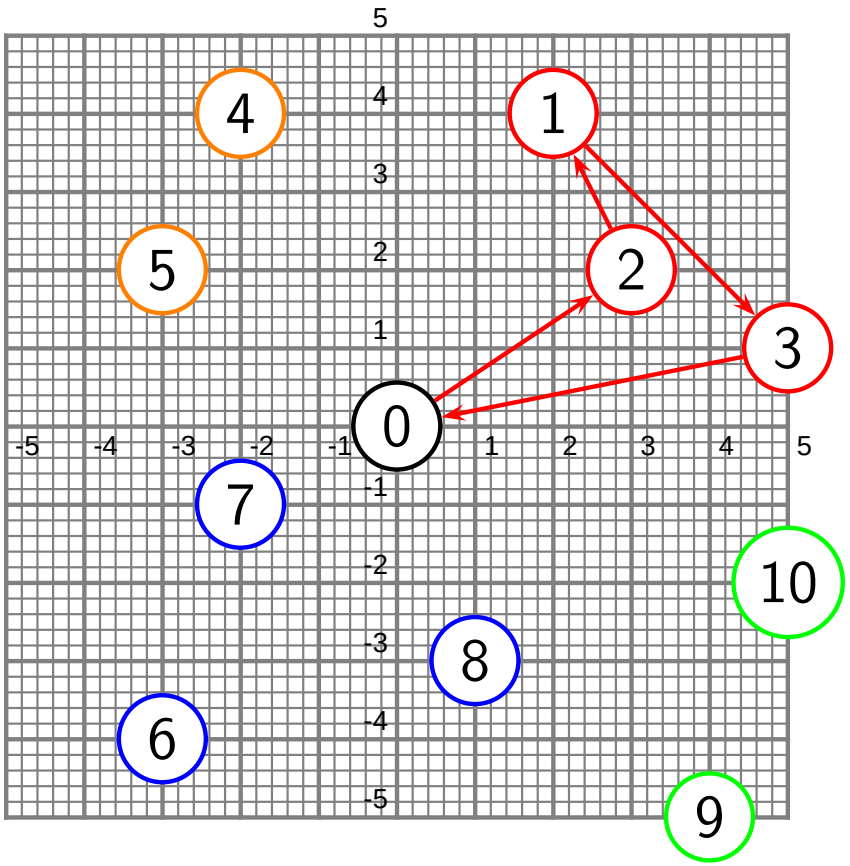


c_{ij}	0	1	2	3	4	5	6	7	8	9	10
0	0.0	4.5	3.6	5.1	4.5	3.6	5.0	2.2	3.2	6.4	5.4
1	4.5	0.0	2.2	4.2	4.0	5.4	9.4	6.4	7.1	9.2	6.7
2	3.6	2.2	0.0	2.2	5.4	6.0	8.5	5.8	5.4	7.1	4.5
3	5.1	4.2	2.2	0.0	7.6	8.1	9.4	7.3	5.7	6.1	3.0
4	4.5	4.0	5.4	7.6	0.0	2.2	8.1	5.0	7.6	10.8	9.2
5	3.6	5.4	6.0	8.1	2.2	0.0	6.0	3.2	6.4	9.9	8.9
6	5.0	9.4	8.5	9.4	8.1	6.0	0.0	3.2	4.1	7.1	8.2
7	2.2	6.4	5.8	7.3	5.0	3.2	3.2	0.0	3.6	7.2	7.1
8	3.2	7.1	5.4	5.7	7.6	6.4	4.1	3.6	0.0	3.6	4.1
9	6.4	9.2	7.1	6.1	10.8	9.9	7.1	7.2	3.6	0.0	3.2
10	5.4	6.7	4.5	3.0	9.2	8.9	8.2	7.1	4.1	3.2	0.0

0-2-1-3-0

Sweep algorithm - Example

Routing (Nearest neighbor)



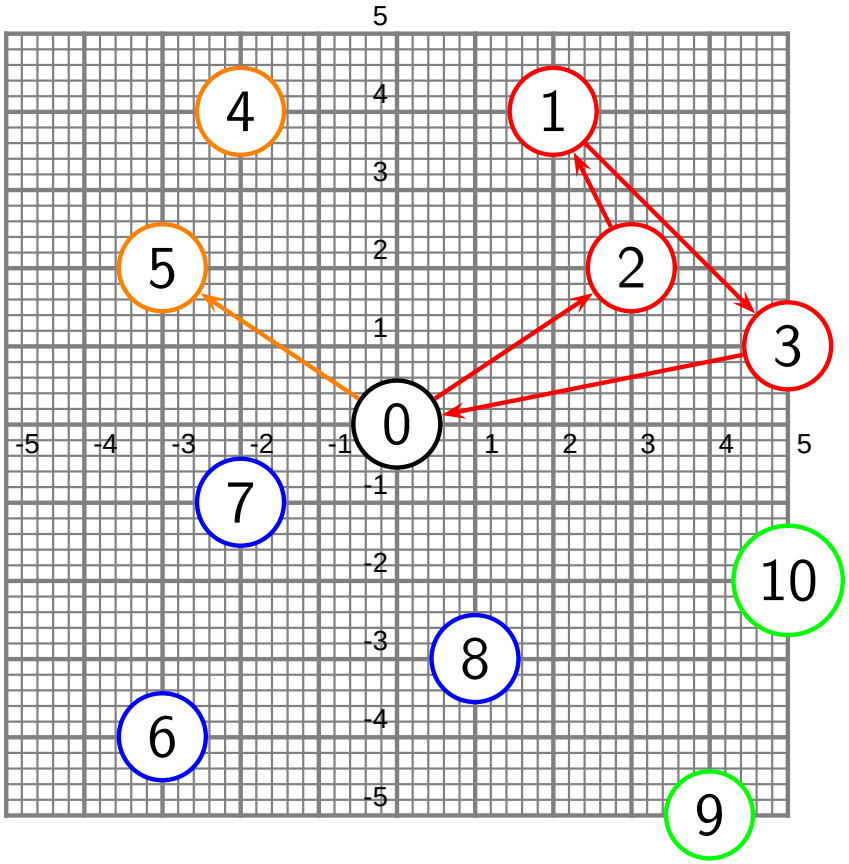
c_{ij}	0	1	2	3	4	5	6	7	8	9	10
0	0.0	4.5	3.6	5.1	4.5	3.6	5.0	2.2	3.2	6.4	5.4
1	4.5	0.0	2.2	4.2	4.0	5.4	9.4	6.4	7.1	9.2	6.7
2	3.6	2.2	0.0	2.2	5.4	6.0	8.5	5.8	5.4	7.1	4.5
3	5.1	4.2	2.2	0.0	7.6	8.1	9.4	7.3	5.7	6.1	3.0
4	4.5	4.0	5.4	7.6	0.0	2.2	8.1	5.0	7.6	10.8	9.2
5	3.6	5.4	6.0	8.1	2.2	0.0	6.0	3.2	6.4	9.9	8.9
6	5.0	9.4	8.5	9.4	8.1	6.0	0.0	3.2	4.1	7.1	8.2
7	2.2	6.4	5.8	7.3	5.0	3.2	3.2	0.0	3.6	7.2	7.1
8	3.2	7.1	5.4	5.7	7.6	6.4	4.1	3.6	0.0	3.6	4.1
9	6.4	9.2	7.1	6.1	10.8	9.9	7.1	7.2	3.6	0.0	3.2
10	5.4	6.7	4.5	3.0	9.2	8.9	8.2	7.1	4.1	3.2	0.0

0-2-1-3-0

0

Sweep algorithm - Example

Routing (Nearest neighbor)



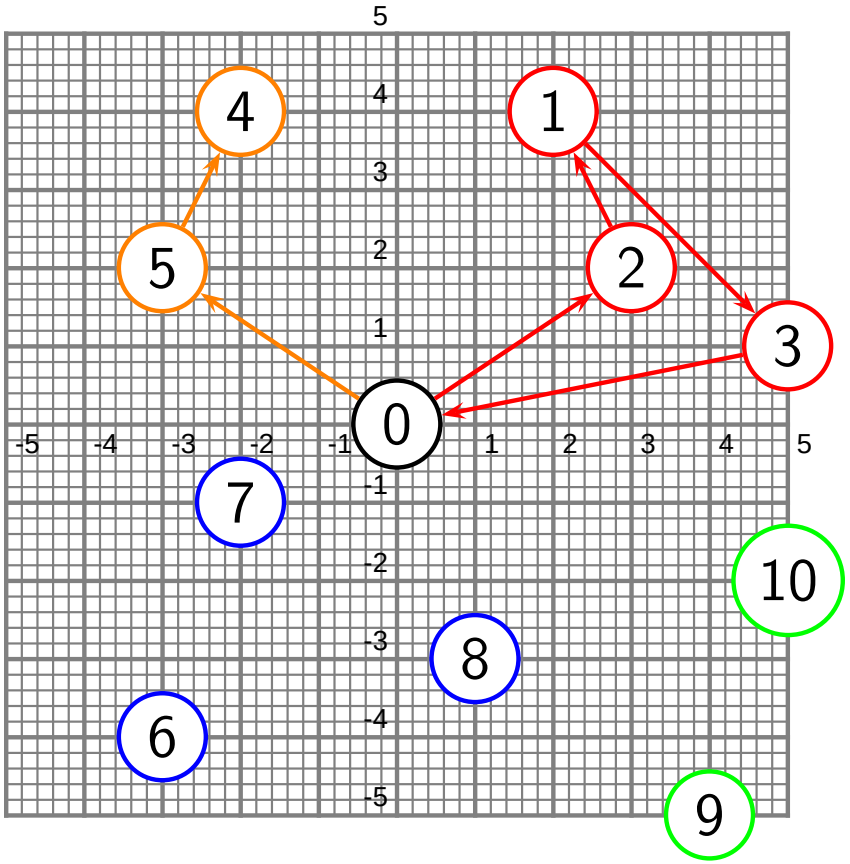
c_{ij}	0	1	2	3	4	5	6	7	8	9	10
0	0.0	4.5	3.6	5.1	4.5	3.6	5.0	2.2	3.2	6.4	5.4
1	4.5	0.0	2.2	4.2	4.0	5.4	9.4	6.4	7.1	9.2	6.7
2	3.6	2.2	0.0	2.2	5.4	6.0	8.5	5.8	5.4	7.1	4.5
3	5.1	4.2	2.2	0.0	7.6	8.1	9.4	7.3	5.7	6.1	3.0
4	4.5	4.0	5.4	7.6	0.0	2.2	8.1	5.0	7.6	10.8	9.2
5	3.6	5.4	6.0	8.1	2.2	0.0	6.0	3.2	6.4	9.9	8.9
6	5.0	9.4	8.5	9.4	8.1	6.0	0.0	3.2	4.1	7.1	8.2
7	2.2	6.4	5.8	7.3	5.0	3.2	3.2	0.0	3.6	7.2	7.1
8	3.2	7.1	5.4	5.7	7.6	6.4	4.1	3.6	0.0	3.6	4.1
9	6.4	9.2	7.1	6.1	10.8	9.9	7.1	7.2	3.6	0.0	3.2
10	5.4	6.7	4.5	3.0	9.2	8.9	8.2	7.1	4.1	3.2	0.0

0-2-1-3-0

0-5

Sweep algorithm - Example

Routing (Nearest neighbor)



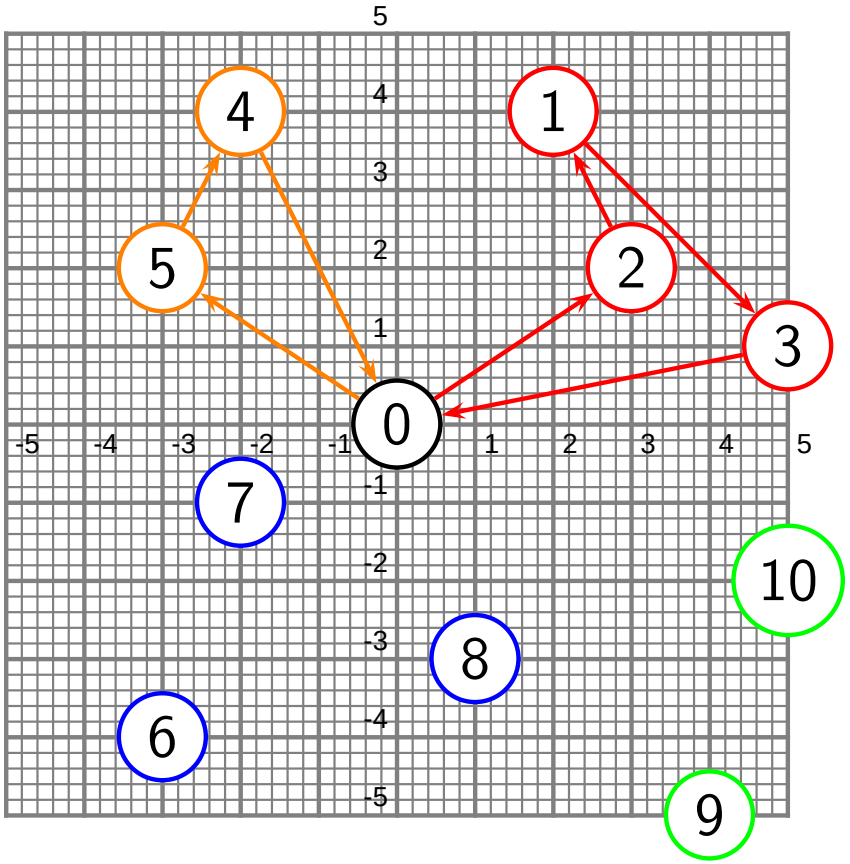
c_{ij}	0	1	2	3	4	5	6	7	8	9	10
0	0.0	4.5	3.6	5.1	4.5	3.6	5.0	2.2	3.2	6.4	5.4
1	4.5	0.0	2.2	4.2	4.0	5.4	9.4	6.4	7.1	9.2	6.7
2	3.6	2.2	0.0	2.2	5.4	6.0	8.5	5.8	5.4	7.1	4.5
3	5.1	4.2	2.2	0.0	7.6	8.1	9.4	7.3	5.7	6.1	3.0
4	4.5	4.0	5.4	7.6	0.0	2.2	8.1	5.0	7.6	10.8	9.2
5	3.6	5.4	6.0	8.1	2.2	0.0	6.0	3.2	6.4	9.9	8.9
6	5.0	9.4	8.5	9.4	8.1	6.0	0.0	3.2	4.1	7.1	8.2
7	2.2	6.4	5.8	7.3	5.0	3.2	3.2	0.0	3.6	7.2	7.1
8	3.2	7.1	5.4	5.7	7.6	6.4	4.1	3.6	0.0	3.6	4.1
9	6.4	9.2	7.1	6.1	10.8	9.9	7.1	7.2	3.6	0.0	3.2
10	5.4	6.7	4.5	3.0	9.2	8.9	8.2	7.1	4.1	3.2	0.0

0-2-1-3-0

0-5-4

Sweep algorithm - Example

Routing (Nearest neighbor)



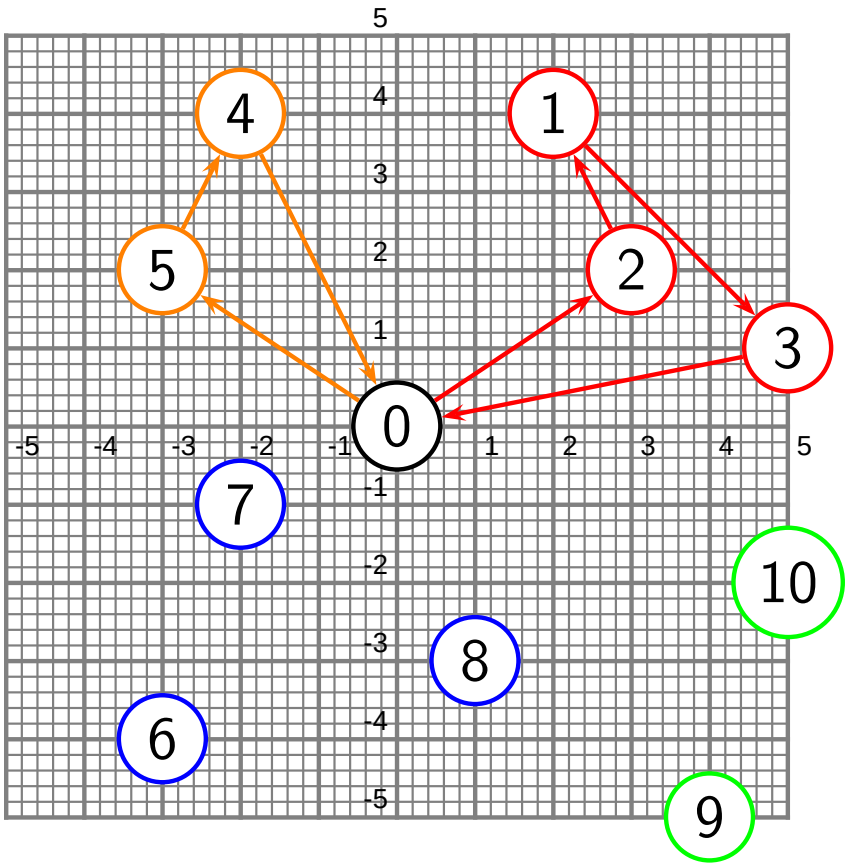
c_{ij}	0	1	2	3	4	5	6	7	8	9	10
0	0.0	4.5	3.6	5.1	4.5	3.6	5.0	2.2	3.2	6.4	5.4
1	4.5	0.0	2.2	4.2	4.0	5.4	9.4	6.4	7.1	9.2	6.7
2	3.6	2.2	0.0	2.2	5.4	6.0	8.5	5.8	5.4	7.1	4.5
3	5.1	4.2	2.2	0.0	7.6	8.1	9.4	7.3	5.7	6.1	3.0
4	4.5	4.0	5.4	7.6	0.0	2.2	8.1	5.0	7.6	10.8	9.2
5	3.6	5.4	6.0	8.1	2.2	0.0	6.0	3.2	6.4	9.9	8.9
6	5.0	9.4	8.5	9.4	8.1	6.0	0.0	3.2	4.1	7.1	8.2
7	2.2	6.4	5.8	7.3	5.0	3.2	3.2	0.0	3.6	7.2	7.1
8	3.2	7.1	5.4	5.7	7.6	6.4	4.1	3.6	0.0	3.6	4.1
9	6.4	9.2	7.1	6.1	10.8	9.9	7.1	7.2	3.6	0.0	3.2
10	5.4	6.7	4.5	3.0	9.2	8.9	8.2	7.1	4.1	3.2	0.0

0-2-1-3-0

0-5-4-0

Sweep algorithm - Example

Routing (Nearest neighbor)



c_{ij}	0	1	2	3	4	5	6	7	8	9	10
0	0.0	4.5	3.6	5.1	4.5	3.6	5.0	2.2	3.2	6.4	5.4
1	4.5	0.0	2.2	4.2	4.0	5.4	9.4	6.4	7.1	9.2	6.7
2	3.6	2.2	0.0	2.2	5.4	6.0	8.5	5.8	5.4	7.1	4.5
3	5.1	4.2	2.2	0.0	7.6	8.1	9.4	7.3	5.7	6.1	3.0
4	4.5	4.0	5.4	7.6	0.0	2.2	8.1	5.0	7.6	10.8	9.2
5	3.6	5.4	6.0	8.1	2.2	0.0	6.0	3.2	6.4	9.9	8.9
6	5.0	9.4	8.5	9.4	8.1	6.0	0.0	3.2	4.1	7.1	8.2
7	2.2	6.4	5.8	7.3	5.0	3.2	3.2	0.0	3.6	7.2	7.1
8	3.2	7.1	5.4	5.7	7.6	6.4	4.1	3.6	0.0	3.6	4.1
9	6.4	9.2	7.1	6.1	10.8	9.9	7.1	7.2	3.6	0.0	3.2
10	5.4	6.7	4.5	3.0	9.2	8.9	8.2	7.1	4.1	3.2	0.0

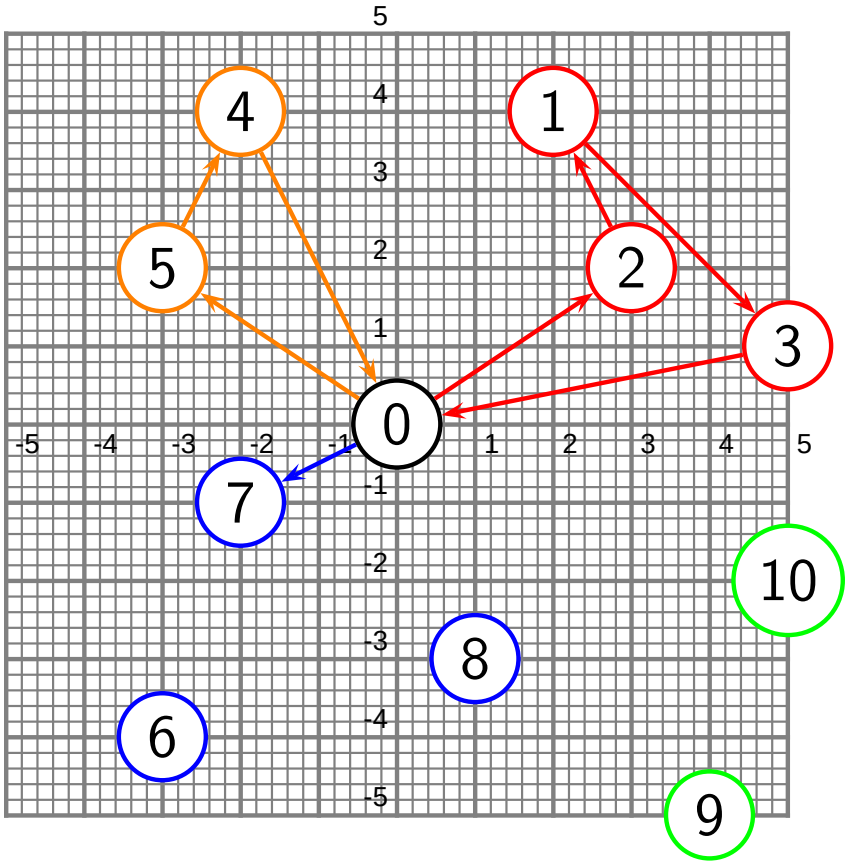
0-2-1-3-0

0-5-4-0

0

Sweep algorithm - Example

Routing (Nearest neighbor)



c_{ij}	0	1	2	3	4	5	6	7	8	9	10
0	0.0	4.5	3.6	5.1	4.5	3.6	5.0	2.2	3.2	6.4	5.4
1	4.5	0.0	2.2	4.2	4.0	5.4	9.4	6.4	7.1	9.2	6.7
2	3.6	2.2	0.0	2.2	5.4	6.0	8.5	5.8	5.4	7.1	4.5
3	5.1	4.2	2.2	0.0	7.6	8.1	9.4	7.3	5.7	6.1	3.0
4	4.5	4.0	5.4	7.6	0.0	2.2	8.1	5.0	7.6	10.8	9.2
5	3.6	5.4	6.0	8.1	2.2	0.0	6.0	3.2	6.4	9.9	8.9
6	5.0	9.4	8.5	9.4	8.1	6.0	0.0	3.2	4.1	7.1	8.2
7	2.2	6.4	5.8	7.3	5.0	3.2	3.2	0.0	3.6	7.2	7.1
8	3.2	7.1	5.4	5.7	7.6	6.4	4.1	3.6	0.0	3.6	4.1
9	6.4	9.2	7.1	6.1	10.8	9.9	7.1	7.2	3.6	0.0	3.2
10	5.4	6.7	4.5	3.0	9.2	8.9	8.2	7.1	4.1	3.2	0.0

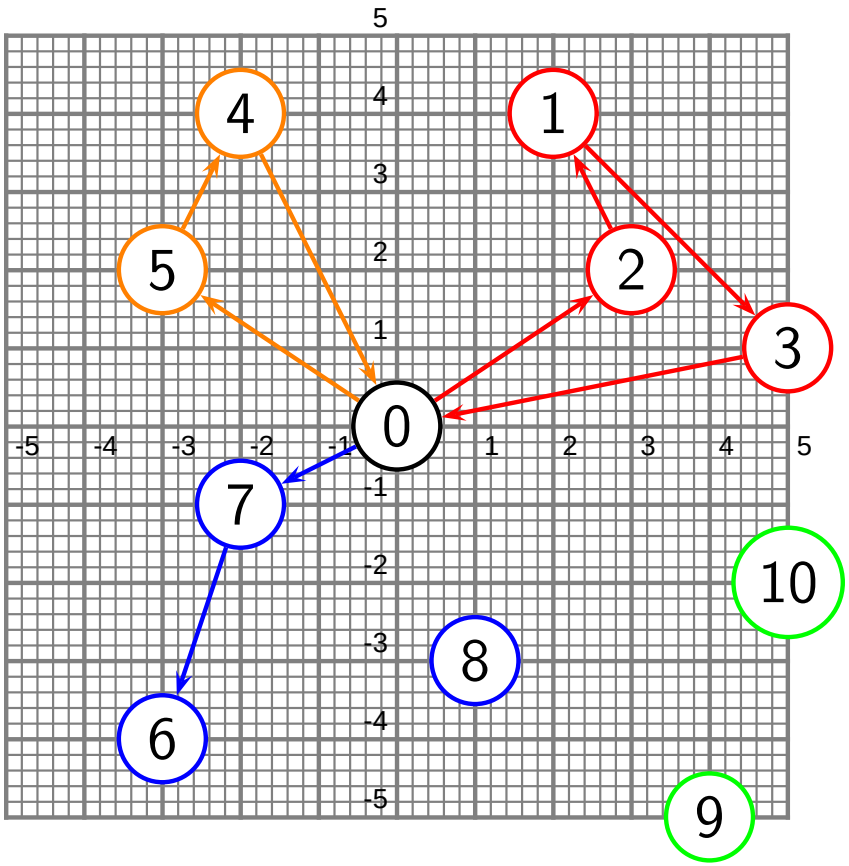
0-2-1-3-0

0-5-4-0

0-7

Sweep algorithm - Example

Routing (Nearest neighbor)



c_{ij}	0	1	2	3	4	5	6	7	8	9	10
0	0.0	4.5	3.6	5.1	4.5	3.6	5.0	2.2	3.2	6.4	5.4
1	4.5	0.0	2.2	4.2	4.0	5.4	9.4	6.4	7.1	9.2	6.7
2	3.6	2.2	0.0	2.2	5.4	6.0	8.5	5.8	5.4	7.1	4.5
3	5.1	4.2	2.2	0.0	7.6	8.1	9.4	7.3	5.7	6.1	3.0
4	4.5	4.0	5.4	7.6	0.0	2.2	8.1	5.0	7.6	10.8	9.2
5	3.6	5.4	6.0	8.1	2.2	0.0	6.0	3.2	6.4	9.9	8.9
6	5.0	9.4	8.5	9.4	8.1	6.0	0.0	3.2	4.1	7.1	8.2
7	2.2	6.4	5.8	7.3	5.0	3.2	3.2	0.0	3.6	7.2	7.1
8	3.2	7.1	5.4	5.7	7.6	6.4	4.1	3.6	0.0	3.6	4.1
9	6.4	9.2	7.1	6.1	10.8	9.9	7.1	7.2	3.6	0.0	3.2
10	5.4	6.7	4.5	3.0	9.2	8.9	8.2	7.1	4.1	3.2	0.0

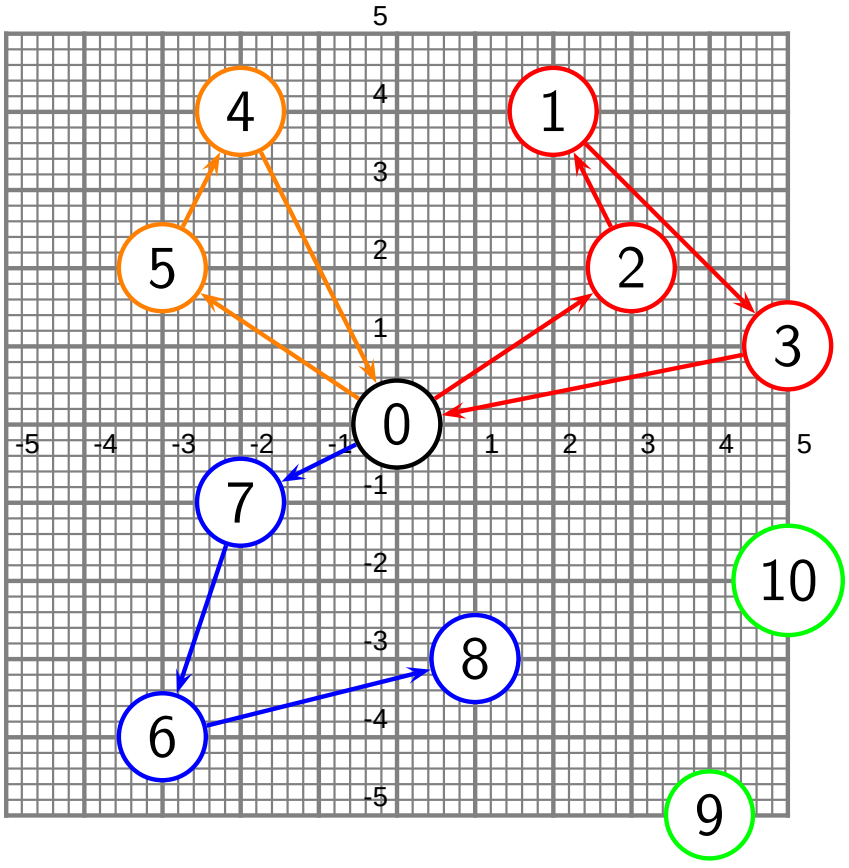
0-2-1-3-0

0-5-4-0

0-7-6

Sweep algorithm - Example

Routing (Nearest neighbor)

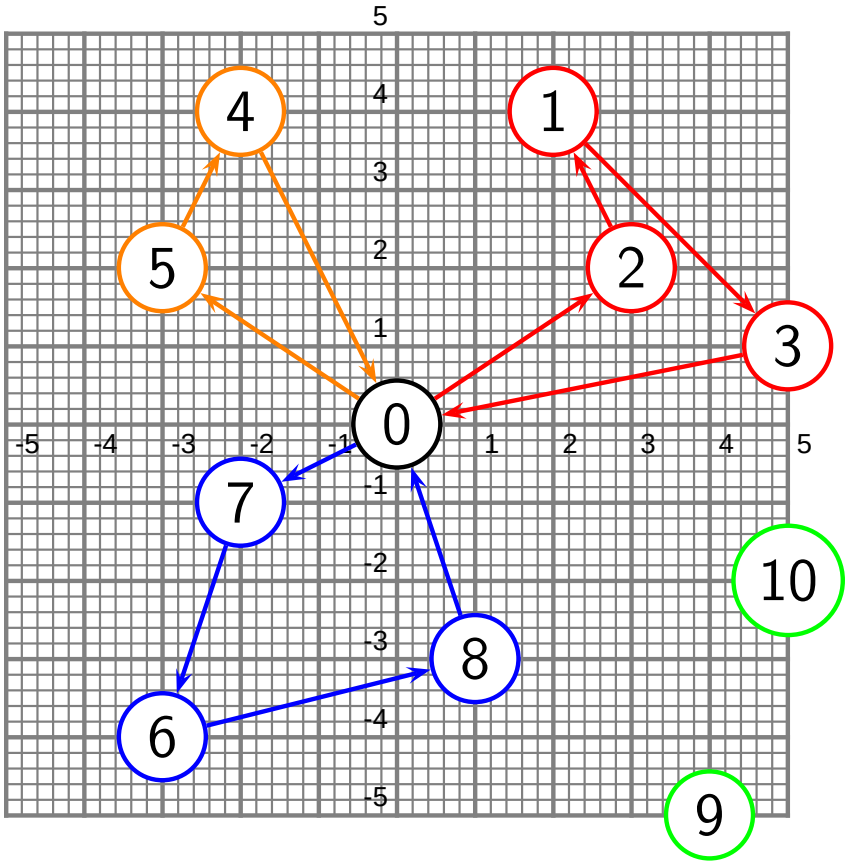


c_{ij}	0	1	2	3	4	5	6	7	8	9	10
0	0.0	4.5	3.6	5.1	4.5	3.6	5.0	2.2	3.2	6.4	5.4
1	4.5	0.0	2.2	4.2	4.0	5.4	9.4	6.4	7.1	9.2	6.7
2	3.6	2.2	0.0	2.2	5.4	6.0	8.5	5.8	5.4	7.1	4.5
3	5.1	4.2	2.2	0.0	7.6	8.1	9.4	7.3	5.7	6.1	3.0
4	4.5	4.0	5.4	7.6	0.0	2.2	8.1	5.0	7.6	10.8	9.2
5	3.6	5.4	6.0	8.1	2.2	0.0	6.0	3.2	6.4	9.9	8.9
6	5.0	9.4	8.5	9.4	8.1	6.0	0.0	3.2	4.1	7.1	8.2
7	2.2	6.4	5.8	7.3	5.0	3.2	3.2	0.0	3.6	7.2	7.1
8	3.2	7.1	5.4	5.7	7.6	6.4	4.1	3.6	0.0	3.6	4.1
9	6.4	9.2	7.1	6.1	10.8	9.9	7.1	7.2	3.6	0.0	3.2
10	5.4	6.7	4.5	3.0	9.2	8.9	8.2	7.1	4.1	3.2	0.0

0-2-1-3-0
0-5-4-0
0-7-6-8

Sweep algorithm - Example

Routing (Nearest neighbor)



c_{ij}	0	1	2	3	4	5	6	7	8	9	10
0	0.0	4.5	3.6	5.1	4.5	3.6	5.0	2.2	3.2	6.4	5.4
1	4.5	0.0	2.2	4.2	4.0	5.4	9.4	6.4	7.1	9.2	6.7
2	3.6	2.2	0.0	2.2	5.4	6.0	8.5	5.8	5.4	7.1	4.5
3	5.1	4.2	2.2	0.0	7.6	8.1	9.4	7.3	5.7	6.1	3.0
4	4.5	4.0	5.4	7.6	0.0	2.2	8.1	5.0	7.6	10.8	9.2
5	3.6	5.4	6.0	8.1	2.2	0.0	6.0	3.2	6.4	9.9	8.9
6	5.0	9.4	8.5	9.4	8.1	6.0	0.0	3.2	4.1	7.1	8.2
7	2.2	6.4	5.8	7.3	5.0	3.2	3.2	0.0	3.6	7.2	7.1
8	3.2	7.1	5.4	5.7	7.6	6.4	4.1	3.6	0.0	3.6	4.1
9	6.4	9.2	7.1	6.1	10.8	9.9	7.1	7.2	3.6	0.0	3.2
10	5.4	6.7	4.5	3.0	9.2	8.9	8.2	7.1	4.1	3.2	0.0

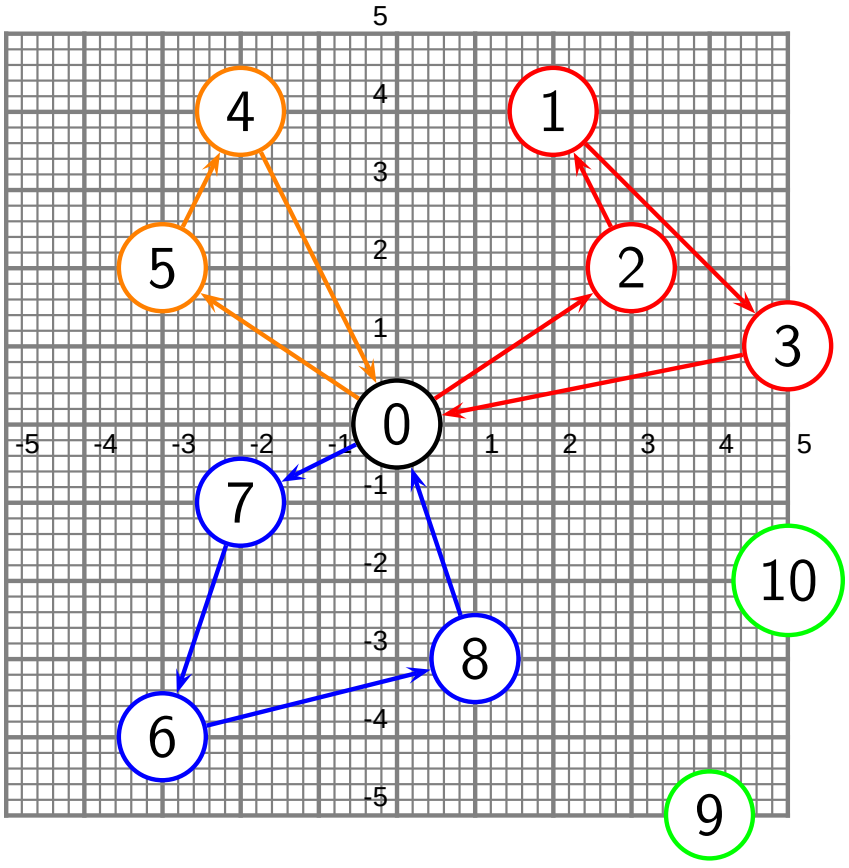
0-2-1-3-0

0-5-4-0

0-7-6-8-0

Sweep algorithm - Example

Routing (Nearest neighbor)

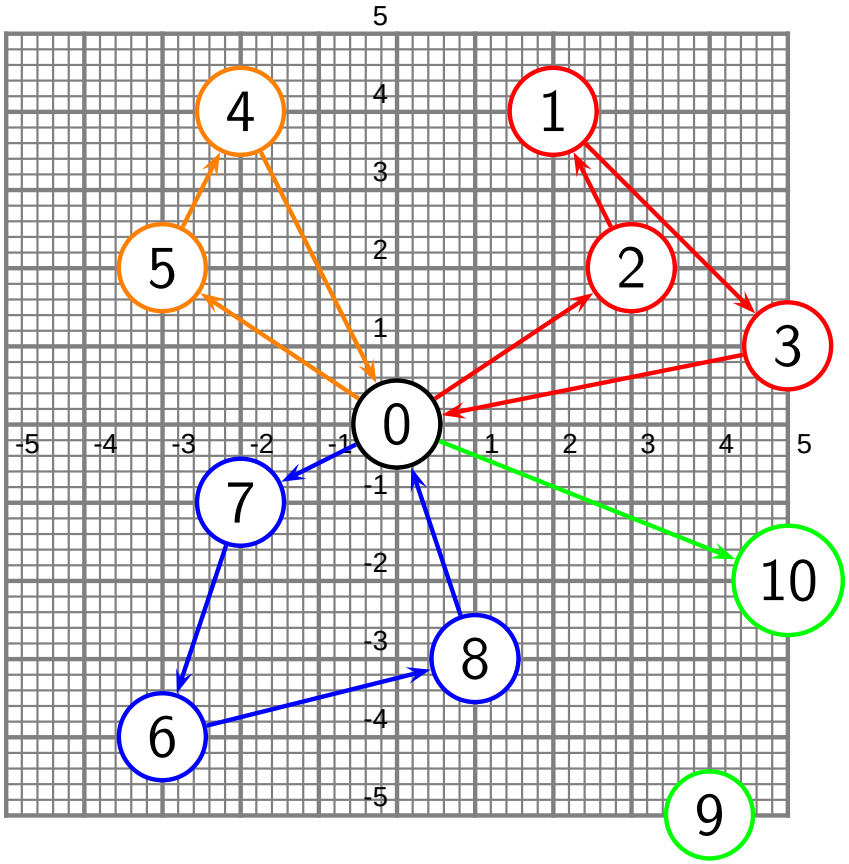


c_{ij}	0	1	2	3	4	5	6	7	8	9	10
0	0.0	4.5	3.6	5.1	4.5	3.6	5.0	2.2	3.2	6.4	5.4
1	4.5	0.0	2.2	4.2	4.0	5.4	9.4	6.4	7.1	9.2	6.7
2	3.6	2.2	0.0	2.2	5.4	6.0	8.5	5.8	5.4	7.1	4.5
3	5.1	4.2	2.2	0.0	7.6	8.1	9.4	7.3	5.7	6.1	3.0
4	4.5	4.0	5.4	7.6	0.0	2.2	8.1	5.0	7.6	10.8	9.2
5	3.6	5.4	6.0	8.1	2.2	0.0	6.0	3.2	6.4	9.9	8.9
6	5.0	9.4	8.5	9.4	8.1	6.0	0.0	3.2	4.1	7.1	8.2
7	2.2	6.4	5.8	7.3	5.0	3.2	3.2	0.0	3.6	7.2	7.1
8	3.2	7.1	5.4	5.7	7.6	6.4	4.1	3.6	0.0	3.6	4.1
9	6.4	9.2	7.1	6.1	10.8	9.9	7.1	7.2	3.6	0.0	3.2
10	5.4	6.7	4.5	3.0	9.2	8.9	8.2	7.1	4.1	3.2	0.0

0-2-1-3-0
0-5-4-0
0-7-6-8-0
0

Sweep algorithm - Example

Routing (Nearest neighbor)

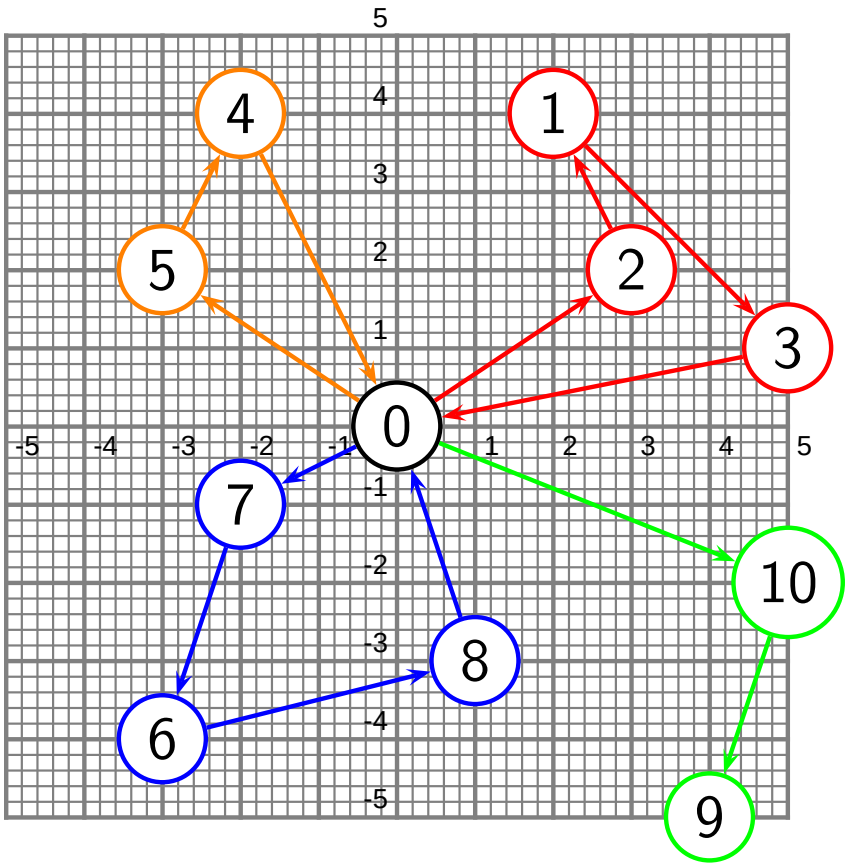


c_{ij}	0	1	2	3	4	5	6	7	8	9	10
0	0.0	4.5	3.6	5.1	4.5	3.6	5.0	2.2	3.2	6.4	5.4
1	4.5	0.0	2.2	4.2	4.0	5.4	9.4	6.4	7.1	9.2	6.7
2	3.6	2.2	0.0	2.2	5.4	6.0	8.5	5.8	5.4	7.1	4.5
3	5.1	4.2	2.2	0.0	7.6	8.1	9.4	7.3	5.7	6.1	3.0
4	4.5	4.0	5.4	7.6	0.0	2.2	8.1	5.0	7.6	10.8	9.2
5	3.6	5.4	6.0	8.1	2.2	0.0	6.0	3.2	6.4	9.9	8.9
6	5.0	9.4	8.5	9.4	8.1	6.0	0.0	3.2	4.1	7.1	8.2
7	2.2	6.4	5.8	7.3	5.0	3.2	3.2	0.0	3.6	7.2	7.1
8	3.2	7.1	5.4	5.7	7.6	6.4	4.1	3.6	0.0	3.6	4.1
9	6.4	9.2	7.1	6.1	10.8	9.9	7.1	7.2	3.6	0.0	3.2
10	5.4	6.7	4.5	3.0	9.2	8.9	8.2	7.1	4.1	3.2	0.0

0-2-1-3-0
0-5-4-0
0-7-6-8-0
0-10

Sweep algorithm - Example

Routing (Nearest neighbor)

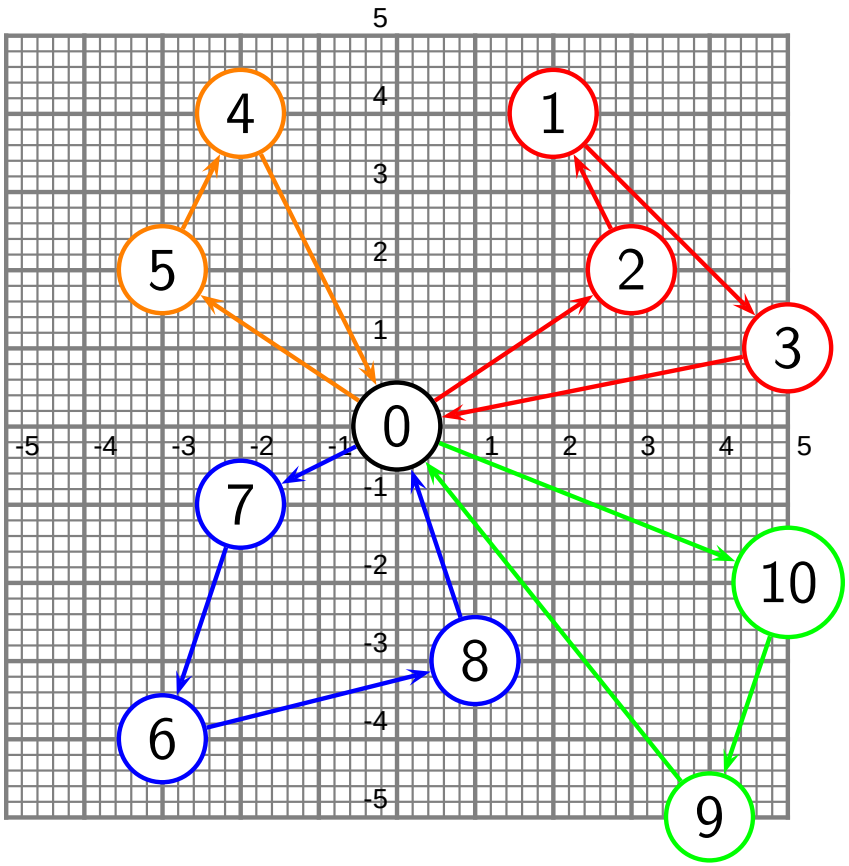


c_{ij}	0	1	2	3	4	5	6	7	8	9	10
0	0.0	4.5	3.6	5.1	4.5	3.6	5.0	2.2	3.2	6.4	5.4
1	4.5	0.0	2.2	4.2	4.0	5.4	9.4	6.4	7.1	9.2	6.7
2	3.6	2.2	0.0	2.2	5.4	6.0	8.5	5.8	5.4	7.1	4.5
3	5.1	4.2	2.2	0.0	7.6	8.1	9.4	7.3	5.7	6.1	3.0
4	4.5	4.0	5.4	7.6	0.0	2.2	8.1	5.0	7.6	10.8	9.2
5	3.6	5.4	6.0	8.1	2.2	0.0	6.0	3.2	6.4	9.9	8.9
6	5.0	9.4	8.5	9.4	8.1	6.0	0.0	3.2	4.1	7.1	8.2
7	2.2	6.4	5.8	7.3	5.0	3.2	3.2	0.0	3.6	7.2	7.1
8	3.2	7.1	5.4	5.7	7.6	6.4	4.1	3.6	0.0	3.6	4.1
9	6.4	9.2	7.1	6.1	10.8	9.9	7.1	7.2	3.6	0.0	3.2
10	5.4	6.7	4.5	3.0	9.2	8.9	8.2	7.1	4.1	3.2	0.0

0-2-1-3-0
0-5-4-0
0-7-6-8-0
0-10-9

Sweep algorithm - Example

Routing (Nearest neighbor)



c_{ij}	0	1	2	3	4	5	6	7	8	9	10
0	0.0	4.5	3.6	5.1	4.5	3.6	5.0	2.2	3.2	6.4	5.4
1	4.5	0.0	2.2	4.2	4.0	5.4	9.4	6.4	7.1	9.2	6.7
2	3.6	2.2	0.0	2.2	5.4	6.0	8.5	5.8	5.4	7.1	4.5
3	5.1	4.2	2.2	0.0	7.6	8.1	9.4	7.3	5.7	6.1	3.0
4	4.5	4.0	5.4	7.6	0.0	2.2	8.1	5.0	7.6	10.8	9.2
5	3.6	5.4	6.0	8.1	2.2	0.0	6.0	3.2	6.4	9.9	8.9
6	5.0	9.4	8.5	9.4	8.1	6.0	0.0	3.2	4.1	7.1	8.2
7	2.2	6.4	5.8	7.3	5.0	3.2	3.2	0.0	3.6	7.2	7.1
8	3.2	7.1	5.4	5.7	7.6	6.4	4.1	3.6	0.0	3.6	4.1
9	6.4	9.2	7.1	6.1	10.8	9.9	7.1	7.2	3.6	0.0	3.2
10	5.4	6.7	4.5	3.0	9.2	8.9	8.2	7.1	4.1	3.2	0.0

0-2-1-3-0
0-5-4-0
0-7-6-8-0
0-10-9-0

Fisher and Jaikumar Algorithm

★ Not directly applicable to VRP with distance constraint.
★ Assumption: size of the vehicle fleet is given.

- **Clustering**
  - **Step 1** Seed selection: choose seed vertices $j_k \in V$ to initialize each cluster k.
  - **Step 2** Allocation of customers to seeds: Compute the cost a_{ik} of allocating customer i to cluster k:
    $a_{ik} = \min\{c_{0i} + c_{ij_k} + c_{j_k 0}, c_{0j_k} + c_{j_k i} + c_{i0}\} - (c_{0j_k} + c_{j_k 0})$.
  - **Step 3** Generalized assignment: Solve a GAP with a_{ik}, weights d_i (demand) and vehicle capacity C to obtain the clusters.
- **Routing** Solve a TSP for each cluster.

Fisher and Jaikumar Algorithm

The generalized assignment problem (GAP)

$$x_{ik} = \begin{cases} 1, \text{ if } i \text{ assigned to cluster } k, \\ 0, \text{ otherwise.} \end{cases} \tag{21}$$

$$\min \sum_i \sum_k a_{ik} x_{ik} \tag{22}$$

$$\sum_k x_{ik} = 1 \qquad \forall i \tag{23}$$

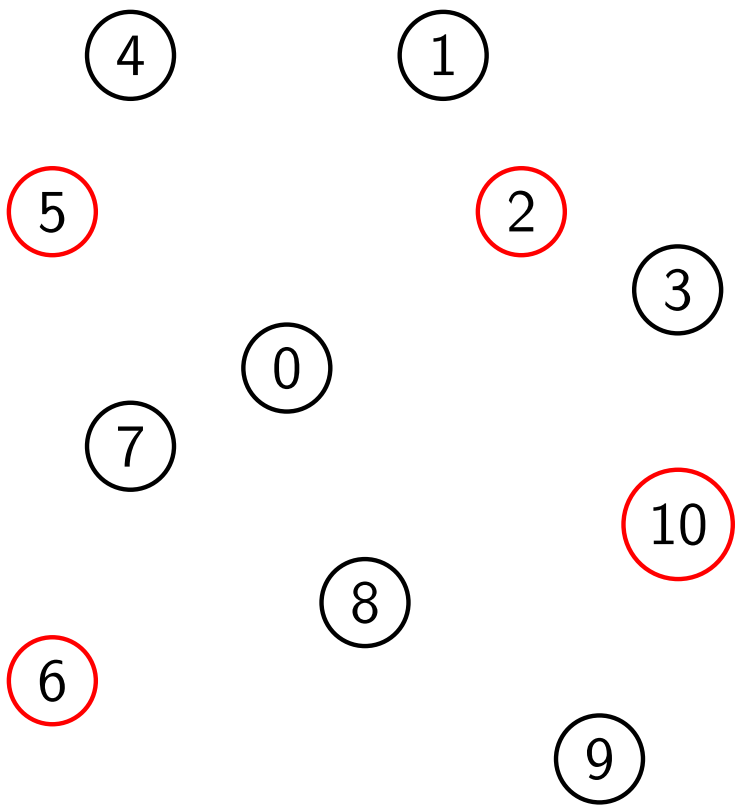
$$\sum_i d_i x_{ik} \leq C \qquad \forall k \tag{24}$$

$$x_{ik} \in \{0, 1\} \tag{25}$$

★ The GAP is more difficult than the AP.

Fisher and Jaikumar Algorithm - Example

Clustering

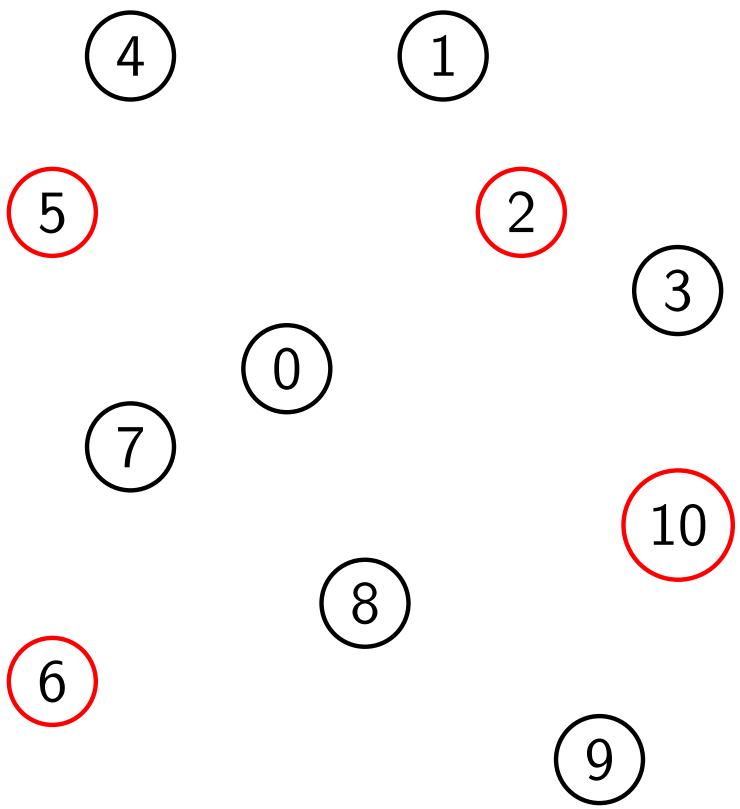


4 vehicles

seed vertices: 2,5,6,10

Fisher and Jaikumar Algorithm - Example

Clustering



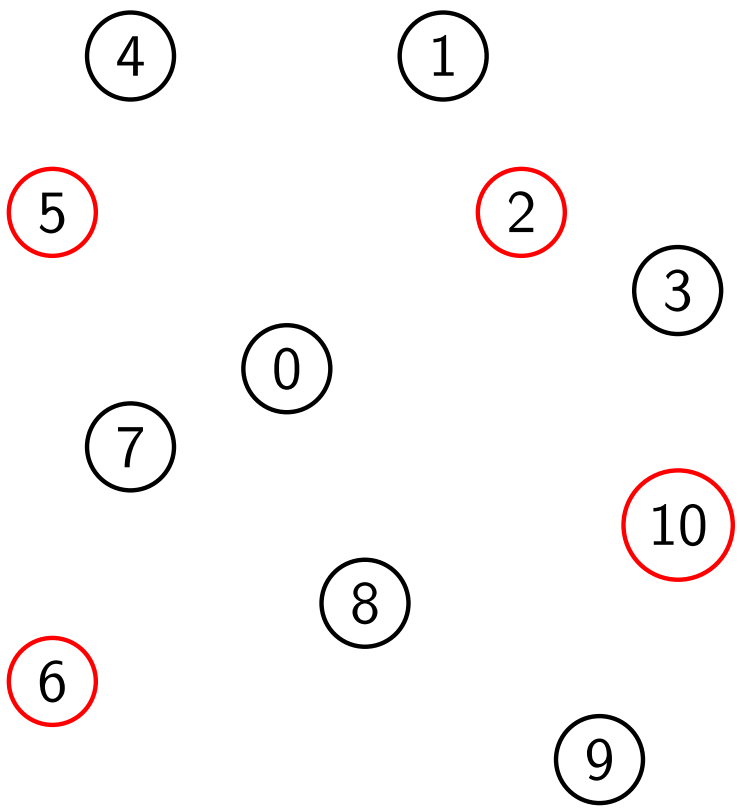
4 vehicles

seed vertices: 2,5,6,10

compute a_{ik} values

Fisher and Jaikumar Algorithm - Example

Clustering



4 vehicles

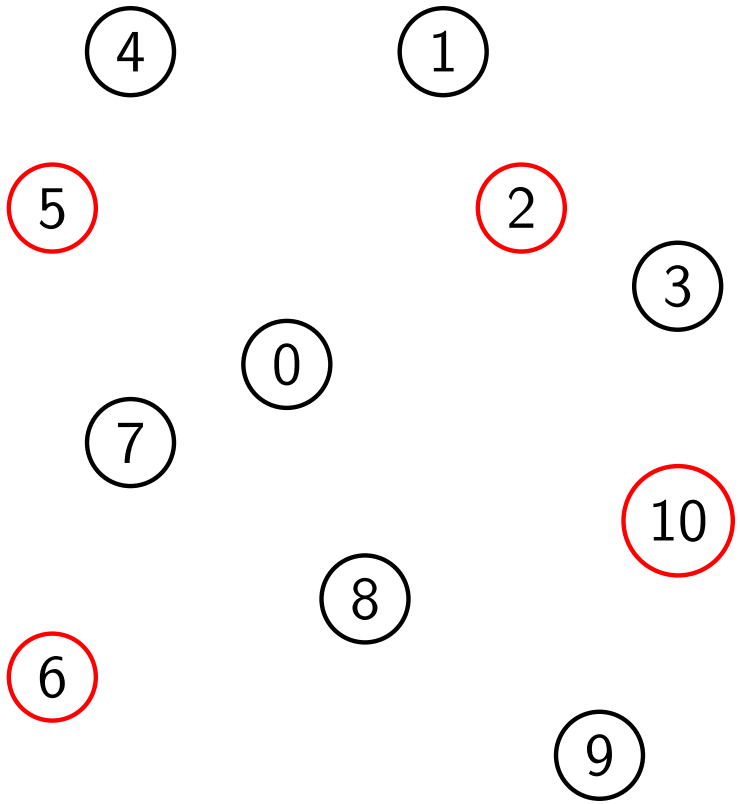
seed vertices: 2,5,6,10

compute a_{ik} values

a_{ik}	2	5	6	10
1	3.1	6.3	8.9	5.8
2	0.0	6.0	7.1	2.7
3	3.7	9.6	9.5	2.7
4	6.3	3.1	7.5	8.3
5	6.0	0.0	4.6	7.2
6	9.9	7.4	0.0	7.9
7	4.5	1.8	0.4	3.9
8	4.9	6.0	2.3	1.9
9	9.9	12.7	8.5	4.2
10	6.3	10.7	8.6	0.0

Fisher and Jaikumar Algorithm - Example

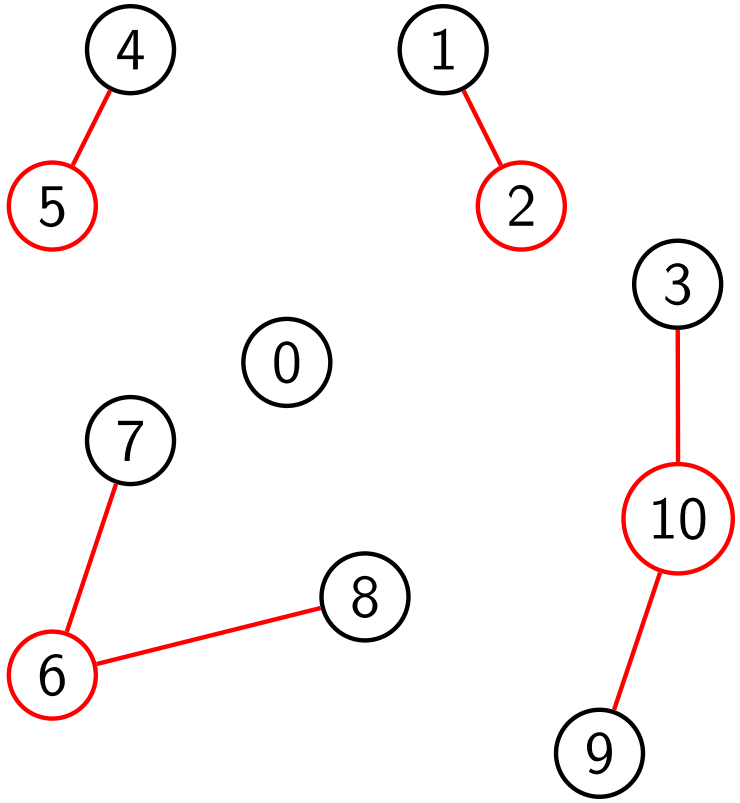
Clustering



solve the GAP
(e.g. with Excel solver)

Fisher and Jaikumar Algorithm - Example

Clustering



solve the GAP

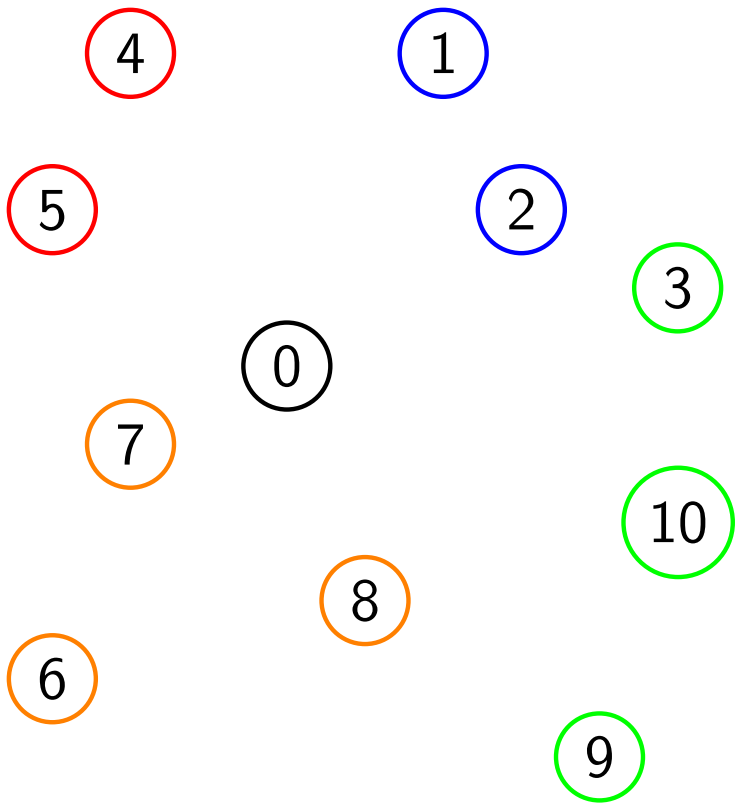
(e.g. with Excel solver)

x_{ik}	2	5	6	10
1	1	0	0	0
2	1	0	0	0
3	0	0	0	1
4	0	1	0	0
5	0	1	0	0
6	0	0	1	0
7	0	0	1	0
8	0	0	1	0
9	0	0	0	1
10	0	0	0	1

★ The GAP can also be solved heuristically (e.g. greedy).

Fisher and Jaikumar Algorithm - Example

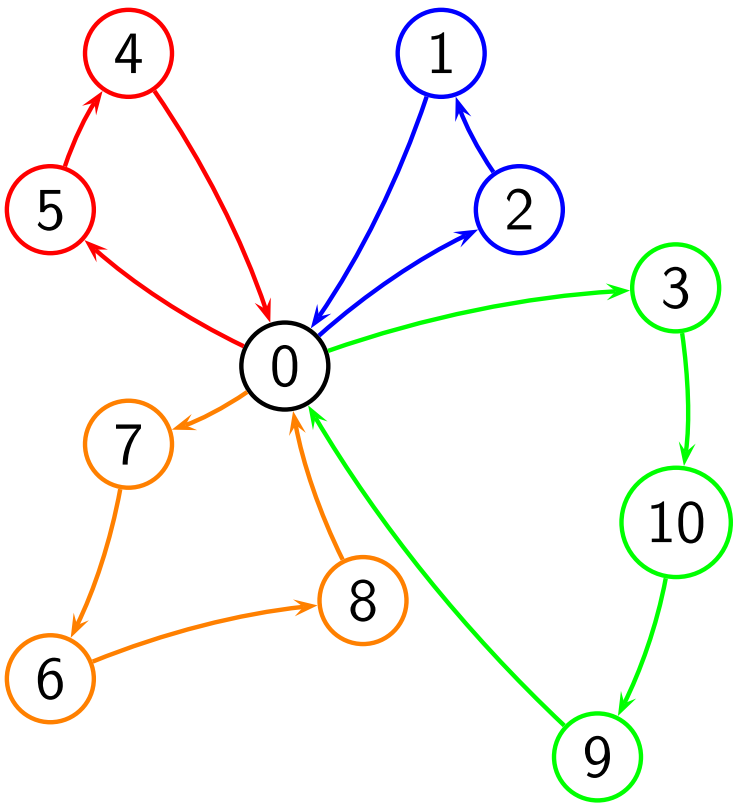
Routing (e.g. nearest neighbor)



c_{ij}	0	1	2	3	4	5	6	7	8	9	10
0	0.0	4.5	3.6	5.1	4.5	3.6	5.0	2.2	3.2	6.4	5.4
1	4.5	0.0	2.2	4.2	4.0	5.4	9.4	6.4	7.1	9.2	6.7
2	3.6	2.2	0.0	2.2	5.4	6.0	8.5	5.8	5.4	7.1	4.5
3	5.1	4.2	2.2	0.0	7.6	8.1	9.4	7.3	5.7	6.1	3.0
4	4.5	4.0	5.4	7.6	0.0	2.2	8.1	5.0	7.6	10.8	9.2
5	3.6	5.4	6.0	8.1	2.2	0.0	6.0	3.2	6.4	9.9	8.9
6	5.0	9.4	8.5	9.4	8.1	6.0	0.0	3.2	4.1	7.1	8.2
7	2.2	6.4	5.8	7.3	5.0	3.2	3.2	0.0	3.6	7.2	7.1
8	3.2	7.1	5.4	5.7	7.6	6.4	4.1	3.6	0.0	3.6	4.1
9	6.4	9.2	7.1	6.1	10.8	9.9	7.1	7.2	3.6	0.0	3.2
10	5.4	6.7	4.5	3.0	9.2	8.9	8.2	7.1	4.1	3.2	0.0

Fisher and Jaikumar Algorithm - Example

Routing (e.g. nearest neighbor)



c_{ij}	0	1	2	3	4	5	6	7	8	9	10
0	0.0	4.5	3.6	5.1	4.5	3.6	5.0	2.2	3.2	6.4	5.4
1	4.5	0.0	2.2	4.2	4.0	5.4	9.4	6.4	7.1	9.2	6.7
2	3.6	2.2	0.0	2.2	5.4	6.0	8.5	5.8	5.4	7.1	4.5
3	5.1	4.2	2.2	0.0	7.6	8.1	9.4	7.3	5.7	6.1	3.0
4	4.5	4.0	5.4	7.6	0.0	2.2	8.1	5.0	7.6	10.8	9.2
5	3.6	5.4	6.0	8.1	2.2	0.0	6.0	3.2	6.4	9.9	8.9
6	5.0	9.4	8.5	9.4	8.1	6.0	0.0	3.2	4.1	7.1	8.2
7	2.2	6.4	5.8	7.3	5.0	3.2	3.2	0.0	3.6	7.2	7.1
8	3.2	7.1	5.4	5.7	7.6	6.4	4.1	3.6	0.0	3.6	4.1
9	6.4	9.2	7.1	6.1	10.8	9.9	7.1	7.2	3.6	0.0	3.2
10	5.4	6.7	4.5	3.0	9.2	8.9	8.2	7.1	4.1	3.2	0.0

Petal algorithms

1. **Route generation**
   Generate a bundle of routes (one route is called a **petal**), using constructive methods (e.g. sweep (starting from each vertex once), randomized sequential insertion, etc.)
2. **Route selection**
   Select the best combination of routes by solving a set partitioning problem.

Petal algorithms

The set partitioning problem

$$x_r = \begin{cases} 1, \text{ if petal/route } r \text{ selected,} \\ 0, \text{ otherwise.} \end{cases}$$

$a_{ir} =$ parameter $(0/1)$ indicating whether customer i is on route r

$c_r =$ costs of petal r

$$\min \sum_r c_r x_r \tag{26}$$

$$\sum_r a_{ir} x_r = 1 \qquad \forall i \tag{27}$$

$$x_r \in \{0, 1\} \qquad \forall r \tag{28}$$

Route first cluster second heuristics

- **Routing**
  Use a TSP solution method to generate one giant tour through all the customers.
- **Partitioning**
  Insert the depot in such a way that feasible VRP tours are obtained (respecting the capacity and/or the distance constraint). Exact and heuristic methods possible.

Route first cluster second heuristics

- **Routing**
  Use a TSP solution method to generate one giant tour through all the customers.
- **Partitioning**
  Insert the depot in such a way that feasible VRP tours are obtained (respecting the capacity and/or the distance constraint). Exact and heuristic methods possible.

★ Only applicable if the there is no constraint on the number of vehicles.

Route first cluster second heuristics - Example

Assume the following TSP tour through all vertices:

0 - 1 - 2 - 3 - 4 - 5 - 6 - 7 - 8 - 9 - 10 - 0

	0	1	2	3	4	5	6	7	8	9	10
d_i	0	2	1	1	2	1	1	2	1	1	2

Route first cluster second heuristics - Example

Assume the following TSP tour through all vertices:

0 - 1 - 2 - 3 - 4 - 5 - 6 - 7 - 8 - 9 - 10 - 0

	0	1	2	3	4	5	6	7	8	9	10
d_i	0	2	1	1	2	1	1	2	1	1	2

Option 1 (heuristic)
Insert the depot whenever necessary:

Route first cluster second heuristics - Example

Assume the following TSP tour through all vertices:

0 - 1 - 2 - 3 - 4 - 5 - 6 - 7 - 8 - 9 - 10 - 0

	0	1	2	3	4	5	6	7	8	9	10
d_i	0	2	1	1	2	1	1	2	1	1	2

Option 1 (heuristic)

Insert the depot whenever necessary:

0 - 1 - 2 - 3 - 0 - 4 - 5 - 6 - 0 - 7 - 8 - 9 - 0 - 10 - 0

Route first cluster second heuristics - Example

Assume the following TSP tour through all vertices:

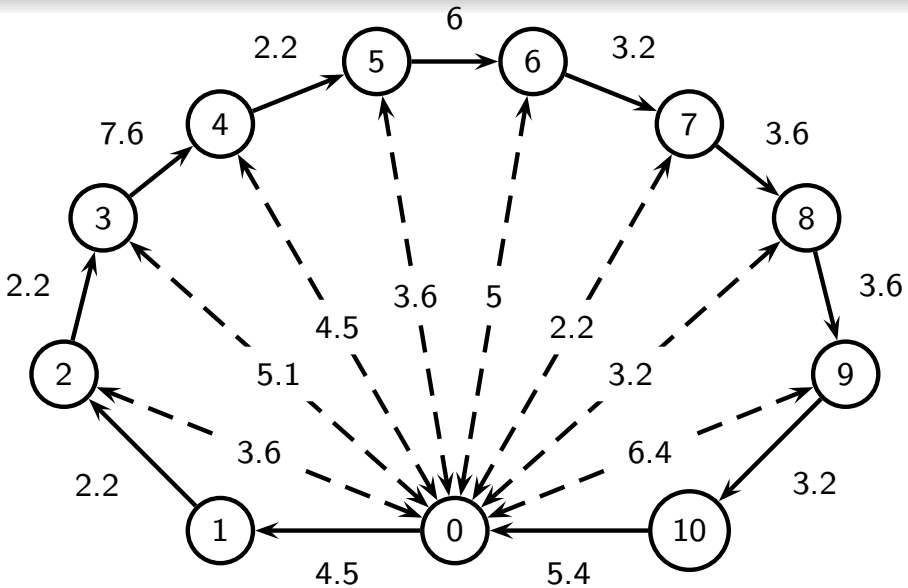
0 - 1 - 2 - 3 - 4 - 5 - 6 - 7 - 8 - 9 - 10 - 0

	0	1	2	3	4	5	6	7	8	9	10
d_i	0	2	1	1	2	1	1	2	1	1	2

Option 1 (heuristic)
Insert the depot whenever necessary:
0 - 1 - 2 - 3 - 0 - 4 - 5 - 6 - 0 - 7 - 8 - 9 - 0 - 10 - 0

Option 2 (exact)
Find the best partitioning (shortest path algorithms!):

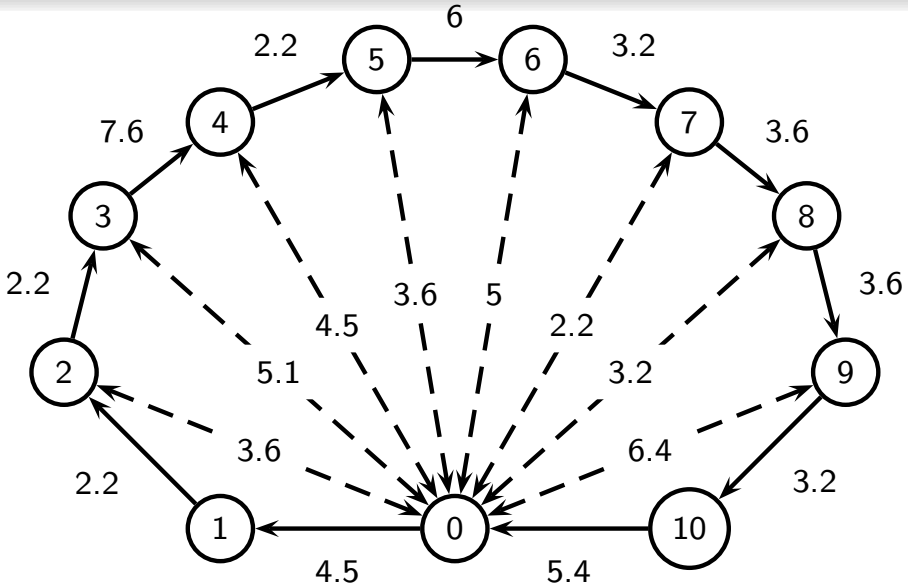
Route first cluster second heuristics - Example



	d_i
1	2
2	1
3	1
4	2
5	1
6	1
7	2
8	1
9	1
10	2

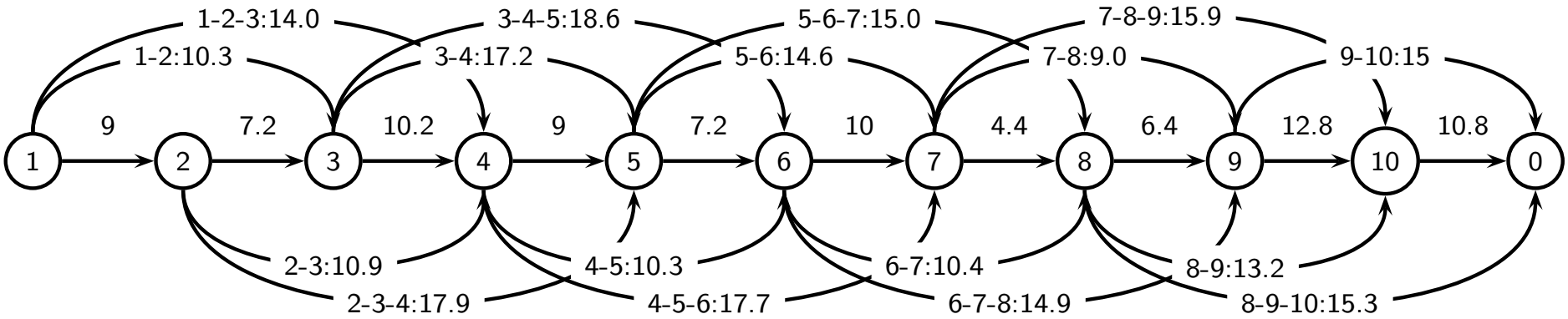
$C = 4$

Route first cluster second heuristics - Example



	d_i
1	2
2	1
3	1
4	2
5	1
6	1
7	2
8	1
9	1
10	2

$C = 4$



arc weights from node i to node k: $c_{0i} + c_{j0} + l_{ij}$, with l_{ij} costs of going from i to j in TSP tour and j direct predecessor of k in TSP tour.

Route first cluster second heuristics - Example

Find the shortest path in the auxiliary graph (Dijkstra!)

n	D[1], V[1]	D[2], V[2]	D[3], V[3]	D[4], V[4]	D[5], V[5]	D[6], V[6]	D[7], V[7]	D[8] V[8]	D[9] V[9]	D[10] V[10]	D[0] V[0]	visited

Route first cluster second heuristics - Example

Find the shortest path in the auxiliary graph (Dijkstra!)

n	D[1], V[1]	D[2], V[2]	D[3], V[3]	D[4], V[4]	D[5], V[5]	D[6], V[6]	D[7], V[7]	D[8] V[8]	D[9] V[9]	D[10] V[10]	D[0] V[0]	visited
0	0;1	9;1	10.3;1	14;1	-;1	-;1	-;1	-;1	-;1	-;1	-;1	1; 0.0

Route first cluster second heuristics - Example

Find the shortest path in the auxiliary graph (Dijkstra!)

n	D[1], V[1]	D[2], V[2]	D[3], V[3]	D[4], V[4]	D[5], V[5]	D[6], V[6]	D[7], V[7]	D[8] V[8]	D[9] V[9]	D[10] V[10]	D[0] V[0]	visited
0	0;1	9;1	10.3;1	14;1	-;1	-;1	-;1	-;1	-;1	-;1	-;1	1; 0.0
1		9;1	10.3;1	14;1	-;1	-;1	-;1	-;1	-;1	-;1	-;1	2; 9.0

Route first cluster second heuristics - Example

Find the shortest path in the auxiliary graph (Dijkstra!)

n	D[1], V[1]	D[2], V[2]	D[3], V[3]	D[4], V[4]	D[5], V[5]	D[6], V[6]	D[7], V[7]	D[8] V[8]	D[9] V[9]	D[10] V[10]	D[0] V[0]	visited
0	0;1	9;1	10.3;1	14;1	-;1	-;1	-;1	-;1	-;1	-;1	-;1	1; 0.0
1		9;1	10.3;1	14;1	-;1	-;1	-;1	-;1	-;1	-;1	-;1	2; 9.0
2			10.3;1	14;1	26.9;2	-;1	-;1	-;1	-;1	-;1	-;1	3; 10.3

Route first cluster second heuristics - Example

Find the shortest path in the auxiliary graph (Dijkstra!)

n	D[1], V[1]	D[2], V[2]	D[3], V[3]	D[4], V[4]	D[5], V[5]	D[6], V[6]	D[7], V[7]	D[8] V[8]	D[9] V[9]	D[10] V[10]	D[0] V[0]	visited
0	0;1	9;1	10.3;1	14;1	-;1	-;1	-;1	-;1	-;1	-;1	-;1	1; 0.0
1		9;1	10.3;1	14;1	-;1	-;1	-;1	-;1	-;1	-;1	-;1	2; 9.0
2			10.3;1	14;1	26.9;2	-;1	-;1	-;1	-;1	-;1	-;1	3; 10.3
3				14;1	26.9;2	28.9;3	-;1	-;1	-;1	-;1	-;1	4; 14.0

Route first cluster second heuristics - Example

Find the shortest path in the auxiliary graph (Dijkstra!)

n	D[1], V[1]	D[2], V[2]	D[3], V[3]	D[4], V[4]	D[5], V[5]	D[6], V[6]	D[7], V[7]	D[8] V[8]	D[9] V[9]	D[10] V[10]	D[0] V[0]	visited
0	0;1	9;1	10.3;1	14;1	-;1	-;1	-;1	-;1	-;1	-;1	-;1	1; 0.0
1		9;1	10.3;1	14;1	-;1	-;1	-;1	-;1	-;1	-;1	-;1	2; 9.0
2			10.3;1	14;1	26.9;2	-;1	-;1	-;1	-;1	-;1	-;1	3; 10.3
3				14;1	26.9;2	28.9;3	-;1	-;1	-;1	-;1	-;1	4; 14.0
4					23;4	24.3;4	31.7;4	-;1	-;1	-;1	-;1	5; 23.0

Route first cluster second heuristics - Example

Find the shortest path in the auxiliary graph (Dijkstra!)

n	D[1], V[1]	D[2], V[2]	D[3], V[3]	D[4], V[4]	D[5], V[5]	D[6], V[6]	D[7], V[7]	D[8] V[8]	D[9] V[9]	D[10] V[10]	D[0] V[0]	visited
0	0;1	9;1	10.3;1	14;1	-;1	-;1	-;1	-;1	-;1	-;1	-;1	1; 0.0
1		9;1	10.3;1	14;1	-;1	-;1	-;1	-;1	-;1	-;1	-;1	2; 9.0
2			10.3;1	14;1	26.9;2	-;1	-;1	-;1	-;1	-;1	-;1	3; 10.3
3				14;1	26.9;2	28.9;3	-;1	-;1	-;1	-;1	-;1	4; 14.0
4					23;4	24.3;4	31.7;4	-;1	-;1	-;1	-;1	5; 23.0
5						24.3;4	31.7;4	38;5	-;1	-;1	-;1	6; 24.3

Route first cluster second heuristics - Example

Find the shortest path in the auxiliary graph (Dijkstra!)

n	D[1], V[1]	D[2], V[2]	D[3], V[3]	D[4], V[4]	D[5], V[5]	D[6], V[6]	D[7], V[7]	D[8] V[8]	D[9] V[9]	D[10] V[10]	D[0] V[0]	visited
0	0;1	9;1	10.3;1	14;1	-;1	-;1	-;1	-;1	-;1	-;1	-;1	1; 0.0
1		9;1	10.3;1	14;1	-;1	-;1	-;1	-;1	-;1	-;1	-;1	2; 9.0
2			10.3;1	14;1	26.9;2	-;1	-;1	-;1	-;1	-;1	-;1	3; 10.3
3				14;1	26.9;2	28.9;3	-;1	-;1	-;1	-;1	-;1	4; 14.0
4					23;4	24.3;4	31.7;4	-;1	-;1	-;1	-;1	5; 23.0
5						24.3;4	31.7;4	38;5	-;1	-;1	-;1	6; 24.3
6							31.7;4	34.7;6	39.2;6	-;1	-;1	7; 31.7

Route first cluster second heuristics - Example

Find the shortest path in the auxiliary graph (Dijkstra!)

n	D[1], V[1]	D[2], V[2]	D[3], V[3]	D[4], V[4]	D[5], V[5]	D[6], V[6]	D[7], V[7]	D[8] V[8]	D[9] V[9]	D[10] V[10]	D[0] V[0]	visited
0	0;1	9;1	10.3;1	14;1	-;1	-;1	-;1	-;1	-;1	-;1	-;1	1; 0.0
1		9;1	10.3;1	14;1	-;1	-;1	-;1	-;1	-;1	-;1	-;1	2; 9.0
2			10.3;1	14;1	26.9;2	-;1	-;1	-;1	-;1	-;1	-;1	3; 10.3
3				14;1	26.9;2	28.9;3	-;1	-;1	-;1	-;1	-;1	4; 14.0
4					23;4	24.3;4	31.7;4	-;1	-;1	-;1	-;1	5; 23.0
5						24.3;4	31.7;4	38;5	-;1	-;1	-;1	6; 24.3
6							31.7;4	34.7;6	39.2;6	-;1	-;1	7; 31.7
7								34.7;6	39.2;6	47.6;7	-;1	8; 34.7

Route first cluster second heuristics - Example

Find the shortest path in the auxiliary graph (Dijkstra!)

n	D[1], V[1]	D[2], V[2]	D[3], V[3]	D[4], V[4]	D[5], V[5]	D[6], V[6]	D[7], V[7]	D[8] V[8]	D[9] V[9]	D[10] V[10]	D[0] V[0]	visited
0	0;1	9;1	10.3;1	14;1	-;1	-;1	-;1	-;1	-;1	-;1	-;1	1; 0.0
1		9;1	10.3;1	14;1	-;1	-;1	-;1	-;1	-;1	-;1	-;1	2; 9.0
2			10.3;1	14;1	26.9;2	-;1	-;1	-;1	-;1	-;1	-;1	3; 10.3
3				14;1	26.9;2	28.9;3	-;1	-;1	-;1	-;1	-;1	4; 14.0
4					23;4	24.3;4	31.7;4	-;1	-;1	-;1	-;1	5; 23.0
5						24.3;4	31.7;4	38;5	-;1	-;1	-;1	6; 24.3
6							31.7;4	34.7;6	39.2;6	-;1	-;1	7; 31.7
7								34.7;6	39.2;6	47.6;7	-;1	8; 34.7
8									39.2;6	47.6;7	50;8	9; 39.2

Route first cluster second heuristics - Example

Find the shortest path in the auxiliary graph (Dijkstra!)

n	D[1], V[1]	D[2], V[2]	D[3], V[3]	D[4], V[4]	D[5], V[5]	D[6], V[6]	D[7], V[7]	D[8] V[8]	D[9] V[9]	D[10] V[10]	D[0] V[0]	visited
0	0;1	9;1	10.3;1	14;1	-;1	-;1	-;1	-;1	-;1	-;1	-;1	1; 0.0
1		9;1	10.3;1	14;1	-;1	-;1	-;1	-;1	-;1	-;1	-;1	2; 9.0
2			10.3;1	14;1	26.9;2	-;1	-;1	-;1	-;1	-;1	-;1	3; 10.3
3				14;1	26.9;2	28.9;3	-;1	-;1	-;1	-;1	-;1	4; 14.0
4					23;4	24.3;4	31.7;4	-;1	-;1	-;1	-;1	5; 23.0
5						24.3;4	31.7;4	38;5	-;1	-;1	-;1	6; 24.3
6							31.7;4	34.7;6	39.2;6	-;1	-;1	7; 31.7
7								34.7;6	39.2;6	47.6;7	-;1	8; 34.7
8									39.2;6	47.6;7	50;8	9; 39.2
9										47.6;7	50;8	10; 47.6

Route first cluster second heuristics - Example

Find the shortest path in the auxiliary graph (Dijkstra!)

n	D[1], V[1]	D[2], V[2]	D[3], V[3]	D[4], V[4]	D[5], V[5]	D[6], V[6]	D[7], V[7]	D[8] V[8]	D[9] V[9]	D[10] V[10]	D[0] V[0]	visited
0	0;1	9;1	10.3;1	14;1	-;1	-;1	-;1	-;1	-;1	-;1	-;1	1; 0.0
1		9;1	10.3;1	14;1	-;1	-;1	-;1	-;1	-;1	-;1	-;1	2; 9.0
2			10.3;1	14;1	26.9;2	-;1	-;1	-;1	-;1	-;1	-;1	3; 10.3
3				14;1	26.9;2	28.9;3	-;1	-;1	-;1	-;1	-;1	4; 14.0
4					23;4	24.3;4	31.7;4	-;1	-;1	-;1	-;1	5; 23.0
5						24.3;4	31.7;4	38;5	-;1	-;1	-;1	6; 24.3
6							31.7;4	34.7;6	39.2;6	-;1	-;1	7; 31.7
7								34.7;6	39.2;6	47.6;7	-;1	8; 34.7
8									39.2;6	47.6;7	50;8	9; 39.2
9										47.6;7	50;8	10; 47.6
10											50;8	0; 50

Route first cluster second heuristics - Example

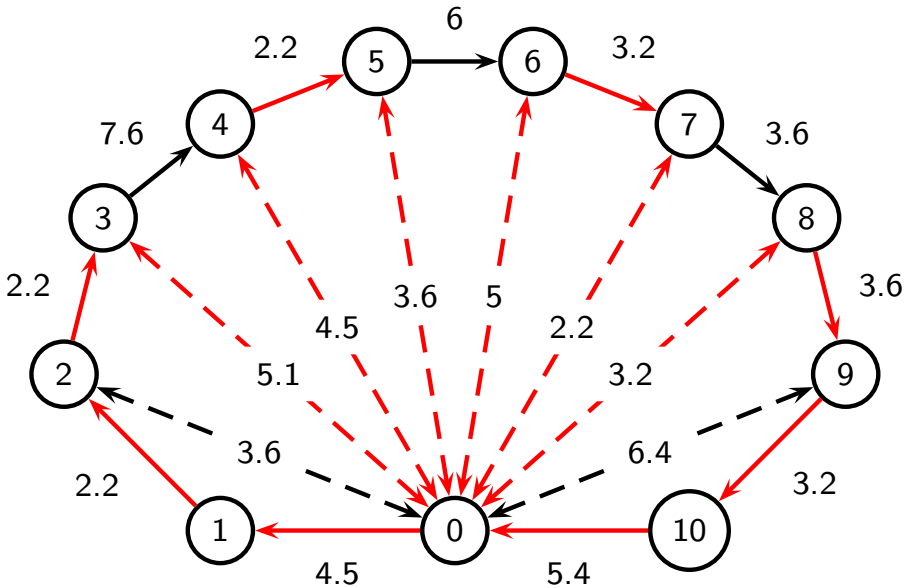
Find the shortest path in the auxiliary graph (Dijkstra!)

n	D[1], V[1]	D[2], V[2]	D[3], V[3]	D[4], V[4]	D[5], V[5]	D[6], V[6]	D[7], V[7]	D[8] V[8]	D[9] V[9]	D[10] V[10]	D[0] V[0]	visited
0	0;1	9;1	10.3;1	14;1	-;1	-;1	-;1	-;1	-;1	-;1	-;1	1; 0.0
1		9;1	10.3;1	14;1	-;1	-;1	-;1	-;1	-;1	-;1	-;1	2; 9.0
2			10.3;1	14;1	26.9;2	-;1	-;1	-;1	-;1	-;1	-;1	3; 10.3
3				14;1	26.9;2	28.9;3	-;1	-;1	-;1	-;1	-;1	4; 14.0
4					23;4	24.3;4	31.7;4	-;1	-;1	-;1	-;1	5; 23.0
5						24.3;4	31.7;4	38;5	-;1	-;1	-;1	6; 24.3
6							31.7;4	34.7;6	39.2;6	-;1	-;1	7; 31.7
7								34.7;6	39.2;6	47.6;7	-;1	8; 34.7
8									39.2;6	47.6;7	50;8	9; 39.2
9										47.6;7	50;8	10; 47.6
10											50;8	0; 50

The shortest path:

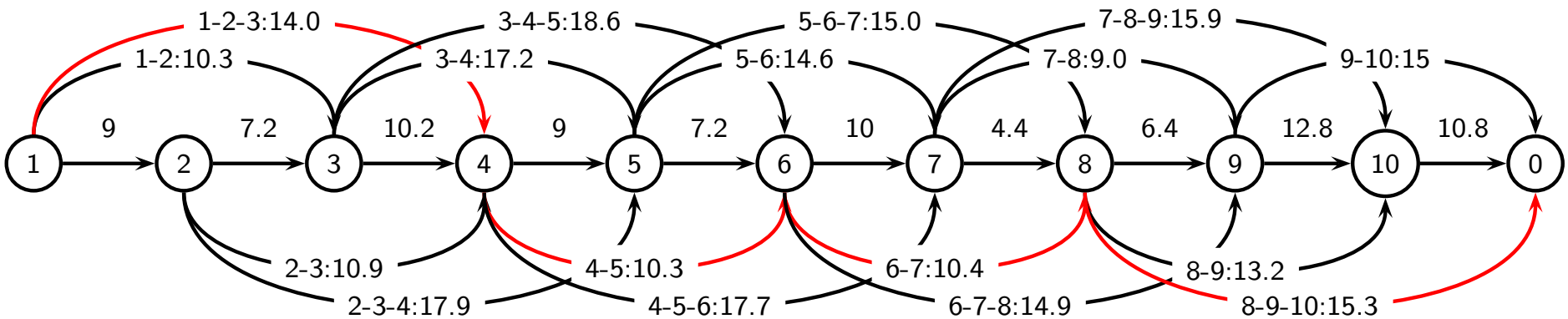
1 - 4 - 6 - 8 - 0

Route first cluster second heuristics - Example



	d_i
1	2
2	1
3	1
4	2
5	1
6	1
7	2
8	1
9	1
10	2

$C = 4$



References

Paolo Toth, and Daniele Vigo (2002) The Vehicle Routing Problem, SIAM. (Chapters 1 and 5)

W. Domschke (1997) 'Logistik: Rundreisen und Touren' Oldenbourg.