Freight Network Models

CIVE 461: Urban Transportation Planning

Basics of Network Models

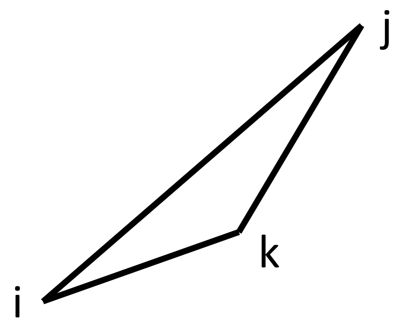
- Types of problems:
  - Calculating travel times & distances where travel is restricted by the network
  - Vehicle routing, collection, & distribution
  - Site selection & location of facilities (not covered here)

Node Covering Problems

Traveling Salesman Problem

Traveling Salesman Problem

- Find the shortest cycle starting & ending at node O that visits each node A, B, C, D, etc. at least once
- TSP1 simplification
  - Given a starting point (depot)
  - Visit n-1 points
  - Network completely connected
  - Network satisfies triangular inequality: l(i,j) <= l(i,k) + l (k,j)
  - Distance matrix is symmetric



Solving TSP1

- 3-step processing – each step applies a well-known algorithm
- Final network graph H consists of minimum spanning tree (MST) + min. length pairwise matching
- Heuristic solution (good, but not necessarily optimal)
- Theorem: $L(H) < 1.5$ L(TST, or Traveling Salesman Theorem result)

Example TSP1 Problem

Distance Matrix

From\ To	1	2	3	4	5	6	7
1	0	25	43	57	43	61	29
2	25	0	29	34	43	68	49
3	43	29	0	52	72	96	72
4	57	34	52	0	45	71	71
5	43	43	72	45	0	27	36
6	61	68	96	71	27	0	40
7	29	49	72	71	36	40	0

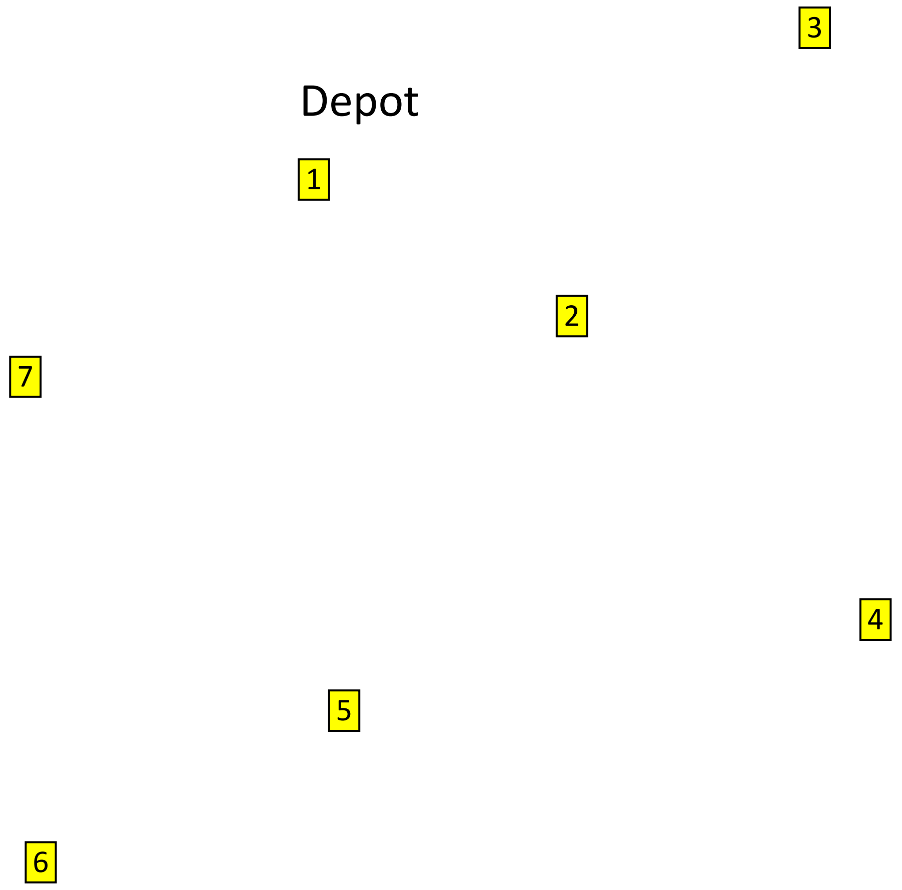
Heuristic Algorithm for TSP1

Step 1: Find the MST

Step 2: Minimum length pairwise matching of odd-degree nodes. Add these links to the network

Step 3: Draw Eulerian Circuit

Step 4: For nodes that are visited more than once, improve by taking advantage of the triangular inequality



Example TSP1 Problem

Distance Matrix

From\ To	1	2	3	4	5	6	7
1	0	25	43	57	43	61	29
2	25	0	29	34	43	68	49
3	43	29	0	52	72	96	72
4	57	34	52	0	45	71	71
5	43	43	72	45	0	27	36
6	61	68	96	71	27	0	40
7	29	49	72	71	36	40	0

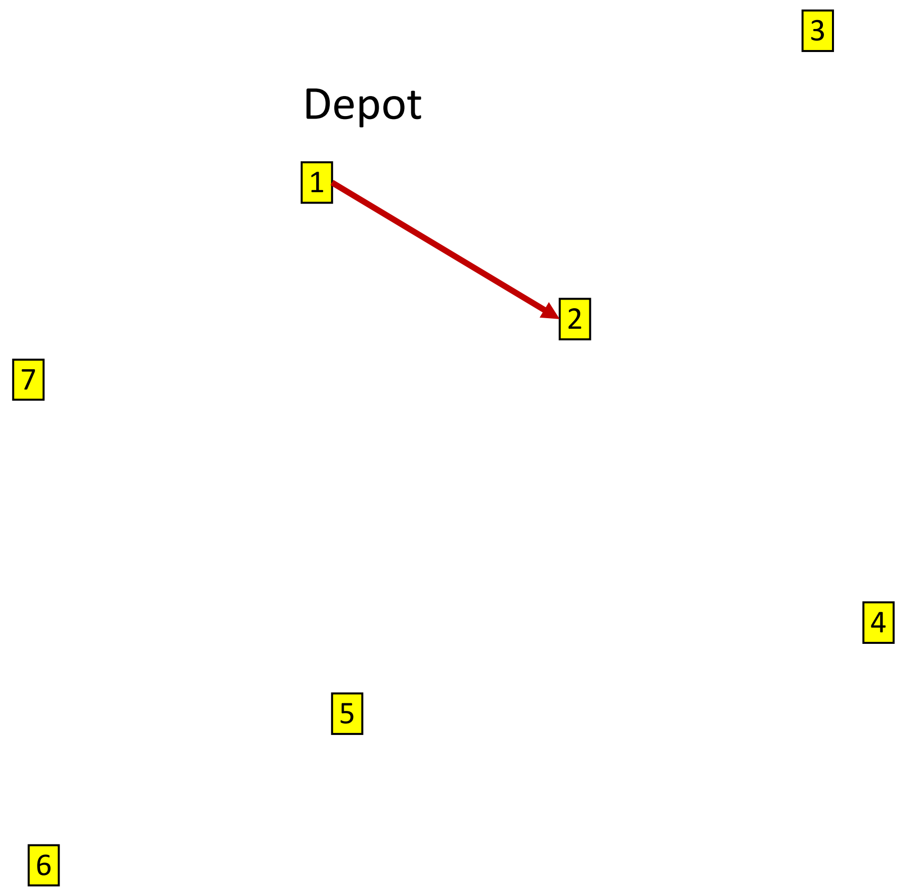
Heuristic Algorithm for TSP1

Step 1: Find the MST

Step 2: Minimum length pairwise matching of odd-degree nodes. Add these links to the network

Step 3: Draw Eulerian Circuit

Step 4: For nodes that are visited more than once, improve by taking advantage of the triangular inequality



Example TSP1 Problem

Distance Matrix

From\ To	1	2	3	4	5	6	7
1	0	25	43	57	43	61	29
2	25	0	29	34	43	68	49
3	43	29	0	52	72	96	72
4	57	34	52	0	45	71	71
5	43	43	72	45	0	27	36
6	61	68	96	71	27	0	40
7	29	49	72	71	36	40	0

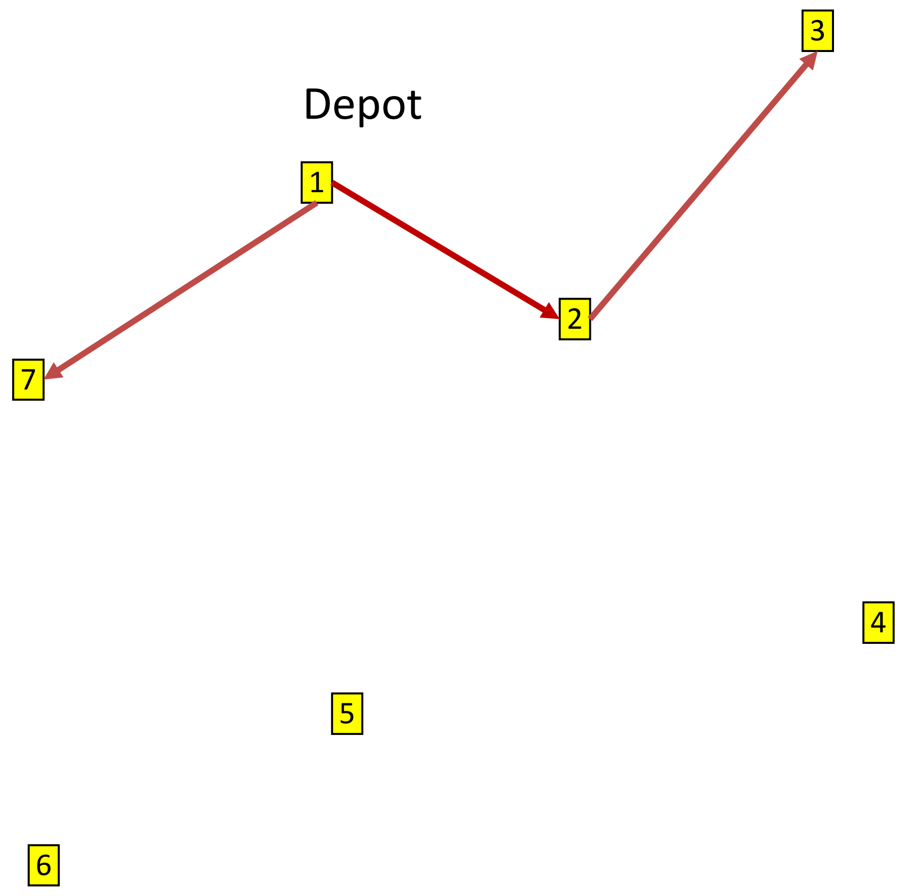
Heuristic Algorithm for TSP1

Step 1: Find the MST

Step 2: Minimum length pairwise matching of odd-degree nodes. Add these links to the network

Step 3: Draw Eulerian Circuit

Step 4: For nodes that are visited more than once, improve by taking advantage of the triangular inequality



Example TSP1 Problem

Distance Matrix

From\ To	1	2	3	4	5	6	7
1	0	25	43	57	43	61	29
2	25	0	29	34	43	68	49
3	43	29	0	52	72	96	72
4	57	34	52	0	45	71	71
5	43	43	72	45	0	27	36
6	61	68	96	71	27	0	40
7	29	49	72	71	36	40	0

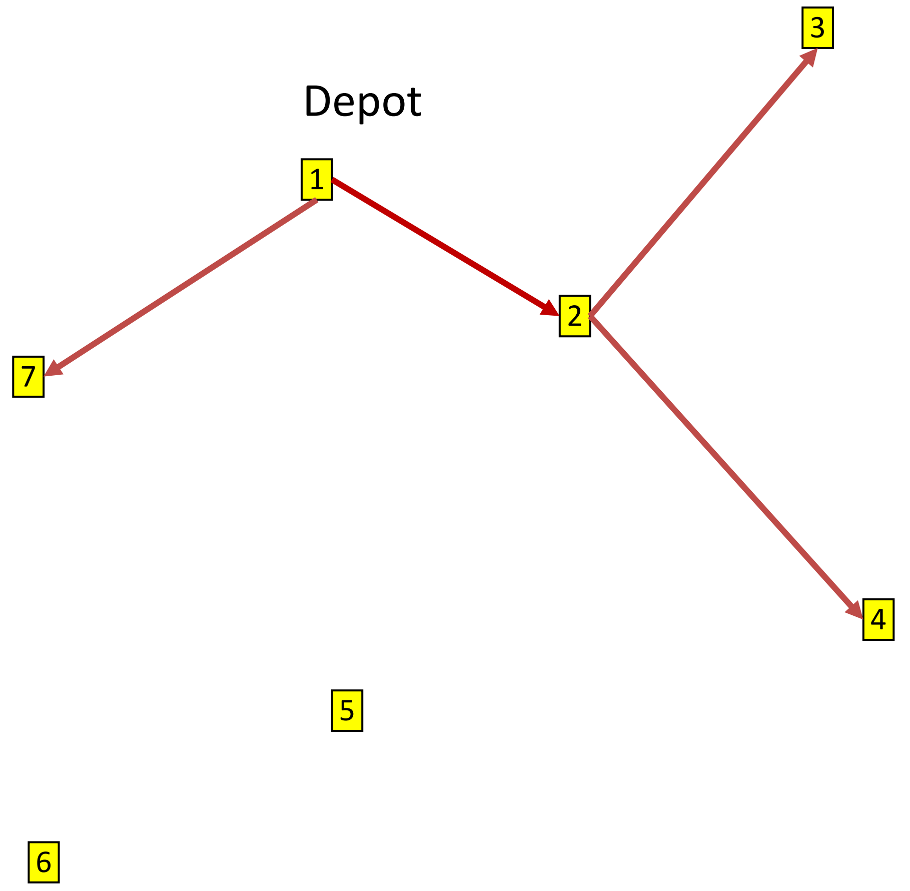
Heuristic Algorithm for TSP1

Step 1: Find the MST

Step 2: Minimum length pairwise matching of odd-degree nodes. Add these links to the network

Step 3: Draw Eulerian Circuit

Step 4: For nodes that are visited more than once, improve by taking advantage of the triangular inequality



Example TSP1 Problem

Distance Matrix

From\ To	1	2	3	4	5	6	7
1	0	25	43	57	43	61	29
2	25	0	29	34	43	68	49
3	43	29	0	52	72	96	72
4	57	34	52	0	45	71	71
5	43	43	72	45	0	27	36
6	61	68	96	71	27	0	40
7	29	49	72	71	36	40	0

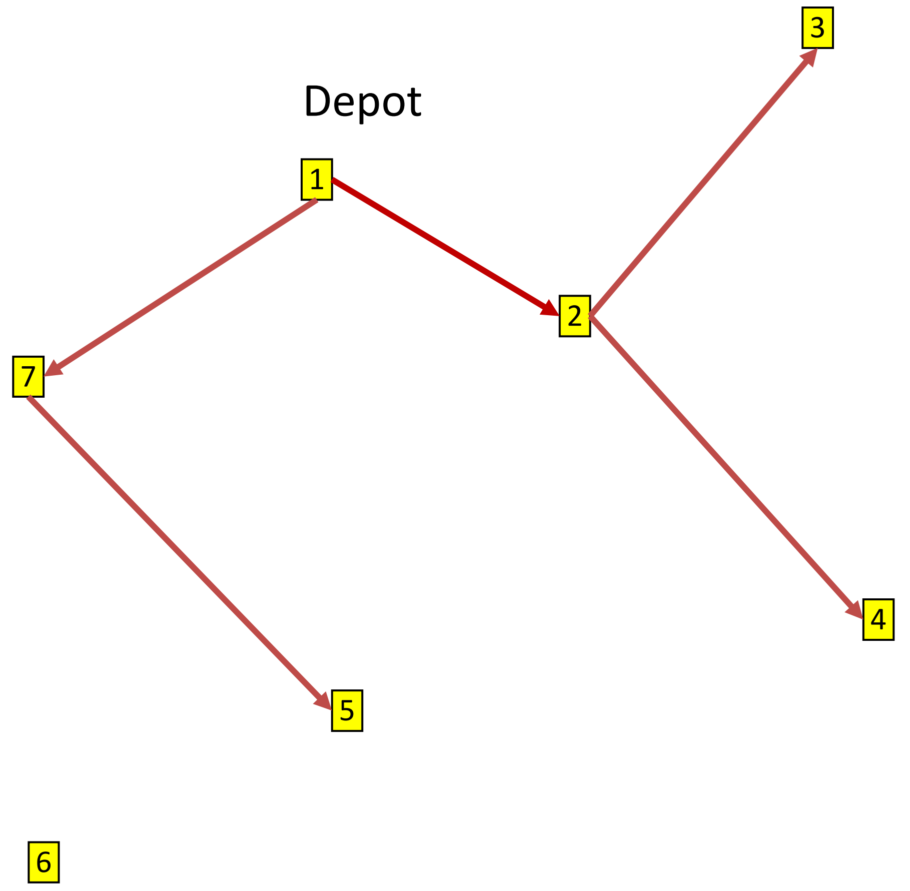
Heuristic Algorithm for TSP1

Step 1: Find the MST

Step 2: Minimum length pairwise matching of odd-degree nodes. Add these links to the network

Step 3: Draw Eulerian Circuit

Step 4: For nodes that are visited more than once, improve by taking advantage of the triangular inequality



Example TSP1 Problem

Distance Matrix

From\ To	1	2	3	4	5	6	7
1	0	25	43	57	43	61	29
2	25	0	29	34	43	68	49
3	43	29	0	52	72	96	72
4	57	34	52	0	45	71	71
5	43	43	72	45	0	27	36
6	61	68	96	71	27	0	40
7	29	49	72	71	36	40	0

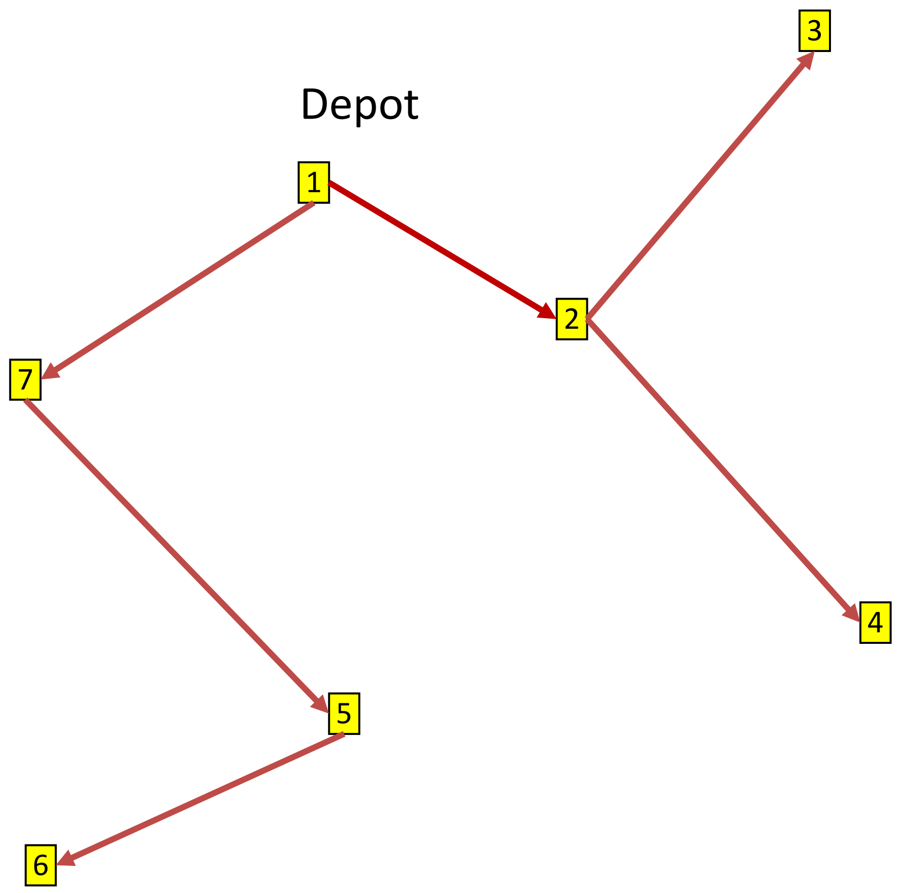
Heuristic Algorithm for TSP1

Step 1: Find the MST

Step 2: Minimum length pairwise matching of odd-degree nodes. Add these links to the network

Step 3: Draw Eulerian Circuit

Step 4: For nodes that are visited more than once, improve by taking advantage of the triangular inequality



Example TSP1 Problem

Distance Matrix

From\ To	1	2	3	4	5	6	7
1	0	25	43	57	43	61	29
2	25	0	29	34	43	68	49
3	43	29	0	52	72	96	72
4	57	34	52	0	45	71	71
5	43	43	72	45	0	27	36
6	61	68	96	71	27	0	40
7	29	49	72	71	36	40	0

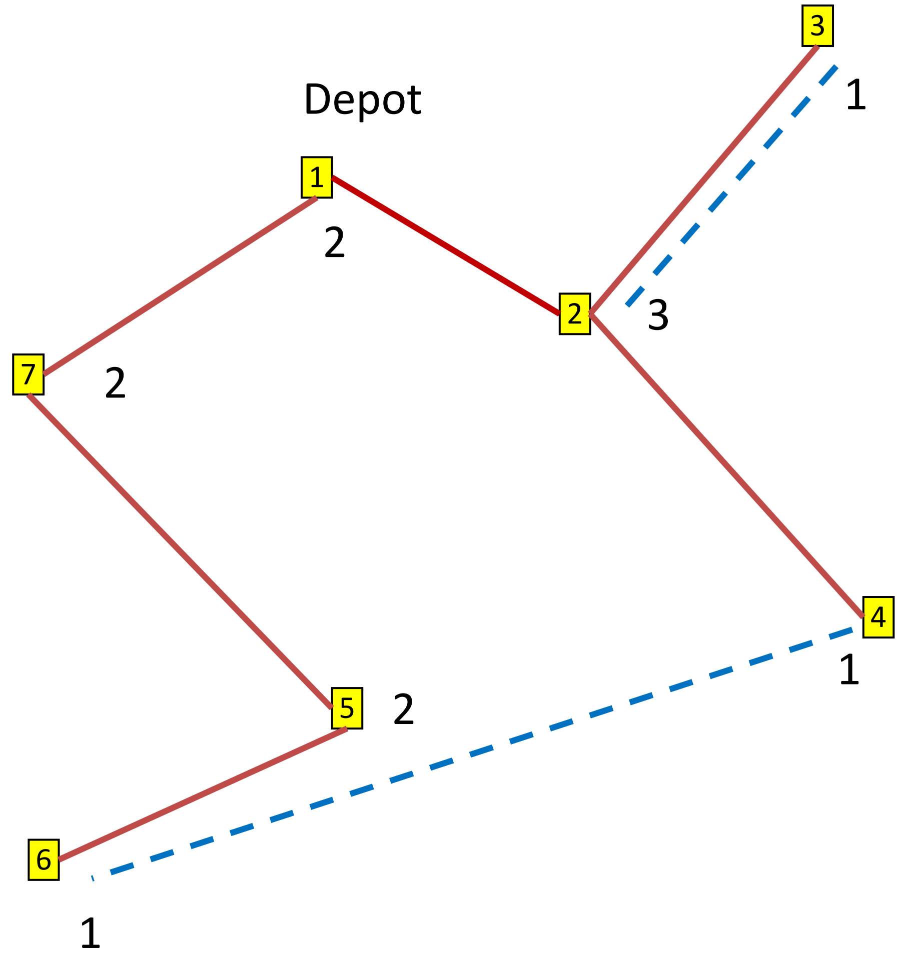
Heuristic Algorithm for TSP1

Step 1: Find the MST

Step 2: Minimum length pairwise matching of odd-degree nodes. Add these links to the network

Step 3: Draw Eulerian Circuit

Step 4: For nodes that are visited more than once, improve by taking advantage of the triangular inequality



Example TSP1 Problem

Distance Matrix

From\ To	1	2	3	4	5	6	7
1	0	25	43	57	43	61	29
2	25	0	29	34	43	68	49
3	43	29	0	52	72	96	72
4	57	34	52	0	45	71	71
5	43	43	72	45	0	27	36
6	61	68	96	71	27	0	40
7	29	49	72	71	36	40	0

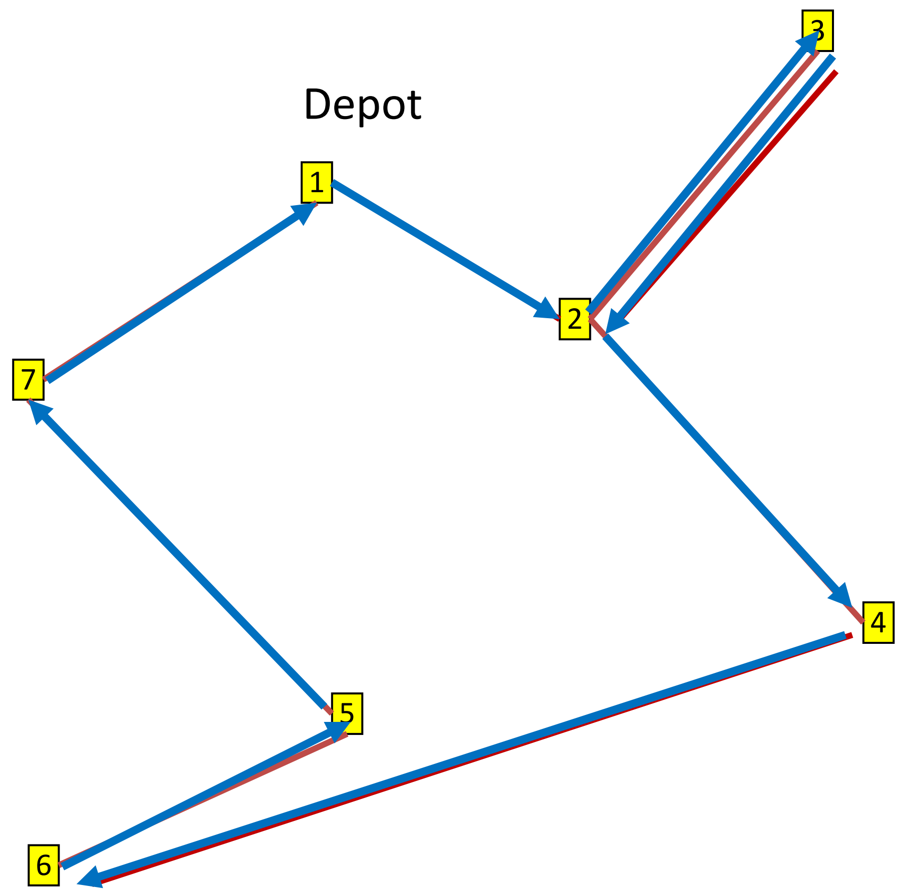
Heuristic Algorithm for TSP1

Step 1: Find the MST

Step 2: Minimum length pairwise matching of odd-degree nodes. Add these links to the network

Step 3: Draw Eulerian Circuit

Step 4: For nodes that are visited more than once, improve by taking advantage of the triangular inequality



Example TSP1 Problem

Distance Matrix

From\ To	1	2	3	4	5	6	7
1	0	25	43	57	43	61	29
2	25	0	29	34	43	68	49
3	43	29	0	52	72	96	72
4	57	34	52	0	45	71	71
5	43	43	72	45	0	27	36
6	61	68	96	71	27	0	40
7	29	49	72	71	36	40	0

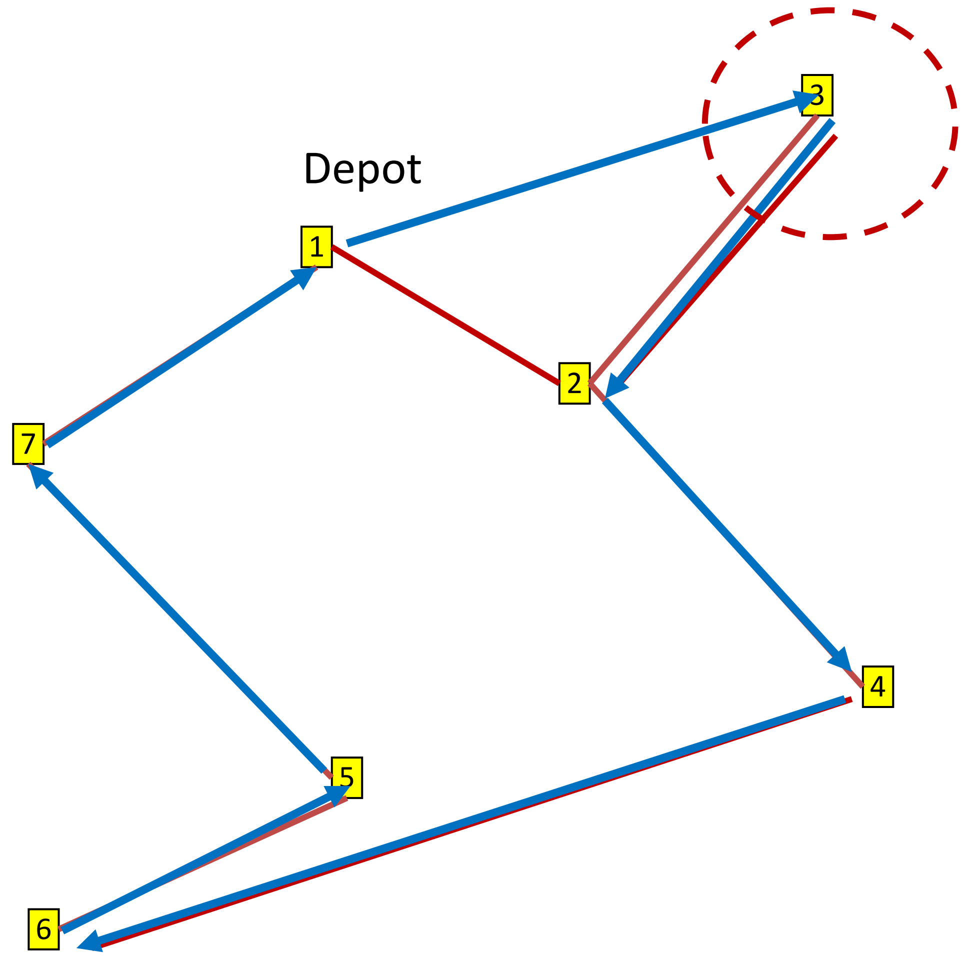
Heuristic Algorithm for TSP1

Step 1: Find the MST

Step 2: Minimum length pairwise matching of odd-degree nodes. Add these links to the network

Step 3: Draw Eulerian Circuit

Step 4: For nodes that are visited more than once, improve by taking advantage of the triangular inequality



Example TSP1 Optimal Solution

Distance Matrix

From\ To	1	2	3	4	5	6	7
1	0	25	43	57	43	61	29
2	25	0	29	34	43	68	49
3	43	29	0	52	72	96	72
4	57	34	52	0	45	71	71
5	43	43	72	45	0	27	36
6	61	68	96	71	27	0	40
7	29	49	72	71	36	40	0

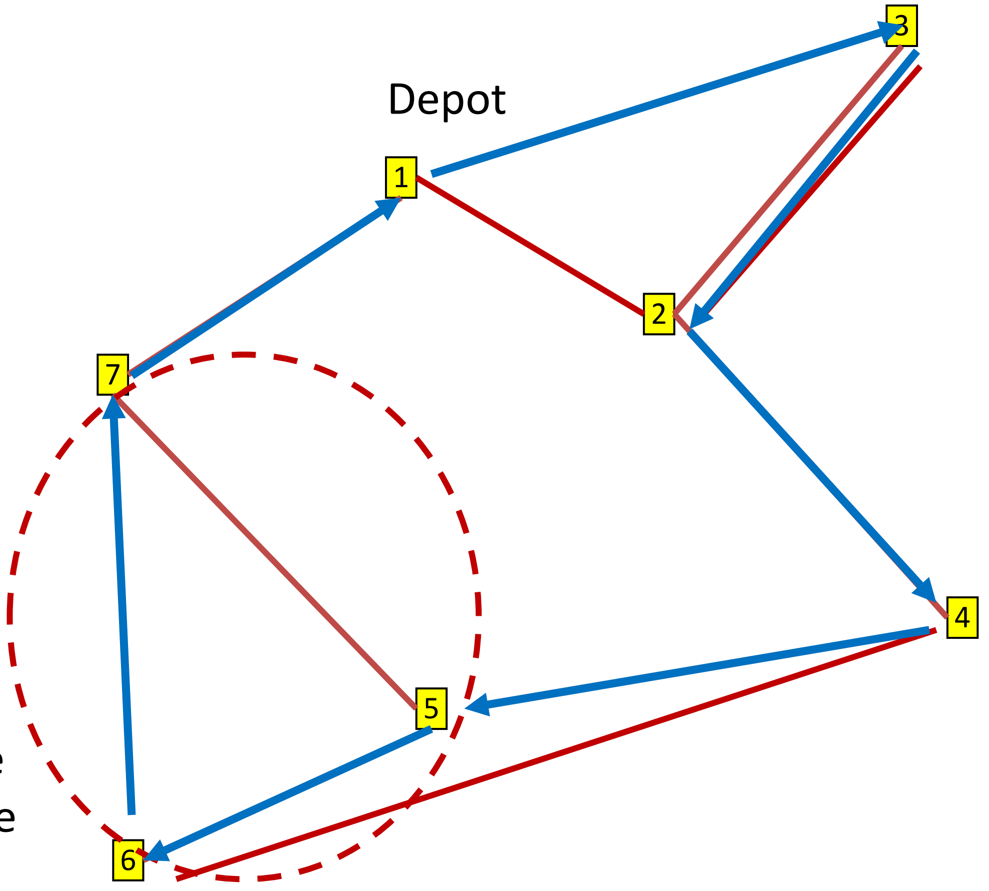
Heuristic Algorithm for TSP1

Step 1: Find the MST

Step 2: Minimum length pairwise matching of odd-degree nodes. Add these links to the network

Step 3: Draw Eulerian Circuit

Step 4: For nodes that are visited more than once, improve by taking advantage of the triangular inequality



Multi-Route Problems

- Actual situations: several vehicles share provision of service in an area
- Mostly heuristic algorithms; two approaches:
  - **Partition region into smaller districts:** design optimal single routes for each district
  - **Design single route for whole area:** subdivide route into no. of sub-routes each covered by diff. vehicle

Multi-Route Node Covering

- Basis for classification of node covering problems:
  - Number of vehicles
  - Number of tour origins/depots
  - Existence of constraints on vehicle capacity, max. tour length, ..

- Classical TSP: single vehicle, single origin, no constraints

- **m-TSP**:
  - m distinct tours
  - Single common origin

- **VRP** (vehicle routing problems)
  - Constraints on capacity or max. distance
  - Need to minimize total system cost

m-TSP

- Design of:
  - m distinct tours
  - Collectively visit each demand point at least once
  - Use a single common origin/destination

- Procedure:
  - Replace origin by m exact copies
  - Assign “infinite” lengths to connections between “origins”
  - Solve as classical (m+n) point TSP
  - Merge copies of origin => m diff. tours

Example m-TSP Problem

Distance Matrix

From/ To	1a	1b	2	3	4	5	6	7
1a	∞	∞	25	43	57	43	61	29
1b	∞	∞	25	43	57	43	61	29
2	25	25	∞	29	34	43	68	49
3	43	43	29	∞	52	72	96	72
4	57	57	34	52	∞	45	71	71
5	43	43	43	72	45	∞	27	36
6	61	61	68	96	71	27	∞	40
7	29	29	49	72	71	36	40	∞

A) Add m "copies" of the origin with infinite length between origins

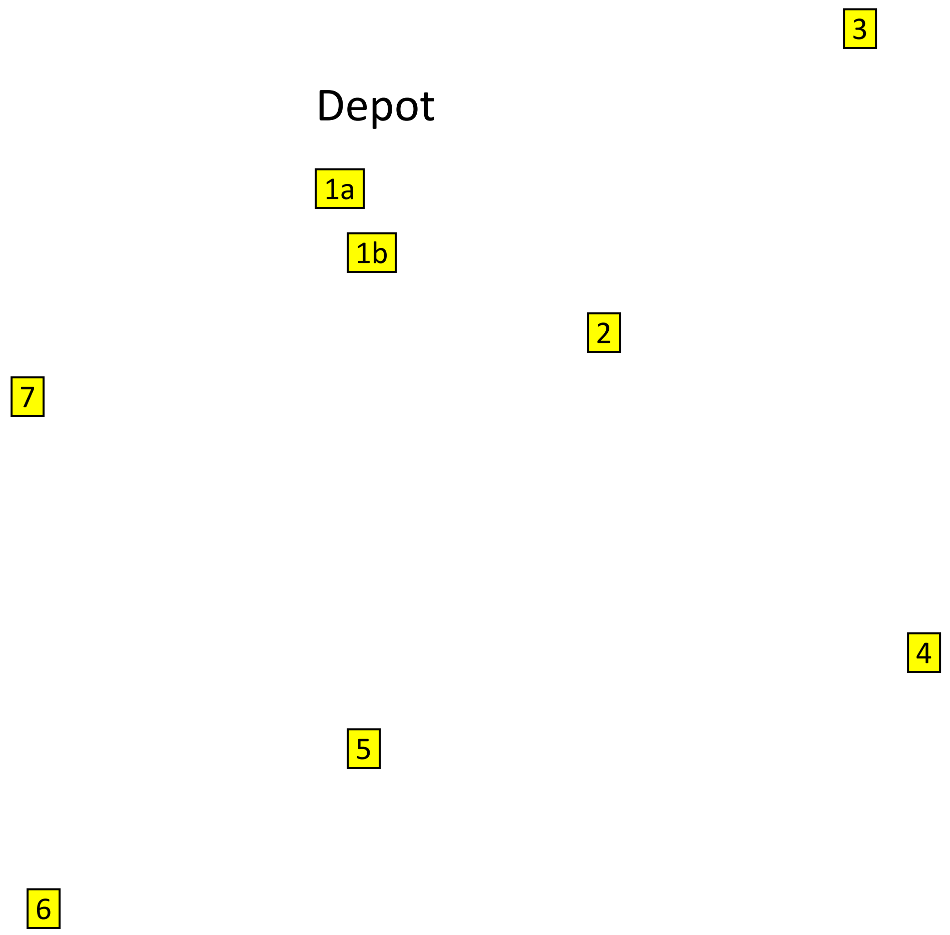
B) Use Heuristic Algorithm for TSP1

Step 1: Find the MST

Step 2: Minimum length pairwise matching of odd-degree nodes. Add these links to the network

Step 3: Draw Eulerian Circuit

Step 4: For nodes that are visited more than once, improve by taking advantage of triangular inequality



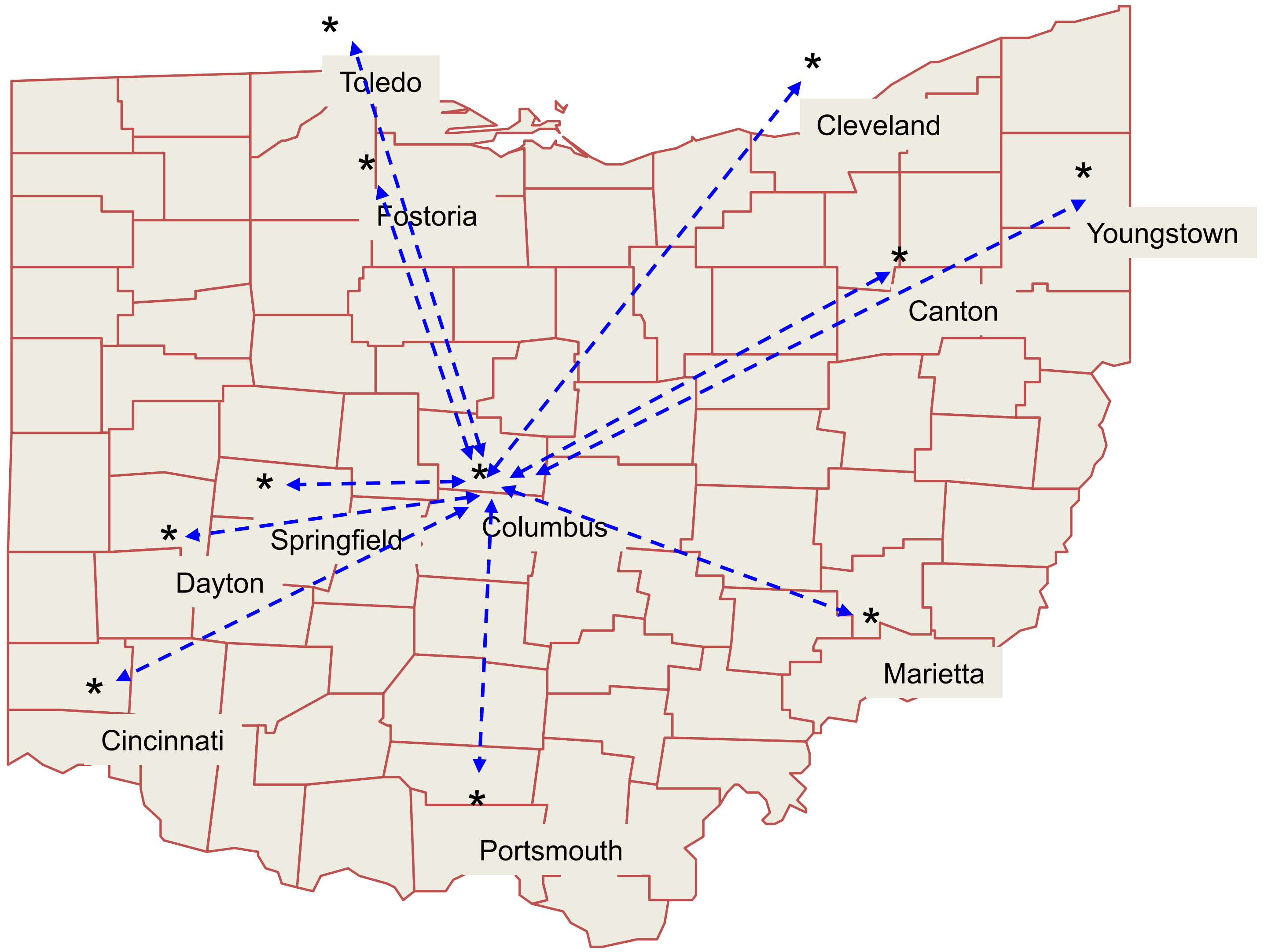
Single-Depot VRP

- Two algorithms:

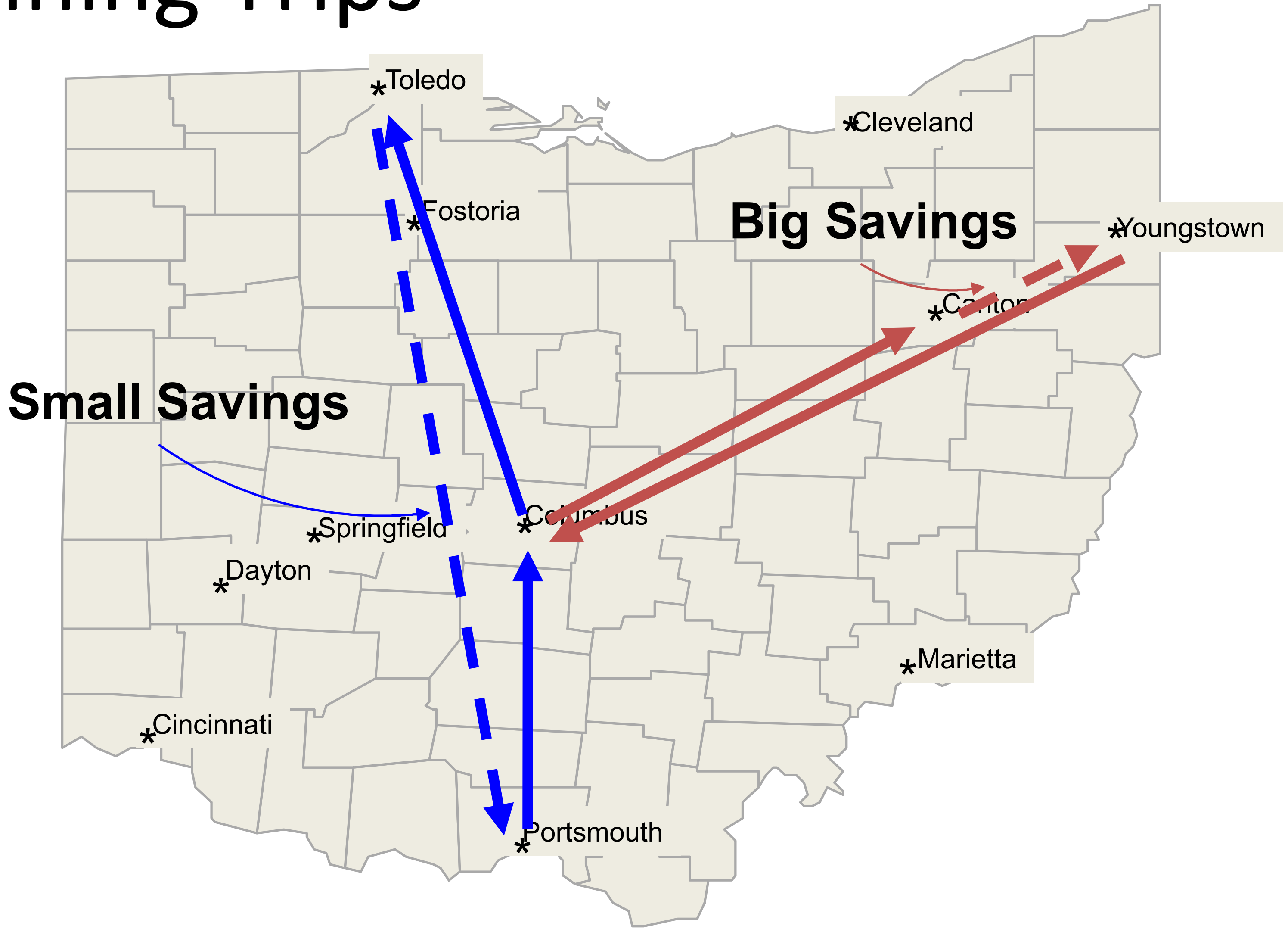
- Clarke-Wright savings algorithm – basic idea:
  - Depot D, n demand points
  - Initial solution: use n vehicles, one per demand point
  - “save” by combining two points

- Sweep algorithm – basic idea:
  - “cluster first, route second”

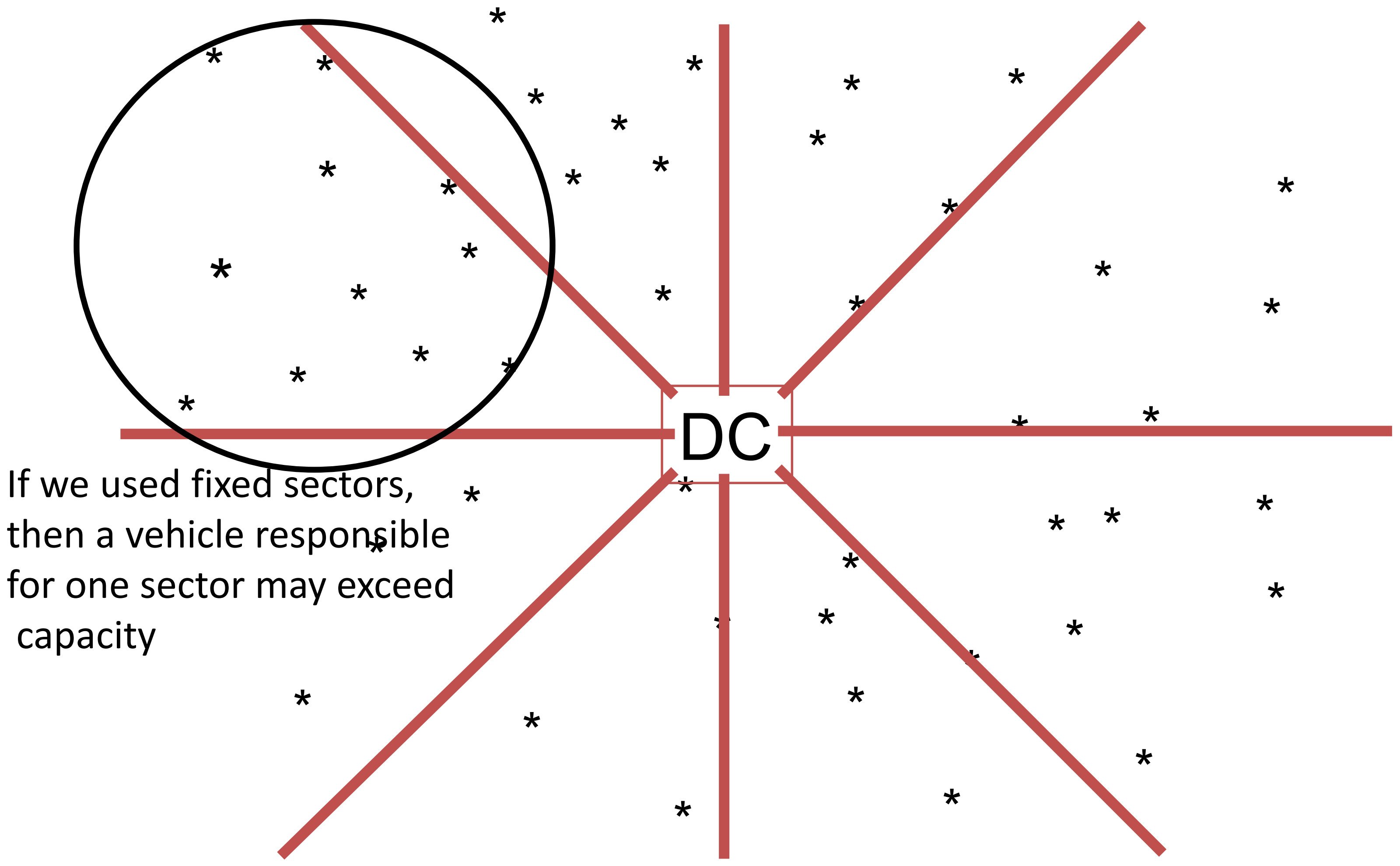
Clarke-Wright Algorithm



Combining Trips



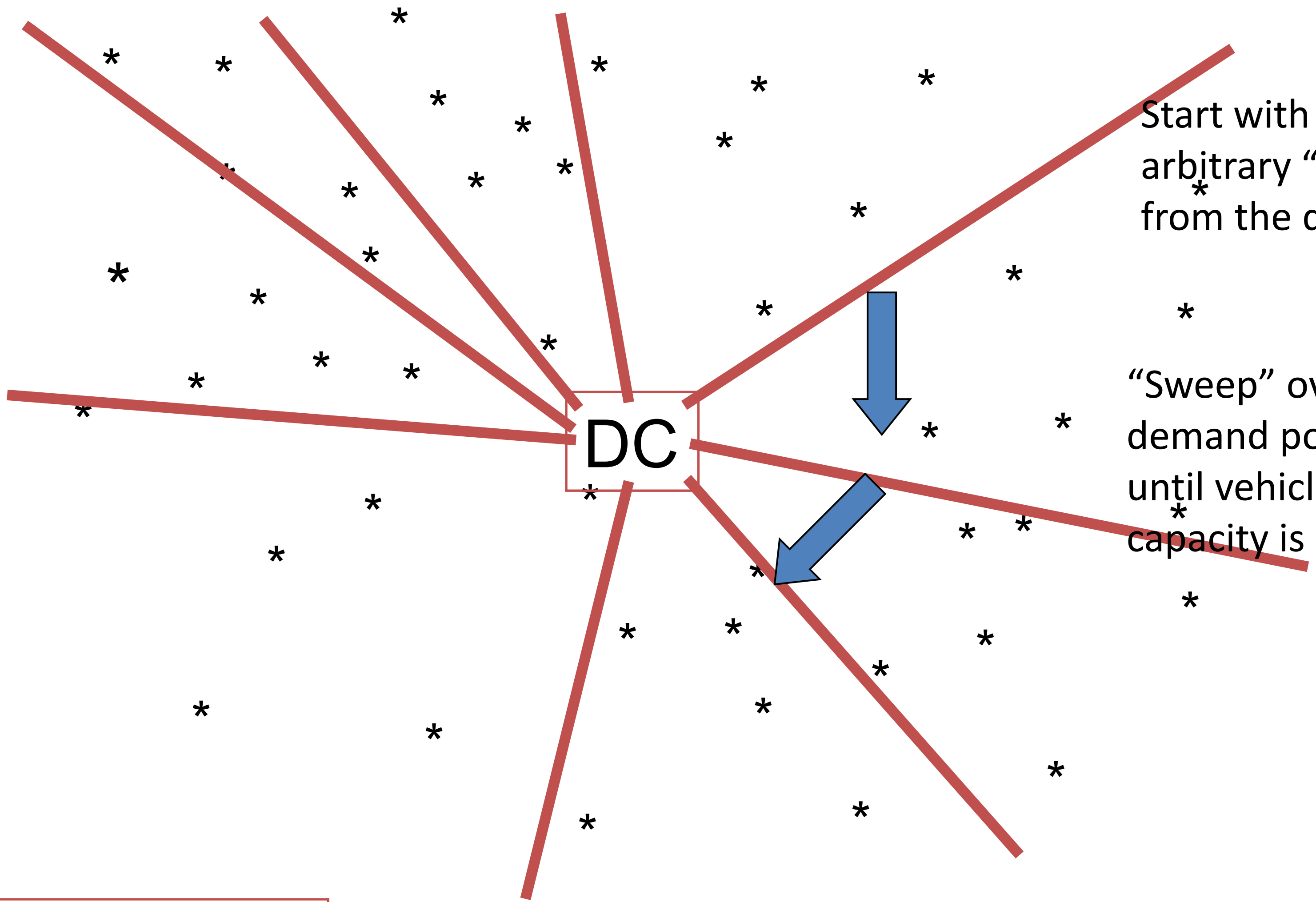
Assignment of Customers to Sectors



If we used fixed sectors, then a vehicle responsible for one sector may exceed capacity

DC = Distribution Center

Sweep Algorithm

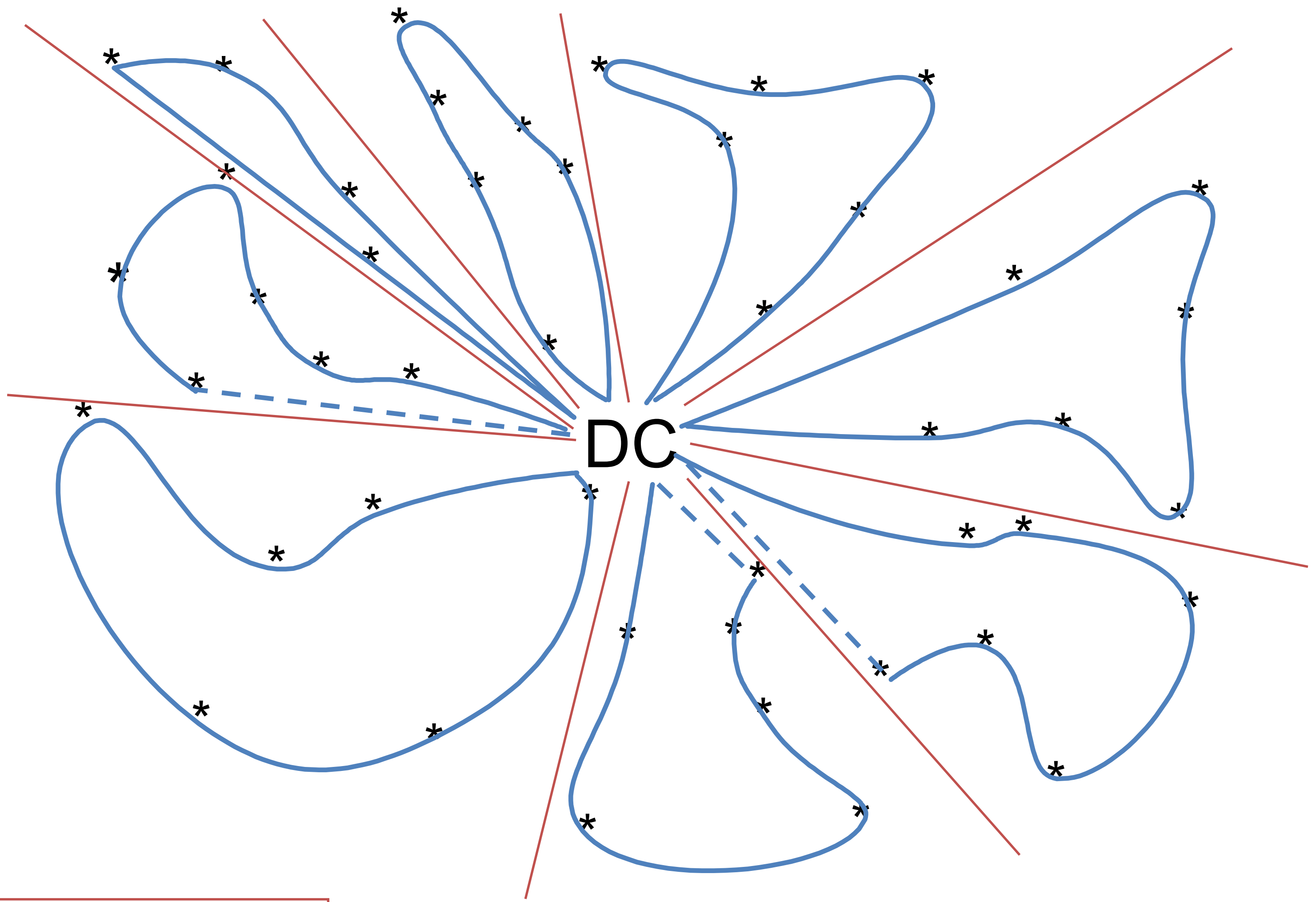


Start with an arbitrary "ray" from the depot

"Sweep" over demand points until vehicle capacity is met

DC = Distribution Center

Routing of Individual Vehicles



DC = Distribution Center