An Introduction to Algorithms

MIT Center for
Transportation & Logistics

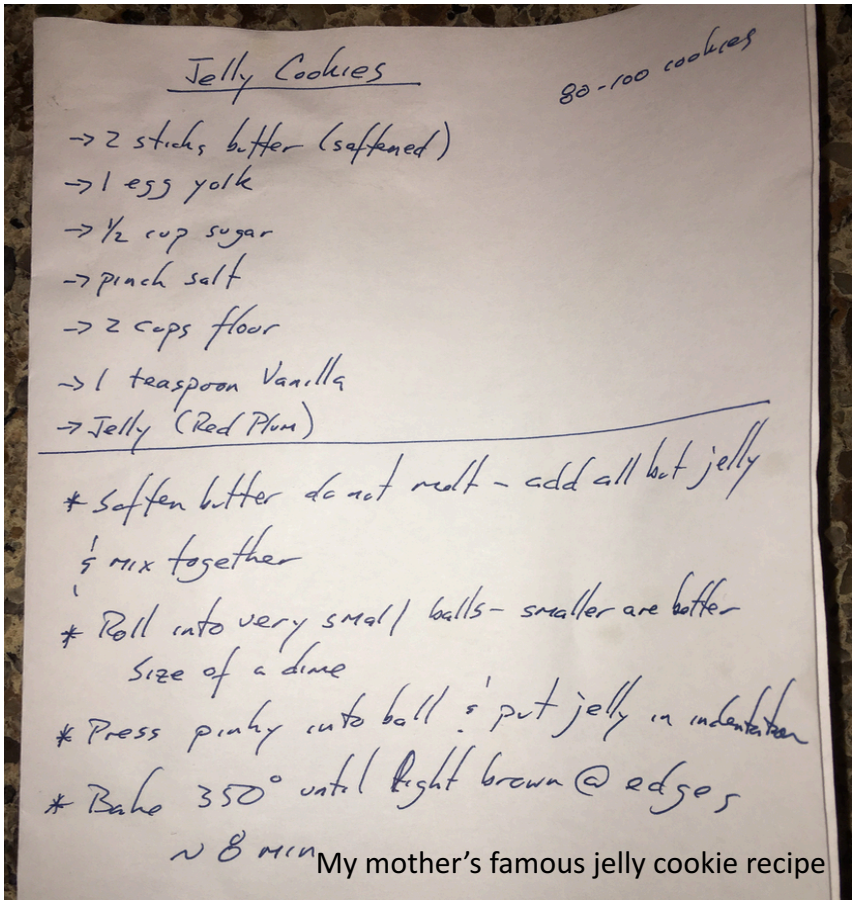
ctl.mit.edu

al·go·rithm ('algə,riTHəm)

a process or set of rules to be followed in calculations or other problem-solving operations, especially by a computer.

```
1  function Dijkstra(Graph, source):
2
3      create vertex set Q
4
5      for each vertex v in Graph:
6          dist[v] ← INFINITY
7          prev[v] ← UNDEFINED
8          add v to Q
9
10     dist[source] ← 0
11
12     while Q is not empty:
13         u ← vertex in Q with min dist[u]
14         remove u from Q
15
16         for each neighbor v of u:
17             alt ← dist[u] + length(u, v)
18             if alt < dist[v]:
19                 dist[v] ← alt
20                 prev[v] ← u
21
22     return dist[], prev[]
```

Psuedo code for Dijkstra's Algorithm for shortest path.



My mother's famous jelly cookie recipe

Desired Properties of an Algorithm

- should be unambiguous,
- require a defined set of inputs,
- produce a defined set of outputs, and
- should terminate and produce a result, always stopping after a finite time.

Algorithm Example: find_max

- Inputs:
  - L = array of N integer variables
  - v(i) = value of the i^{th} variable in the list
- Algorithm:
  1. set max = 0 and i = 1
  2. select item i in the list
  3. if v(i)>max, then set max=v(i)
  4. if i<N, then set i = i+1 and go to step 2, otherwise go to step 5
  5. end
- Output:
  - maximum value in array L (max)

L = [4, 2, 18, 7, 6]
N = 5
v(1)=4, v(2)=2, etc.

i	v()	max
1	4	0
1	4	4
2	2	4
3	18	18
4	7	18
5	6	18

max = 18.

Evaluation of find_max()

- Is it unambiguous?
  - Yes. Each step is quite simple and straightforward.
- Does it have defined inputs and outputs?
  - Yes.
- Is it guaranteed to terminate?
  - Yes. The list L is of finite length, so after looking at every element of the list the algorithm will stop.
- Does it produce the correct result?
  - Yes.

Agenda for Lesson

- Explore different algorithms - focus on networks
  - Shortest Path Problem
    - Dijkstra's Algorithm
  - Traveling Salesman Problem
    - Nearest Neighbor (construction)
    - 2-Opt (improvement)
  - Vehicle Routing Problem
    - Sweep Heuristic
    - Clark-Wright or Savings Algorithm
    - Mixed Integer Linear Program (MILP)

Shortest Path Problem

Shortest Path Problem (SPP)

- What is the objective?
  - Find the shortest path in a network between two nodes
- Why is it important?
  - Result is used as base for other analysis
  - Connects physical to operational network
- What are the primary approaches?
  - Standard Linear Programming (LP)
  - Specialized Algorithms (Dijkstra's Algorithm)

Minimize: $\sum_i \sum_j c_{ij} x_{ij}$

Subject to:

$$\sum_i x_{ji} = 1 \qquad \forall\ j = s$$

$$\sum_i x_{ji} - \sum_i x_{ij} = 0 \qquad \forall\ j \neq s, j \neq t$$

$$\sum_i x_{ij} = 1 \qquad \forall\ j = t$$

$$x_{ij} \geq 0$$

where:

x_{ij} = Number of units flowing from node i to node j

c_{ij} = Cost per unit for flow from node i to node j

s = Source node – where flow starts

t = Terminal node – where flow ends

Dijkstra's Algorithm

L(j) = length of path from source node s to node j
P(j) = preceding node for j in the shortest path
S(j) = 1 if node j has been visited, = 0 otherwise
d(ij) = distance or cost from node i to node j

- Inputs:
  - Connected graph with nodes and arcs with positive costs, d(ij)
  - Source (s) and Terminal (t) nodes
- Algorithm:
  1. for all nodes in graph, set L()=∞, P()=Null, S()=0
  2. set s to i, S(i)=1, and L(i)=0
  3. For all nodes, j, directly connected (adjacent) to node i;
     if L(j) > L(i) + d(ij), then set L(j) = L(i) + d(ij) and P(j)=i
  4. For all nodes where S()=0,
     select the node with lowest L() and set it to i, set S(i)=1
  5. Is this node t, the terminal node? If so, go to end
     If not, go to step 3
  6. end – return L(t)
- Output:
  - L(t) and P array

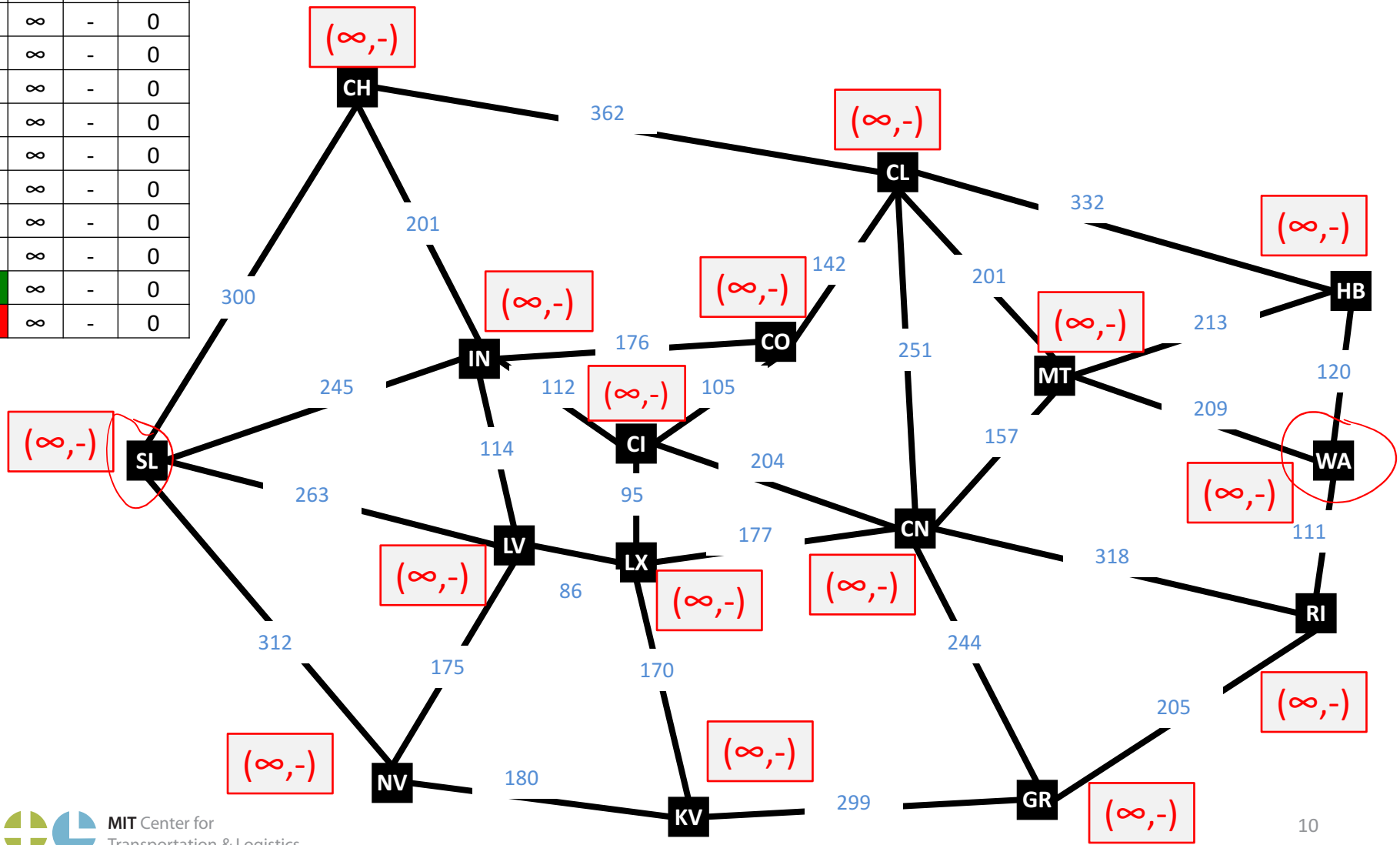
To find path from s to t, start at the end.
- Find P(t) – say it is j
- If j=source node, stop, otherwise, find P(j)
- keep tracing preceding nodes until you reach source node

Example: Dijkstra's Algorithm

Shortest Path (SL to WA)

initialize all nodes

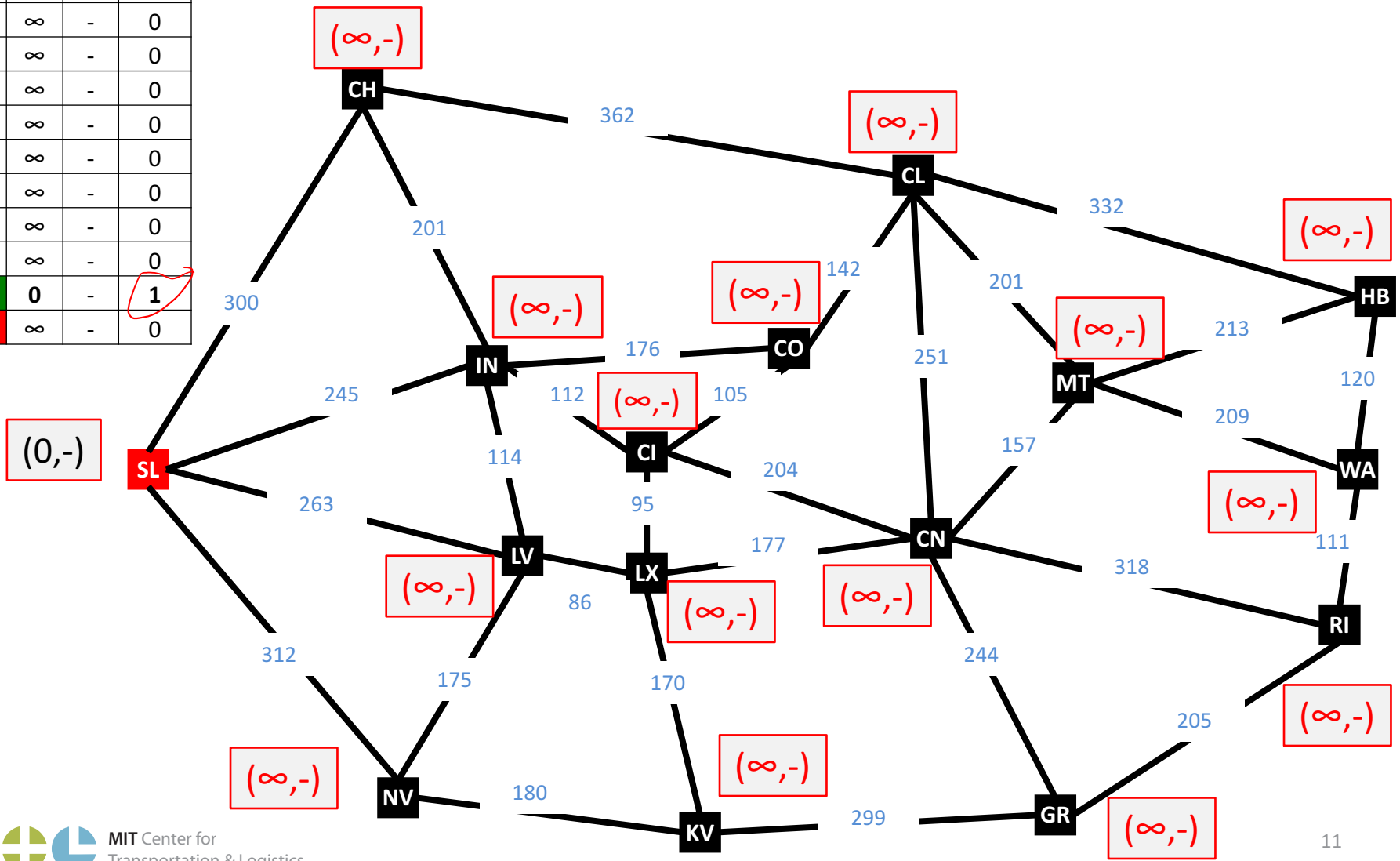
	L()	P()	S()
CH	∞	-	0
CI	∞	-	0
CL	∞	-	0
CN	∞	-	0
CO	∞	-	0
GR	∞	-	0
HB	∞	-	0
IN	∞	-	0
KV	∞	-	0
LV	∞	-	0
LX	∞	-	0
MT	∞	-	0
NV	∞	-	0
RI	∞	-	0
SL	∞	-	0
WA	∞	-	0



Shortest Path (SL to WA)

select SL as active node

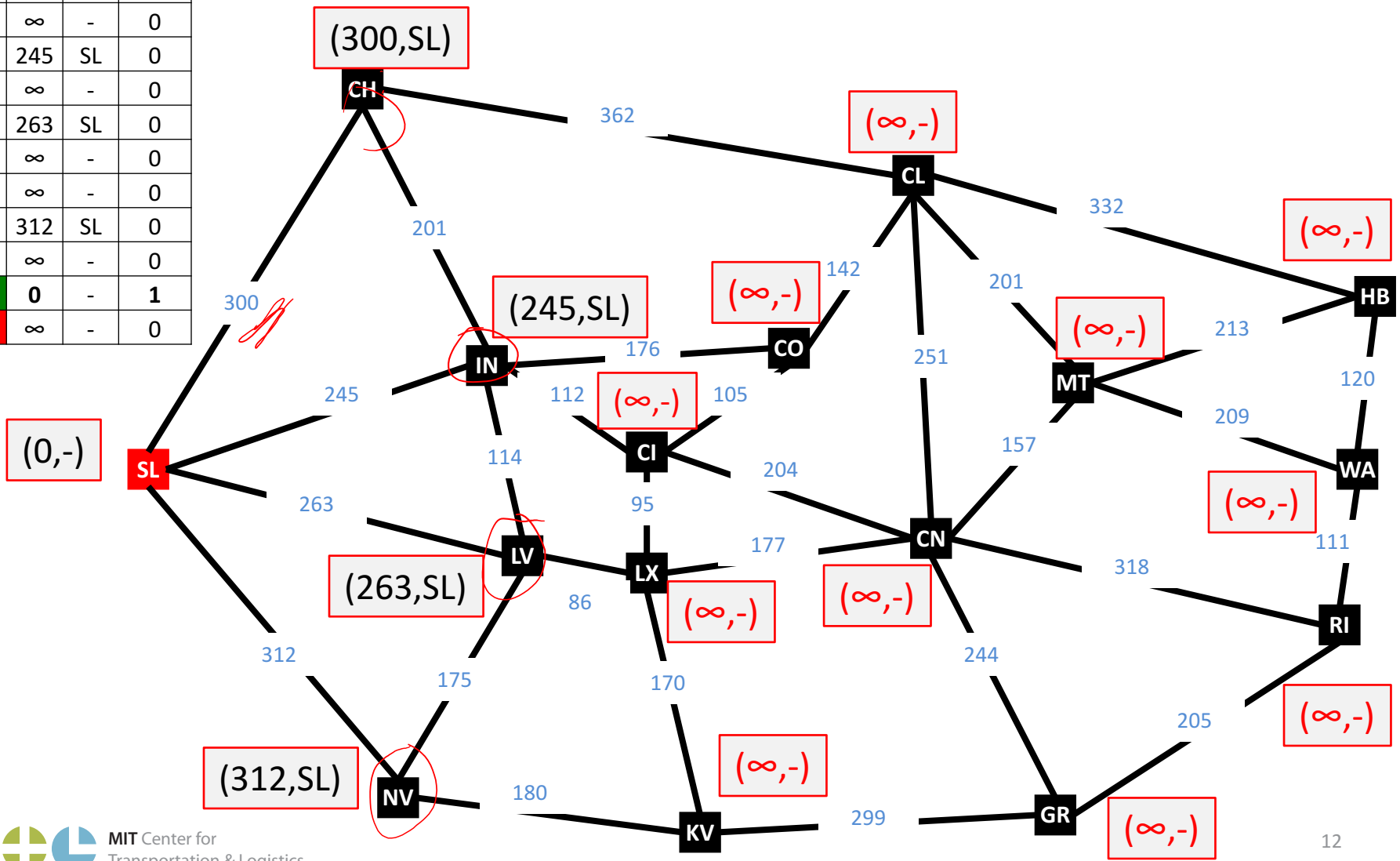
	L()	P()	S()
CH	∞	-	0
CI	∞	-	0
CL	∞	-	0
CN	∞	-	0
CO	∞	-	0
GR	∞	-	0
HB	∞	-	0
IN	∞	-	0
KV	∞	-	0
LV	∞	-	0
LX	∞	-	0
MT	∞	-	0
NV	∞	-	0
RI	∞	-	0
SL	**0**	-	**1**
WA	∞	-	0



Shortest Path (SL to WA)

adjust all adjacent nodes

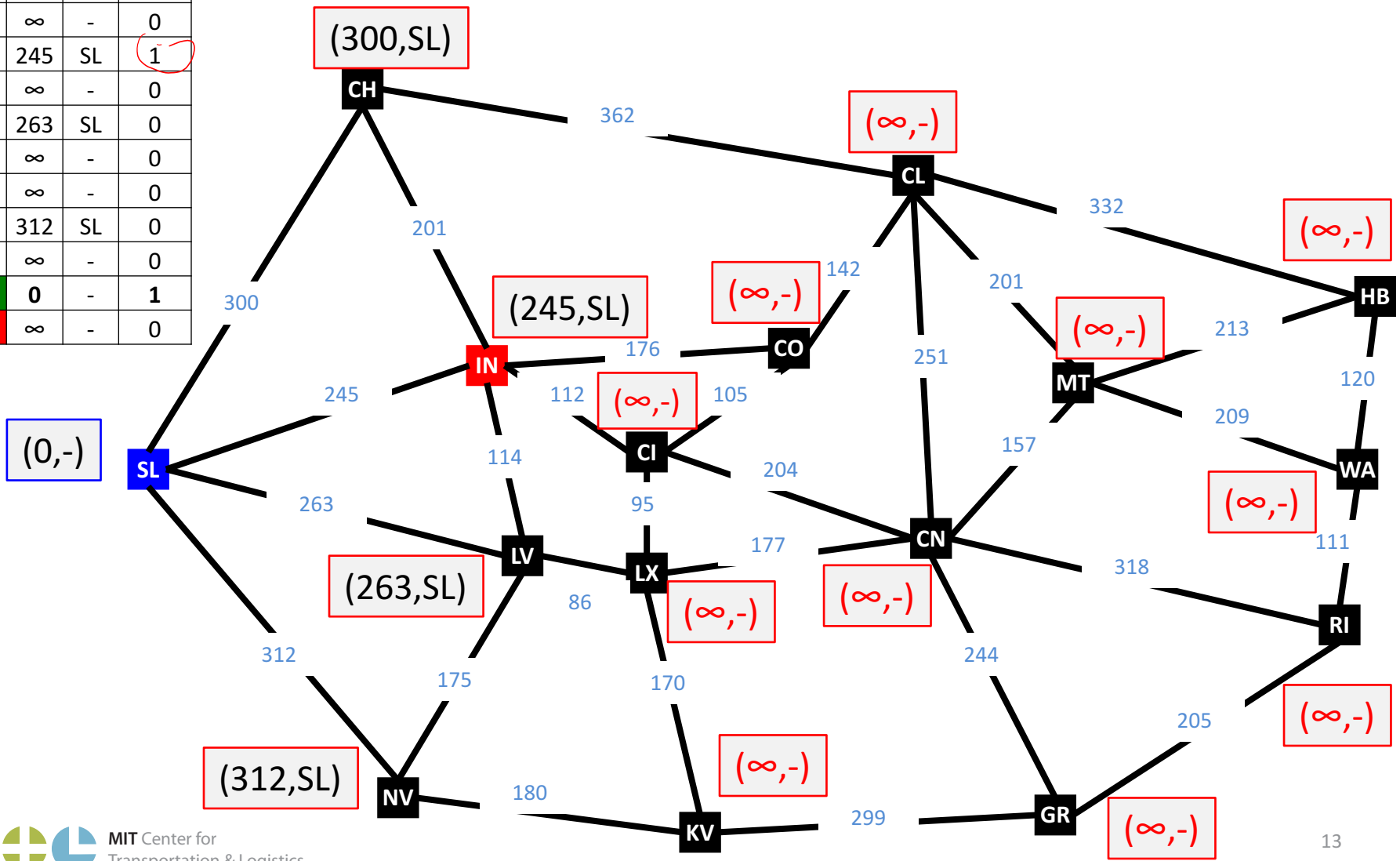
	L()	P()	S()
CH	300	SL	0
CI	∞	-	0
CL	∞	-	0
CN	∞	-	0
CO	∞	-	0
GR	∞	-	0
HB	∞	-	0
IN	245	SL	0
KV	∞	-	0
LV	263	SL	0
LX	∞	-	0
MT	∞	-	0
NV	312	SL	0
RI	∞	-	0
SL	**0**	-	**1**
WA	∞	-	0



Shortest Path (SL to WA)

select IN as active node

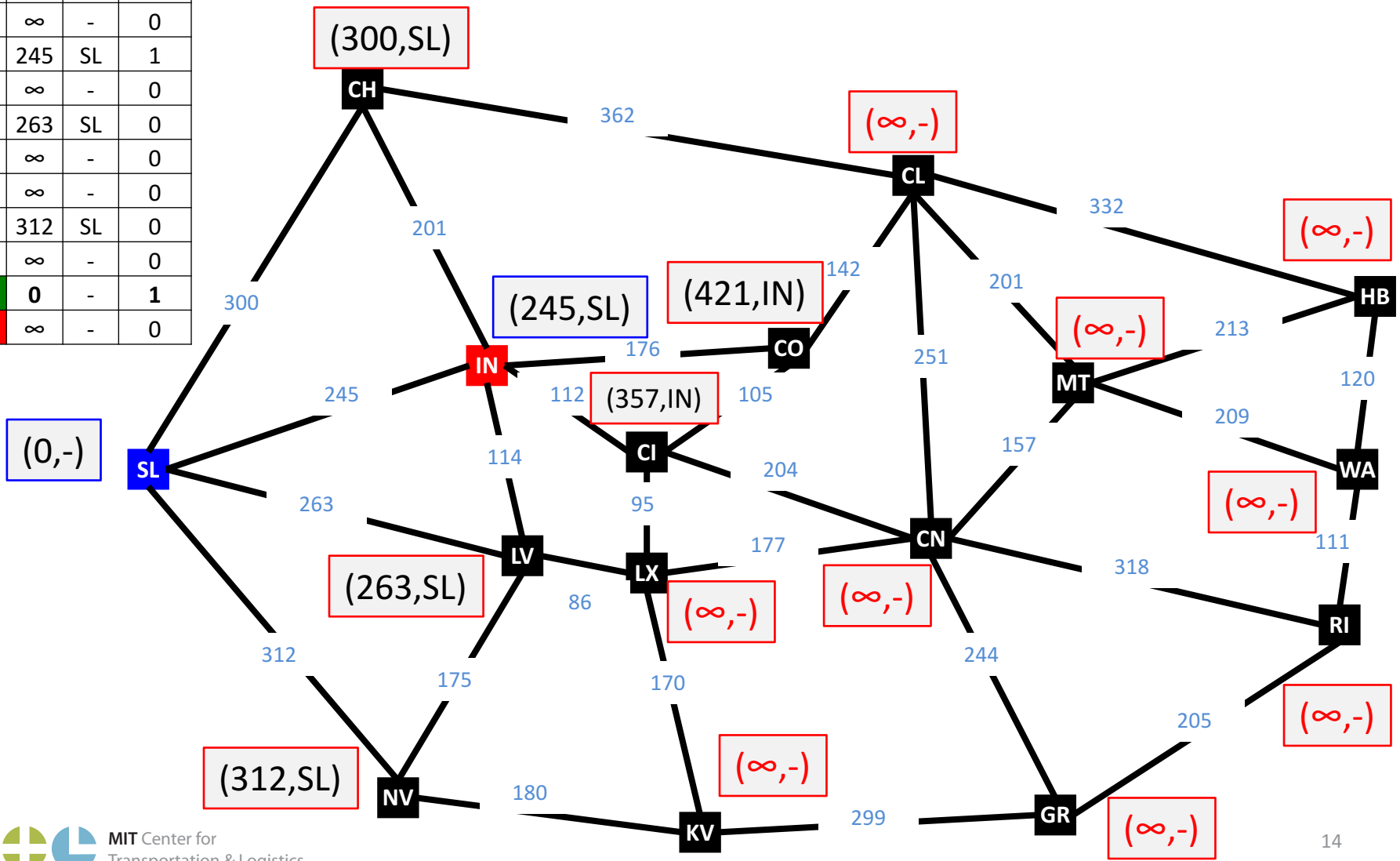
	L()	P()	S()
CH	300	SL	0
CI	∞	-	0
CL	∞	-	0
CN	∞	-	0
CO	∞	-	0
GR	∞	-	0
HB	∞	-	0
IN	245	SL	1
KV	∞	-	0
LV	263	SL	0
LX	∞	-	0
MT	∞	-	0
NV	312	SL	0
RI	∞	-	0
SL	**0**	-	**1**
WA	∞	-	0



Shortest Path (SL to WA)

adjust adjacent nodes to IN

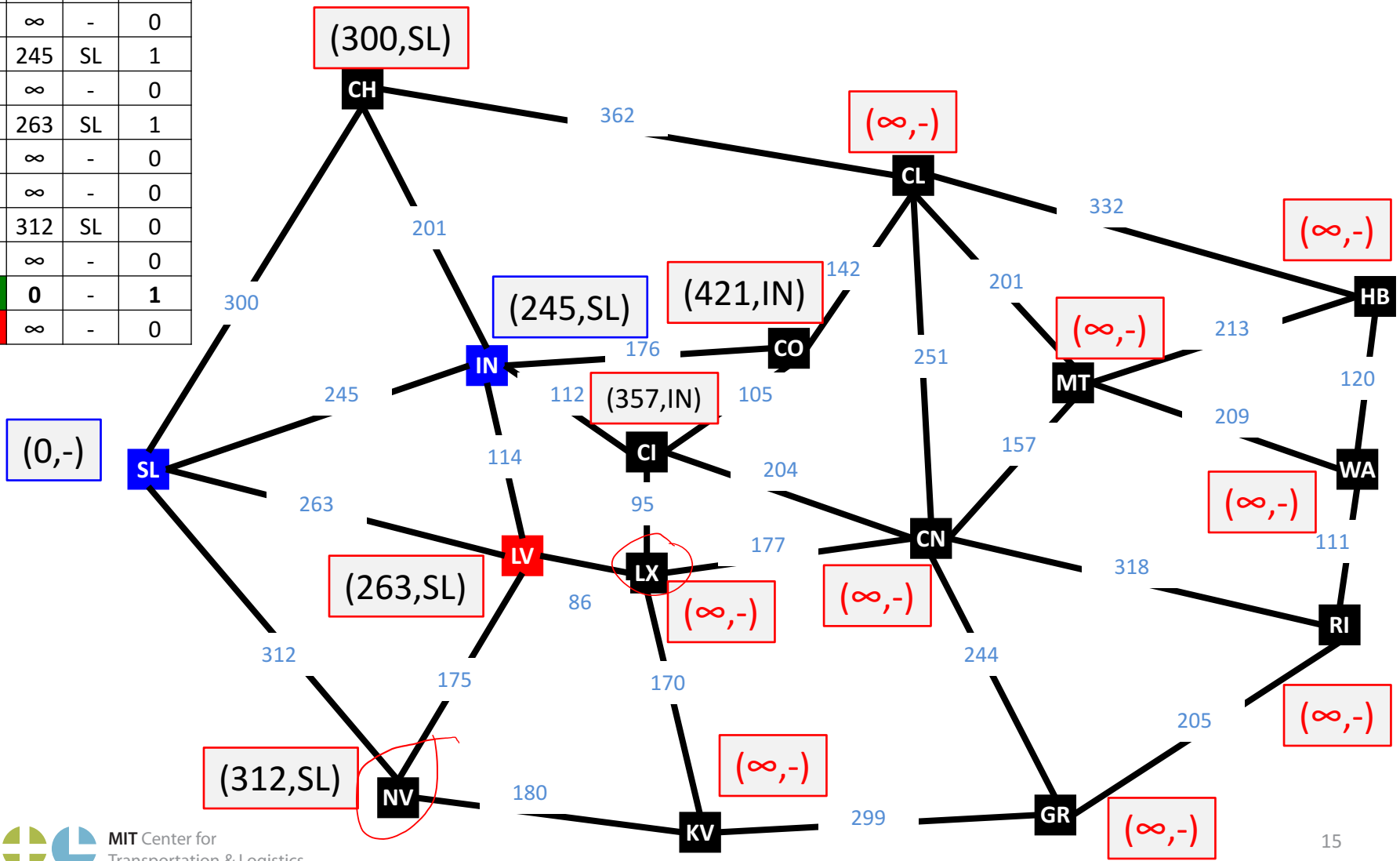
	L()	P()	S()
CH	300	SL	0
CI	357	IN	0
CL	∞	-	0
CN	∞	-	0
CO	421	IN	0
GR	∞	-	0
HB	∞	-	0
IN	245	SL	1
KV	∞	-	0
LV	263	SL	0
LX	∞	-	0
MT	∞	-	0
NV	312	SL	0
RI	∞	-	0
SL	**0**	-	**1**
WA	∞	-	0



Shortest Path (SL to WA)

select LV as active node

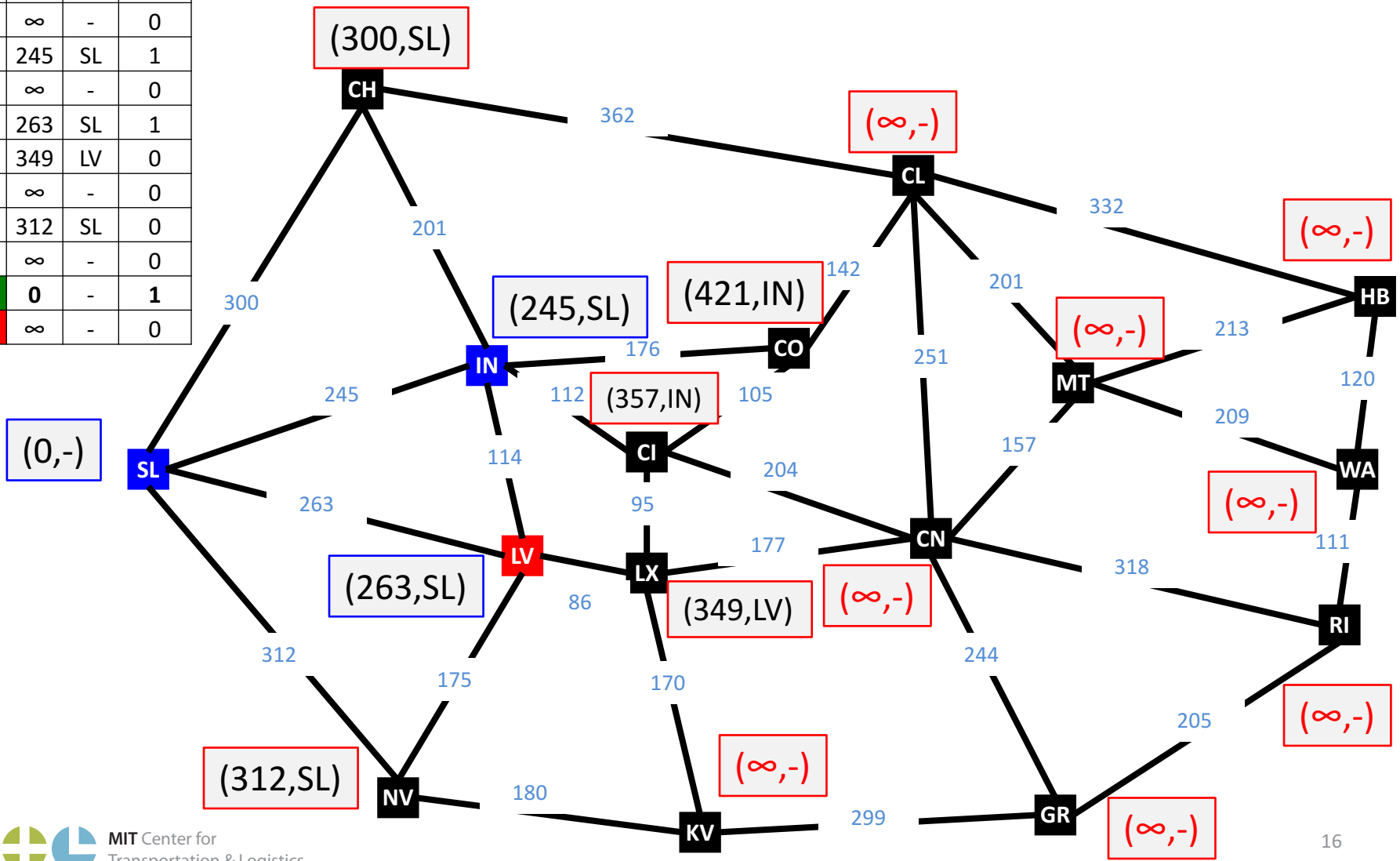
	L()	P()	S()
CH	300	SL	0
CI	357	IN	0
CL	∞	-	0
CN	∞	-	0
CO	421	IN	1
GR	∞	-	0
HB	∞	-	0
IN	245	SL	1
KV	∞	-	0
LV	263	SL	1
LX	∞	-	0
MT	∞	-	0
NV	312	SL	0
RI	∞	-	0
SL	**0**	-	**1**
WA	∞	-	0



Shortest Path (SL to WA)

adjust adjacent nodes to LV

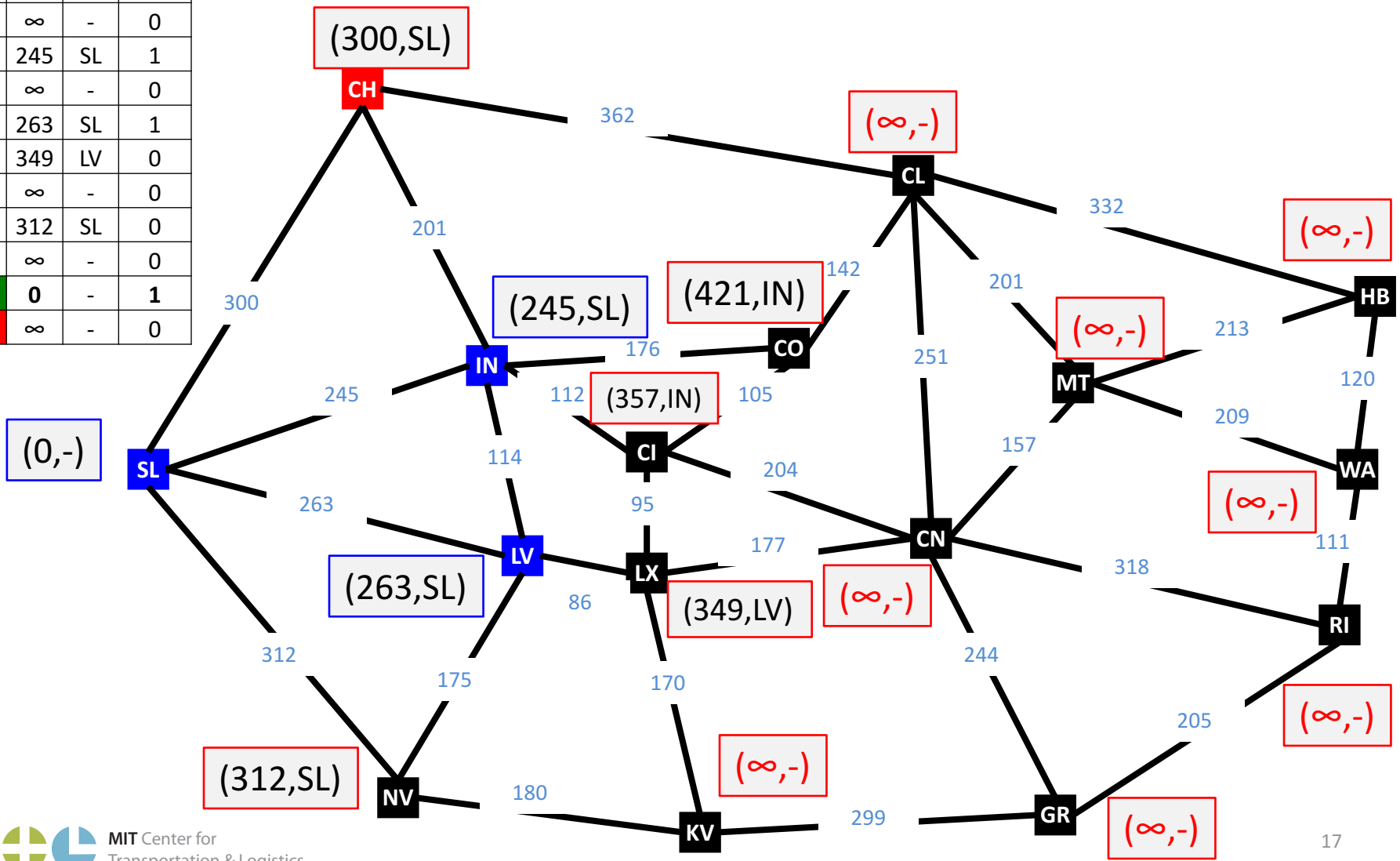
	L()	P()	S()
CH	300	SL	0
CI	357	IN	0
CL	∞	-	0
CN	∞	-	0
CO	421	IN	0
GR	∞	-	0
HB	∞	-	0
IN	245	SL	1
KV	∞	-	0
LV	263	SL	1
LX	349	LV	0
MT	∞	-	0
NV	312	SL	0
RI	∞	-	0
SL	**0**	-	**1**
WA	∞	-	0



Shortest Path (SL to WA)

select CH as active node

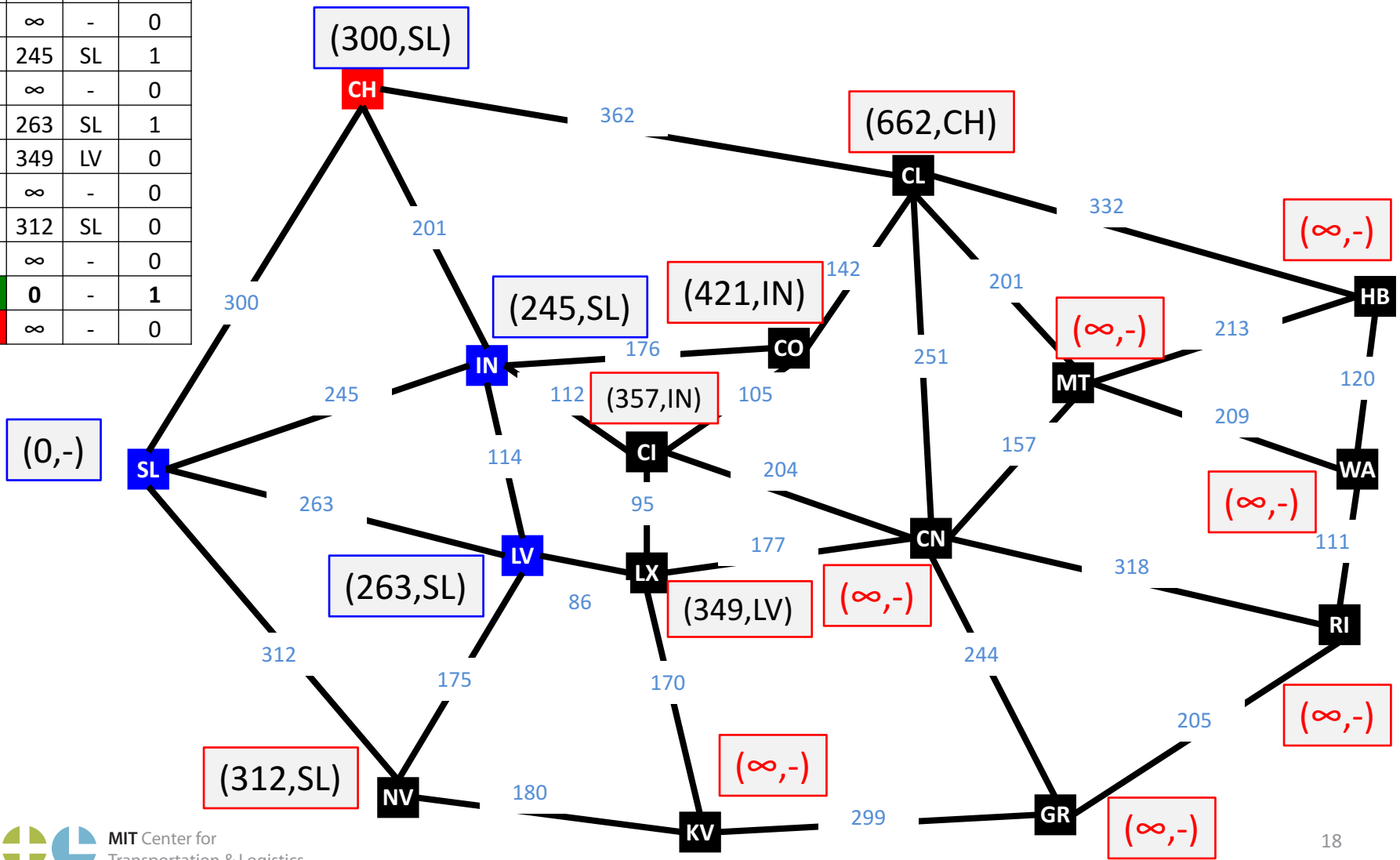
	L()	P()	S()
CH	300	SL	1
CI	357	IN	0
CL	∞	-	0
CN	∞	-	0
CO	421	IN	0
GR	∞	-	0
HB	∞	-	0
IN	245	SL	1
KV	∞	-	0
LV	263	SL	1
LX	349	LV	0
MT	∞	-	0
NV	312	SL	0
RI	∞	-	0
SL	**0**	-	**1**
WA	∞	-	0



Shortest Path (SL to WA)

adjust adjacent nodes to CH

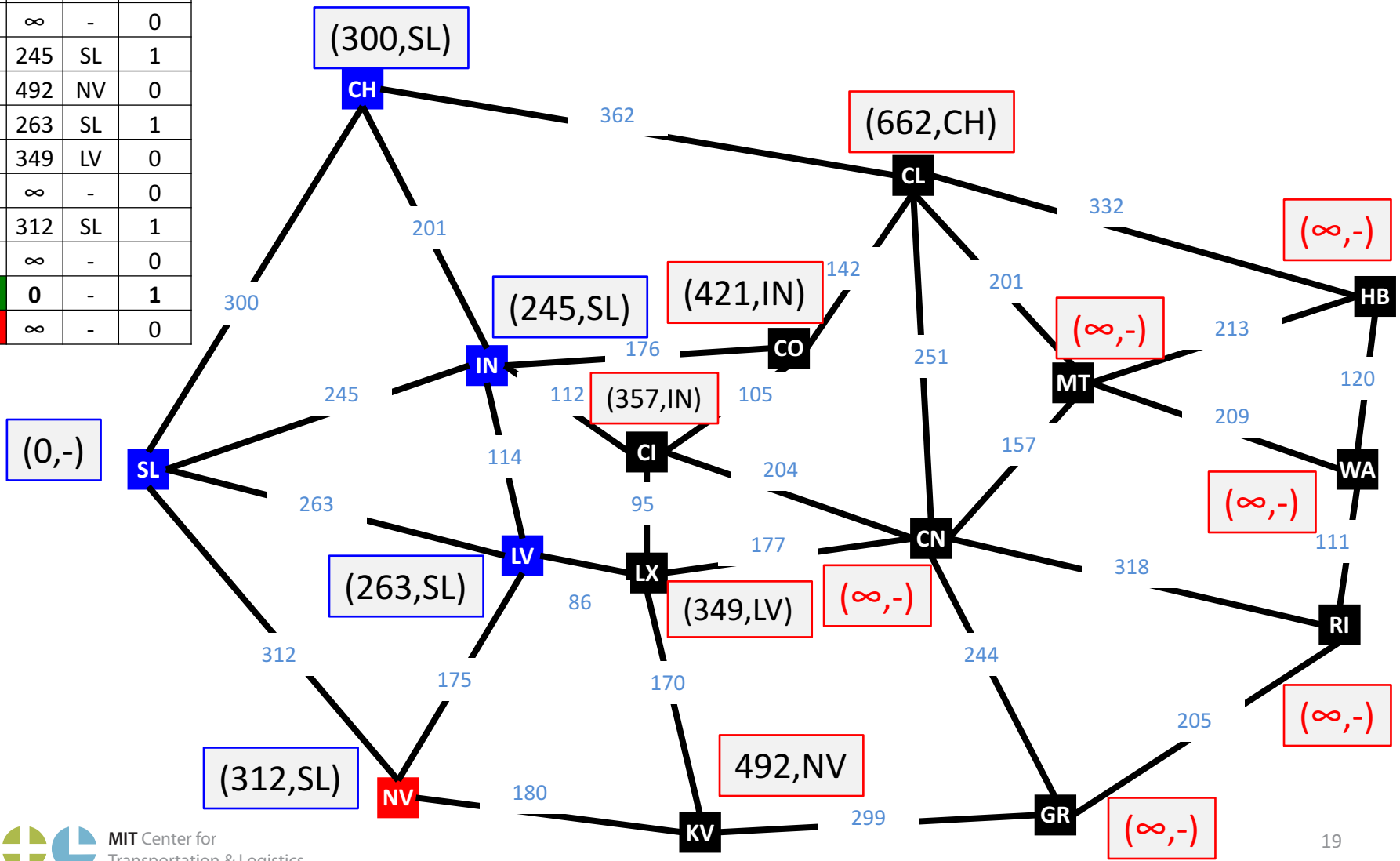
	L()	P()	S()
CH	300	SL	1
CI	357	IN	0
CL	662	CH	0
CN	∞	-	0
CO	421	IN	0
GR	∞	-	0
HB	∞	-	0
IN	245	SL	1
KV	∞	-	0
LV	263	SL	1
LX	349	LV	0
MT	∞	-	0
NV	312	SL	0
RI	∞	-	0
SL	**0**	-	**1**
WA	∞	-	0



Shortest Path (SL to WA)

	L()	P()	S()
CH	300	SL	1
CI	357	IN	0
CL	662	CH	0
CN	∞	-	0
CO	421	IN	0
GR	∞	-	0
HB	∞	-	0
IN	245	SL	1
KV	492	NV	0
LV	263	SL	1
LX	349	LV	0
MT	∞	-	0
NV	312	SL	1
RI	∞	-	0
SL	**0**	-	**1**
WA	∞	-	0

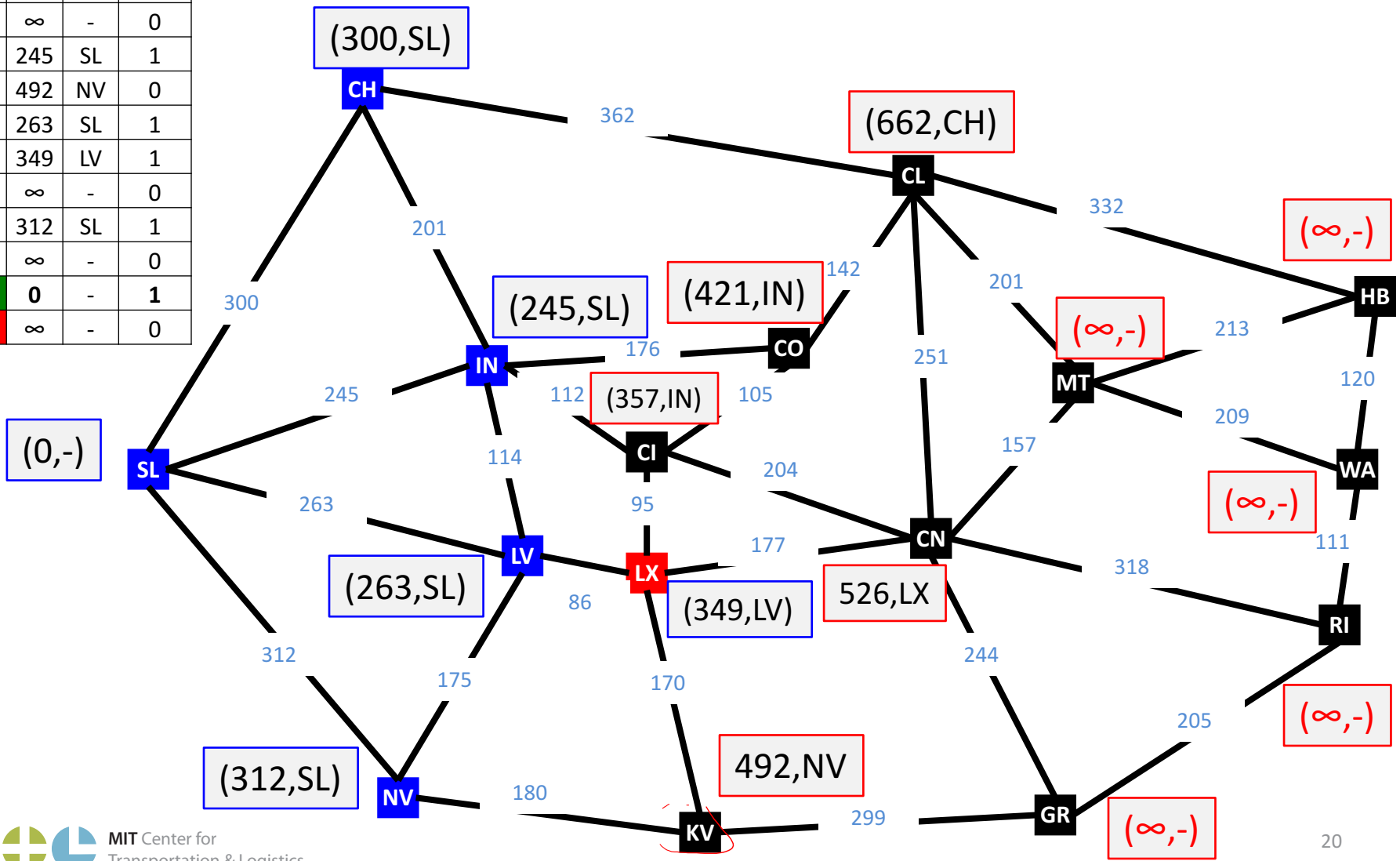
select NV as active node
update adjacent nodes



Shortest Path (SL to WA)

	L()	P()	S()
CH	300	SL	1
CI	357	IN	0
CL	662	CH	0
CN	526	LX	0
CO	421	IN	0
GR	∞	-	0
HB	∞	-	0
IN	245	SL	1
KV	492	NV	0
LV	263	SL	1
LX	349	LV	1
MT	∞	-	0
NV	312	SL	1
RI	∞	-	0
SL	**0**	-	**1**
WA	∞	-	0

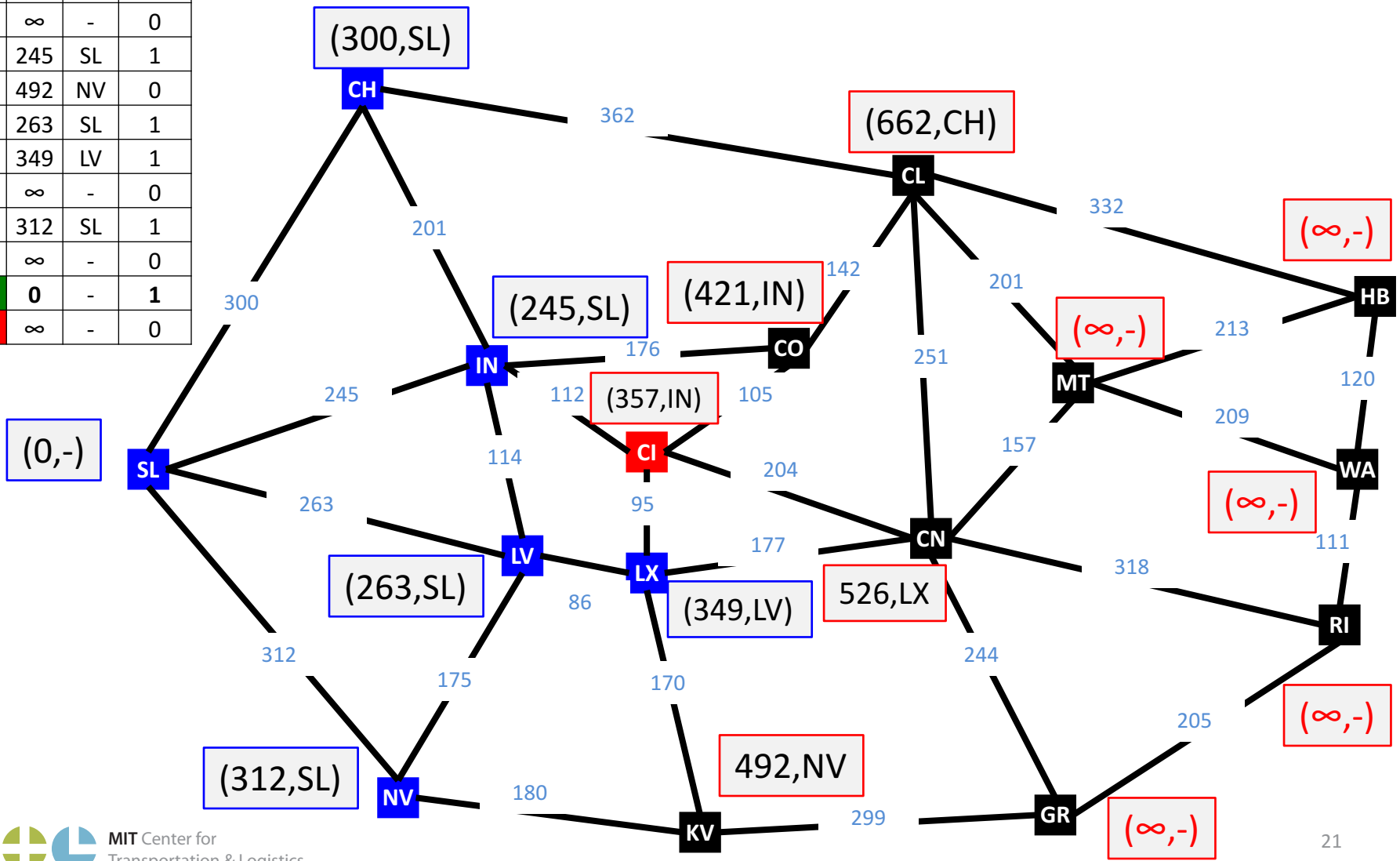
select LX as active node
update adjacent nodes



Shortest Path (SL to WA)

select CI as active node
update adjacent nodes

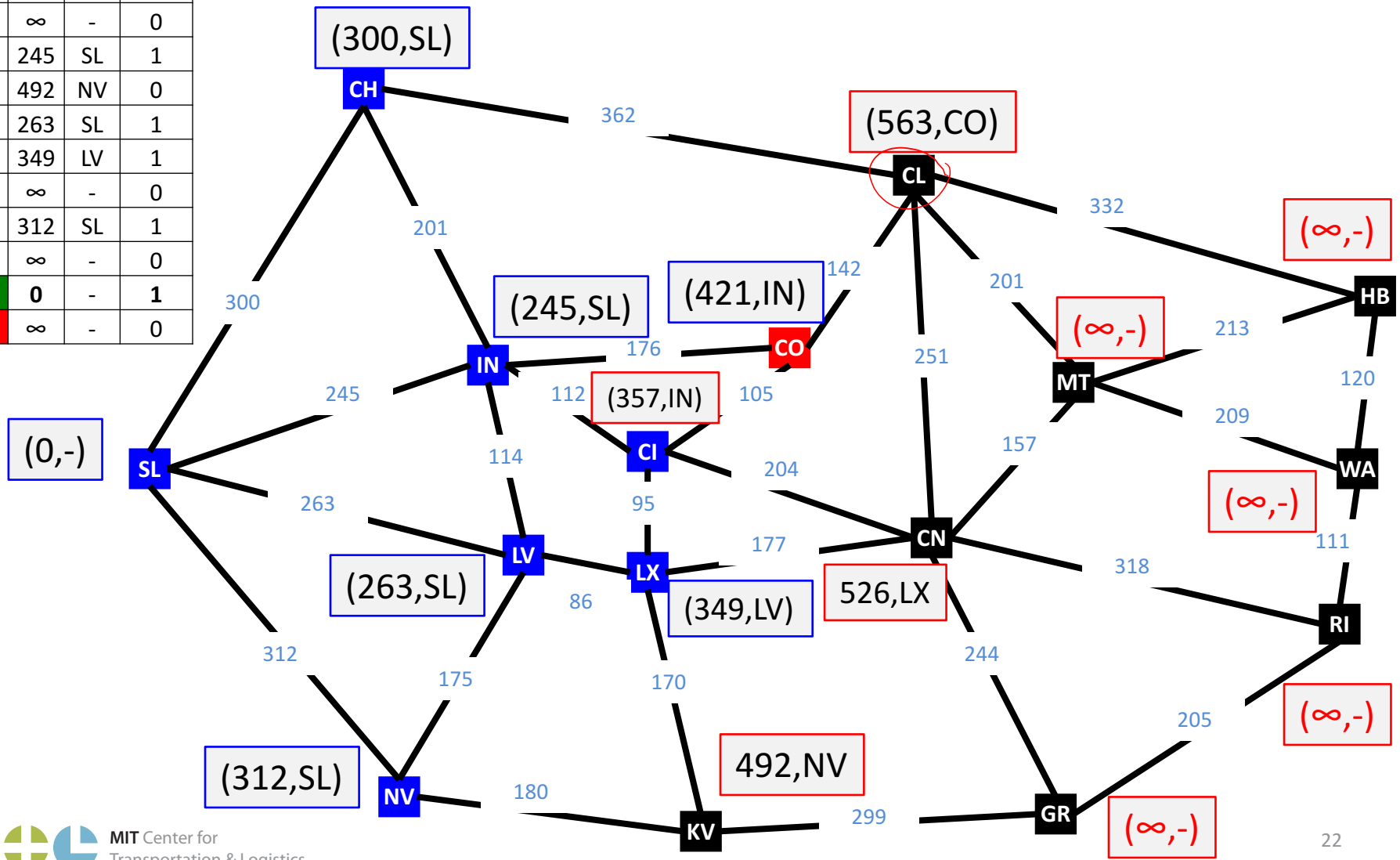
	L()	P()	S()
CH	300	SL	1
CI	357	IN	1
CL	662	CH	0
CN	526	LX	0
CO	421	IN	0
GR	∞	-	0
HB	∞	-	0
IN	245	SL	1
KV	492	NV	0
LV	263	SL	1
LX	349	LV	1
MT	∞	-	0
NV	312	SL	1
RI	∞	-	0
SL	**0**	-	**1**
WA	∞	-	0



Shortest Path (SL to WA)

	L()	P()	S()
CH	300	SL	1
CI	357	IN	1
CL	563	CL	0
CN	526	LX	0
CO	421	IN	1
GR	∞	-	0
HB	∞	-	0
IN	245	SL	1
KV	492	NV	0
LV	263	SL	1
LX	349	LV	1
MT	∞	-	0
NV	312	SL	1
RI	∞	-	0
SL	**0**	-	**1**
WA	∞	-	0

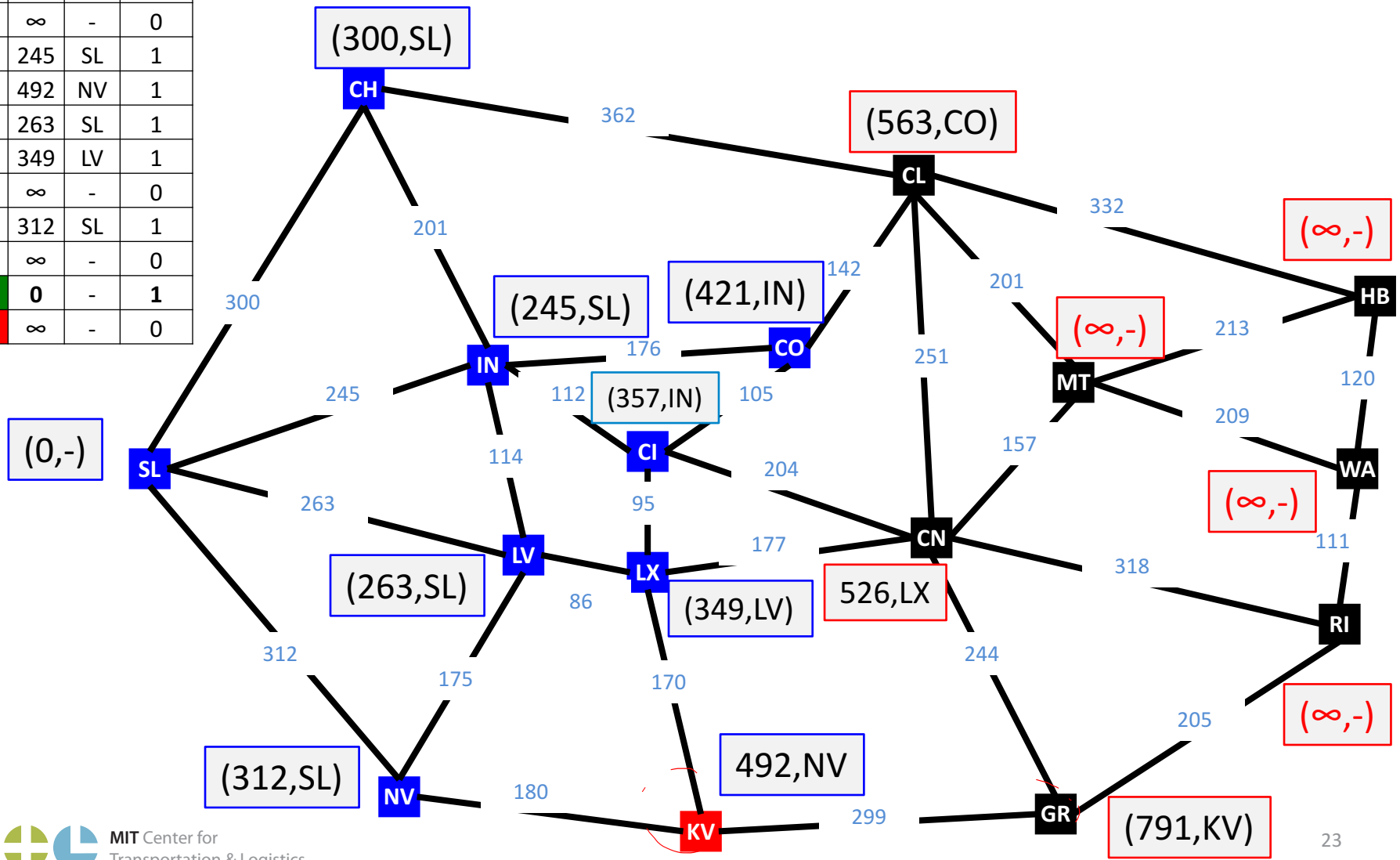
select CO as active node
update adjacent nodes



Shortest Path (SL to WA)

select KV as active node
update adjacent nodes

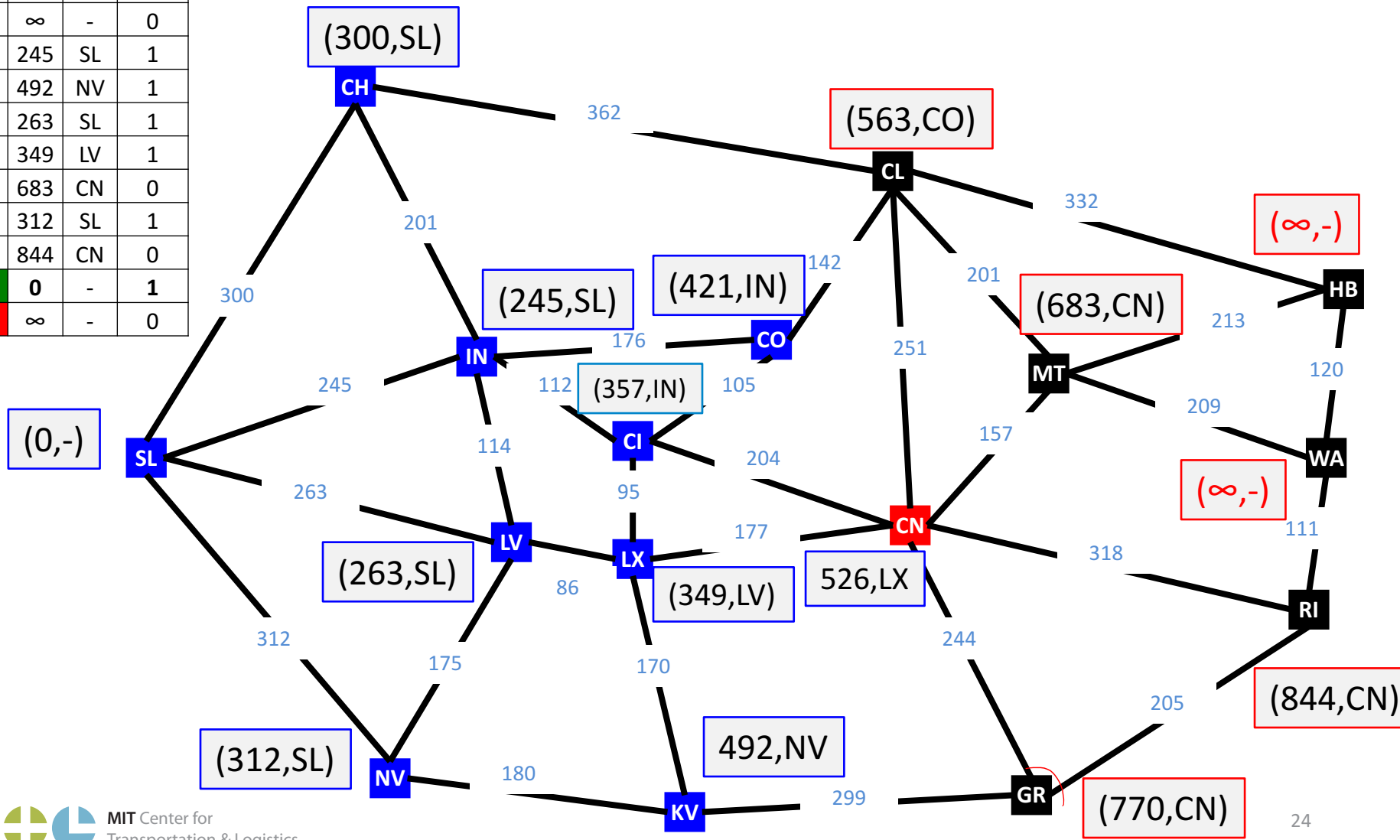
	L()	P()	S()
CH	300	SL	1
CI	357	IN	1
CL	563	CL	0
CN	526	LX	0
CO	421	IN	1
GR	791	KV	0
HB	∞	-	0
IN	245	SL	1
KV	492	NV	1
LV	263	SL	1
LX	349	LV	1
MT	∞	-	0
NV	312	SL	1
RI	∞	-	0
SL	**0**	-	**1**
WA	∞	-	0



Shortest Path (SL to WA)

	L()	P()	S()
CH	300	SL	1
CI	357	IN	1
CL	563	CL	0
CN	526	LX	1
CO	421	IN	1
GR	770	CN	0
HB	∞	-	0
IN	245	SL	1
KV	492	NV	1
LV	263	SL	1
LX	349	LV	1
MT	683	CN	0
NV	312	SL	1
RI	844	CN	0
SL	**0**	-	**1**
WA	∞	-	0

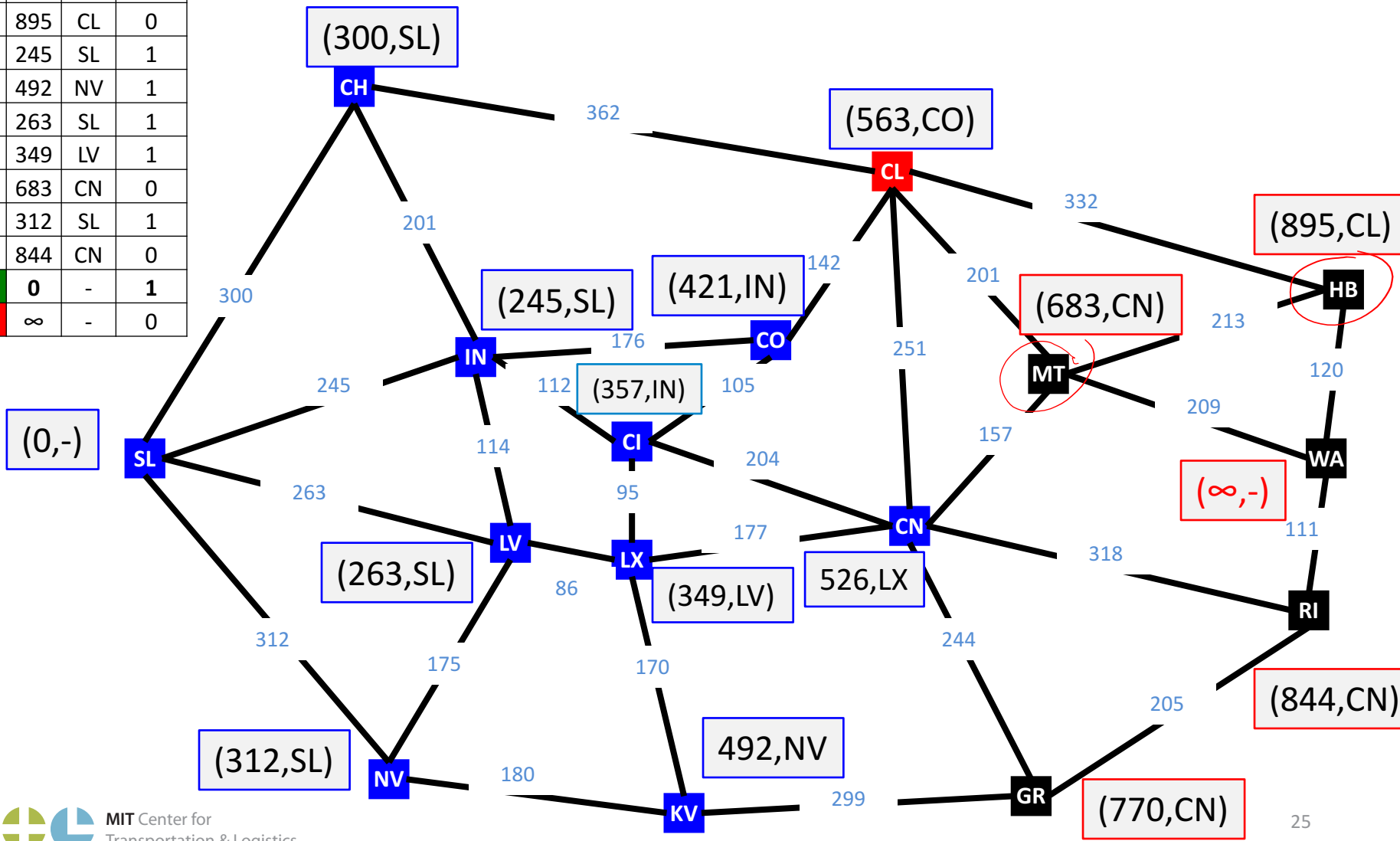
select CN as active node
update adjacent nodes



Shortest Path (SL to WA)

	L()	P()	S()
CH	300	SL	1
CI	357	IN	1
CL	563	CL	1
CN	526	LX	1
CO	421	IN	1
GR	770	CN	0
HB	895	CL	0
IN	245	SL	1
KV	492	NV	1
LV	263	SL	1
LX	349	LV	1
MT	683	CN	0
NV	312	SL	1
RI	844	CN	0
SL	**0**	-	**1**
WA	∞	-	0

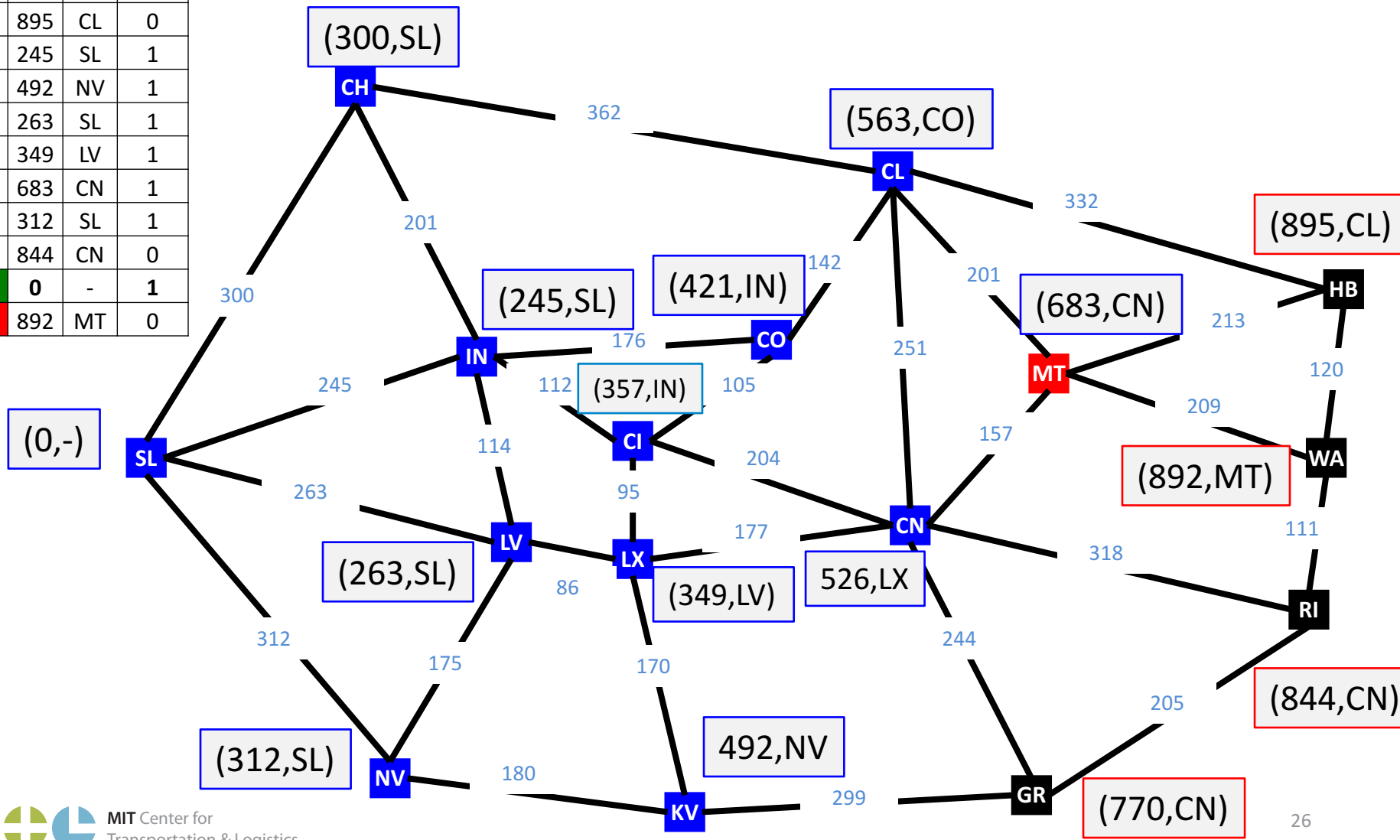
select CL as active node
update adjacent nodes



Shortest Path (SL to WA)

	L()	P()	S()
CH	300	SL	1
CI	357	IN	1
CL	563	CL	1
CN	526	LX	1
CO	421	IN	1
GR	770	CN	0
HB	895	CL	0
IN	245	SL	1
KV	492	NV	1
LV	263	SL	1
LX	349	LV	1
MT	683	CN	1
NV	312	SL	1
RI	844	CN	0
SL	**0**	-	**1**
WA	892	MT	0

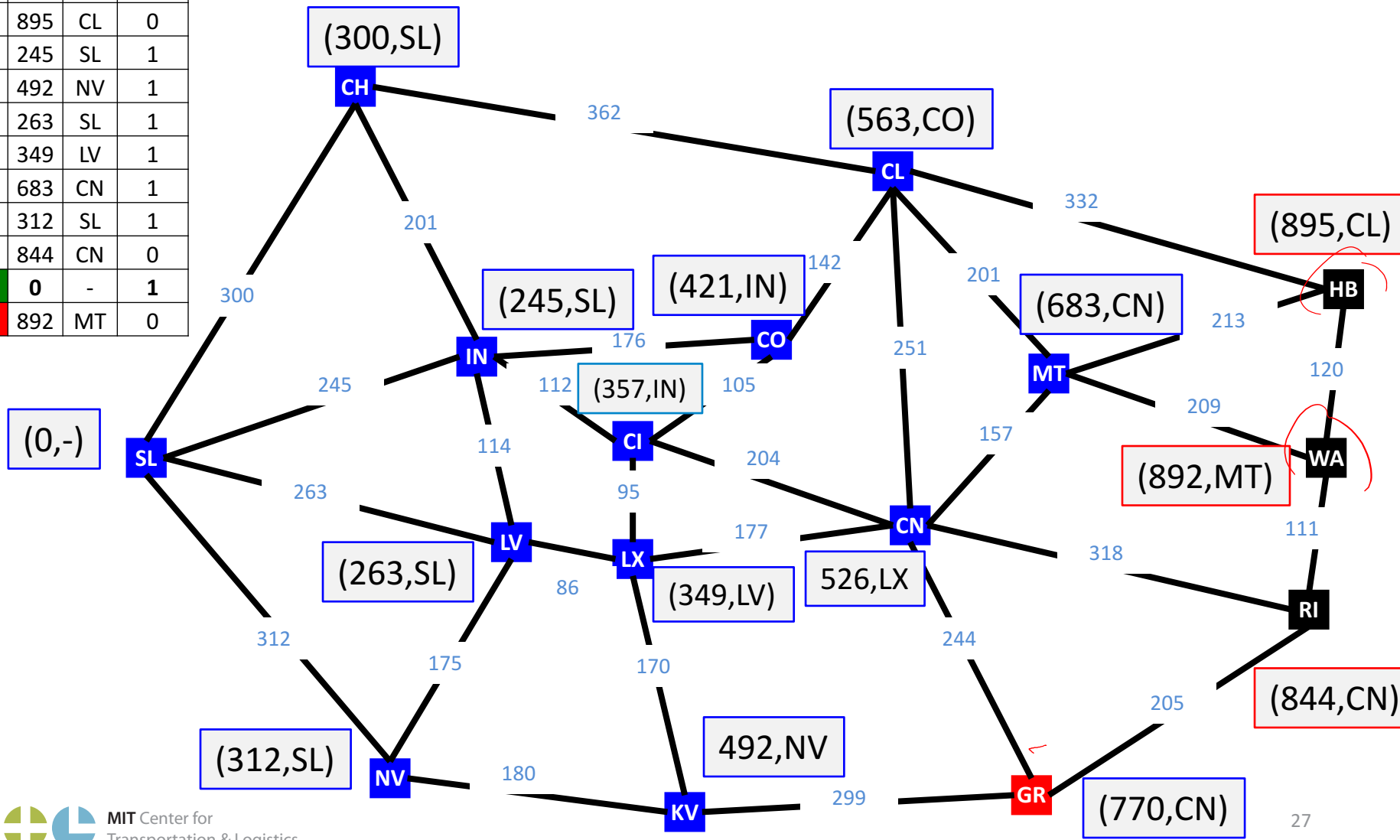
select MT as active node
update adjacent nodes



Shortest Path (SL to WA)

	L()	P()	S()
CH	300	SL	1
CI	357	IN	1
CL	563	CL	1
CN	526	LX	1
CO	421	IN	1
GR	770	CN	1
HB	895	CL	0
IN	245	SL	1
KV	492	NV	1
LV	263	SL	1
LX	349	LV	1
MT	683	CN	1
NV	312	SL	1
RI	844	CN	0
SL	**0**	-	**1**
WA	892	MT	0

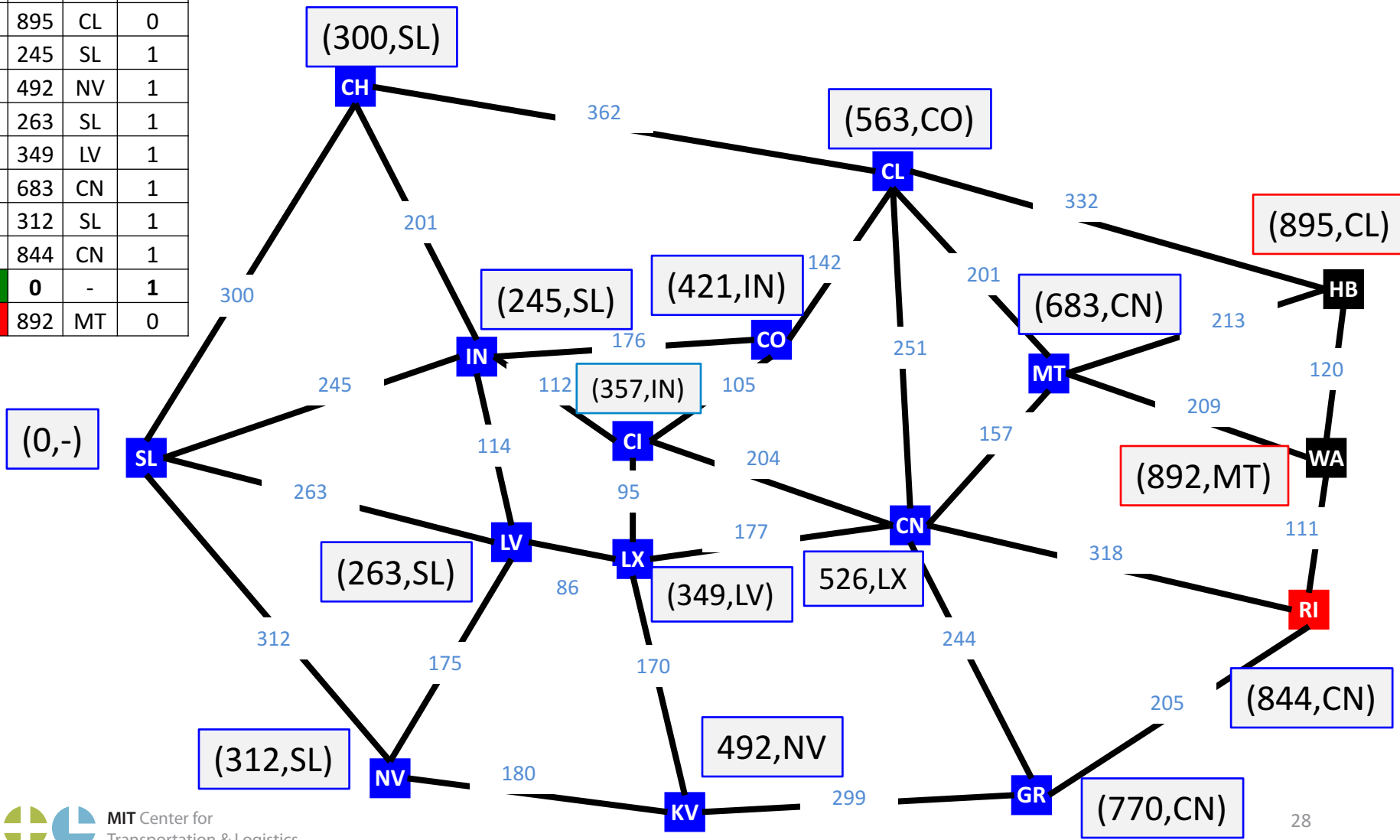
select GR as active node
update adjacent nodes



Shortest Path (SL to WA)

	L()	P()	S()
CH	300	SL	1
CI	357	IN	1
CL	563	CL	1
CN	526	LX	1
CO	421	IN	1
GR	770	CN	1
HB	895	CL	0
IN	245	SL	1
KV	492	NV	1
LV	263	SL	1
LX	349	LV	1
MT	683	CN	1
NV	312	SL	1
RI	844	CN	1
SL	**0**	-	**1**
WA	892	MT	0

select RI as active node
update adjacent nodes

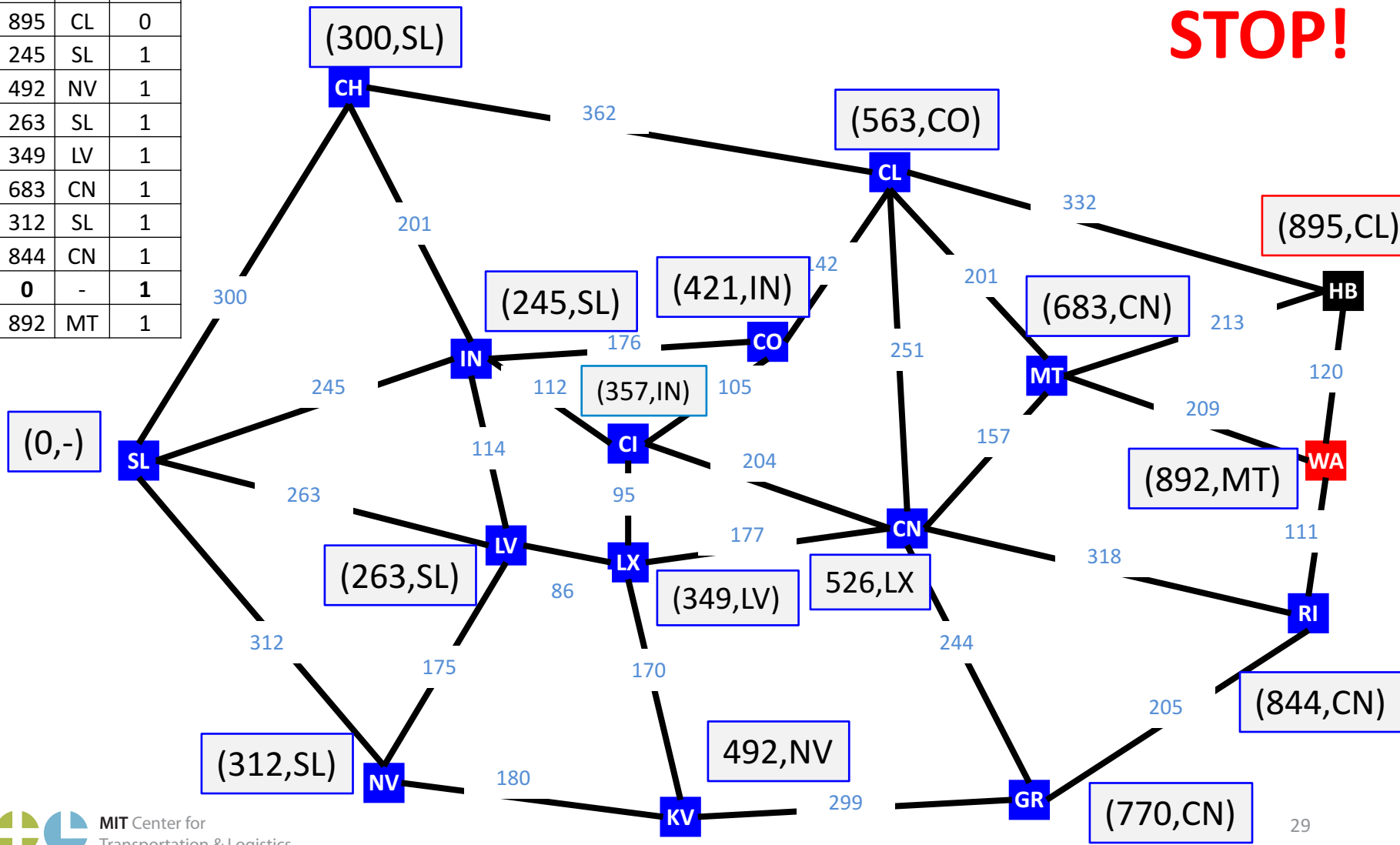


Shortest Path (SL to WA)

	L()	P()	S()
CH	300	SL	1
CI	357	IN	1
CL	563	CL	1
CN	526	LX	1
CO	421	IN	1
GR	770	CN	1
HB	895	CL	0
IN	245	SL	1
KV	492	NV	1
LV	263	SL	1
LX	349	LV	1
MT	683	CN	1
NV	312	SL	1
RI	844	CN	1
SL	**0**	-	**1**
WA	892	MT	1

select WA as active node
update adjacent nodes

STOP!



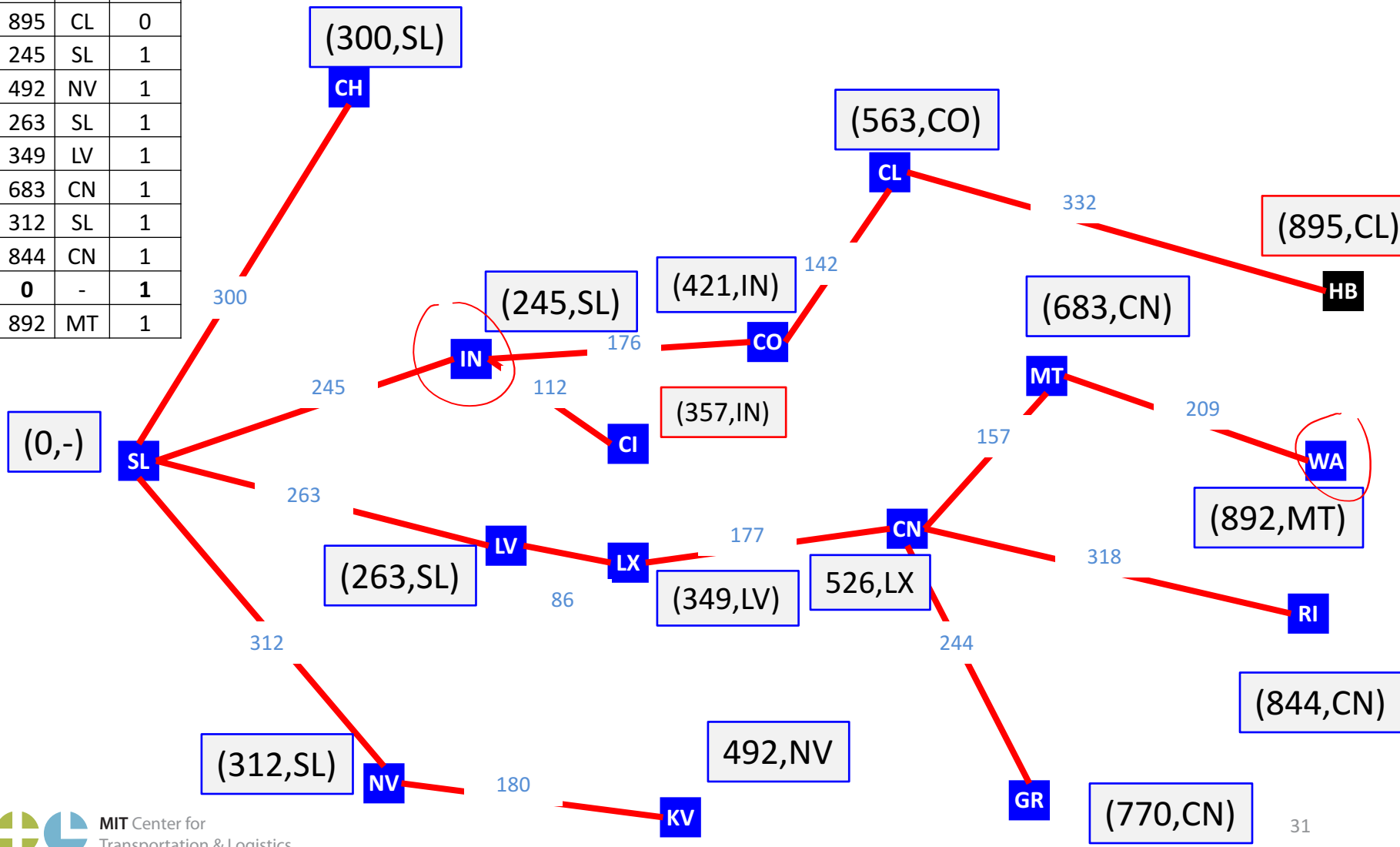
	L()	P()	S()
CH	300	SL	1
CI	357	IN	1
CL	563	CL	1
CN	526	LX	1
CO	421	IN	1
GR	770	CN	1
HB	895	CL	0
IN	245	SL	1
KV	492	NV	1
LV	263	SL	1
LX	349	LV	1
MT	683	CN	1
NV	312	SL	1
RI	844	CN	1
SL	**0**	-	**1**
WA	892	MT	1

Shortest Path from SL-WA

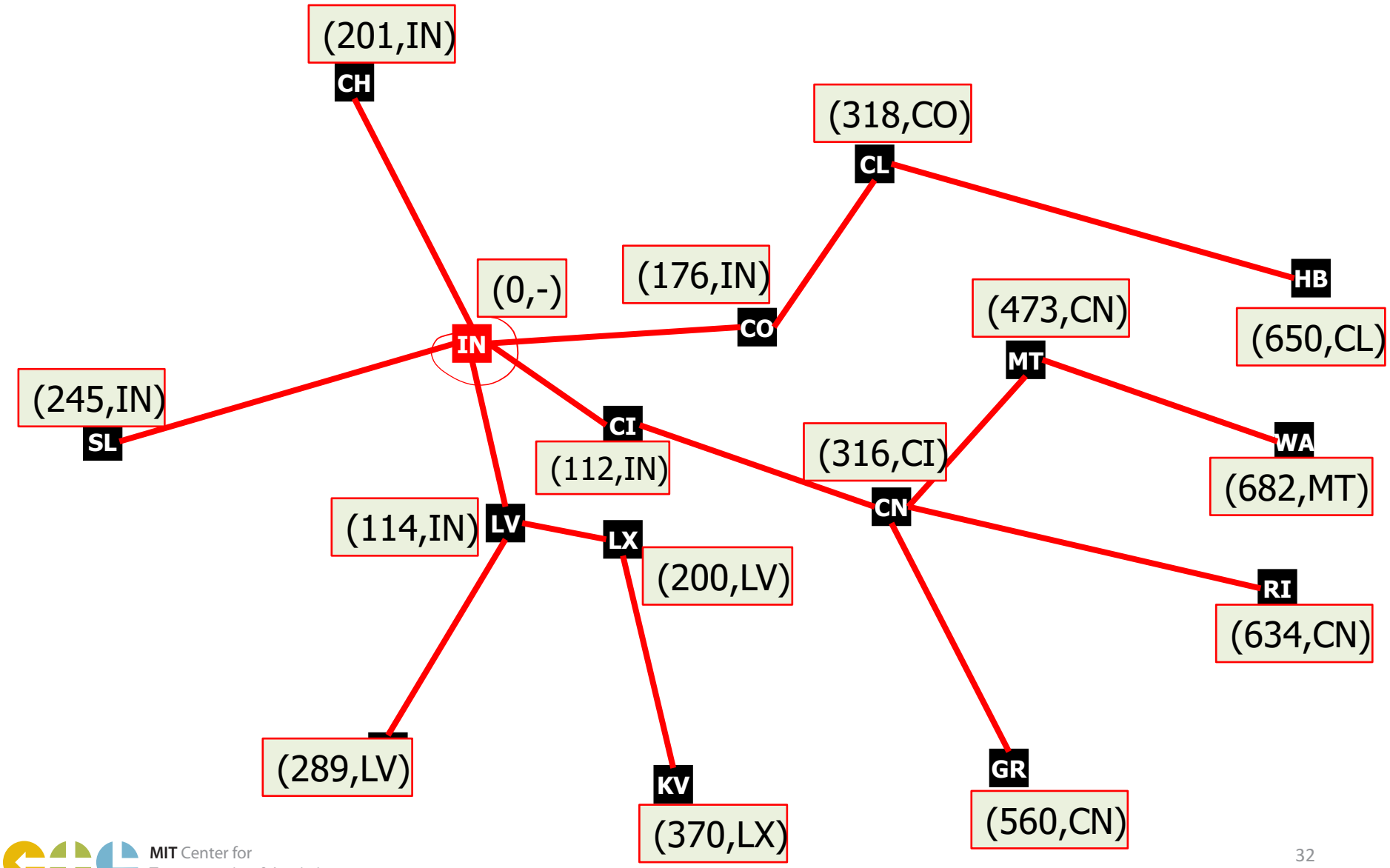
- So, what is our shortest path?
- Start from terminal and work backwards!
  - Terminal Node = WA
  - P(WA) = MT
  - P(MT) = CN
  - P(CN) = LX
  - P(LX) = LV
  - P(LV) = SL = Start Node
- Path is:
  - SL-LV-LX-CN-MT-WA for 892 miles

Shortest Path Tree from SL

	L()	P()	S()
CH	300	SL	1
CI	357	IN	1
CL	563	CL	1
CN	526	LX	1
CO	421	IN	1
GR	770	CN	1
HB	895	CL	0
IN	245	SL	1
KV	492	NV	1
LV	263	SL	1
LX	349	LV	1
MT	683	CN	1
NV	312	SL	1
RI	844	CN	1
SL	**0**	-	**1**
WA	892	MT	1



Final Shortest Path Tree from IN



Traveling Salesman Problem

Traveling Salesman Problem (TSP)

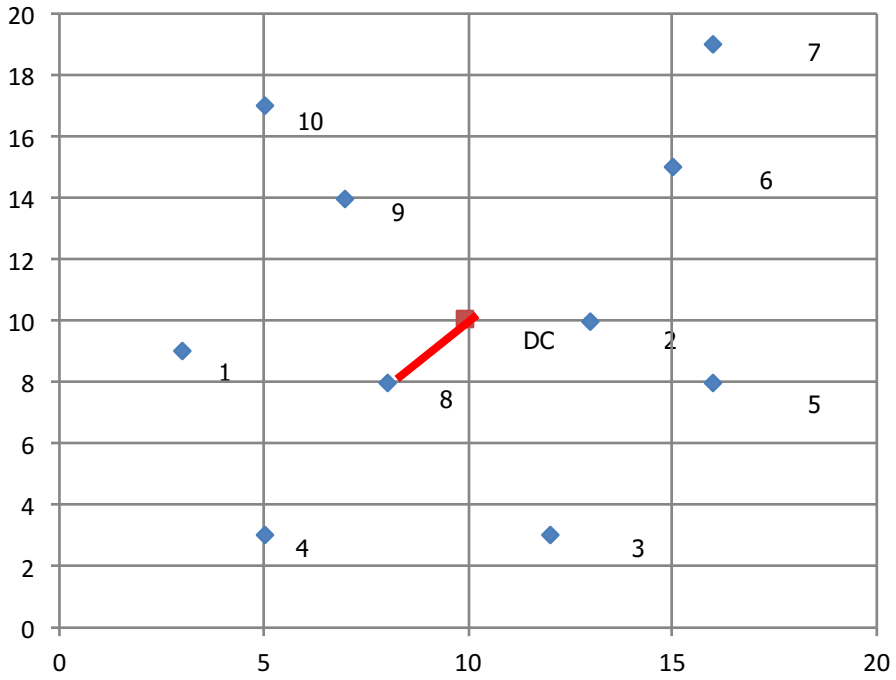
Starting from an origin node, find the minimum distance required to visit each node once and only once and return to the origin.

Nearest Neighbor Heuristic

1. Select any node to be the active node
2. Connect the active node to the closest unconnected node, make that the new active node.
3. If there are more unconnected nodes go to step 2, otherwise connect to the starting node and end.

Dist	1	2	3	4	5	6	7	8	9	10	DC
1	0.0	10.0	10.8	6.3	13.0	13.4	16.4	5.1	6.4	8.2	7.1
2	10.0	0.0	7.1	10.6	3.6	5.4	9.5	5.4	7.2	10.6	3.0
3	10.8	7.1	0.0	7.0	6.4	12.4	16.5	6.4	12.1	15.7	7.3
4	6.3	10.6	7.0	0.0	12.1	15.6	19.4	5.8	11.2	14.0	8.6
5	13.0	3.6	6.4	12.1	0.0	7.1	11.0	8.0	10.8	14.2	6.3
6	13.4	5.4	12.4	15.6	7.1	0.0	4.1	9.9	8.1	10.2	7.1
7	16.4	9.5	16.5	19.4	11.0	4.1	0.0	13.6	10.3	11.2	10.8
8	5.1	5.4	6.4	5.8	8.0	9.9	13.6	0.0	6.1	9.5	2.8
9	6.4	7.2	12.1	11.2	10.8	8.1	10.3	6.1	0.0	3.6	5.0
10	8.2	10.6	15.7	14.0	14.2	10.2	11.2	9.5	3.6	0.0	8.6
DC	7.1	3.0	7.3	8.6	6.3	7.1	10.8	2.8	5.0	8.6	0.0

Tour: DC-8



Traveling Salesman Problem (TSP)

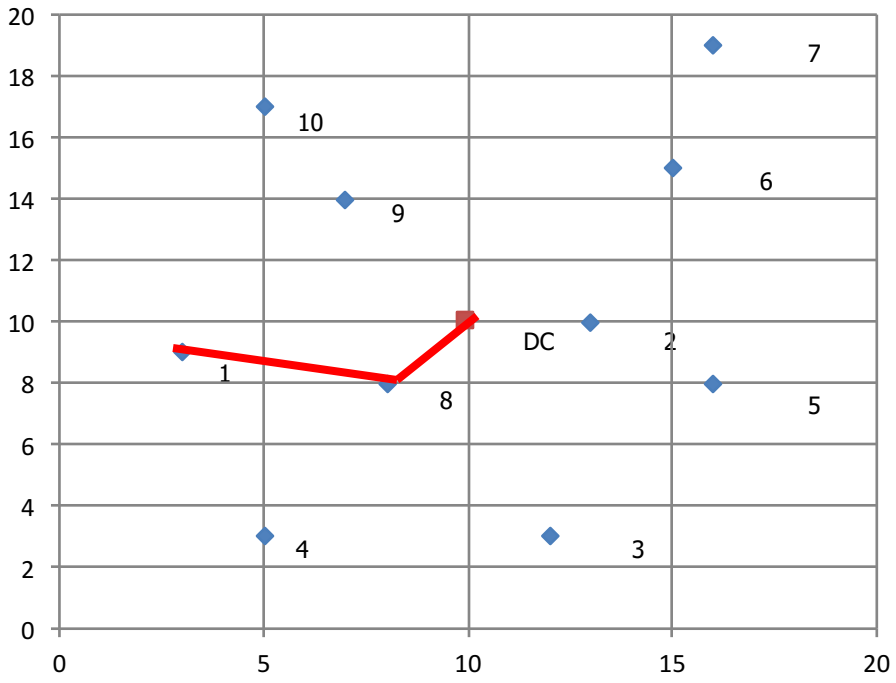
Starting from an origin node, find the minimum distance required to visit each node once and only once and return to the origin.

Nearest Neighbor Heuristic

1. Select any node to be the active node
2. Connect the active node to the closest unconnected node, make that the new active node.
3. If there are more unconnected nodes go to step 2, otherwise connect to the starting node and end.

Dist	1	2	3	4	5	6	7	8	9	10	DC
1	0.0	10.0	10.8	6.3	13.0	13.4	16.4	5.1	6.4	8.2	7.1
2	10.0	0.0	7.1	10.6	3.6	5.4	9.5	5.4	7.2	10.6	3.0
3	10.8	7.1	0.0	7.0	6.4	12.4	16.5	6.4	12.1	15.7	7.3
4	6.3	10.6	7.0	0.0	12.1	15.6	19.4	5.8	11.2	14.0	8.6
5	13.0	3.6	6.4	12.1	0.0	7.1	11.0	8.0	10.8	14.2	6.3
6	13.4	5.4	12.4	15.6	7.1	0.0	4.1	9.9	8.1	10.2	7.1
7	16.4	9.5	16.5	19.4	11.0	4.1	0.0	13.6	10.3	11.2	10.8
8	5.1	5.4	6.4	5.8	8.0	9.9	13.6	0.0	6.1	9.5	2.8
9	6.4	7.2	12.1	11.2	10.8	8.1	10.3	6.1	0.0	3.6	5.0
10	8.2	10.6	15.7	14.0	14.2	10.2	11.2	9.5	3.6	0.0	8.6
DC	7.1	3.0	7.3	8.6	6.3	7.1	10.8	2.8	5.0	8.6	0.0

Tour: DC-8



Traveling Salesman Problem (TSP)

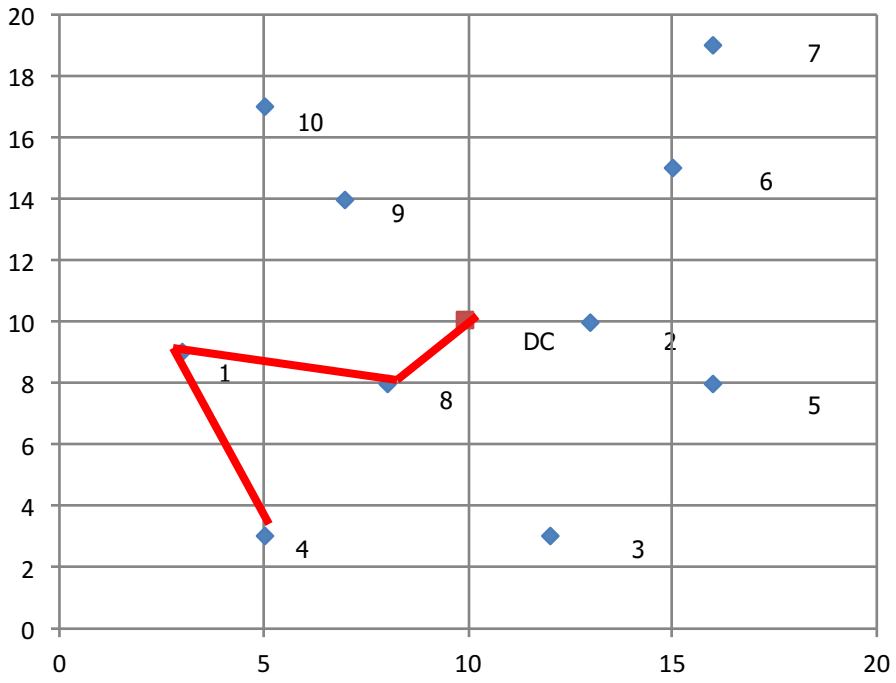
Starting from an origin node, find the minimum distance required to visit each node once and only once and return to the origin.

Nearest Neighbor Heuristic

1. Select any node to be the active node
2. Connect the active node to the closest unconnected node, make that the new active node.
3. If there are more unconnected nodes go to step 2, otherwise connect to the starting node and end.

Dist	1	2	3	4	5	6	7	8	9	10	DC
1	0.0	10.0	10.8	6.3	13.0	13.4	16.4	5.1	6.4	8.2	7.1
2	10.0	0.0	7.1	10.6	3.6	5.4	9.5	5.4	7.2	10.6	3.0
3	10.8	7.1	0.0	7.0	6.4	12.4	16.5	6.4	12.1	15.7	7.3
4	6.3	10.6	7.0	0.0	12.1	15.6	19.4	5.8	11.2	14.0	8.6
5	13.0	3.6	6.4	12.1	0.0	7.1	11.0	8.0	10.8	14.2	6.3
6	13.4	5.4	12.4	15.6	7.1	0.0	4.1	9.9	8.1	10.2	7.1
7	16.4	9.5	16.5	19.4	11.0	4.1	0.0	13.6	10.3	11.2	10.8
8	5.1	5.4	6.4	5.8	8.0	9.9	13.6	0.0	6.1	9.5	2.8
9	6.4	7.2	12.1	11.2	10.8	8.1	10.3	6.1	0.0	3.6	5.0
10	8.2	10.6	15.7	14.0	14.2	10.2	11.2	9.5	3.6	0.0	8.6
DC	7.1	3.0	7.3	8.6	6.3	7.1	10.8	2.8	5.0	8.6	0.0

Tour: DC-8-1



Traveling Salesman Problem (TSP)

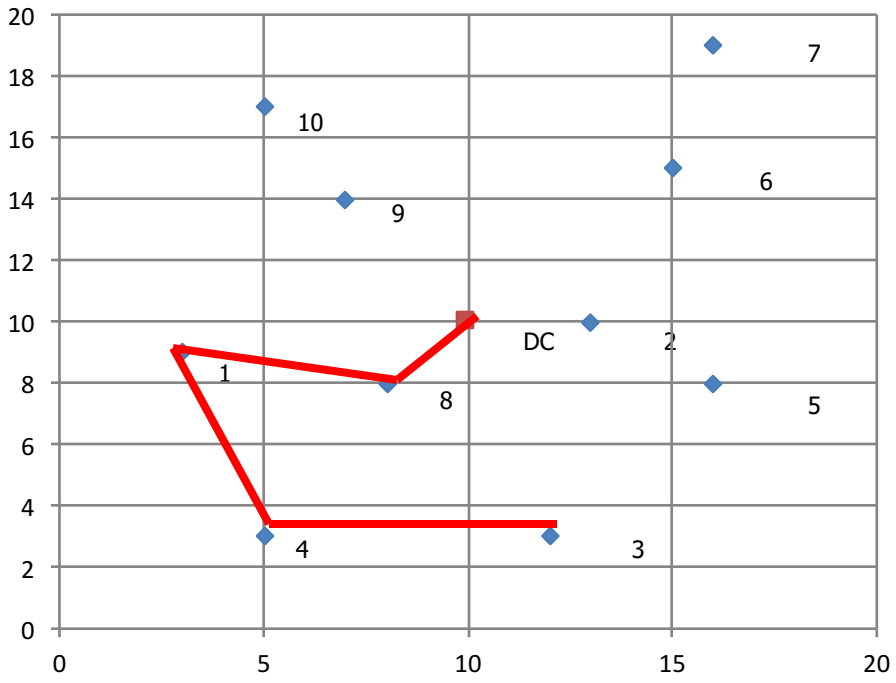
Starting from an origin node, find the minimum distance required to visit each node once and only once and return to the origin.

Nearest Neighbor Heuristic

1. Select any node to be the active node
2. Connect the active node to the closest unconnected node, make that the new active node.
3. If there are more unconnected nodes go to step 2, otherwise connect to the starting node and end.

Dist	1	2	3	4	5	6	7	8	9	10	DC
1	0.0	10.0	10.8	6.3	13.0	13.4	16.4	5.1	6.4	8.2	7.1
2	10.0	0.0	7.1	10.6	3.6	5.4	9.5	5.4	7.2	10.6	3.0
3	10.8	7.1	0.0	7.0	6.4	12.4	16.5	6.4	12.1	15.7	7.3
4	6.3	10.6	7.0	0.0	12.1	15.6	19.4	5.8	11.2	14.0	8.6
5	13.0	3.6	6.4	12.1	0.0	7.1	11.0	8.0	10.8	14.2	6.3
6	13.4	5.4	12.4	15.6	7.1	0.0	4.1	9.9	8.1	10.2	7.1
7	16.4	9.5	16.5	19.4	11.0	4.1	0.0	13.6	10.3	11.2	10.8
8	5.1	5.4	6.4	5.8	8.0	9.9	13.6	0.0	6.1	9.5	2.8
9	6.4	7.2	12.1	11.2	10.8	8.1	10.3	6.1	0.0	3.6	5.0
10	8.2	10.6	15.7	14.0	14.2	10.2	11.2	9.5	3.6	0.0	8.6
DC	7.1	3.0	7.3	8.6	6.3	7.1	10.8	2.8	5.0	8.6	0.0

Tour: DC-8-1-4



Traveling Salesman Problem (TSP)

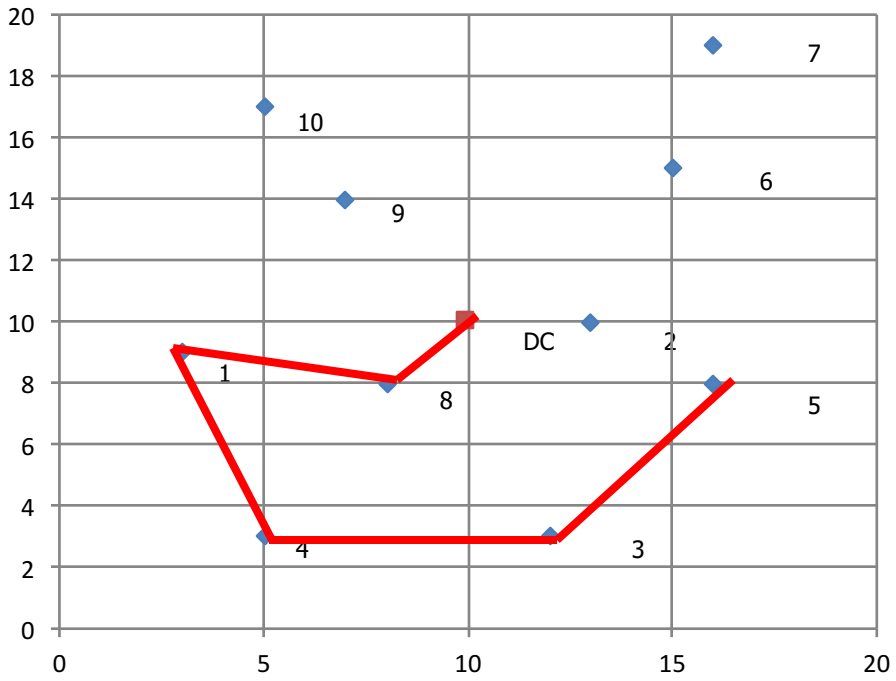
Starting from an origin node, find the minimum distance required to visit each node once and only once and return to the origin.

Nearest Neighbor Heuristic

1. Select any node to be the active node
2. Connect the active node to the closest unconnected node, make that the new active node.
3. If there are more unconnected nodes go to step 2, otherwise connect to the starting node and end.

Dist	1	2	3	4	5	6	7	8	9	10	DC
1	0.0	10.0	10.8	6.3	13.0	13.4	16.4	5.1	6.4	8.2	7.1
2	10.0	0.0	7.1	10.6	3.6	5.4	9.5	5.4	7.2	10.6	3.0
3	10.8	7.1	0.0	7.0	6.4	12.4	16.5	6.4	12.1	15.7	7.3
4	6.3	10.6	7.0	0.0	12.1	15.6	19.4	5.8	11.2	14.0	8.6
5	13.0	3.6	6.4	12.1	0.0	7.1	11.0	8.0	10.8	14.2	6.3
6	13.4	5.4	12.4	15.6	7.1	0.0	4.1	9.9	8.1	10.2	7.1
7	16.4	9.5	16.5	19.4	11.0	4.1	0.0	13.6	10.3	11.2	10.8
8	5.1	5.4	6.4	5.8	8.0	9.9	13.6	0.0	6.1	9.5	2.8
9	6.4	7.2	12.1	11.2	10.8	8.1	10.3	6.1	0.0	3.6	5.0
10	8.2	10.6	15.7	14.0	14.2	10.2	11.2	9.5	3.6	0.0	8.6
DC	7.1	3.0	7.3	8.6	6.3	7.1	10.8	2.8	5.0	8.6	0.0

Tour: DC-8-1-4-3



Traveling Salesman Problem (TSP)

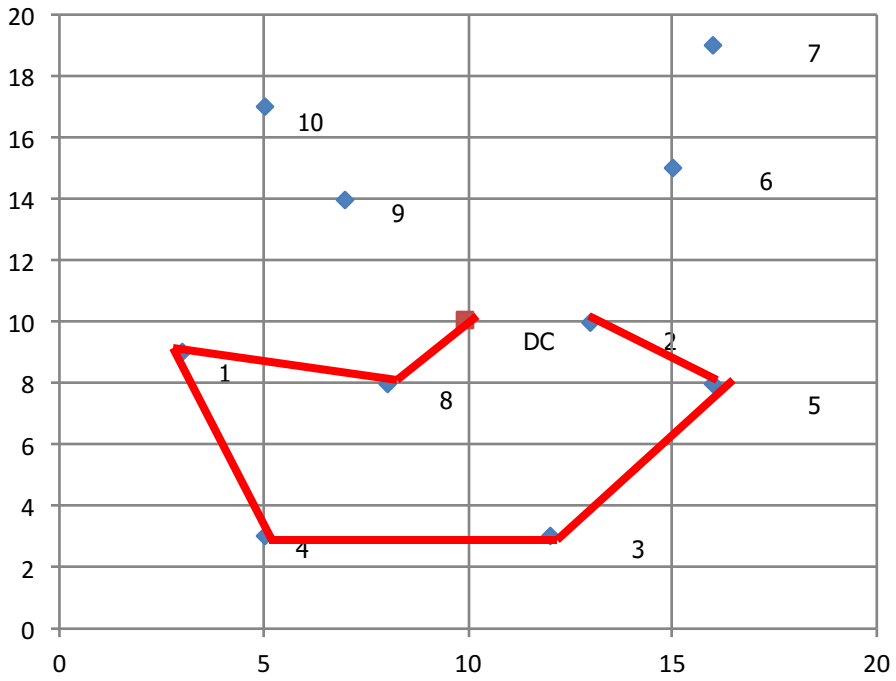
Starting from an origin node, find the minimum distance required to visit each node once and only once and return to the origin.

Nearest Neighbor Heuristic

1. Select any node to be the active node
2. Connect the active node to the closest unconnected node, make that the new active node.
3. If there are more unconnected nodes go to step 2, otherwise connect to the starting node and end.

Dist	1	2	3	4	5	6	7	8	9	10	DC
1	0.0	10.0	10.8	6.3	13.0	13.4	16.4	5.1	6.4	8.2	7.1
2	10.0	0.0	7.1	10.6	3.6	5.4	9.5	5.4	7.2	10.6	3.0
3	10.8	7.1	0.0	7.0	6.4	12.4	16.5	6.4	12.1	15.7	7.3
4	6.3	10.6	7.0	0.0	12.1	15.6	19.4	5.8	11.2	14.0	8.6
5	13.0	3.6	6.4	12.1	0.0	7.1	11.0	8.0	10.8	14.2	6.3
6	13.4	5.4	12.4	15.6	7.1	0.0	4.1	9.9	8.1	10.2	7.1
7	16.4	9.5	16.5	19.4	11.0	4.1	0.0	13.6	10.3	11.2	10.8
8	5.1	5.4	6.4	5.8	8.0	9.9	13.6	0.0	6.1	9.5	2.8
9	6.4	7.2	12.1	11.2	10.8	8.1	10.3	6.1	0.0	3.6	5.0
10	8.2	10.6	15.7	14.0	14.2	10.2	11.2	9.5	3.6	0.0	8.6
DC	7.1	3.0	7.3	8.6	6.3	7.1	10.8	2.8	5.0	8.6	0.0

Tour: DC-8-1-4-3-5



Traveling Salesman Problem (TSP)

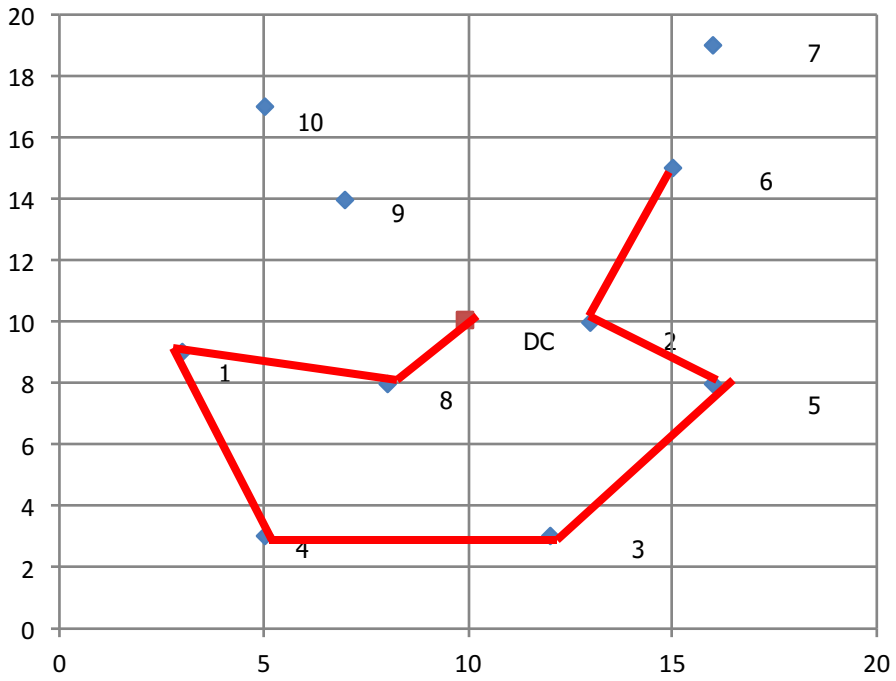
Starting from an origin node, find the minimum distance required to visit each node once and only once and return to the origin.

Nearest Neighbor Heuristic

1. Select any node to be the active node
2. Connect the active node to the closest unconnected node, make that the new active node.
3. If there are more unconnected nodes go to step 2, otherwise connect to the starting node and end.

Dist	1	2	3	4	5	6	7	8	9	10	DC
1	0.0	10.0	10.8	6.3	13.0	13.4	16.4	5.1	6.4	8.2	7.1
2	10.0	0.0	7.1	10.6	3.6	5.4	9.5	5.4	7.2	10.6	3.0
3	10.8	7.1	0.0	7.0	6.4	12.4	16.5	6.4	12.1	15.7	7.3
4	6.3	10.6	7.0	0.0	12.1	15.6	19.4	5.8	11.2	14.0	8.6
5	13.0	3.6	6.4	12.1	0.0	7.1	11.0	8.0	10.8	14.2	6.3
6	13.4	5.4	12.4	15.6	7.1	0.0	4.1	9.9	8.1	10.2	7.1
7	16.4	9.5	16.5	19.4	11.0	4.1	0.0	13.6	10.3	11.2	10.8
8	5.1	5.4	6.4	5.8	8.0	9.9	13.6	0.0	6.1	9.5	2.8
9	6.4	7.2	12.1	11.2	10.8	8.1	10.3	6.1	0.0	3.6	5.0
10	8.2	10.6	15.7	14.0	14.2	10.2	11.2	9.5	3.6	0.0	8.6
DC	7.1	3.0	7.3	8.6	6.3	7.1	10.8	2.8	5.0	8.6	0.0

Tour: DC-8-1-4-3-5-2



Traveling Salesman Problem (TSP)

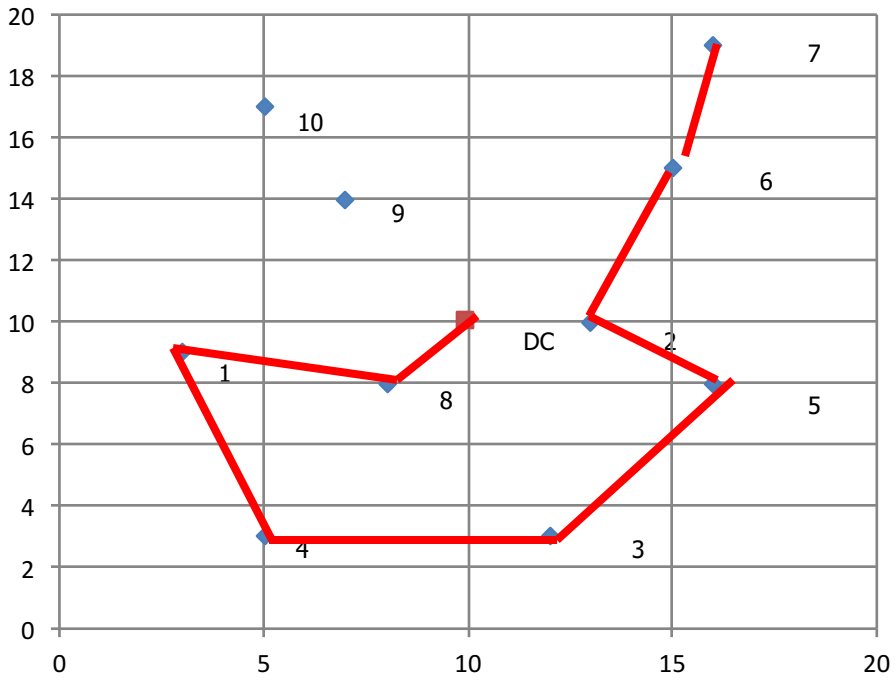
Starting from an origin node, find the minimum distance required to visit each node once and only once and return to the origin.

Nearest Neighbor Heuristic

1. Select any node to be the active node
2. Connect the active node to the closest unconnected node, make that the new active node.
3. If there are more unconnected nodes go to step 2, otherwise connect to the starting node and end.

Dist	1	2	3	4	5	6	7	8	9	10	DC
1	0.0	10.0	10.8	6.3	13.0	13.4	16.4	5.1	6.4	8.2	7.1
2	10.0	0.0	7.1	10.6	3.6	5.4	9.5	5.4	7.2	10.6	3.0
3	10.8	7.1	0.0	7.0	6.4	12.4	16.5	6.4	12.1	15.7	7.3
4	6.3	10.6	7.0	0.0	12.1	15.6	19.4	5.8	11.2	14.0	8.6
5	13.0	3.6	6.4	12.1	0.0	7.1	11.0	8.0	10.8	14.2	6.3
6	13.4	5.4	12.4	15.6	7.1	0.0	4.1	9.9	8.1	10.2	7.1
7	16.4	9.5	16.5	19.4	11.0	4.1	0.0	13.6	10.3	11.2	10.8
8	5.1	5.4	6.4	5.8	8.0	9.9	13.6	0.0	6.1	9.5	2.8
9	6.4	7.2	12.1	11.2	10.8	8.1	10.3	6.1	0.0	3.6	5.0
10	8.2	10.6	15.7	14.0	14.2	10.2	11.2	9.5	3.6	0.0	8.6
DC	7.1	3.0	7.3	8.6	6.3	7.1	10.8	2.8	5.0	8.6	0.0

Tour: DC-8-1-4-3-5-2-6



Traveling Salesman Problem (TSP)

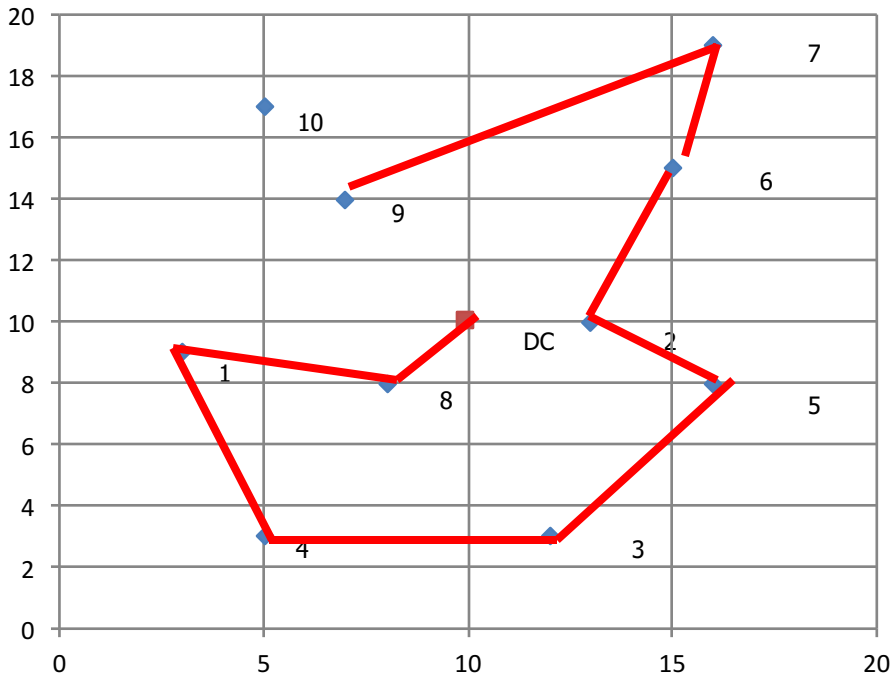
Starting from an origin node, find the minimum distance required to visit each node once and only once and return to the origin.

Nearest Neighbor Heuristic

1. Select any node to be the active node
2. Connect the active node to the closest unconnected node, make that the new active node.
3. If there are more unconnected nodes go to step 2, otherwise connect to the starting node and end.

Dist	1	2	3	4	5	6	7	8	9	10	DC
1	0.0	10.0	10.8	6.3	13.0	13.4	16.4	5.1	6.4	8.2	7.1
2	10.0	0.0	7.1	10.6	3.6	5.4	9.5	5.4	7.2	10.6	3.0
3	10.8	7.1	0.0	7.0	6.4	12.4	16.5	6.4	12.1	15.7	7.3
4	6.3	10.6	7.0	0.0	12.1	15.6	19.4	5.8	11.2	14.0	8.6
5	13.0	3.6	6.4	12.1	0.0	7.1	11.0	8.0	10.8	14.2	6.3
6	13.4	5.4	12.4	15.6	7.1	0.0	4.1	9.9	8.1	10.2	7.1
7	16.4	9.5	16.5	19.4	11.0	4.1	0.0	13.6	10.3	11.2	10.8
8	5.1	5.4	6.4	5.8	8.0	9.9	13.6	0.0	6.1	9.5	2.8
9	6.4	7.2	12.1	11.2	10.8	8.1	10.3	6.1	0.0	3.6	5.0
10	8.2	10.6	15.7	14.0	14.2	10.2	11.2	9.5	3.6	0.0	8.6
DC	7.1	3.0	7.3	8.6	6.3	7.1	10.8	2.8	5.0	8.6	0.0

Tour: DC-8-1-4-3-5-2-6-7



Traveling Salesman Problem (TSP)

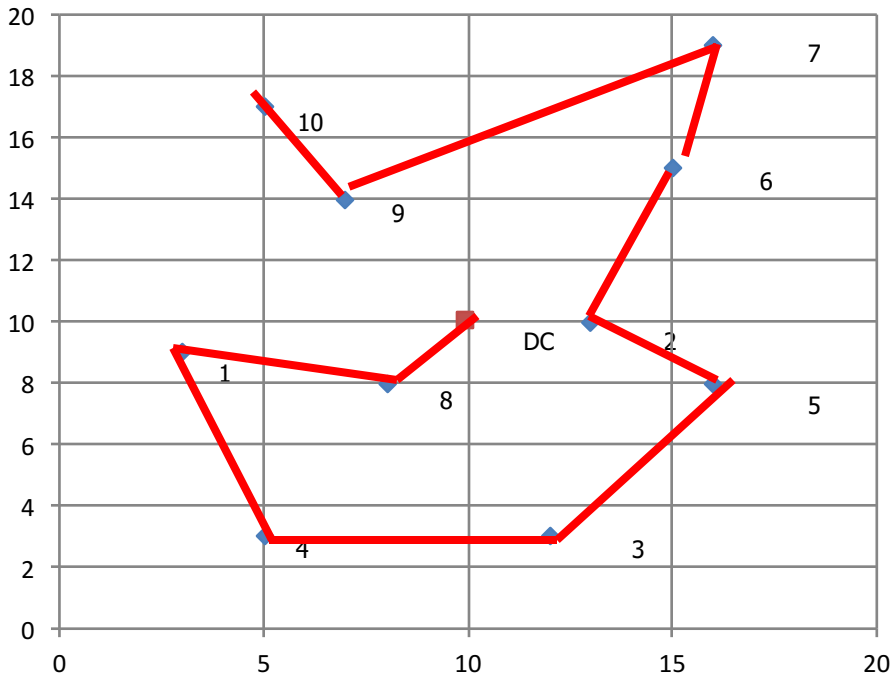
Starting from an origin node, find the minimum distance required to visit each node once and only once and return to the origin.

Nearest Neighbor Heuristic

1. Select any node to be the active node
2. Connect the active node to the closest unconnected node, make that the new active node.
3. If there are more unconnected nodes go to step 2, otherwise connect to the starting node and end.

Dist	1	2	3	4	5	6	7	8	9	10	DC
1	0.0	10.0	10.8	6.3	13.0	13.4	16.4	5.1	6.4	8.2	7.1
2	10.0	0.0	7.1	10.6	3.6	5.4	9.5	5.4	7.2	10.6	3.0
3	10.8	7.1	0.0	7.0	6.4	12.4	16.5	6.4	12.1	15.7	7.3
4	6.3	10.6	7.0	0.0	12.1	15.6	19.4	5.8	11.2	14.0	8.6
5	13.0	3.6	6.4	12.1	0.0	7.1	11.0	8.0	10.8	14.2	6.3
6	13.4	5.4	12.4	15.6	7.1	0.0	4.1	9.9	8.1	10.2	7.1
7	16.4	9.5	16.5	19.4	11.0	4.1	0.0	13.6	10.3	11.2	10.8
8	5.1	5.4	6.4	5.8	8.0	9.9	13.6	0.0	6.1	9.5	2.8
9	6.4	7.2	12.1	11.2	10.8	8.1	10.3	6.1	0.0	3.6	5.0
10	8.2	10.6	15.7	14.0	14.2	10.2	11.2	9.5	3.6	0.0	8.6
DC	7.1	3.0	7.3	8.6	6.3	7.1	10.8	2.8	5.0	8.6	0.0

Tour: DC-8-1-4-3-5-2-6-7-9



Traveling Salesman Problem (TSP)

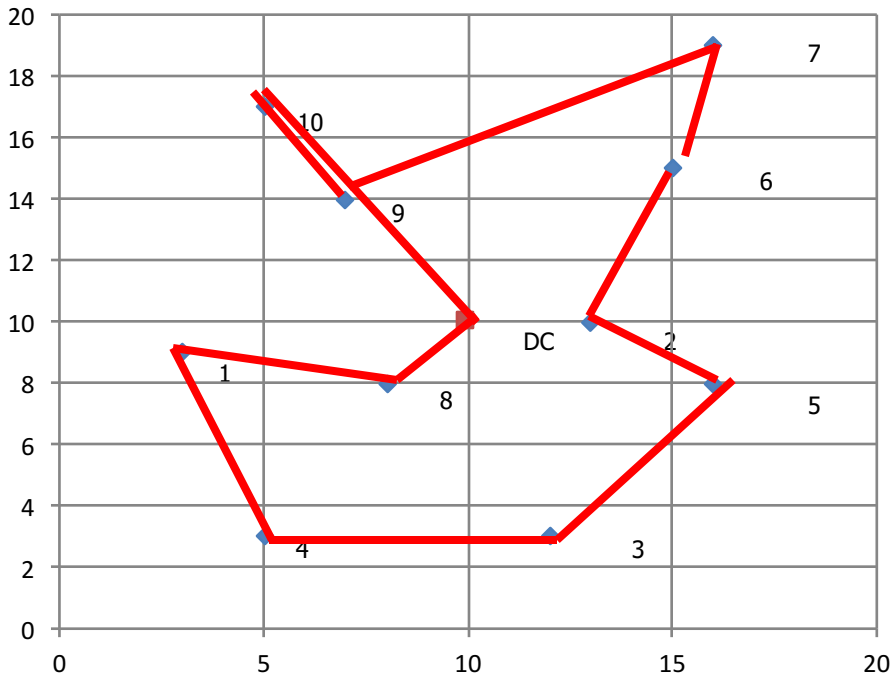
Starting from an origin node, find the minimum distance required to visit each node once and only once and return to the origin.

Nearest Neighbor Heuristic

1. Select any node to be the active node
2. Connect the active node to the closest unconnected node, make that the new active node.
3. If there are more unconnected nodes go to step 2, otherwise connect to the starting node and end.

Dist	1	2	3	4	5	6	7	8	9	10	DC
1	0.0	10.0	10.8	6.3	13.0	13.4	16.4	5.1	6.4	8.2	7.1
2	10.0	0.0	7.1	10.6	3.6	5.4	9.5	5.4	7.2	10.6	3.0
3	10.8	7.1	0.0	7.0	6.4	12.4	16.5	6.4	12.1	15.7	7.3
4	6.3	10.6	7.0	0.0	12.1	15.6	19.4	5.8	11.2	14.0	8.6
5	13.0	3.6	6.4	12.1	0.0	7.1	11.0	8.0	10.8	14.2	6.3
6	13.4	5.4	12.4	15.6	7.1	0.0	4.1	9.9	8.1	10.2	7.1
7	16.4	9.5	16.5	19.4	11.0	4.1	0.0	13.6	10.3	11.2	10.8
8	5.1	5.4	6.4	5.8	8.0	9.9	13.6	0.0	6.1	9.5	2.8
9	6.4	7.2	12.1	11.2	10.8	8.1	10.3	6.1	0.0	3.6	5.0
10	8.2	10.6	15.7	14.0	14.2	10.2	11.2	9.5	3.6	0.0	8.6
DC	7.1	3.0	7.3	8.6	6.3	7.1	10.8	2.8	5.0	8.6	0.0

Tour: DC-8-1-4-3-5-2-6-7-9-10

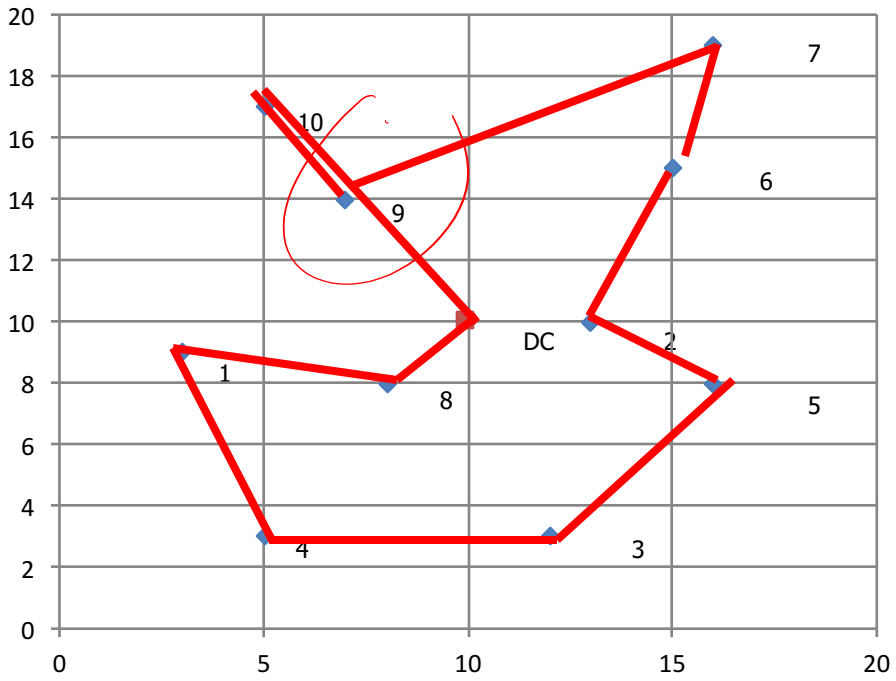


Traveling Salesman Problem (TSP)

Starting from an origin node, find the minimum distance required to visit each node once and only once and return to the origin.

Nearest Neighbor Heuristic
1. Select any node to be the active node
2. Connect the active node to the closest unconnected node, make that the new active node.
3. If there are more unconnected nodes go to step 2, otherwise connect to the starting node and end.

Dist	1	2	3	4	5	6	7	8	9	10	DC
1	0.0	10.0	10.8	6.3	13.0	13.4	16.4	5.1	6.4	8.2	7.1
2	10.0	0.0	7.1	10.6	3.6	5.4	9.5	5.4	7.2	10.6	3.0
3	10.8	7.1	0.0	7.0	6.4	12.4	16.5	6.4	12.1	15.7	7.3
4	6.3	10.6	7.0	0.0	12.1	15.6	19.4	5.8	11.2	14.0	8.6
5	13.0	3.6	6.4	12.1	0.0	7.1	11.0	8.0	10.8	14.2	6.3
6	13.4	5.4	12.4	15.6	7.1	0.0	4.1	9.9	8.1	10.2	7.1
7	16.4	9.5	16.5	19.4	11.0	4.1	0.0	13.6	10.3	11.2	10.8
8	5.1	5.4	6.4	5.8	8.0	9.9	13.6	0.0	6.1	9.5	2.8
9	6.4	7.2	12.1	11.2	10.8	8.1	10.3	6.1	0.0	3.6	5.0
10	8.2	10.6	15.7	14.0	14.2	10.2	11.2	9.5	3.6	0.0	8.6
DC	7.1	3.0	7.3	8.6	6.3	7.1	10.8	2.8	5.0	8.6	0.0



Tour: DC-8-1-4-3-5-2-6-7-9-10-DC
Length: 63.2

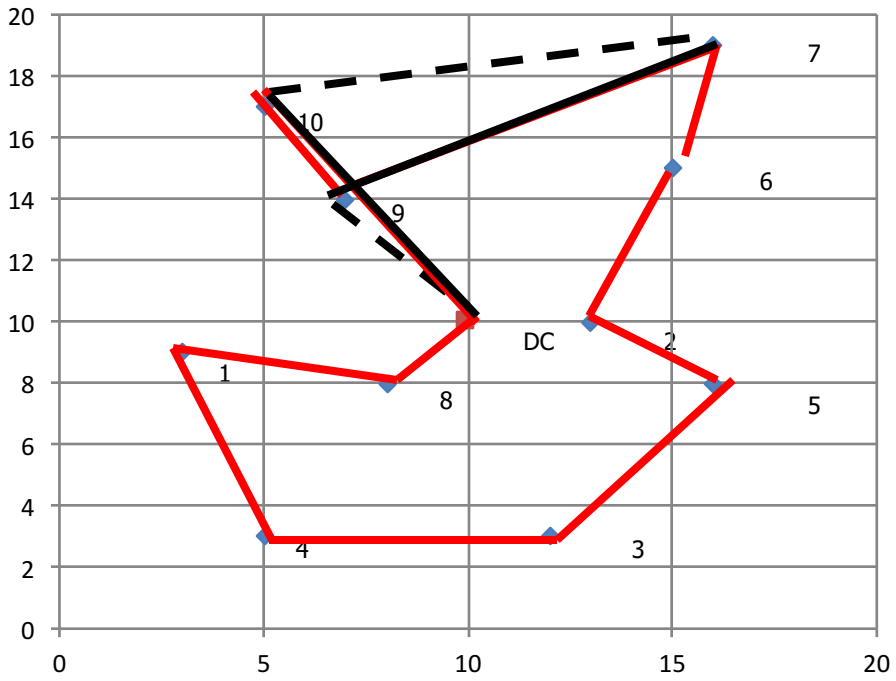
Improvement Heuristic: 2-Opt

Traveling Salesman Problem (TSP)

Starting from an origin node, find the minimum distance required to visit each node once and only once and return to the origin.

2-Opt Heuristic
1. Identify pairs of arcs (i-j and k-l), where d(ij) + d(kl) > d(ik) + d(jl) – usually where they cross
2. Select the pair with the largest difference, and re-connect the arcs (i-k and j-l)
3. Continue until there are no more crossed arcs.

Dist	1	2	3	4	5	6	7	8	9	10	DC
1	0.0	10.0	10.8	6.3	13.0	13.4	16.4	5.1	6.4	8.2	7.1
2	10.0	0.0	7.1	10.6	3.6	5.4	9.5	5.4	7.2	10.6	3.0
3	10.8	7.1	0.0	7.0	6.4	12.4	16.5	6.4	12.1	15.7	7.3
4	6.3	10.6	7.0	0.0	12.1	15.6	19.4	5.8	11.2	14.0	8.6
5	13.0	3.6	6.4	12.1	0.0	7.1	11.0	8.0	10.8	14.2	6.3
6	13.4	5.4	12.4	15.6	7.1	0.0	4.1	9.9	8.1	10.2	7.1
7	16.4	9.5	16.5	19.4	11.0	4.1	0.0	13.6	10.3	11.2	10.8
8	5.1	5.4	6.4	5.8	8.0	9.9	13.6	0.0	6.1	9.5	2.8
9	6.4	7.2	12.1	11.2	10.8	8.1	10.3	6.1	0.0	3.6	5.0
10	8.2	10.6	15.7	14.0	14.2	10.2	11.2	9.5	3.6	0.0	8.6
DC	7.1	3.0	7.3	8.6	6.3	7.1	10.8	2.8	5.0	8.6	0.0



Arcs 7-9 and 10-DC cross

$d_{(79)} + d_{(10\text{-}DC)} = 18.9 > d_{(7\text{-}10)} + d_{(9\text{-}DC)} = 16.2$

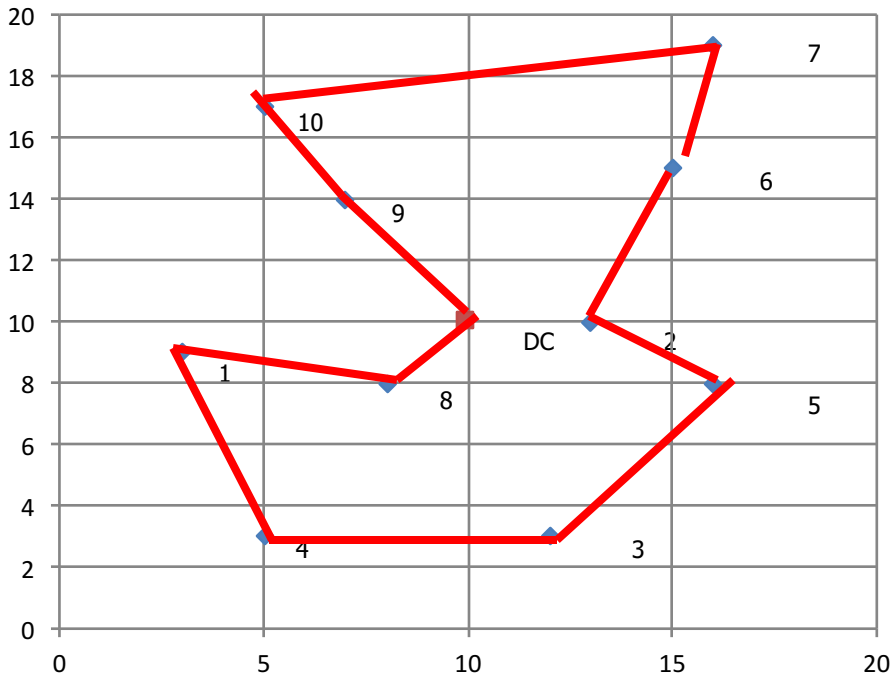
Re-connect arcs 7-10 and 9-DC

Traveling Salesman Problem (TSP)

Starting from an origin node, find the minimum distance required to visit each node once and only once and return to the origin.

2-Opt Heuristic
1. Identify pairs of arcs (i-j and k-l), where d(ij) + d(kl) > d(ik) + d(jl) – usually where they cross
2. Select the pair with the largest difference, and re-connect the arcs (i-k and j-l)
3. Continue until there are no more crossed arcs.

Dist	1	2	3	4	5	6	7	8	9	10	DC
1	0.0	10.0	10.8	6.3	13.0	13.4	16.4	5.1	6.4	8.2	7.1
2	10.0	0.0	7.1	10.6	3.6	5.4	9.5	5.4	7.2	10.6	3.0
3	10.8	7.1	0.0	7.0	6.4	12.4	16.5	6.4	12.1	15.7	7.3
4	6.3	10.6	7.0	0.0	12.1	15.6	19.4	5.8	11.2	14.0	8.6
5	13.0	3.6	6.4	12.1	0.0	7.1	11.0	8.0	10.8	14.2	6.3
6	13.4	5.4	12.4	15.6	7.1	0.0	4.1	9.9	8.1	10.2	7.1
7	16.4	9.5	16.5	19.4	11.0	4.1	0.0	13.6	10.3	11.2	10.8
8	5.1	5.4	6.4	5.8	8.0	9.9	13.6	0.0	6.1	9.5	2.8
9	6.4	7.2	12.1	11.2	10.8	8.1	10.3	6.1	0.0	3.6	5.0
10	8.2	10.6	15.7	14.0	14.2	10.2	11.2	9.5	3.6	0.0	8.6
DC	7.1	3.0	7.3	8.6	6.3	7.1	10.8	2.8	5.0	8.6	0.0



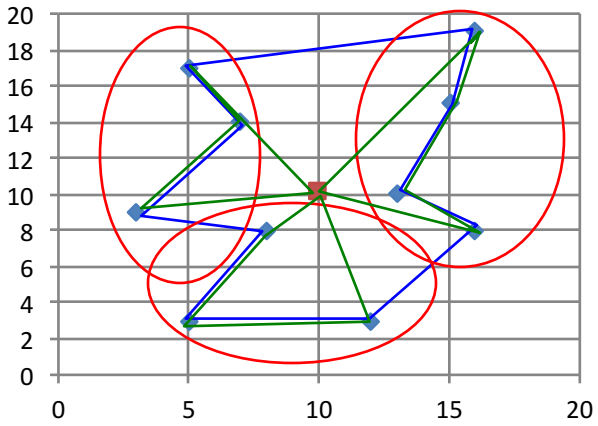
Tour: DC-8-1-4-3-5-2-6-7-10-9-DC
Tour length reduces from 63.2 to 60.5

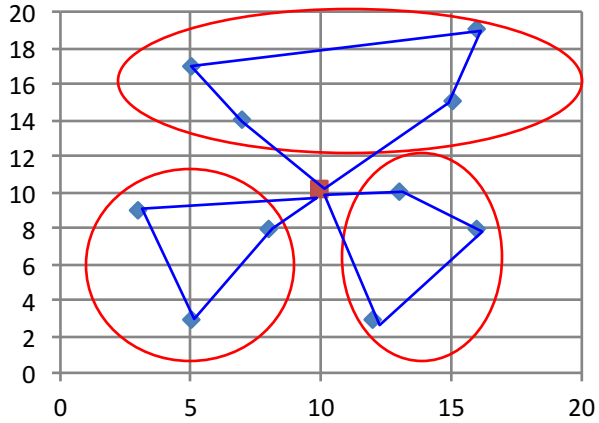
Vehicle Routing Problem

Vehicle Routing Problem (VRP)

Find minimum cost tours from single origin to multiple destinations with varying demand using multiple capacitated vehicles.

- Heuristics
  - Route first Cluster second
    - Any earlier TSP heuristic can be used
  - Cluster first Route second
    - Sweep Algorithm
    - Savings (Clarke-Wright)





- Optimal
  - Mixed Integer Linear Program (MILP)
  - Select optimal routes from potential set

VRP Sweep Heuristic

Vehicle Routing Problem – Sweep

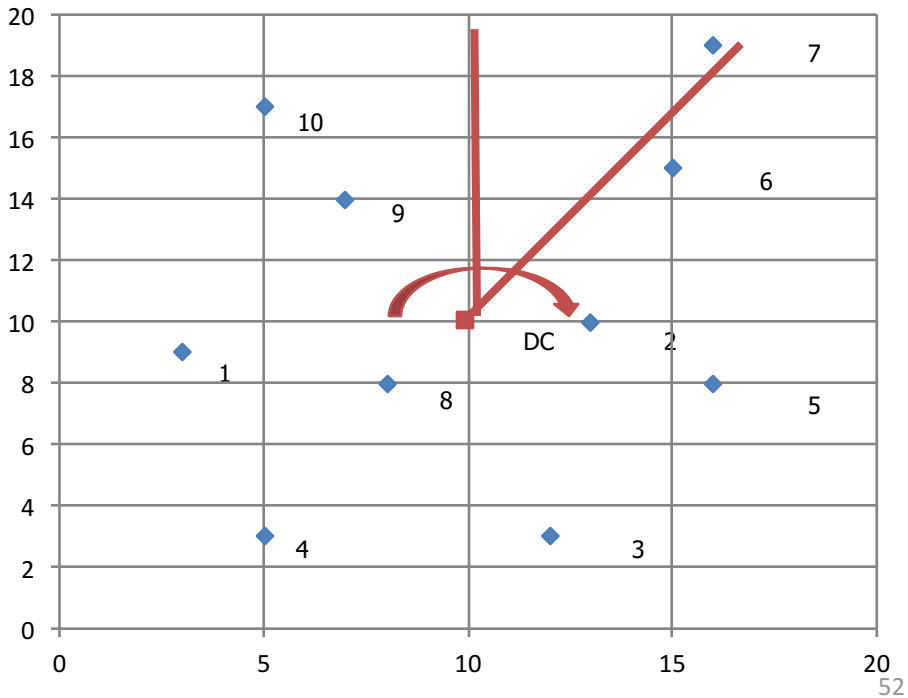
Find minimum cost tours from DC to 10 destinations with demand as shown using up to 4 vehicles of capacity of 200 units.

Sweep Heuristic

1. Form a ray from the DC and select an angle and direction (CW vs CCW) to start
2. Select a new vehicle, j, that is empty, w_j=0, and has capacity, c_j.
3. Rotate the ray in selected direction until it hits a customer node, i, or reaches the starting point (go to step 5).
4. If the demand at i (D_i) plus current load already in the vehicle (w_j) is less than the vehicle capacity, add it to the vehicle, $w_j=D_i + w_j$ and go to step 3. Otherwise, close this vehicle, and go to step 2 to start a new tour.
5. Solve the TSP for each independent vehicle tour.

Node ID	Demand (units)
1	22
2	18
3	53
4	54
5	31
6	80
7	78
8	84
9	85
10	42
DC	

j	w_j	$cluster_j$
1	78	7-



Vehicle Routing Problem – Sweep

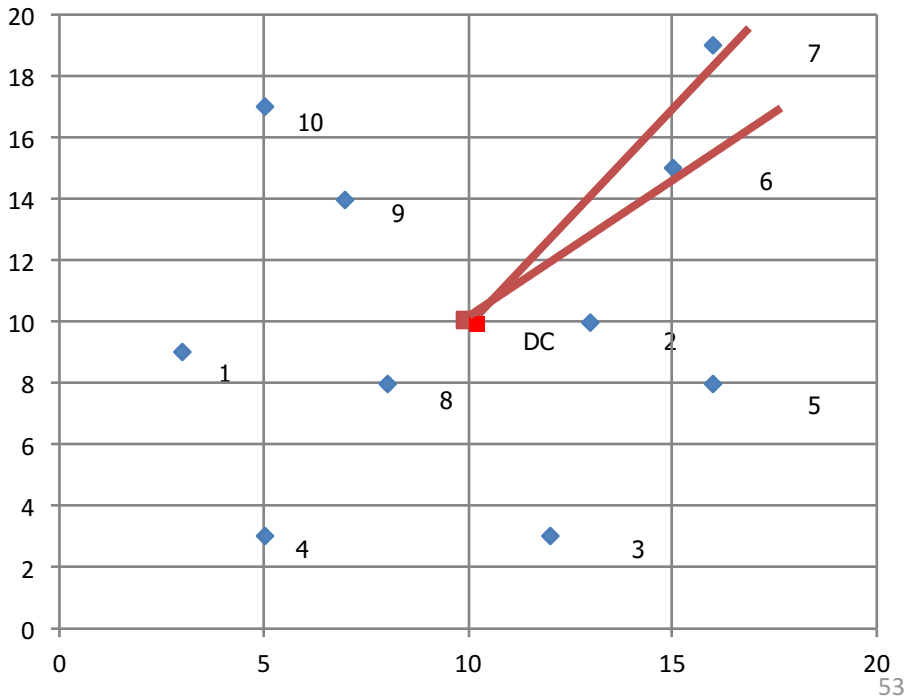
Find minimum cost tours from DC to 10 destinations with demand as shown using up to 4 vehicles of capacity of 200 units.

Sweep Heuristic

1. Form a ray from the DC and select an angle and direction (CW vs CCW) to start
2. Select a new vehicle, j, that is empty, w_j=0, and has capacity, c_j.
3. Rotate the ray in selected direction until it hits a customer node, i, or reaches the starting point (go to step 5).
4. If the demand at i (D_i) plus current load already in the vehicle (w_j) is less than the vehicle capacity, add it to the vehicle, w_j=D_i + w_j and go to step 3. Otherwise, close this vehicle, and go to step 2 to start a new tour.
5. Solve the TSP for each independent vehicle tour.

Node ID	Demand (units)
1	22
2	18
3	53
4	54
5	31
6	80
7	78
8	84
9	85
10	42
DC	

j	w_j	$cluster_j$
1	138	7-6-



Vehicle Routing Problem – Sweep

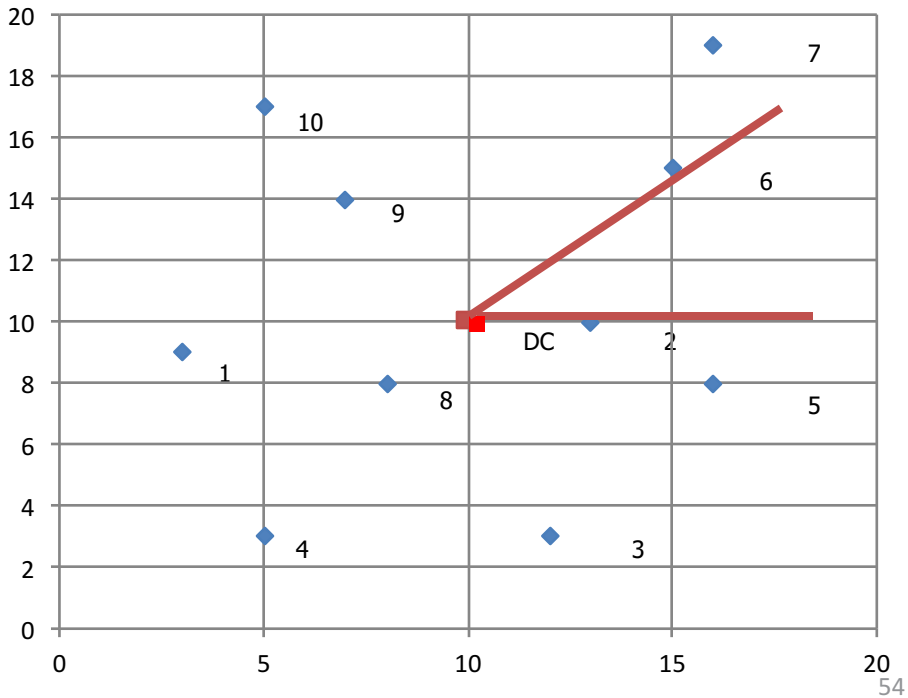
Find minimum cost tours from DC to 10 destinations with demand as shown using up to 4 vehicles of capacity of 200 units.

Sweep Heuristic

1. Form a ray from the DC and select an angle and direction (CW vs CCW) to start
2. Select a new vehicle, j, that is empty, w_j=0, and has capacity, c_j.
3. Rotate the ray in selected direction until it hits a customer node, i, or reaches the starting point (go to step 5).
4. If the demand at i (D_i) plus current load already in the vehicle (w_j) is less than the vehicle capacity, add it to the vehicle, w_j=D_i + w_j and go to step 3. Otherwise, close this vehicle, and go to step 2 to start a new tour.
5. Solve the TSP for each independent vehicle tour.

Node ID	Demand (units)
1	22
2	18
3	53
4	54
5	31
6	80
7	78
8	84
9	85
10	42
DC	

j	w_j	$cluster_j$
1	178	7-6-2-



Vehicle Routing Problem – Sweep

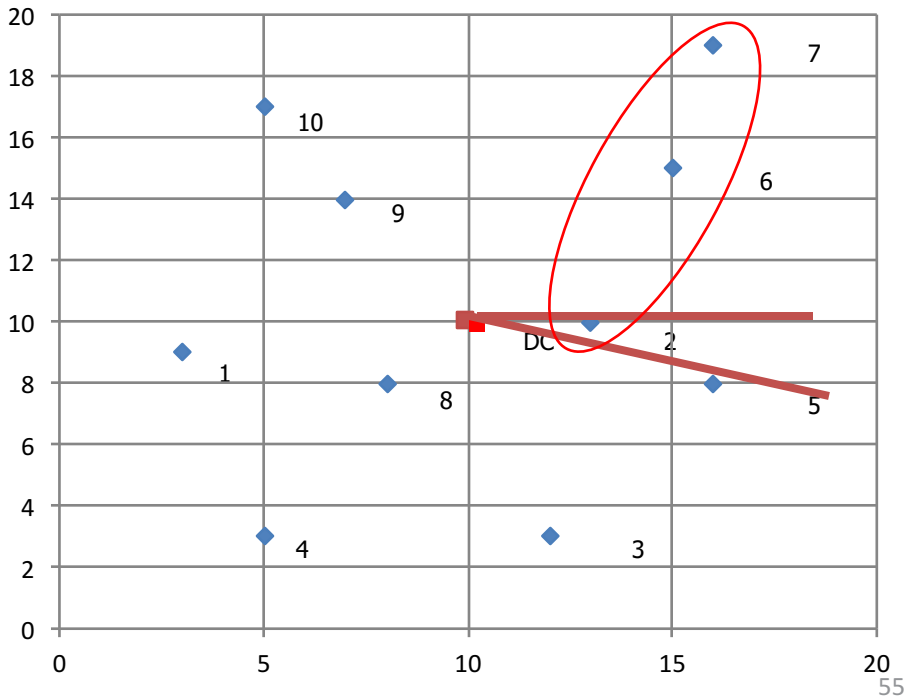
Find minimum cost tours from DC to 10 destinations with demand as shown using up to 4 vehicles of capacity of 200 units.

Sweep Heuristic

1. Form a ray from the DC and select an angle and direction (CW vs CCW) to start
2. Select a new vehicle, j, that is empty, w_j=0, and has capacity, c_j.
3. Rotate the ray in selected direction until it hits a customer node, i, or reaches the starting point (go to step 5).
4. If the demand at i (D_i) plus current load already in the vehicle (w_j) is less than the vehicle capacity, add it to the vehicle, $w_j=D_i + w_j$ and go to step 3. Otherwise, close this vehicle, and go to step 2 to start a new tour.
5. Solve the TSP for each independent vehicle tour.

Node ID	Demand (units)
1	22
2	18
3	53
4	54
5	31
6	80
7	78
8	84
9	85
10	42
DC	

j	w_j	$cluster_j$
1	176	7-6-2-
		176 + 31 > 200
2	31	5-



Vehicle Routing Problem – Sweep

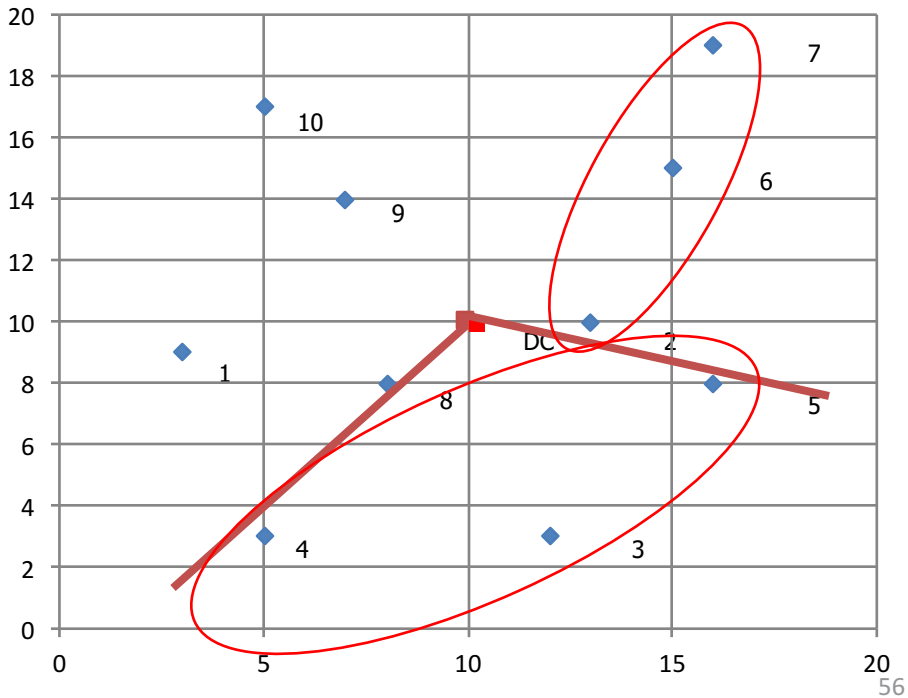
Find minimum cost tours from DC to 10 destinations with demand as shown using up to 4 vehicles of capacity of 200 units.

Sweep Heuristic

1. Form a ray from the DC and select an angle and direction (CW vs CCW) to start
2. Select a new vehicle, j, that is empty, w_j=0, and has capacity, c_j.
3. Rotate the ray in selected direction until it hits a customer node, i, or reaches the starting point (go to step 5).
4. If the demand at i (D_i) plus current load already in the vehicle (w_j) is less than the vehicle capacity, add it to the vehicle, w_j=D_i + w_j and go to step 3. Otherwise, close this vehicle, and go to step 2 to start a new tour.
5. Solve the TSP for each independent vehicle tour.

Node ID	Demand (units)
1	22
2	18
3	53
4	54
5	31
6	80
7	78
8	84
9	85
10	42
DC	

j	w_j	$cluster_j$
1	176	7-6- 2-
		176 + 31 > 200
2	138	5-3-4
		138+ 84 > 200
3	84	8-



Vehicle Routing Problem – Sweep

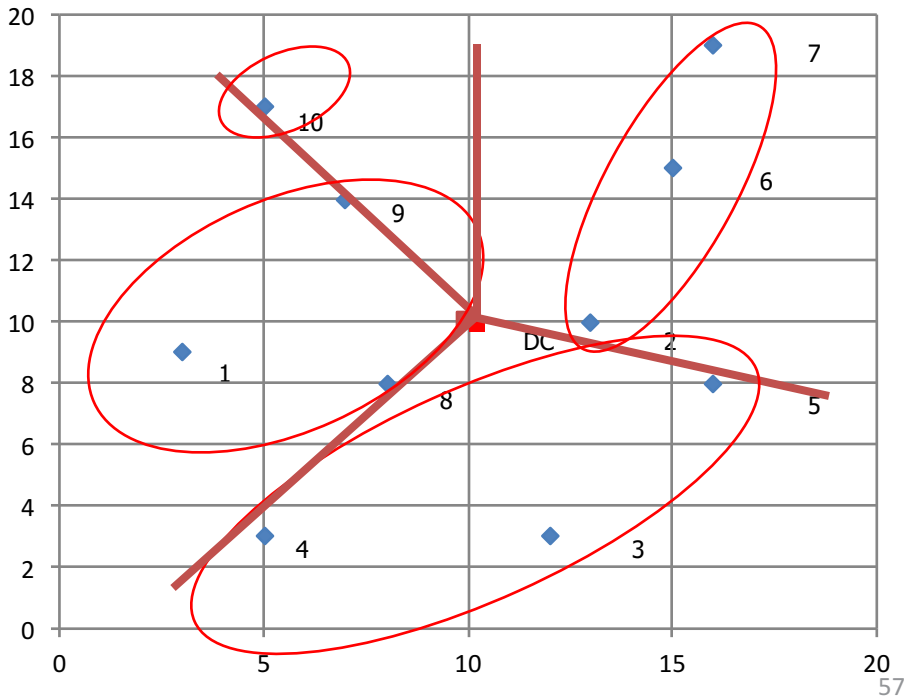
Find minimum cost tours from DC to 10 destinations with demand as shown using up to 4 vehicles of capacity of 200 units.

Sweep Heuristic

1. Form a ray from the DC and select an angle and direction (CW vs CCW) to start
2. Select a new vehicle, j, that is empty, w_j=0, and has capacity, c_j.
3. Rotate the ray in selected direction until it hits a customer node, i, or reaches the starting point (go to step 5).
4. If the demand at i (D_i) plus current load already in the vehicle (w_j) is less than the vehicle capacity, add it to the vehicle, w_j=D_i + w_j and go to step 3. Otherwise, close this vehicle, and go to step 2 to start a new tour.
5. Solve the TSP for each independent vehicle tour.

Node ID	Demand (units)
1	22
2	18
3	53
4	54
5	31
6	80
7	78
8	84
9	85
10	42
DC	

j	w_j	$cluster_j$
1	176	7-6- 2-
		176 + 31 > 200
2	138	5-3-4
		138+ 84 > 200
3	191	8-1-9
		191 + 42 > 200
4	42	10



Vehicle Routing Problem – Sweep

Find minimum cost tours from DC to 10 destinations with demand as shown using up to 4 vehicles of capacity of 200 units.

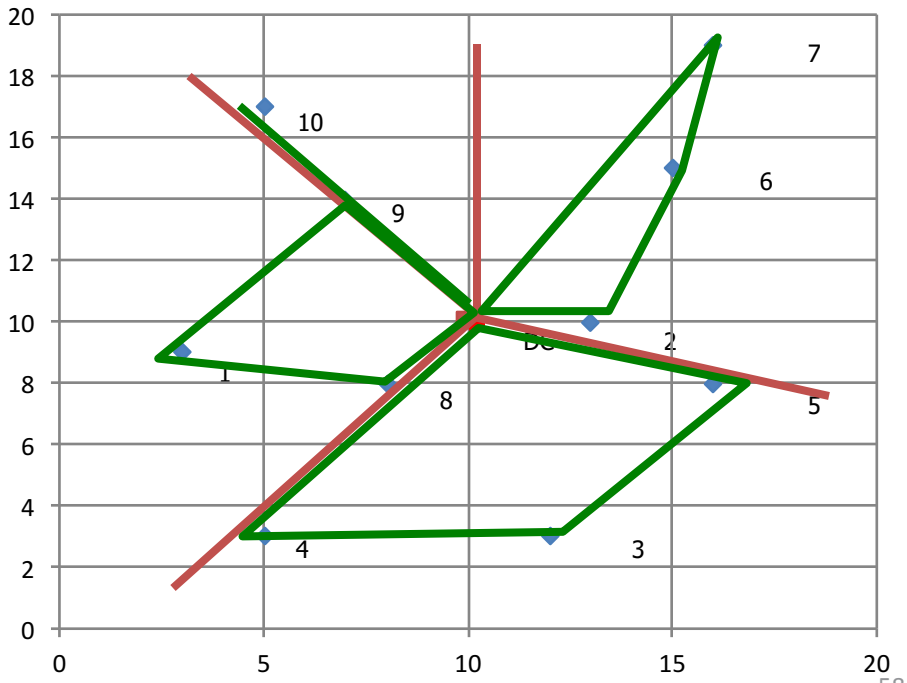
Sweep Heuristic

1. Form a ray from the DC and select an angle and direction (CW vs CCW) to start
2. Select a new vehicle, j, that is empty, w_j=0, and has capacity, c_j.
3. Rotate the ray in selected direction until it hits a customer node, i, or reaches the starting point (go to step 5).
4. If the demand at i (D_i) plus current load already in the vehicle (w_j) is less than the vehicle capacity, add it to the vehicle, w_j=D_i + w_j and go to step 3. Otherwise, close this vehicle, and go to step 2 to start a new tour.
5. Solve the TSP for each independent vehicle tour.

Node ID	Demand (units)
1	22
2	18
3	53
4	54
5	31
6	80
7	78
8	84
9	85
10	42
DC	

j	w_j	cluster$_j$	route$_j$
1	176	7-6-2-1	DC-2-6-7-DC
		23.3 miles	
2	138	5-3-4	DC-5-3-4-DC
		28.3 miles	
3	191	8-1-9	DC-8-1-9-DC
		19.3 miles	
4	42	10	DC-10-DC
		17.2 miles	

4 trucks 88.2 miles



Vehicle Routing Problem – Sweep

Find minimum cost tours from DC to 10 destinations with demand as shown using up to 4 vehicles of capacity of 200 units.

Sweep Heuristic

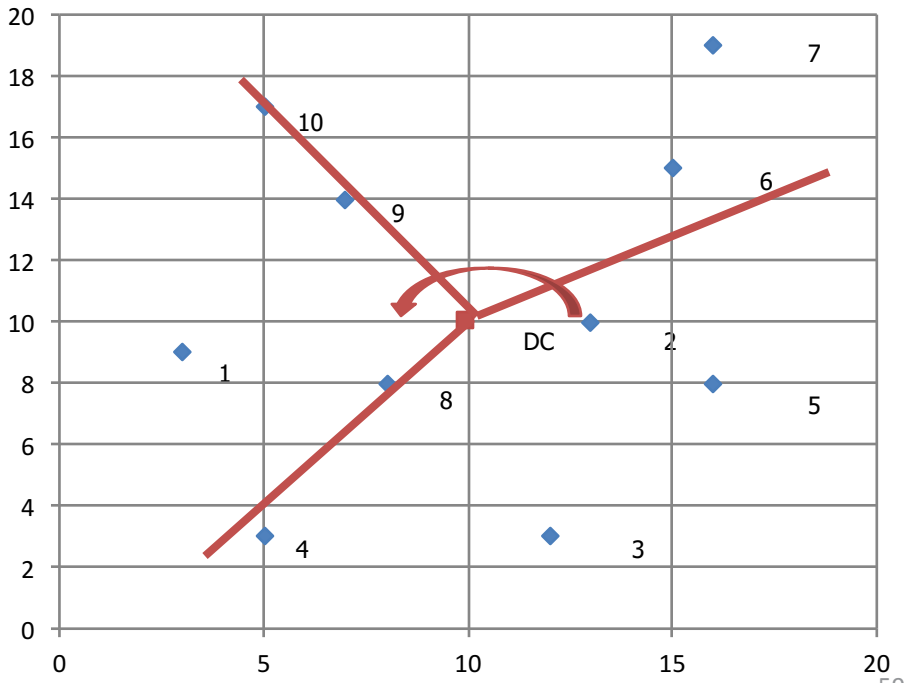
1. Form a ray from the DC and select an angle and direction (CW vs CCW) to start
2. Select a new vehicle, j, that is empty, w_j=0, and has capacity, c_j.
3. Rotate the ray in selected direction until it hits a customer node, i, or reaches the starting point (go to step 5).
4. If the demand at i (D_i) plus current load already in the vehicle (w_j) is less than the vehicle capacity, add it to the vehicle, w_j=D_i + w_j and go to step 3. Otherwise, close this vehicle, and go to step 2 to start a new tour.
5. Solve the TSP for each independent vehicle tour.

Node ID	Demand (units)
1	22
2	18
3	53
4	54
5	31
6	80
7	78
8	84
9	85
10	42
DC	

j	w_j	$route_j$	
1	200	DC-6-7-10-DC	30.6 m
2	191	DC-8-1-9-DC	19.3 m
3	156	DC-2-5-3-4-DC	28.6 m

3 trucks 78.5 miles

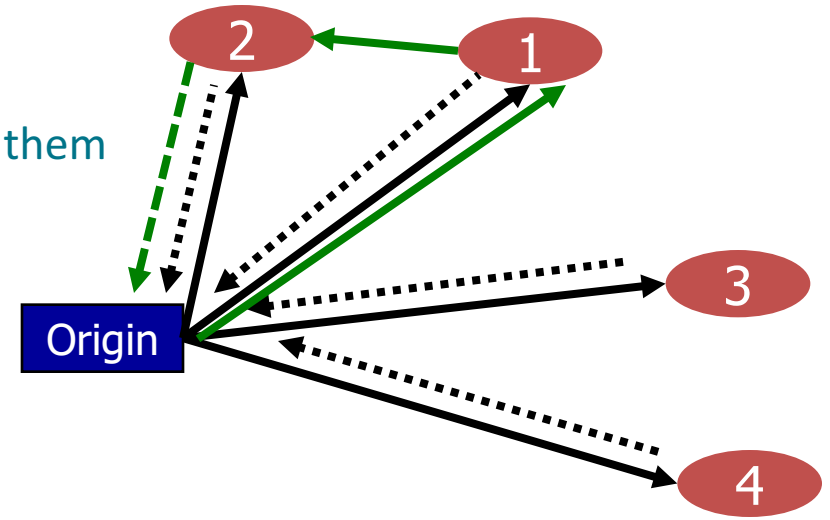
Different starting points and directions can yield different solutions! Best to use a variety or a stack of heuristics.



Clark-Wright Savings Algorithm

General Approach

- Start with a complete solution (out and back)
- Identify nodes to link to form a common tour by calculating the savings:
  - Example: joining node 1 & 2 into a single tour
    - Current tours cost = $2c_{O1} + 2c_{O2}$
    - Joined tour costs = $c_{O1} + c_{12} + c_{2O}$
  - So, if $2c_{O1} + 2c_{O2} > c_{O1} + c_{12} + c_{2O}$ then join them
  - That is: $\mathbf{c_{O1} + c_{2O} - c_{12} > 0}$
- This savings value can be calculated for every pair of nodes
- Run through the nodes pairing the ones with the highest savings first
- Need to make sure vehicle capacity is not violated
- Also, "interior tour" nodes cannot be added – must be on end.



VRP – Clark-Wright Savings

Find minimum cost tours from DC to 10 destinations with demand as shown using <= 4 vehicles of capacity of 200 units.

Savings Heuristic

1. Calculate savings $s_{i,j} = c_{O,i} + c_{O,j} - c_{i,j}$ for every pair (i, j) of demand nodes.
2. Rank and process the savings $s_{i,j}$ in descending order of magnitude.
3. For the savings $s_{i,j}$ under consideration, include arc (i, j) in a route only if:
   - No route or vehicle constraints will be violated by adding it in a route and
   - Nodes i and j are first or last nodes to/from the origin in their current route.
4. If the savings list has not been exhausted, return to Step 3, processing the next entry in the list; otherwise, stop.

Node ID	Demand (units)
1	22
2	18
3	53
4	54
5	31
6	80
7	78
8	84
9	85
10	42
DC	

Example:

$s_{1,9} = c_{O,1} + c_{O,9} - c_{1,9} = 7.1 + 5.0 - 6.4 = 5.7$

Distance Savings	1	2	3	4	5	6	7	8	9	10	DC
1	-	10.0	10.8	6.3	13.0	13.4	16.4	5.1	6.4	8.2	7.1
2	0.1	-	7.1	10.6	3.6	5.4	9.5	5.4	7.2	10.6	3.0
3	3.5	3.2	-	7.0	6.4	12.4	16.5	6.4	12.1	15.7	7.3
4	9.3	1.0	8.9	-	12.1	15.6	19.4	5.8	11.2	14.0	8.6
5	0.4	5.7	7.2	2.8	-	7.1	11.0	8.0	10.8	14.2	6.3
6	0.7	4.7	2.0	0.1	6.3	-	4.1	9.9	8.1	10.2	7.1
7	1.5	4.3	1.6	0	6.1	13.8	-	13.6	10.3	11.2	10.8
8	4.8	0.4	3.7	5.6	1.2	0	0	-	6.1	9.5	2.8
9	5.7	0.8	0.2	2.4	0.5	4.0	5.5	1.7	-	3.6	5.0
10	7.4	1.0	0.2	3.2	0.7	5.5	8.2	1.9	10.0	-	8.6

i-j	s(ij)
7-6	13.8
10-9	10.0
4-1	9.3
4-3	8.9
10-7	8.2
10-1	7.4
5-3	7.2
6-5	6.3
7-5	6.1
5-2	5.7
9-1	5.7
8-4	5.6
9-7	5.5
10-6	5.5
8-1	4.8
6-2	4.7
7-2	4.3
9-6	4.0
8-3	3.7
3-1	3.5
3-2	3.2
10-4	3.2
5-4	2.8
9-4	2.4
6-3	2.0
10-8	1.9
9-8	1.7
7-3	1.6
7-1	1.5
8-5	1.2
4-2	1.0
10-2	1.0

VRP – Clark-Wright Savings

Find minimum cost tours from DC to 10 destinations with demand as shown using <= 4 vehicles of capacity of 200 units.

Node ID	Demand (units)
1	22
2	18
3	53
4	54
5	31
6	80
7	78
8	84
9	85
10	42
DC	

i-j	s(ij)
7-6	13.8
10-9	10.0
4-1	9.3
4-3	8.9
10-7	8.2
10-1	7.4
5-3	7.2
6-5	6.3
7-5	6.1
5-2	5.7
9-1	5.7
8-4	5.6
9-7	5.5
10-6	5.5
8-1	4.8
6-2	4.7
7-2	4.3
9-6	4.0
8-3	3.7
3-1	3.5
3-2	3.2
10-4	3.2
5-4	2.8
9-4	2.4
6-3	2.0
10-8	1.9
9-8	1.7
7-3	1.6
7-1	1.5
8-5	1.2
4-2	1.0
10-2	1.0

Savings Heuristic

1. Calculate savings $s_{i,j} = c_{O,i} + c_{O,j} - c_{i,j}$ for every pair (i, j) of demand nodes.
2. Rank and process the savings $s_{i,j}$ in descending order of magnitude.
3. For the savings $s_{i,j}$ under consideration, include arc (i, j) in a route only if:
   - No route or vehicle constraints will be violated by adding it in a route and
   - Nodes i and j are first or last nodes to/from the origin in their current route.
4. If the savings list has not been exhausted, return to Step 3, processing the next entry in the list; otherwise, stop.

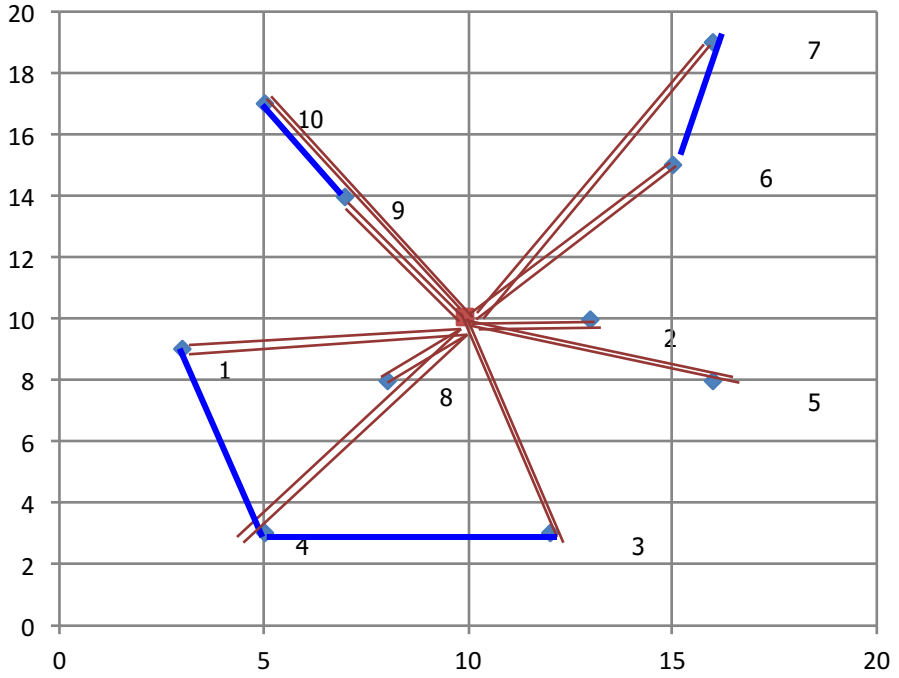
Initial Total Distance = 133.2

Consider joining 7-6
w = 80 + 78= 158 ≤200 **OK**
Distance = 133.2 – 13.8 = 119.4

Consider joining 10-9
w = 85 + 42 = 127 ≤200 **OK**
Distance = 119.4 – 10.0 = 109.4

Consider joining 4-1
w = 54 + 22 = 76 ≤200 **OK**
Distance = 109.4 – 9.3 = 100.1

Consider joining 4-3
w = 76 + 53 = 129 ≤200 **OK**
Distance = 100.1 – 8.9 = 91.2



VRP – Clark-Wright Savings

Find minimum cost tours from DC to 10 destinations with demand as shown using <= 4 vehicles of capacity of 200 units.

Node ID	Demand (units)
1	22
2	18
3	53
4	54
5	31
6	80
7	78
8	84
9	85
10	42
DC	

i-j	s(ij)
~~7-6~~	~~13.8~~
~~10-9~~	~~10.0~~
~~4-1~~	~~9.3~~
~~4-3~~	~~8.9~~
10-7	8.2
10-1	7.4
5-3	7.2
6-5	6.3
7-5	6.1
5-2	5.7
9-1	5.7
8-4	5.6
9-7	5.5
10-6	5.5
8-1	4.8
6-2	4.7
7-2	4.3
9-6	4.0
8-3	3.7
3-1	3.5
3-2	3.2
10-4	3.2
5-4	2.8
9-4	2.4
6-3	2.0
10-8	1.9
9-8	1.7
7-3	1.6
7-1	1.5
8-5	1.2
4-2	1.0
10-2	1.0

Savings Heuristic

1. Calculate savings $s_{i,j} = c_{O,i} + c_{O,j} - c_{i,j}$ for every pair (i, j) of demand nodes.
2. Rank and process the savings $s_{i,j}$ in descending order of magnitude.
3. For the savings $s_{i,j}$ under consideration, include arc (i, j) in a route only if:
   - No route or vehicle constraints will be violated by adding it in a route and
   - Nodes i and j are first or last nodes to/from the origin in their current route.
4. If the savings list has not been exhausted, return to Step 3, processing the next entry in the list; otherwise, stop.

Consider joining 10-7
w = 158 + 127 = 285 >200 **NO**

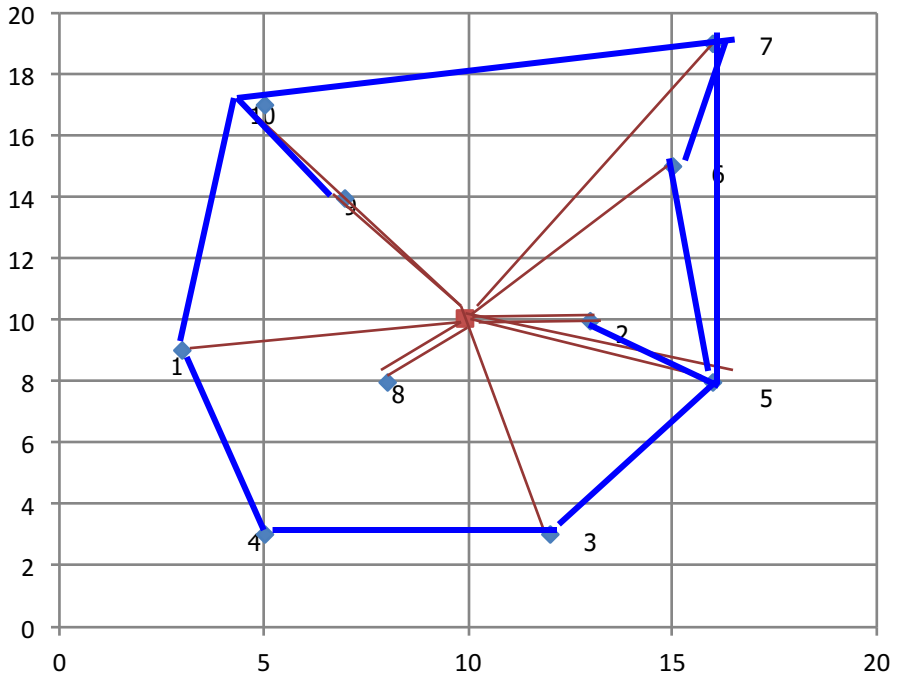
Consider joining 10-1
w = 129 + 127 = 256 >200 **NO**

Consider joining 5-3
w = 129 + 31 = 160 ≤200 **OK**
Distance = 91.2 – 7.2 = 84

Consider joining 6-5
w = 158 + 160 = 318 >200 **NO**

Consider joining 7-5
w = 158 + 160 = 318 >200 **NO**

Consider joining 5-2
w = 160 + 18 = 178 ≤200 **OK**
Distance = 84 – 5.7 = 78.3



VRP – Clark-Wright Savings

Find minimum cost tours from DC to 10 destinations with demand as shown using <= 4 vehicles of capacity of 200 units.

Node ID	Demand (units)
1	22
2	18
3	53
4	54
5	31
6	80
7	78
8	84
9	85
10	42
DC	

i-j	s(ij)
~~7-6~~	~~13.8~~
~~10-9~~	~~10.0~~
~~4-1~~	~~9.3~~
~~4-3~~	~~8.9~~
10-7	8.2
10-1	7.4
5-3	7.2
6-5	6.3
7-5	6.1
5-2	5.7
9-1	5.7
8-4	5.6
9-7	5.5
10-6	5.5
8-1	4.8
6-2	4.7
7-2	4.3
9-6	4.0
8-3	3.7
3-1	3.5
3-2	3.2
10-4	3.2
5-4	2.8
9-4	2.4
6-3	2.0
10-8	1.9
9-8	1.7
7-3	1.6
7-1	1.5
8-5	1.2
4-2	1.0
10-2	1.0

Savings Heuristic

1. Calculate savings $s_{i,j} = c_{O,i} + c_{O,j} - c_{i,j}$ for every pair (i, j) of demand nodes.
2. Rank and process the savings $s_{i,j}$ in descending order of magnitude.
3. For the savings $s_{i,j}$ under consideration, include arc (i, j) in a route only if:
   - No route or vehicle constraints will be violated by adding it in a route and
   - Nodes i and j are first or last nodes to/from the origin in their current route.
4. If the savings list has not been exhausted, return to Step 3, processing the next entry in the list; otherwise, stop.

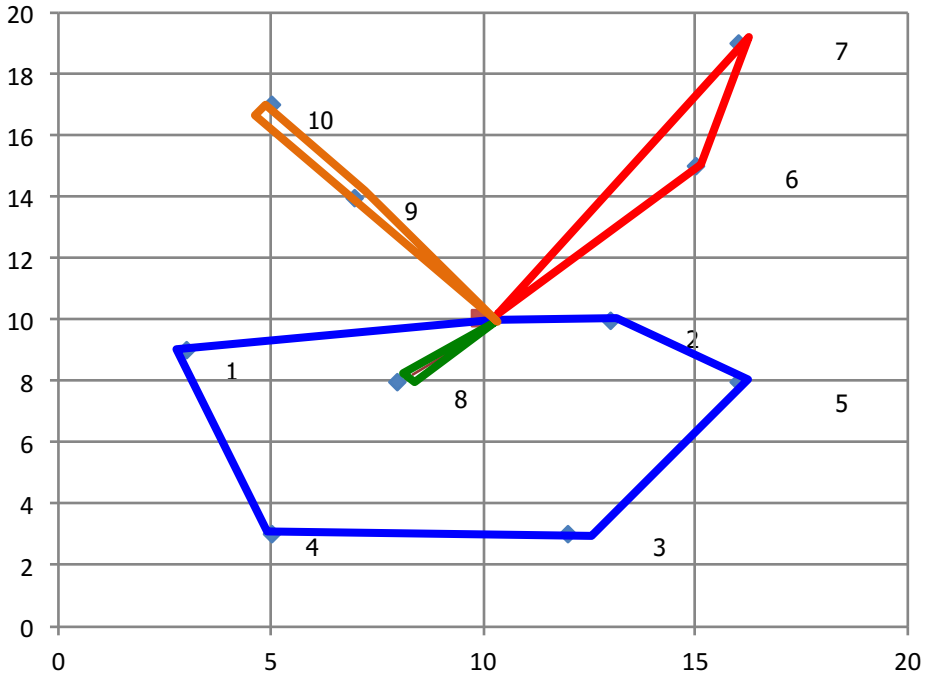
Final Solution:
4 Routes for total 78.3 miles

Blue Route
DC-1-4-3-5-2-DC
178 units @ 33.4 miles

Red Route
DC-6-7-DC
158 units @ 22.0 miles

Orange Route
DC-9-10-DC
127 units @ 17.2 miles

Green Route
DC-8-DC
84 units @ 5.7 miles



Solving VRP with MILP

Solving VRP with MILP

- MILP used to select routes
  - Each column is a route
  - Each row is a node/stop
  - Total cost of each route is included
- Potential routes are an input
- Can consider different costs, not just distance

$$Min \sum_j C_j Y_j$$

$$s.t.$$

$$\sum_{j=1}^{J} a_{ij} Y_j \geq D_i \quad \forall i$$

$$\sum_{j=1}^{J} Y_j \leq V$$

$$Y_j = \{0,1\} \quad \forall j$$

Indices

Demand nodes i

Vehicle routes j

Input Data

C_j = Total cost of route j ($)

D_i = Demand at node i (# of visits)

V = Maximum vehicles

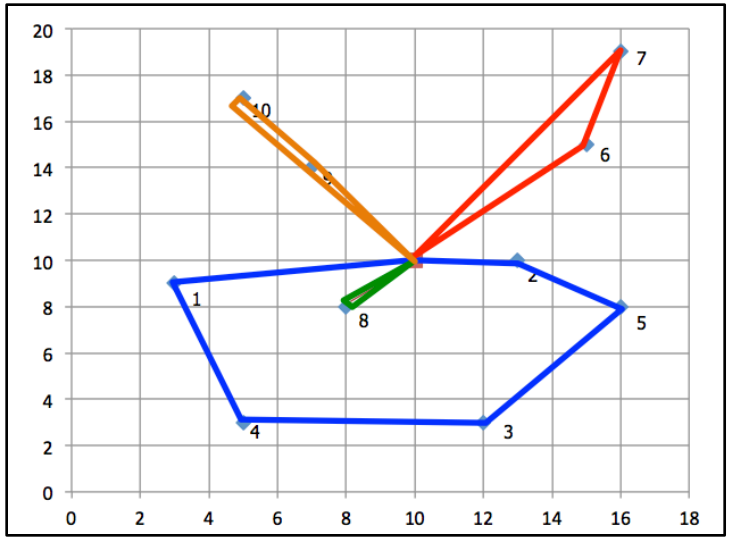
a_{ij} = 1 if node i is in route j;
= 0 otherwise

Decision Variables

Y_j = 1 if route j is used,
=0 otherwise

Total Cost																						
$ 133.2				$ 1.00	Cost per mile																	
				$ -	Cost per vehicle																	
Dec Var	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0				
	Route 1	Route 2	Route 3	Route 4	Route 5	Route 6	Route 7	Route 8	Route 9	Route 10	Route 11	Route 12	Route 13	Route 14	Route 15	Route 16	Route 17	Route 18	Sum		RHS
Stop 1	1	0	0	0	0	0	0	0	0	0	0	1	1	0	1	0	1	0	1	>=	1
Stop 2	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1	>=	1
Stop 3	0	0	1	0	0	0	0	0	0	0	0	0	1	0	1	0	0	1	1	>=	1
Stop 4	0	0	0	1	0	0	0	0	0	0	0	1	1	0	1	0	0	1	1	>=	1
Stop 5	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	1	1	>=	1
Stop 6	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	1	>=	1
Stop 7	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	1	0	0	1	>=	1
Stop 8	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	1	>=	1
Stop 9	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	1	0	1	>=	1
Stop 10	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	1	0	0	1	>=	1
Vehicles	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	10	<=	10
Capacity	22	18	53	54	31	80	78	84	85	42	158	76	129	127	178	200	191	156			
Distance	14.1	6.0	14.6	17.2	12.6	14.1	21.6	5.7	10.0	17.2	22.0	22.0	33.5	17.2	33.4	31	19.3	28.6			

Solving VRP with MILP



- How many potential routes are there?
  - hint: Permutations of n choose k = n!/(n-k)!
  - 0ver 9.8 million!!!
- In practice, routes are generated in a separate “sub-problem” and fed into the MILP “master problem”
- Very active research area – solving very large scale optimization problems

Total Cost			
$ 78.3		$ 1.00	Cost per mile
		$ -	Cost per vehicle

Dec Var	b 0	b 0	b 0	b 0	b 0	b 0	b 0	b 1	b 0	b 0	b 1	b 0	b 0	b 1	b 1	b 0	b 0	b 0			
	Route 1	Route 2	Route 3	Route 4	Route 5	Route 6	Route 7	Route 8	Route 9	Route 10	Route 11	Route 12	Route 13	Route 14	Route 15	Route 16	Route 17	Route 18	Sum		RHS
Stop 1	1	0	0	0	0	0	0	0	0	0	0	1	1	0	1	0	1	0	1	>=	1
Stop 2	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1	>=	1
Stop 3	0	0	1	0	0	0	0	0	0	0	0	0	1	0	1	0	0	1	1	>=	1
Stop 4	0	0	0	1	0	0	0	0	0	0	0	1	1	0	1	0	0	1	1	>=	1
Stop 5	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	1	1	>=	1
Stop 6	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	1	>=	1
Stop 7	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	1	0	0	1	>=	1
Stop 8	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	1	>=	1
Stop 9	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	1	0	1	>=	1
Stop 10	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	1	0	0	1	>=	1
Vehicles	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	4	<=	≤ 10
Capacity	22	18	53	54	31	80	78	84	85	42	158	76	129	127	178	200	191	156			
Distance	14.1	6.0	14.6	17.2	12.6	14.1	21.6	5.7	10.0	17.2	22.0	22.0	33.5	17.2	33.4	31	19.3	28.6			

Key Points from Lesson

Key Points from Lesson

- Introduced algorithms (*you have already used them, actually . . .*)
- Desired Properties of an Algorithm
  - should be unambiguous,
  - require a defined set of inputs,
  - produce a defined set of outputs, and
  - should terminate and produce a result, always stopping after a finite time.
- Demonstrated different network algorithms
  - Shortest Path Problem
    - Dijkstra's Algorithm – label setting – produces a shortest path tree
  - Traveling Salesman Problem
    - Nearest Neighbor – construction algorithm – very fast – no guarantee of quality
    - 2-Opt – improvement algorithm
  - Vehicle Routing Problem (Cluster First – Route Second)
    - Sweep Heuristic – very fast – clusters and requires TSP in routing
    - Clark-Wright – widely used – builds routes in sequence – can include other considerations
    - MILP – requires tours to be generated – can take a long time to solve

Questions, Comments, Suggestions?
Use the Discussion Forum!

"Dexter & Wilson – always taking the shortest path to their food bowls"

MIT Center for
Transportation & Logistics

caplice@mit.edu
ctl.mit.edu