Chapter 4. Transportation Models

This is an introduction chapter quotation. It is offset three inches to the right.

4.1. Vehicle Routing Systems Classification

Single Origin-Destination Routing

Multiple Origin-Destination Routing

Single Vehicle Roundtrip Routing

Vehicle Routing and Scheduling

4.2. Single Origin and Destination Vehicle Routing

Shortest Path Applications

Driving Instructions

You can request driving instructions between any two addresses in the continental United States from a variety of web sites. To respond to your query, the software must find the shortest path between two points on the underlying street network in the United States. The street network is based on the TIGER files, which are published by the U.S. Census Bureau and are completely revised every ten years on a rotating basis for different areas of the country.

Figure 4.1. Route Planning Software Illustration

Time	Mile	Instruction	For
Summary: 13.9 miles (23 minutes)			
5:00 PM	0	Depart 765 Ferst Dr NW, Atlanta, GA 30318 on Ferst Dr NW (North)	0.3 mi
5:01 PM	0.3	Turn LEFT (North) onto Dalney St NW	0.2 mi
5:02 PM	0.5	Turn RIGHT (East) onto 10th St NW	0.5 mi
5:03 PM	0.9	Turn RIGHT (South) onto Ramp	0.1 mi
5:03 PM	1.1	Merge onto I-75 [I-85] (South)	6.8 mi
5:12 PM	7.8	Continue (South) on I-85	3.6 mi
5:16 PM	11.5	At I-85 Exit 72, turn off onto Ramp	0.4 mi
5:17 PM	11.9	Continue (West) on Airport Blvd [S Terminal Pkwy]	1.0 mi
5:20 PM	12.9	Continue (South-West) on Airport Circle	0.2 mi
5:21 PM	13.1	Bear RIGHT (East) onto N Terminal Pkwy	0.6 mi
5:22 PM	13.7	Turn RIGHT (East) onto Local road(s)	0.2 mi
5:23 PM	13.9	Arrive Hartsfield-Atlanta International Airport	

Figure 4.2. Route Planning Driving Instructions

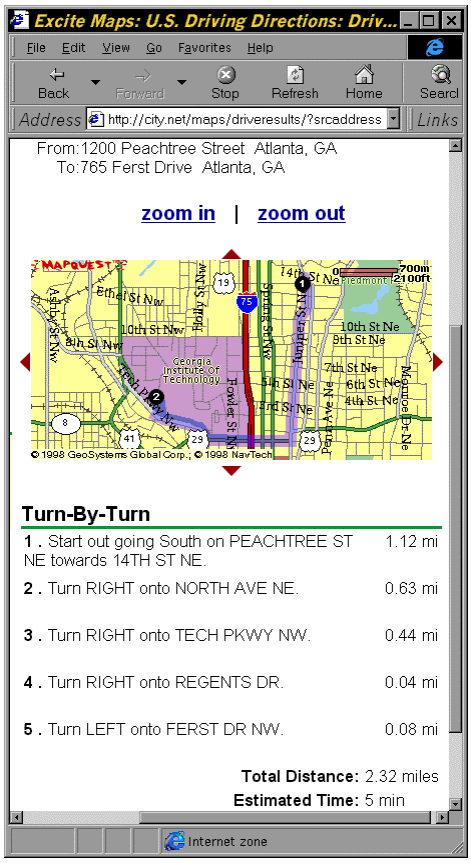


Figure 4.3. Excite Web Site with Shortest Path Driving Instructions

Equipment Replacement

A typical decision to be made in equipment replacement is when to replace a particular machine by a newer model. This decision is usually based on the tradeoff between leasing cost and maintenance cost, where it is assumed that the newer equipment will have a higher leasing cost but a lower maintenance cost and the used equipment will have the opposite cost characteristics. It is also assumed that a machine must be available during the entire planning horizon.

Each machine replacement decision can be represented as an arc from the starting period to the end period of the use of that machine. The least cost equipment replacement schedule can then be found as the shortest path from the start to the end of the planning horizon.

Shortest Path Problem (SPP)

- Network nodes = points to be visited

- Network links = connecting the nodes

Dijkstra's Optimal Algorithm (1959)

100,000 Nodes

Shortest Path Problem (SPP) Variants

One Source to One Sink (s to t)

One Source to All Sinks (s to all)

All Pairs

k Shortest Paths (Sensitivity)

All Non-Negative Costs (Label Setting)

General Costs (Label Correcting)

Longest Path in Acyclic Graphs (PERT and CPM)

Dijkstra's Shortest Path Algorithm

Labeling Algorithms

Temporary Labels = Upper Bound

Permanent Label = Exact Path Length

Reduce Labels by Iterative Procedure

Label Setting

- One temporary label becomes permanent per iteration

Label Correcting

- All temporary labels become permanent at the last iteration

Shortest Path Algorithm Illustration

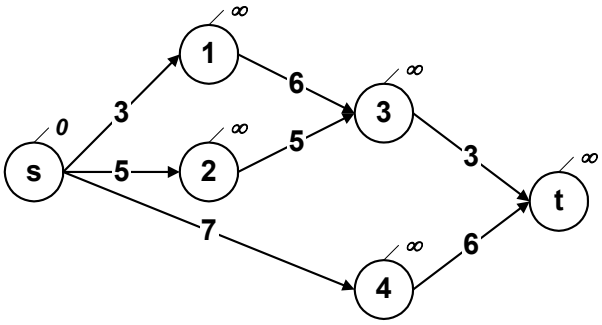


Figure 4.4. Shortest Path Problem

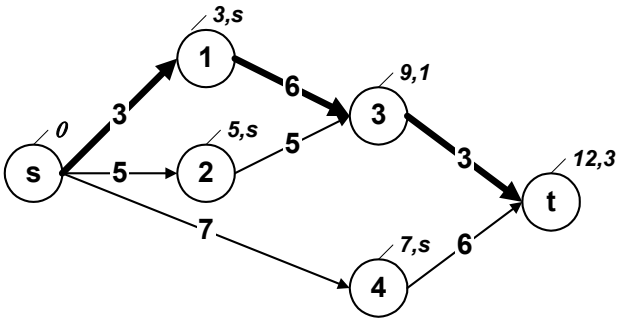


Figure 4.5. Shortest Path Solution

Algorithm Description

1. Set all node labels l(x) = ∞, set l(s) = 0, set all nodes to temporary
2. Find temporary node with min. label l(p)=min{l(x)}
3. For all temporary x ε Γ(p) update labels l(x) = min { l(x), l(p) + c(px)}
4. Mark node p as permanent
5. If all destinations are permanent stop, else go to step 2

Algorithm 4.1. Dijkstra's Shortest Path Algorithm for Dense Graphs

$P = \varnothing,\ T = N,\ l(s) = 0,\ l(i) = \infty\ \ \forall i \in N$

while $|P| < n$ {

$k \leftarrow l(k) = \min\{l(j): j \in T\}$

$P \leftarrow P \cup \{k\},\ T \leftarrow T - \{k\}$

for $j \in \Gamma(k) \cap T$

if $l(j) > l(k) + c_{kj}$

$l(j) = l(k) + c_{kj},\ pred(j) = k$

} *endwhile*

Algorithm Characteristics

Forward Dynamic Programming

Nonnegative Arc "Lengths" or "Costs"

$O(n^2) + O(m)$ or $O(n^2)$ for Fully Dense Graphs

Directed Out-Tree Rooted at s

Node Selection Computationally Most Expensive

100,000 Nodes

Algorithm Example

Table 4.1. Shortest Path Example Distance Matrix

	1	2	3	4	5	6	7	8	9
1		10					3	6	12
2	10		18				2		13
3		18		25		20			7
4			25		5	16	4		
5				5		10			
6			20		10		14	15	9
7		2		4		14			24
8	6				23	15			5
9	12	13				9	24	5	

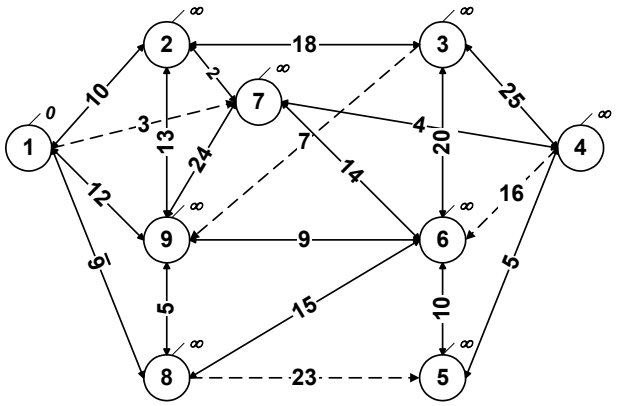


Figure 4.6. Dijkstra's Shortest Path Example Network

The solid edges indicate bi-directional connectors with symmetric distances, the dashed edges indicate asymmetric, one-directional connectors.

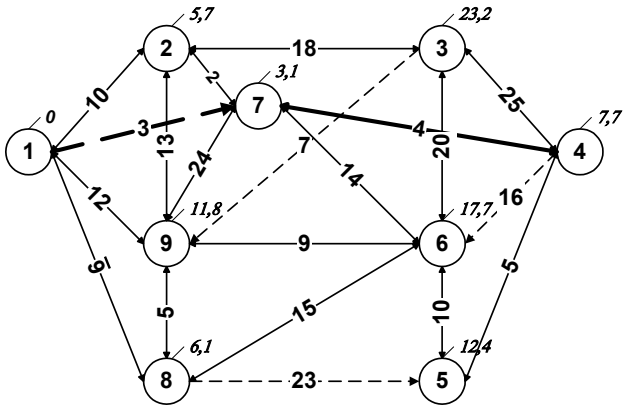


Figure 4.7. Dijkstra's Shortest Path Example Solution

Dijkstra's Algorithm for Sparse Graphs

Heap Implementations (Binary, Fibonacci, Radix,..)

Running Times

O(m+n log n) for Fibonacci Heap

O(m log n) for Binary Heap

Intricate Implementation

Network Flow Mathematical Formulation

$$
\begin{aligned}
Min \quad & \sum_{i}^{N}\sum_{j}^{N} c_{ij}x_{ij} \\
s.t. \quad & \sum_{j}^{N} x_{ij} - \sum_{h}^{N} x_{hi} = b_i \qquad \forall i \\
& 0 \le x_{ij} \le u_{ij} \qquad \forall ij
\end{aligned}
\tag{4.1}
$$

One variable for Each Arc and Commodity (Flow)

One conservation of Flow Constraint for Each Node and Commodity

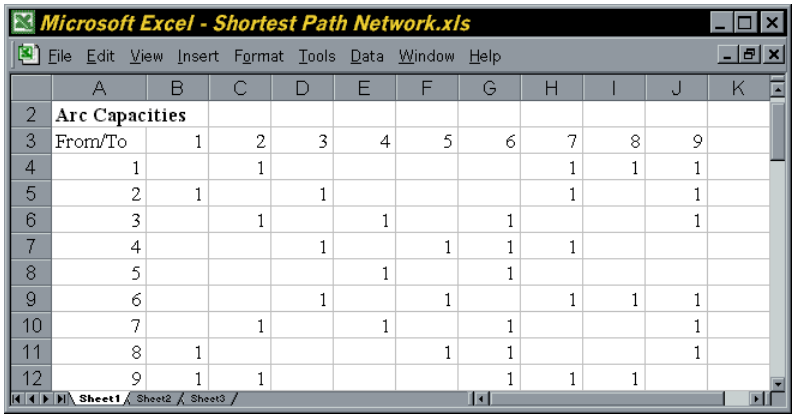
Flows from the Outside (Sign Convention)

Individual and Joint Upper (and Lower) Bounds

The mathematical formulation for the shorted path network shown in Figure 4.4 is given below.

$$
\begin{aligned}
\min \quad & 3x_{s1} + 5x_{s2} + 7x_{s4} + 6x_{13} + 5x_{23} + 3x_{3t} + 6x_{4t} \\
s.t. \quad & 1 - x_{s1} - x_{s2} - x_{s4} = 0 \\
& x_{s1} - x_{13} = 0 \\
& x_{s2} - x_{23} = 0 \\
& x_{13} + x_{23} - x_{3t} = 0 \\
& x_{s4} - x_{4t} = 0 \\
& x_{3t} + x_{4t} - 1 = 0
\end{aligned}
$$

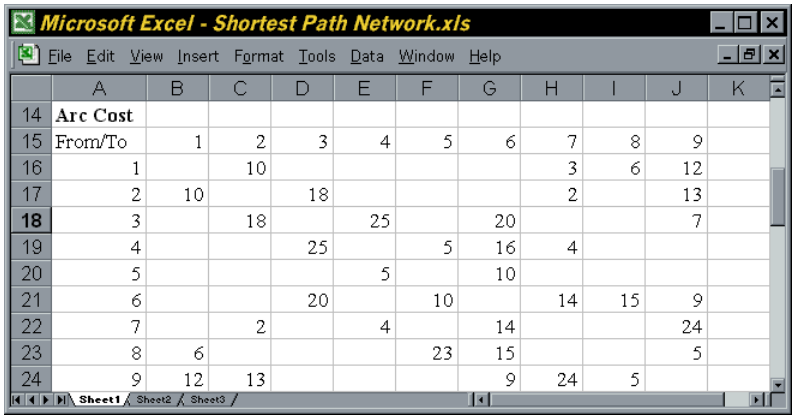
Excel Shortest Path Spreadsheet and Solver



	A	B	C	D	E	F	G	H	I	J	K
2	**Arc Capacities**										
3	From/To	1	2	3	4	5	6	7	8	9	
4	1		1					1	1	1	
5	2	1		1				1		1	
6	3		1		1		1			1	
7	4			1		1	1	1			
8	5				1		1				
9	6			1		1		1	1	1	
10	7		1		1		1			1	
11	8	1				1	1			1	
12	9	1	1				1	1	1		

Sheet1 Sheet2 Sheet3

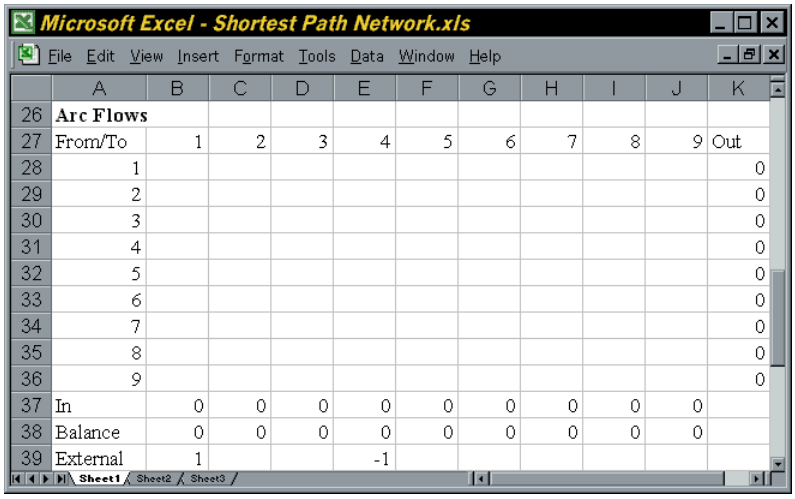
Figure 4.8. Excel Shortest Path Spreadsheet Arc Capacities



	A	B	C	D	E	F	G	H	I	J	K
14	**Arc Cost**										
15	From/To	1	2	3	4	5	6	7	8	9	
16	1		10					3	6	12	
17	2	10		18				2		13	
18	3		18		25		20			7	
19	4			25		5	16	4			
20	5				5		10				
21	6			20		10		14	15	9	
22	7		2		4		14			24	
23	8	6				23	15			5	
24	9	12	13				9	24	5		

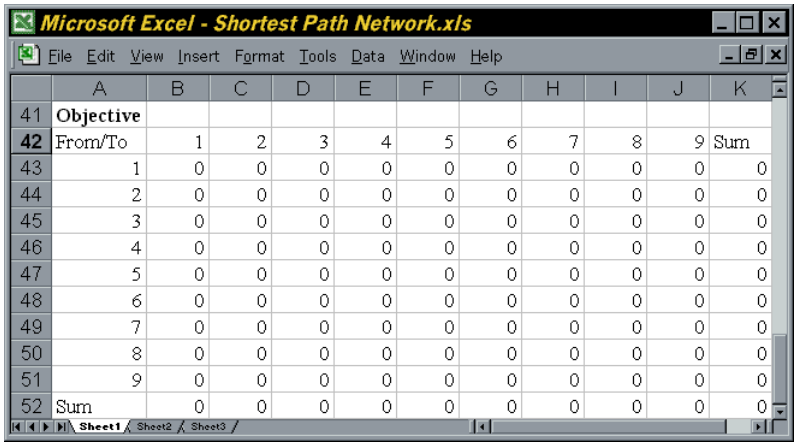
Sheet1 Sheet2 Sheet3

Figure 4.9. Excel Shortest Path Spreadsheet Arc Costs



	A	B	C	D	E	F	G	H	I	J	K
26	**Arc Flows**										
27	From/To	1	2	3	4	5	6	7	8	9	Out
28	1										0
29	2										0
30	3										0
31	4										0
32	5										0
33	6										0
34	7										0
35	8										0
36	9										0
37	In	0	0	0	0	0	0	0	0	0	
38	Balance	0	0	0	0	0	0	0	0	0	
39	External	1			-1						

Figure 4.10. Excel Shortest Path Spreadsheet Initial Flow Balance



	A	B	C	D	E	F	G	H	I	J	K
41	**Objective**										
42	From/To	1	2	3	4	5	6	7	8	9	Sum
43	1	0	0	0	0	0	0	0	0	0	0
44	2	0	0	0	0	0	0	0	0	0	0
45	3	0	0	0	0	0	0	0	0	0	0
46	4	0	0	0	0	0	0	0	0	0	0
47	5	0	0	0	0	0	0	0	0	0	0
48	6	0	0	0	0	0	0	0	0	0	0
49	7	0	0	0	0	0	0	0	0	0	0
50	8	0	0	0	0	0	0	0	0	0	0
51	9	0	0	0	0	0	0	0	0	0	0
52	Sum	0	0	0	0	0	0	0	0	0	0

Figure 4.11. Excel Shortest Path Spreadsheet Initial Objective Function

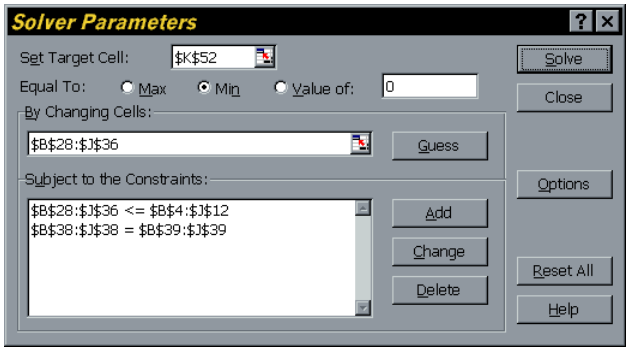


Figure 4.12. Excel Shortest Path Spreadsheet Solver Parameters

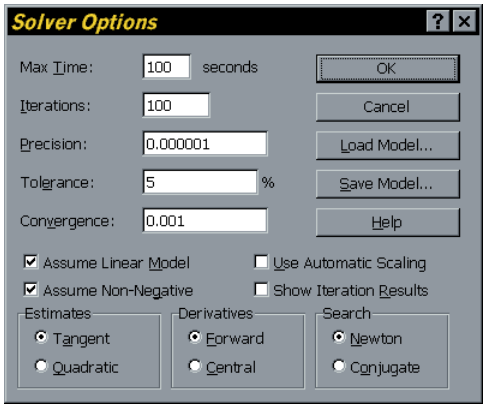


Figure 4.13. Excel Shortest Path Spreadsheet Solver Options

Microsoft Excel - Shortest Path Network.xls

File Edit View Insert Format Tools Data Window Help

	A	B	C	D	E	F	G	H	I	J	K
26	**Arc Flows**										
27	From/To	1	2	3	4	5	6	7	8	9	Out
28	1	0	0	0	0	0	0	1	0	0	1
29	2	0	0	0	0	0	0	0	0	0	0
30	3	0	0	0	0	0	0	0	0	0	0
31	4	0	0	0	0	0	0	0	0	0	0
32	5	0	0	0	0	0	0	0	0	0	0
33	6	0	0	0	0	0	0	0	0	0	0
34	7	0	0	0	1	0	0	0	0	0	1
35	8	0	0	0	0	0	0	0	0	0	0
36	9	0	0	0	0	0	0	0	0	0	0
37	In	0	0	0	1	0	0	1	0	0	
38	Balance	1	0	0	-1	0	0	0	0	0	
39	External	1			-1						

Sheet1 Sheet2 Sheet3

Figure 4.14. Excel Shortest Path Spreadsheet Solution Flows

Microsoft Excel - Shortest Path Network.xls

File Edit View Insert Format Tools Data Window Help

	A	B	C	D	E	F	G	H	I	J	K
41	**Objective**										
42	From/To	1	2	3	4	5	6	7	8	9	Sum
43	1	0	0	0	0	0	0	3	0	0	3
44	2	0	0	0	0	0	0	0	0	0	0
45	3	0	0	0	0	0	0	0	0	0	0
46	4	0	0	0	0	0	0	0	0	0	0
47	5	0	0	0	0	0	0	0	0	0	0
48	6	0	0	0	0	0	0	0	0	0	0
49	7	0	0	0	4	0	0	0	0	0	4
50	8	0	0	0	0	0	0	0	0	0	0
51	9	0	0	0	0	0	0	0	0	0	0
52	Sum	0	0	0	4	0	0	3	0	0	7

Sheet1 Sheet2 Sheet3

Figure 4.15. Excel Shortest Path Spreadsheet Solution Objective Function

Nissen (1999) created a JAVA applet implementing Dijstra's Shortest Path algorithm that nicely illustrates the dynamic programming progression. This applet is located on the personal home page of this student in computer science, so the applet may not be available in the future.

Directed Graph Representations

1. Adjacency Matrix
2. Successor List
3. Successor Linked List

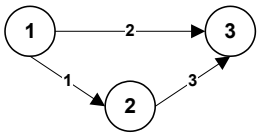


Figure 4.16. Directed Graph Example

Adjacency Matrix

A[i, j] Present If Edge From Node i to Node j

Characteristics

n^2 Storage Memory

Constant O(1) Lookup, Addition, Deletion

Linear O(n) Successor and Predecessor Loops

	1	2	3
1		1	2
2			3
3			

Figure 4.17. Directed Graph Adjacency Matrix Example

Successor Array

Head or First Arc Array.

This array contains the index of the row in the arc array for the first outgoing arc for each node, respectively.

To-Node Array with Every Arc

This array is stored in a single monolithic area of the computer memory.

Characteristics

n + 2m Memory Storage

Linear O(n) Lookup, Linear O(n) Insertion and Deletion

The linear effort O(n) for the insertion and deletion of an arc is generated by the work required to find the arc for deletion or the place for the new arc for insertion. The actual insertion and deletion then requires the extension or contraction of the arc array and the movement of all the rows below the insertion or deletion row. The elements in the head array, which are the row indices into the arc array, for the current origin node and below also have to be adjusted. Since this effort is constant with respect to the number of nodes in the network or O(1) it is ignored in the computational complexity characterization. But the actual work involved is usually much larger than finding the proper place in the arc array. This is an example where for all practical networks it is more important to include the lower power terms and their coefficients in the computational complexity polynomial.

The amount of work involved in inserting or deleting an arc in the middle of the arc array is one of the major disadvantages of this particular data structure for the representation of directed graphs.

Linear O(n) Successor Loop, Linear O(m) Predecessor Loop

Head
1
3
4
4

To	Arc
2	1
3	2
3	3

Figure 4.18. Directed Graph Successor Array Example

Successor Double Linked List

Head or First Arc Array

This array contains the memory address of the row in the arc array for the first outgoing arc for each node, respectively.

To-Node Double Linked List With Every Arc

The predecessor (prev) and successor (next) elements are memory addresses of the preceding or succeeding arc in the network. The memory for each arc is allocated and released on an individual arc basis.

Characteristics

n + 4m Memory Storage

Linear O(n) Lookup, Constant O(1) Insertion, Linear O(n) Deletion

The constant effort O(1) for the insertion and the linear effort O(n) for the deletion of an arc are generated by the work required to find the arc for deletion or the place for the new arc for insertion. The actual insertion and deletion then requires the allocation or release of the memory for a single arc. For insertion the memory address for the origin node has to be modified in the head array and it may have to be modified for arc deletion if the deleted arc was the first outgoing arc from this node. But no other addresses in the head array have to be modified. Since this effort is constant with respect to the number of nodes in the network or O(1) it is ignored in the computational complexity characterization.

Linear O(n) Successor Loop, Linear O(m) Predecessor Loop

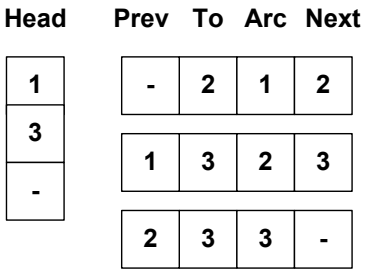


Figure 4.19. Directed Graph Successor List Example

Element Deletion

k to be Deleted Element

- If Pred(k) then Succ(Pred(k)) = Succ(k)
- If Succ(k) then Pred(Succ(k)) = Pred(k)

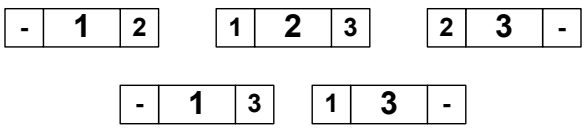


Figure 4.20. Double Linked List Deletion

Element Insertion

k to be Inserted Element after p

- Succ(k) = Succ(p), Pred(k)=p, Succ(p)=k
- If Succ(k) then Pred(Succ(k))=k

k to be inserted Element before s

- Pred(k)=Pred(s), Succ(k)=s, Pred(s)=k
- if Pred(k) then Succ(Pred(k))=k

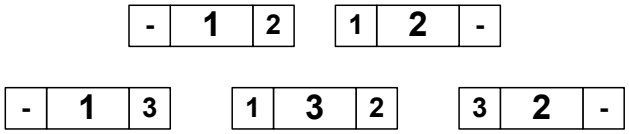


Figure 4.21. Double Linked List Insertion

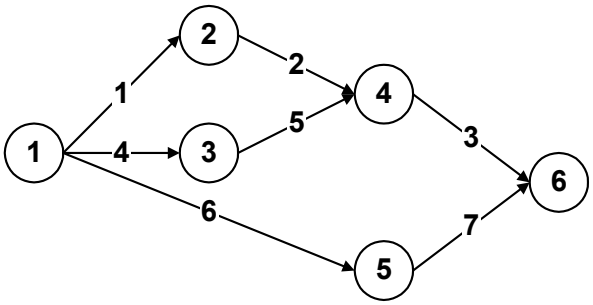


Figure 4.22. Directed Graph Example

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

Figure 4.23. Directed Graph Example Adjacency Matrix

Head		To	Arc
1		2	1
4		3	4
5		5	6
6		4	2
7		4	5
8		6	3
8		6	7

Figure 4.24. Directed Graph Example Successor Array

Head		List			
1		-	2	1	2
4		1	3	4	3
5		2	5	6	4
6		3	4	2	5
7		4	4	5	6
-		5	6	3	7
		6	6	7	-

Figure 4.25. Directed Graph Example Successor Linked List

For further reference see Horowitz and Shani (1984), Sedgewick (1983), and Aho et al. (1983).

4.3. Multiple Origin and Destination Vehicle Routing

Introduction

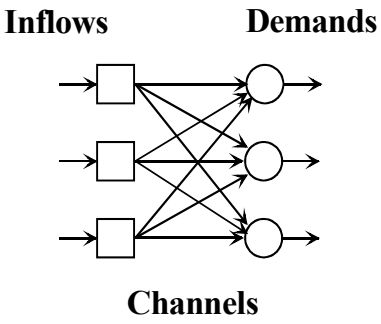


Figure 4.26. Multiple Origin and Destination Network Illustration

Max Flow Network Flow Problem

- Capacity models, public sector

Min Cost Network Flow Problem

- Economic models, private industry

Network Simplex Algorithm

100,000 Channels

Network Variants

Transportation

Transshipment

Min Cut - Max Flow Network

Min Cost Network

Multicommodity Network

Generalized Networks

Network Properties

Unimodularity Property

The unimodularity property, also called the integrality property, states that if the all the external flows and the arc capacities of a network problem are integer numbers, then the optimal solution to this network flow problem will consist of all integer flows. It should be noted, that the unimodularity or integrality property does not hold for generalized networks.

Applications

Tactical Production-Distribution Planning

Parameters and Variables

Products p

Customers j, demand dem_{jp}

Plants i, capacity cap_i

Marginal production cost a_{ip} and resource consumption req_{ip}

Transportation cost c_{ijp} and quantity x_{ijp}

The standard network formulation does not allow having capacities or costs on the flow through nodes, only capacities or costs for flows through arcs. A standard modeling technique to avoid this limitation is to split the original node up into two nodes connected by a single arc. On this arc a flow capacity and flow cost can then be specified. This technique is illustrated in the following figure for the tactical production-distribution planning problem.

Illustration

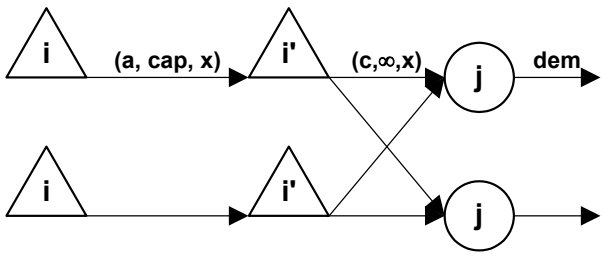


Figure 27. Tactical Production-Distribution Planning Network

Model

$$
\begin{aligned}
Min \quad & \sum_{i=1}^{F}\sum_{j=1}^{C}\sum_{p=1}^{P}\left(a_{ip}+c_{ijp}\right)x_{ijp} \\
s.t. \quad & \sum_{i=1}^{F} x_{ijp} = dem_{jp} && \forall jp \\
& \sum_{j=1}^{C}\sum_{p=1}^{P} req_{ip}\,x_{ijp} \le cap_i && \forall i \\
& x_{ijp} \ge 0
\end{aligned}
\tag{4.2}
$$

If all req_{ip} are equal and can thus be set equal to one, the above formulation is called a network. If not all req_{ip} are equal, the formulation is called a generalized network. The solution times for solving a generalized network are significantly larger than for solving a network of equivalent size. In addition, the integrality property does not longer hold for generalized networks and the optimal solution flows may be fractional.

Operator Scheduling

Parameters and Variables

Time periods with coverage requirements b_i

Operator shifts with costs c_i that cover consecutive time periods

Number of operators for each shift x_i

Model

$$
\begin{aligned}
Min \quad & cx \\
s.t. \quad & Ax \ge b \\
& x \ge 0
\end{aligned}
\tag{4.3}
$$

$$
\begin{bmatrix}
0 & 1 & 0 & 1 & 1 \\
1 & 1 & 0 & 0 & 1 \\
1 & 1 & 0 & 0 & 1 \\
1 & 1 & 0 & 0 & 0 \\
1 & 1 & 1 & 0 & 0
\end{bmatrix} x \ge \begin{bmatrix} 5 \\ 12 \\ 8 \\ 10 \\ 4 \end{bmatrix}
$$

Transformation

Consecutive Ones in Each Column

Add Negative Identity Matrix (Row Surplus Variable)

Add "Zero" Row (Node N+1 Flow Balance Constraint)

Linear Row Operation

For r = N Down To 1

Row[r+1] = Row[r+1] - Row[r]

$$\begin{bmatrix} 0 & 1 & 0 & 1 & 1 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & -1 & 0 & 1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \\ -1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 5 \\ 7 \\ -4 \\ 2 \\ -6 \\ -4 \end{bmatrix}$$

Illustration

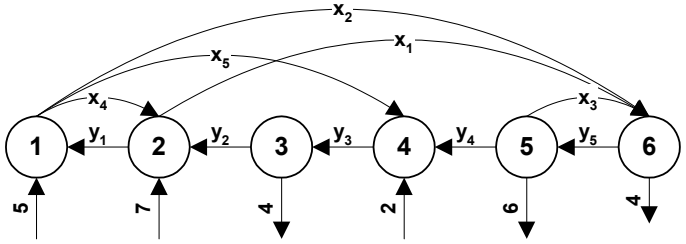


Figure 4.28. Operator Scheduling Network

Successive Shortest Path Algorithm

Algorithm Description

1. Start all flows $x = 0$, all node potentials $\pi = 0$

2. Construct incremental/residual graph

   Same nodes as original graph

   If $x_{ij} > 0$ then add artificial arc ji

   If $x_{ij} = u_{ij}$ then eliminate arc ij or $d_{ij} = \infty$

   If $x_{ij} < u_{ij}$ then $d_{ij} = c_{ij} - \pi_i + \pi_j$

   If $x_{ij} > 0$ then $d_{ji} = -c_{ij} - \pi_j + \pi_i = -d_{ij}$

   Add super source and super sink nodes, add arcs from the super source to the sources and from the sinks to the super sink with cost equal to zero and capacity equal to the remaining supply and demand, respectively.

3. Find shortest path from source to sink

Find shortest path to any sink node k with Dijkstra's algorithm

If no such path, stop, network flow problem is infeasible

4. Compute maximum flow change on the shortest path

on backflow arcs, $\delta_{ij} = x_{ji}$

on regular arcs, $\delta_{ij} = u_{ij} - x_{ij}$

$\delta = \min\{\delta_{ij}\}$

5. Augment flow on the shortest path

on backflow arcs, $x_{ji} = x_{ji} - \delta$

on regular arcs, $x_{ij} = x_{ij} + \delta$

update remaining supply and demand

6. If all remaining demands are zero, stop, network is optimal

7. Update node potentials

k = shortest path sink node and SPL_i is the shortest path length to node i

if node i is permanent, $\pi_i = \pi_i - SPL_i$

if node i is temporary, $\pi_i = \pi_i - SPL_k$

8. Go to step 2

Algorithm Illustration

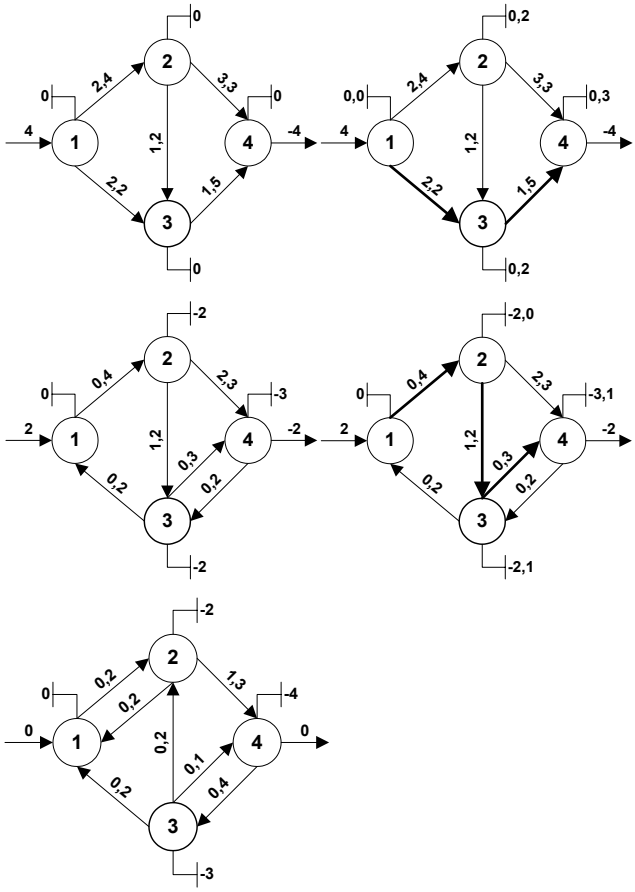


Figure 4.29. Successive Shortest Path Algorithm Illustration

Algorithm Example

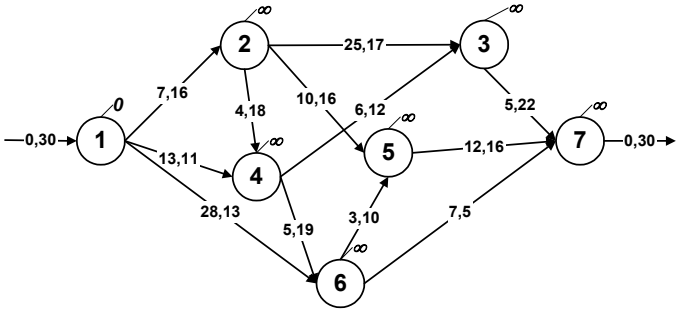


Figure 4.30. Successive Shortest Path Example Data

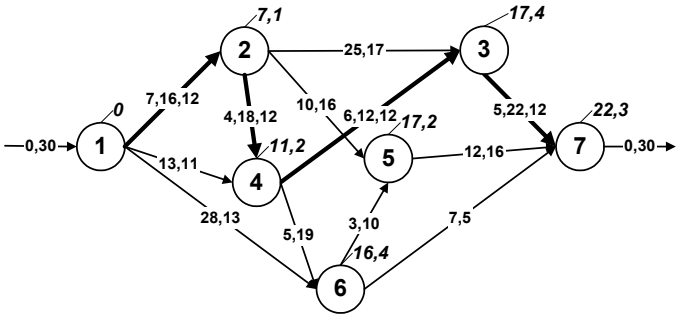


Figure 4.31. Successive Shortest Path Example First Shortest Path

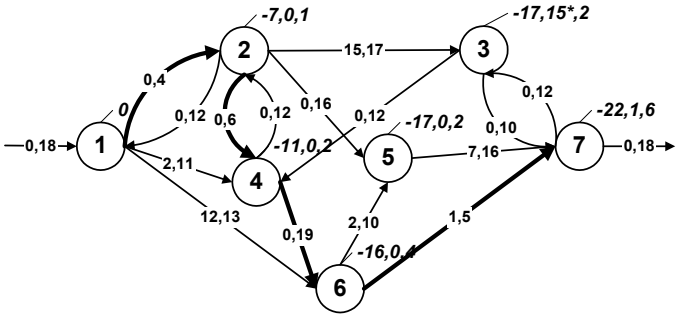


Figure 4.32. Successive Shortest Path Example Second Shortest Path

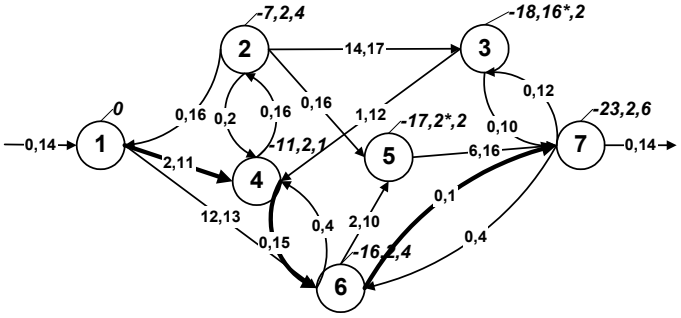


Figure 4.33. Successive Shortest Path Example Third Shortest Path

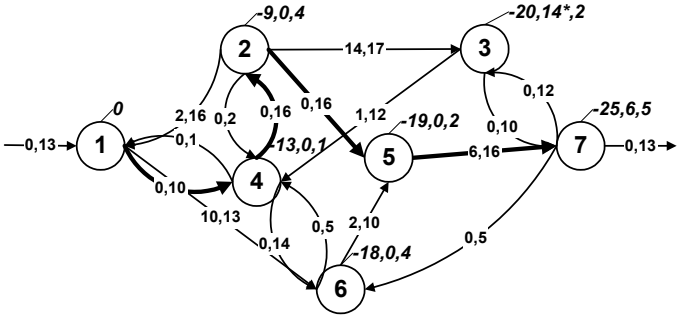


Figure 4.34. Successive Shortest Path Example Fourth Shortest Path

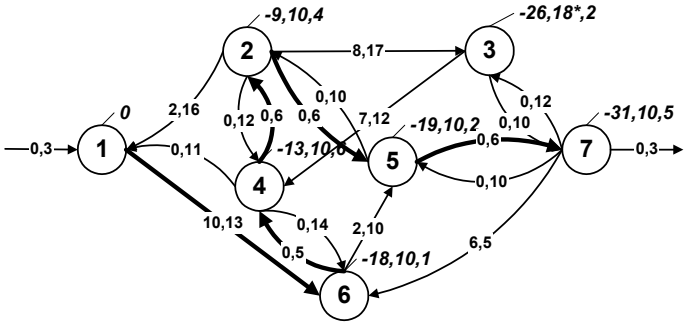


Figure 4.35. Successive Shortest Path Example Fifth Shortest Path

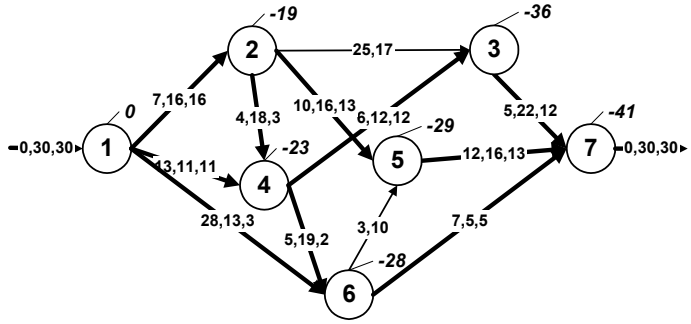
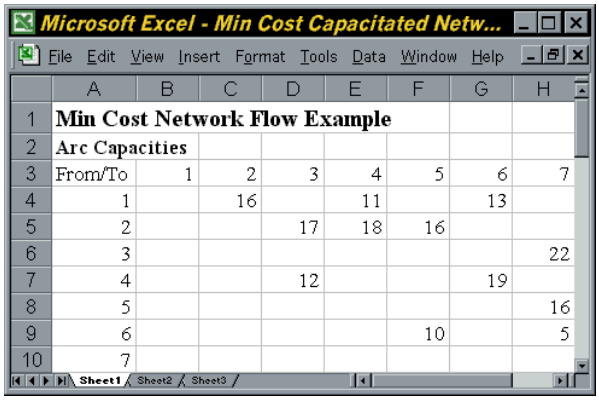


Figure 4.36. Successive Shortest Path Example Sixth Shortest Path



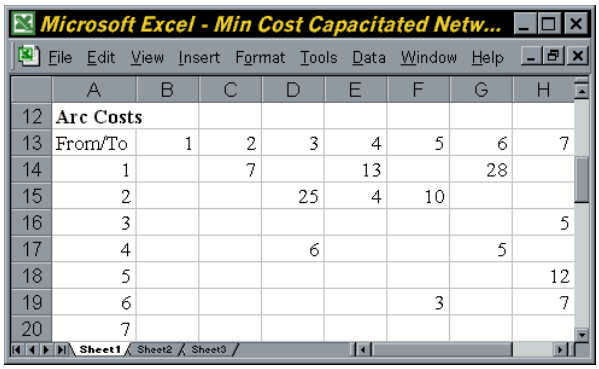
Microsoft Excel - Min Cost Capacitated Netw...

File Edit View Insert Format Tools Data Window Help

	A	B	C	D	E	F	G	H
1	**Min Cost Network Flow Example**							
2	Arc Capacities							
3	From/To	1	2	3	4	5	6	7
4	1		16		11		13	
5	2			17	18	16		
6	3							22
7	4			12			19	
8	5							16
9	6					10		5
10	7							

Sheet1 Sheet2 Sheet3

Figure 4.37. Excel Spreadsheet for Minimum Cost Network Example Arc Capacities



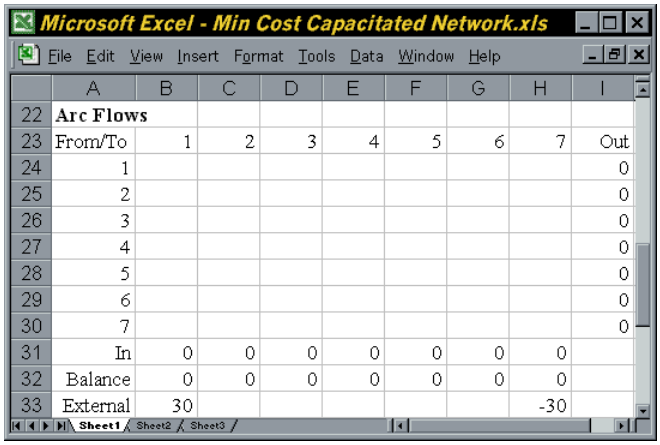
Microsoft Excel - Min Cost Capacitated Netw...

File Edit View Insert Format Tools Data Window Help

	A	B	C	D	E	F	G	H
12	Arc Costs							
13	From/To	1	2	3	4	5	6	7
14	1		7		13		28	
15	2			25	4	10		
16	3							5
17	4			6			5	
18	5							12
19	6					3		7
20	7							

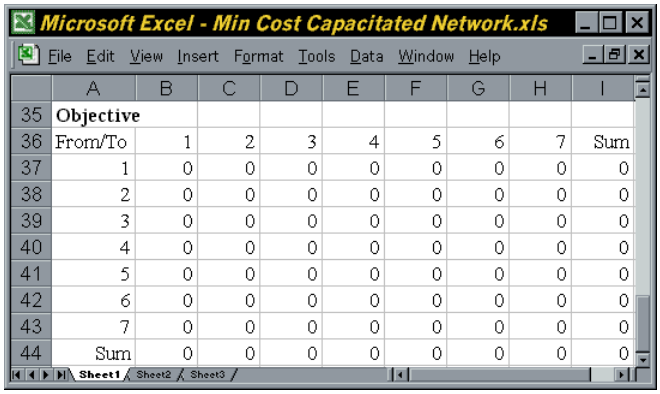
Sheet1 Sheet2 Sheet3

Figure 4.38. Excel Spreadsheet for Minimum Cost Network Example Arc Costs



	A	B	C	D	E	F	G	H	I
22	**Arc Flows**								
23	From/To	1	2	3	4	5	6	7	Out
24	1								0
25	2								0
26	3								0
27	4								0
28	5								0
29	6								0
30	7								0
31	In	0	0	0	0	0	0	0	
32	Balance	0	0	0	0	0	0	0	
33	External	30						-30	

Figure 4.39. Excel Spreadsheet for Minimum Cost Network Example Initial Zero Flows



	A	B	C	D	E	F	G	H	I
35	**Objective**								
36	From/To	1	2	3	4	5	6	7	Sum
37	1	0	0	0	0	0	0	0	0
38	2	0	0	0	0	0	0	0	0
39	3	0	0	0	0	0	0	0	0
40	4	0	0	0	0	0	0	0	0
41	5	0	0	0	0	0	0	0	0
42	6	0	0	0	0	0	0	0	0
43	7	0	0	0	0	0	0	0	0
44	Sum	0	0	0	0	0	0	0	0

Figure 4.40. Excel Spreadsheet for Minimum Cost Network Example Initial Zero Objective

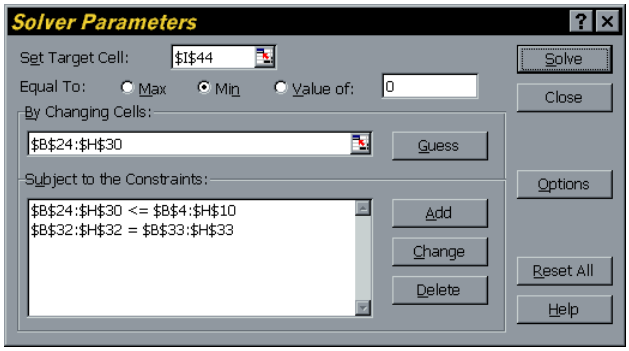
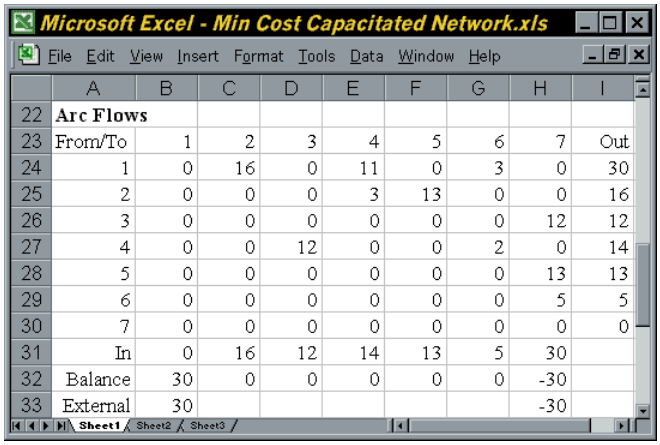


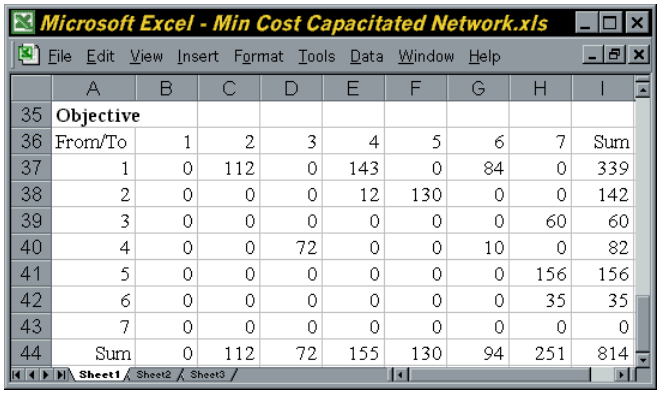
Figure 4.41. Excel Spreadsheet for Minimum Cost Network Example Solver

The flow balance is modeled as an equality constraint. If there is more supply available then there is demand, the flow balance equality constraint is replaced with a less-than-or-equal constraint. This assumes that external inflows (supplies) have a positive sign and external outflows (demands) have a negative sign. An alternative method is to introduce an artificial sink node with as demand the difference between the total supply and the total demand in the network. All source nodes are then connected to this artificial sink node with flow arcs with zero cost and infinite capacity. Since the network is then again balanced, the flow balance is modeled as an equality constraint.



	A	B	C	D	E	F	G	H	I
22	**Arc Flows**								
23	From/To	1	2	3	4	5	6	7	Out
24	1	0	16	0	11	0	3	0	30
25	2	0	0	0	3	13	0	0	16
26	3	0	0	0	0	0	0	12	12
27	4	0	0	12	0	0	2	0	14
28	5	0	0	0	0	0	0	13	13
29	6	0	0	0	0	0	0	5	5
30	7	0	0	0	0	0	0	0	0
31	In	0	16	12	14	13	5	30	
32	Balance	30	0	0	0	0	0	-30	
33	External	30						-30	

Figure 4.42. Excel Spreadsheet for Minimum Cost Network Example Solution Flows



	A	B	C	D	E	F	G	H	I
35	**Objective**								
36	From/To	1	2	3	4	5	6	7	Sum
37	1	0	112	0	143	0	84	0	339
38	2	0	0	0	12	130	0	0	142
39	3	0	0	0	0	0	0	60	60
40	4	0	0	72	0	0	10	0	82
41	5	0	0	0	0	0	0	156	156
42	6	0	0	0	0	0	0	35	35
43	7	0	0	0	0	0	0	0	0
44	Sum	0	112	72	155	130	94	251	814

Figure 4.43. Excel Spreadsheet for Minimum Cost Network Example Solution Objective

Minimum Cost Network Flow Formulation

Primal formulation

Parameters and Variables

x_{ij} flow from node *i* (origin) to node *j* (destination)

c_{ij} cost of transporting one unit of flow from node *i* (origin) to node *j* (destination)

b_i external flow for node *i*, with the sign convention that external inflows (supply) are positive and external outflows (demand) are negative.

u_{ij} arc capacity or upper bound on the flow from node *i* (origin) to node *j* (destination)

Model

Formulation 4.1. Minimum Cost Network Flow Formulation

$$
\begin{array}{lllll}
Min & \sum_{i}^{N}\sum_{j}^{N} c_{ij}x_{ij} & & & \\
s.t. & -\sum_{h}^{N} x_{hi} + \sum_{j}^{N} x_{ij} = b_i & \forall i & [\pi_i] & (4.4) \\
 & 0 \le x_{ij} \le u_{ij} & \forall ij & [\alpha_{ij}] &
\end{array}
$$

Residual Network Construction

$arc(i,j)$ *with* c_{ij} *and* u_{ij}

if $x_{ij} > 0$ *then*

$arc(i,j)$ $c_{ij}^{\pi} = c_{ij}^{orig} + \pi_j - \pi_i$ $\quad r_{ij} = u_{ij} - x_{ij}$

$arc(j,i)$ $c_{ji}^{\pi} = -c_{ij}^{\pi}$ $\quad r_{ji} = x_{ij}$

Optimality Conditions

Reduced costs based on shortest path labels.

$$
\begin{array}{ll}
d_j \le d_i + c_{ij} & \forall (i,j) \\
c_{ij}^{d} = c_{ij} + d_i - d_j \ge 0 & \forall (i,j)
\end{array}
\qquad (4.5)
$$

Reduced costs based on node potentials.

$$
c_{ij}^{\pi} = c_{ij} - \pi_i + \pi_j \ge 0 \qquad \forall (i,j) \qquad (4.6)
$$

Dual formulation

Standardized Primal Formulation

$$
\begin{array}{lllll}
Min & \sum_{i}^{N}\sum_{j}^{N} c_{ij}x_{ij} & & & \\
s.t. & -\sum_{h}^{N} x_{hi} + \sum_{j}^{N} x_{ij} = b_i & \forall i & [\pi_i] & (4.7) \\
 & -x_{ij} \ge -u_{ij} & \forall ij & [\alpha_{ij}] &
\end{array}
$$

Dual formulation

$$\begin{array}{lll} \max & \sum_i^N b_i \pi_i - \sum_i^N \sum_j^N u_{ij} \alpha_{ij} & \\ s.t. & \pi_i - \pi_j - \alpha_{ij} \le c_{ij} & \forall (i,j) \qquad [x_{ij}] \\ & \alpha_{ij} \ge 0 & \\ & \pi_i \quad unrestricted & \end{array} \tag{4.8}$$

Condensed dual formulation

$$\begin{array}{lll} \max & \sum_i^N b_i \pi_i - \sum_i^N \sum_j^N u_{ij} \alpha_{ij} & \\ s.t. & \alpha_{ij} + c_{ij}^{\pi} \ge 0 & \forall (i,j) \qquad [x_{ij}] \\ & \alpha_{ij} \ge 0 & \\ & \pi_i \quad unrestricted & \end{array} \tag{4.9}$$

Complementary slackness conditions.

$$\begin{array}{l} (u_{ij} - x_{ij}^*)\alpha_{ij}^* = 0 \\ (\alpha_{ij}^* + c_{ij}^{\pi})x_{ij}^* = 0 \end{array} \tag{4.10}$$

Optimality conditions.

$$\begin{array}{l} if \;\; c_{ij}^{\pi} > 0 \;\; then \;\; (\alpha_{ij}^* + c_{ij}^{\pi}) > 0 \;\; then \;\; x_{ij}^* = 0 \\ if \;\; 0 < x_{ij}^* < u_{ij} \;\; then \;\; \alpha_{ij}^* = 0 \;\; then \;\; c_{ij}^{\pi} = 0 \\ if \;\; c_{ij}^{\pi} < 0 \;\; then \;\; \alpha_{ij}^* > 0 \;\; then \;\; x_{ij}^* = u_{ij} \end{array} \tag{4.11}$$

4.4. Single Roundtrip Vehicle Routing

Introduction

Traveling Salesman Problem

Specialized Branch and Bound Algorithms

2000 Nodes

Many Heuristic Algorithms

Traveling Salesman Problem Applications

Traveling Salesman

Shortest Hamiltonian Cycle

Knight's Tour

Person-Aboard Order Picking

Running Domestic Errands

Sequencing Jobs in a Paint Booth

Traveling Salesman Problem Definition

Asymmetric Traveling Salesman Problem Formulation

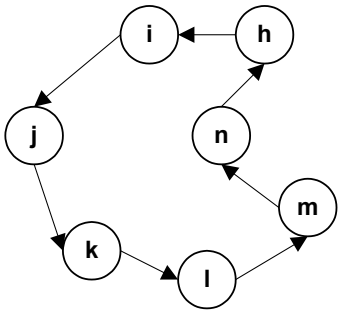


Figure 4.44. Asymmetric Traveling Salesman Problem Illustration

Formulation 4.2. Asymmetric Traveling Salesman Problem

$$
\begin{aligned}
Min \quad & \sum_{i=1}^{N}\sum_{j=1}^{M} c_{ij}x_{ij} & \\
s.t. \quad & \sum_{i=1}^{N} x_{ij} = 1 & \forall j \\
& \sum_{j=1}^{N} x_{ij} = 1 & \forall i \\
& \sum_{i\in S}\sum_{j\in S} x_{ij} \le |S|-1 & \forall S \subset N \\
& x_{ij} \in \{0,1\} &
\end{aligned}
\tag{4.12}
$$

The Asymmetric Traveling Salesman (ATSP) is basically an Assignment Formulation (AP) with additional constraints that eliminate subtours.

Subtour Elimination Constraints

$$\sum_{i \in S} \sum_{j \in S} x_{ij} \leq |S| - 1 \quad \forall S \subset N \tag{4.13}$$

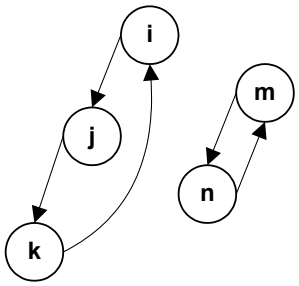


Figure 4.45. Subtour Elimination Illustration

Symmetric Traveling Salesman Formulation

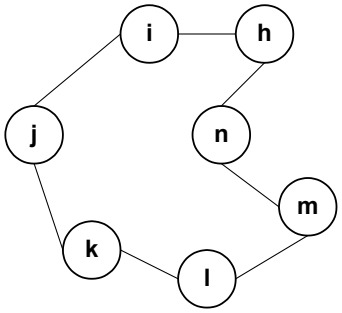


Figure 4.46. Symmetric Traveling Salesman Problem Illustration

Formulation 4.3. Symmetric Traveling Salesman Problem

$$\begin{aligned}
Min \quad & \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} c_{ij} x_{ij} \\
s.t. \quad & \sum_{i=1}^{j-1} x_{ij} + \sum_{k=j+1}^{N} x_{jk} = 2 \qquad \forall j \\
& \sum_{i \in S} \sum_{j \in S, j>i} x_{ij} \leq |S| - 1 \qquad \forall S \subset N \\
& x_{ij} \in \{0,1\}
\end{aligned} \tag{4.14}$$

The Symmetric Traveling Salesman Problem formulation (STSP) is basically a two-matching formulation with side constraints that eliminate subtours.

Simple TSP Heuristics

Heuristic Types

Create an initial tour

- convex hull, sweep, nearest neighbor

Insert remaining free points

- nearest, cheapest, farthest insertion

Improve existing tour

- two, three, or Or exchanges

Construction Heuristics

Nearest Neighbor

The Nearest Neighbor algorithm starts the tour one initial point and then appends the nearest unvisited or free point to the tour. This algorithm was originally described by Rosenkrantz et al. (1977). The initial starting point is an algorithm parameter that you can specify. Since the Nearest Neighbor algorithm executes very fast, a possible alternative would be to start a tour at each point and then to retain the shortest tour among them.

Improvement Heuristics

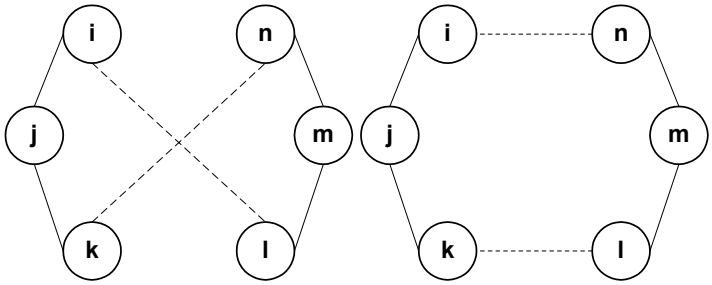


Figure 4.47. Two Exchange Improvement Illustration

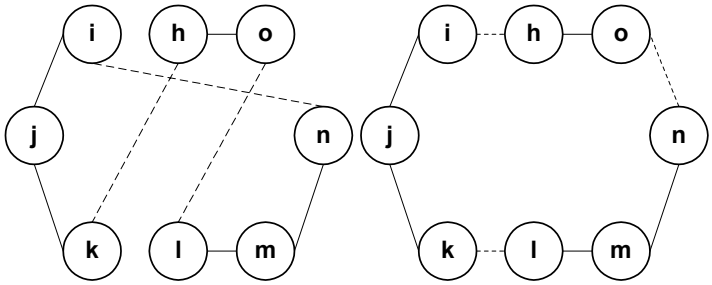


Figure 4.48. Three Exchange Improvement Illustration

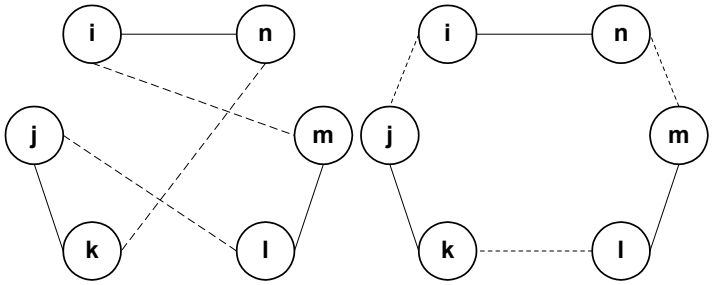


Figure 49. Or (2 Chain) Exchange Improvement Illustration

Improvement Heuristic Classification

Exchange improvement heuristics can be divided into four classes depending on which exchange they test for possible improvement and which exchange they select to execute. For a minimization problem such as the TSP where we want to a tour with the lowest possible length, the categories are

1. First Descent
2. Steepest Descent
3. Simulated Annealing
4. Tabu Search

First Descent

All possible edge exchanges that can result in a new tour are examined in a structured way until an exchange is found that reduces the tour length. This exchange is executed immediately and the process of examining all possible exchanges starts all over. Hence, the first exchange in each iteration that yields a reduction is executed. The process terminates when no further exchanges can be found that yield a cost reduction.

Steepest Descent

All possible edge exchanges that can result in a new tour are examined in a structured way and the exchange that yielded the largest reduction in the tour length is retained. If this exchange reduces the tour length then it is executed and the process of examining all possible exchanges starts all over. Hence, the exchange that yields the strongest reduction in each iteration is executed. The process terminates when no further exchanges can be found that yield a cost reduction.

Simulated Annealing

Both previous improvement algorithms are deterministic, i.e. each algorithm will convert an initial tour into specific final tour. Since they are heuristics, this final tour may not be of high quality. To remedy this problem, a probabilistic exchange improvement algorithm was developed. There exists an analogy between the optimization method of simulated annealing and the laws of thermodynamics, specifically with the way in which liquids freeze and crystallize or metals cool and anneal.

The simulated annealing algorithm selects a set of edges for exchange evaluation at random. If the exchange yields a cost reduction, then the exchange is executed immediately. If the exchange yields a cost increase, then the exchange is executed with probability P, which is computed in function of the cost increase Δ and the temperature T. T is a search control parameter that is systematically reduced during the algorithm execution.

$$\begin{aligned} &if \quad \Delta < 0 \quad P[Exch] = 1 \\ &if \quad \Delta \geq 0 \quad P[Exch] = e^{-\Delta/T} \end{aligned} \tag{4.15}$$

This allows early on exchanges with large cost increases. As the temperature is reduced, the number of such exchanges and the size of the allowed cost increases are gradually reduced. The objective of these non-improving exchanges is to avoid a first descent into a local minimum. The process repeats itself until no further improvements can be made. Since the exchanges were selected at random, the improvement algorithm may generate a different final tour if run from the same initial tour if different seeds are used to generate different pseudo-random number streams for sampling the probability function of P.

For further information on two and three exchanges see Goetschalckx (1992). For further information on simulated annealing see Kirkpatrick et al. (1983) and Vechi and Kirkpatrick (1983).

Computational processing time increases sharply with the amount of improvement processing.

Tabu Search

Insertion Heuristics

Insertion heuristics must make two type of decisions and also decide which decision to make first. The two decision are which point to insert next and where to insert this point. Many variants exists depending on how and in what sequence those two questions are answered.

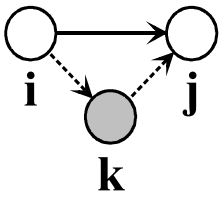


Figure 4.50. Node Insertion

Cheapest Insertion

Cheapest insertion first determines for every remaining free or unvisited point where the optimal link is to insert this point. This corresponds to the inner minimization in the equation (4.16). The insertion penalty is sum of the distance to the free point minus the distance of the link that will be removed. Cheapest insertion then selects the point to insert as the point with the minimum insertion penalty.

$$\min_{k}\left\{\min_{ij}\left\{\delta_{kij} = c_{ik} + c_{kj} - c_{ij}\right\}\right\} \tag{4.16}$$

Priciest Insertion

Priciest insertion first determines for every remaining free or unvisited point where the optimal link is to insert this point. This corresponds to the inner minimization in the equation (4.17) and is identical to the minimization process of the cheapest insertion algorithm. The insertion penalty is sum of the distance to the free point minus the distance of the link that will be removed. Priciest insertion then selects the point to insert as the point with the maximum insertion penalty.

$$\max_{k}\left\{\min_{ij}\left\{\delta_{kij} = c_{ik} + c_{kj} - c_{ij}\right\}\right\} \tag{4.17}$$

Nearest Insertion

Nearest insertion determines first the free point to insert by finding the free point closest to a point on the tour. The algorithm in essence performs a mini-min operation on the distance from a free point to a point on the tour.

$$\min_{k \notin T, j \in T} \{c_{kj}\} = \min_{k \notin T} \left\{ \min_{j \in T} c_{kj} \right\} \tag{4.18}$$

Nearest insertion then determines the best link to insert this point. This process is identical to the minimization process of the cheapest and farthest insertion algorithms.

$$\min_{(i,j) \in T} \{\delta_{ijk} = c_{ik} + c_{kj} - c_{ij}\} \tag{4.19}$$

Farthest Insertion

Farthest insertion determines first for every free point the smallest distance to any point already on the tour. Then it inserts the free point with the maximum smallest distance to a point on the tour. The algorithm in essence performs a maxi-min operation on the distance from a free point to a point on the tour.

$$\max_{k \notin T} \left\{ \min_{j \in T} c_{kj} \right\} \tag{4.20}$$

Farthest insertion then determines the best link to insert this point. This process is identical to the minimization process of the cheapest and farthest insertion algorithms.

$$\min_{(i,j) \in T} \{\delta_{ijk} = c_{ik} + c_{kj} - c_{ij}\} \tag{4.21}$$

Nearest Addition

Nearest addition determines first the free point to insert by finding the free point closest to a point on the tour.

$$\min_{k \notin T, j \in T} \{c_{kj}\} \tag{4.22}$$

Nearest addition then determines the best link to insert this point by examining the two links on the tour incident to the tour point the free point was closest to. This is a more restricted search that the link determination step in the cheapest and farthest insertion algorithms.

$$\min\{\delta_{ijk} = c_{ik} + c_{kj} - c_{ij}, \delta_{jkm} = c_{jk} + c_{km} - c_{jm}\} \tag{4.23}$$

Clarke and Wright Savings Heuristic

Clarke and Wright (1964) developed a construction procedure that extends a partial route or route primitive on its two end points. Conceptually the algorithm defines a base point and constructs an Eulerian tour that visits each of the other points and the returns to the base point. The Eulerian tour is

then reduced in length by finding and executing the shortcut with the largest savings. The savings are computed as the sum of the distances to the base point of the two points minus the distance between the two points.

$$\max_{i,j}\left\{s_{ij} = c_{i0} + c_{0j} - c_{ij}\right\} \tag{4.24}$$

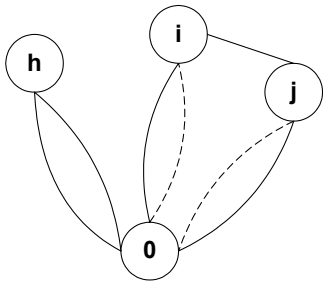


Figure 4.51. Clarke and Wright Tour Extension Illustration

Once two points have been joined by a shortcut they are never again separated again by the Clarke and Wright algorithm. The serial variant of the algorithm extends the single partial route at its end points, which are connected to the base point. The next point is then selected by finding the point with the largest savings shortcut to the current end points of the partial tour. The parallel variant of the algorithm creates a number of partial tours that are concatenated when two endpoint of two different tours are connected. The serial variant is easier to program and requires fewer calculations to update the savings after two points have been connected by a shortcut.

$$\max_{i,j}\left\{\max_{h}\left\{s_{ih} = c_{i0} + c_{h0} - c_{ih}\right\}\right\} \tag{4.25}$$

Algorithm 4.2 Clarke and Wright Savings Algorithm (TSP Serial Variant)

1. Select base point {0}
2. Construct a tour primitive by finding the two points with the largest savings shortcut
3. While not all points have added to the partial tour
4. Update computation of savings of combining tours
5. Append point with largest savings shortcut to endpoints of the partial tour

Spacefilling Curve

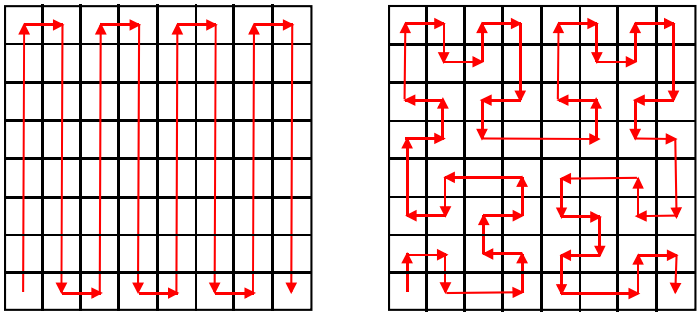


Figure 4.52 . Spacefilling Curve Examples (Serpentine and Hilbert 19)

Optimization Based Heuristics

Minimum Spanning Tree and Matching Heuristic

Lagrangean Relaxation and 1-Tree Heuristic

Held and Karp (1970, 1971) developed a solution for the symmetrical traveling salesman problem based on the notion of a 1-tree relaxation. A 1-tree is a minimum spanning tree of a set of points excluding a single base point plus the two shortest edges connecting the base point to the minimum spanning tree. The number of edges in the 1-tree is $((N-1)-1)+2=N$. The length of the 1-tree is a lower bound on the length of the shortest Hamiltonian cycle through all the points. Since a 1-tree is a (minimum) spanning tree plus two additional edges from a single point to the tree, a 1-tree contains a single cycle. If the 1-tree is a cycle, then it is the solution to the traveling salesman problem. The 1-tree is a cycle if the node degree of all the nodes is equal to two. The Lagrangean relaxation relaxes the constraint that the node degree of each node must be equal to two. Since the constraint is an equality constraint, the corresponding Lagrangean multiplier is unrestricted in sign. The master Lagrangean dual problem is solved with subgradient optimization. The bound provided by the Lagrangean dual was used in a branch-and-bound scheme to solve the TSP. The Lagrangean relaxation is written as

$$\begin{aligned} &Min && \sum_{i=1}^{N-1}\sum_{j=i+1}^{N} c_{ij}x_{ij} + \sum_{j=1}^{N}\lambda_j \cdot \left(\sum_{i=1}^{j-1} x_{ij} + \sum_{k=j+1}^{N} x_{jk} - 2 \right) \\ &s.t. && x_{ij} \in \{0,1\},\ x_{ij} \in 1-tree \end{aligned} \tag{4.26}$$

After rearranging the terms in the objective function, the Lagrangean relaxation becomes

$$\begin{aligned} &Min \quad \sum_{i=1}^{N-1}\sum_{j=i+1}^{N}\left(c_{ij}+\lambda_i+\lambda_j\right)x_{ij}-2\sum_{j=1}^{N}\lambda_j \\ &s.t. \quad x_{ij}\in\{0,1\},\ x_{ij}\in 1-tree \end{aligned} \tag{4.27}$$

The subgradient or improvement direction at a particular solution for each Lagrangean multiplier λ_j is given by

$$\sum_{i=1}^{j-1}x_{ij}+\sum_{k=j+1}^{N}x_{jk}-2=nd_j-2 \qquad \left[\lambda_j\right] \tag{4.28}$$

Held et al. (1974) proposed the following update procedure in their solution of the symmetrical traveling salesman problem.

$$\begin{aligned} &\lambda^{k+1}=\lambda^k+t^k\cdot x^s\left(\lambda^k\right) \\ &t^k=w^k\frac{\hat{z}-z\left(\lambda^k\right)}{\left\|x^s\left(\lambda^k\right)\right\|^2} \\ &w^0=2 \\ &w^{k+m}=w^k/2 \end{aligned} \tag{4.29}$$

Where $\hat{z}$ is the tour length of the best primal feasible solution found so far or incumbent. The weight parameter *w* is cut in half after a fixed number of *m* iterations. This implementation of subgradient optimization is known for its rapid initial convergence, but also for its zigzagging behavior and instability in the neighborhood of the optimal λ^*.

Typically, the incumbent solution $\hat{z}$ is computed initially with a primal heuristic and may or may not be updated during the execution of the Lagrangean optimization. Examples of primal heuristics for the TSP are nearest neighbor and sweep, followed by improvement algorithms such as two-exchange and three-exchange.

1-Tree Relaxation Algorithm for a TSP Bound

1. Initialize all node degree penalties λ to zero, set $w=2, k=0$
2. Compute adjusted distances with $c_{ij}^{\lambda}=c_{ij}+\lambda_i+\lambda_j$
3. Construct minimum spanning tree on $\{\mathbf{N}\}-P_b$, where P_b is the base point, using the adjusted distances c_{ij}^{λ}

4. Connect the base point with the two shortest edges to spanning tree using the adjusted distances c_{ij}^{λ}

5. If all the node degrees are equal to two, stop.

   Else update node degree penalties with the subgradient method.

   $$\begin{array}{lll} \text{if } nd_j = 2 & \lambda_j \text{ remains unchanged} & \lambda_j^{k+1} = \lambda_j^k + t^k \cdot \left(nd_j - 2\right) \\ \text{if } nd_j < 2 & \lambda_j \text{ is decreased} & \\ \text{if } nd_j > 2 & \lambda_j \text{ is increased} & t^k = w \dfrac{\hat{z} - z\left(\lambda^k\right)}{\left\| nd_j - 2 \right\|^2} \end{array}$$

6. $k = k + 1$, If $\left(k \,\text{modulo}\, m\right) = 0$ then $w = w/2$

7. Go to Step 2

TSP Heuristics Example

Table 4.2. Point Locations for the TSP Example

#	x	y
1	0	0
2	100	600
3	400	400
4	500	700
5	900	400
6	800	900

Table 4.3. Euclidean Distances for the TSP Example

	1	2	3	4	5	6
1	0	608	566	860	985	1204
2	608	0	361	412	825	762
3	566	361	0	316	500	640
4	860	412	316	0	500	361
5	985	825	500	500	0	510
6	1204	762	640	361	510	0

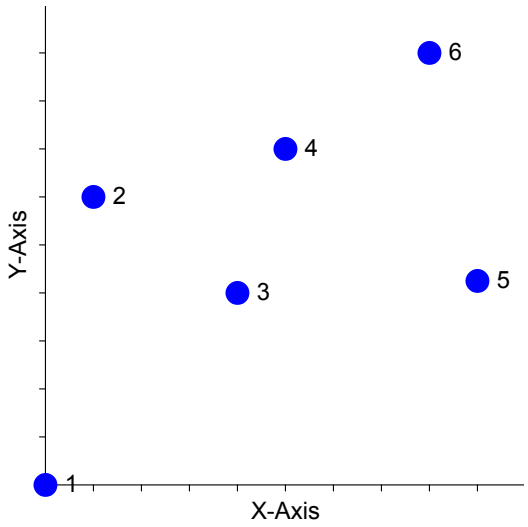


Figure 4.53. Traveling Salesman Example

Nearest Neighbor Construction

Start Point 3

Min {566, 361, 316, 500, 640}=316 (4)

Min {860, 412, 500, 361} = 361 (6)

Min {1204, 762, 510} = 510 (5)

Min {985, 825} = 825 (2)

Min {608} = 608 (1)

Total Tour Length = 3186

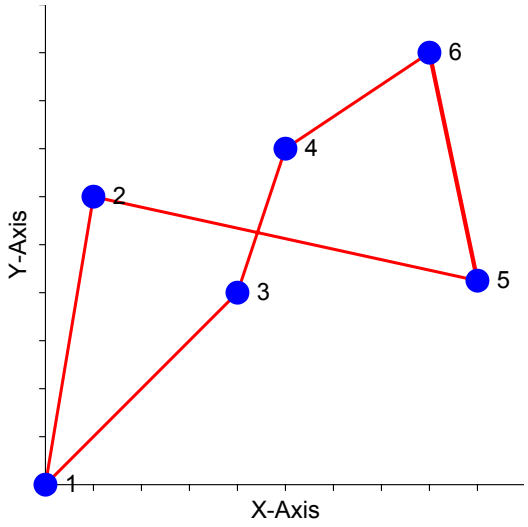


Figure 4.54. Nearest Neightbor TSP Tour

Two Exchange Improvement

Original Tour Length = 3186

Crossing Edges (3-4) and (2-5) in Geometric TSP

Exchange Edges (3-4) and (2-5) with (2-4) and (3-5)

Savings = 316 + 825 - 412 - 500 = 229

Improved Tour Length = 2957

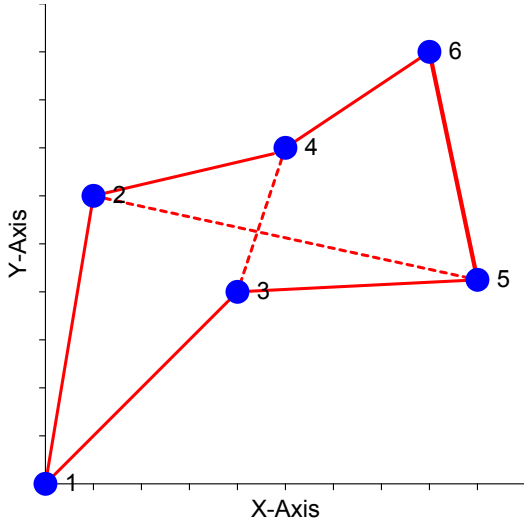


Figure 4.55. TSP Two Exchange

Quad Tour Skeleton Construction

Tour length = 2699

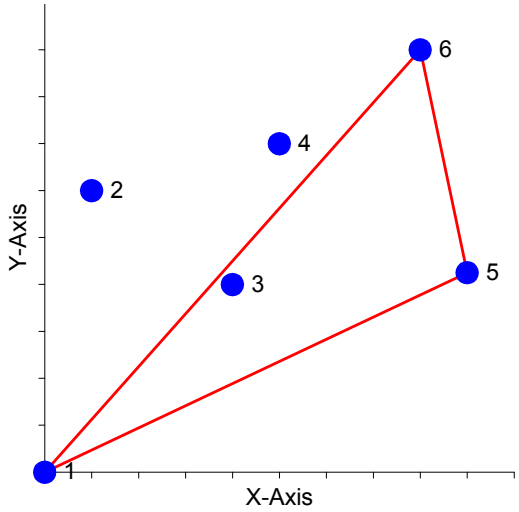


Figure 4.56. Quad Tour Skeleton

Convex Hull Tour Skeleton Construction

Tour length is 2865.

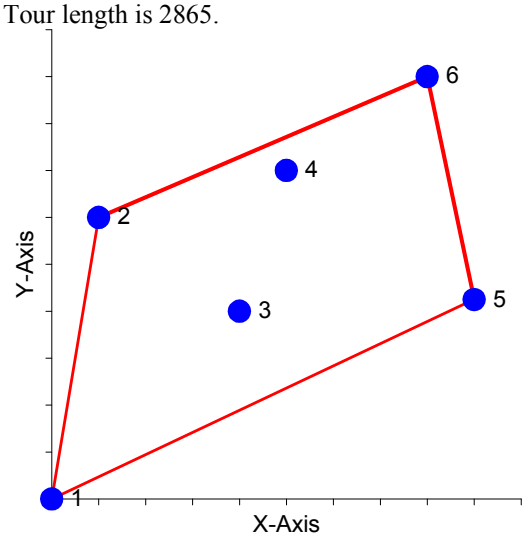


Figure 4.57. TSP Convex Hull Tour Skeleton

Cheapest Insertion

Point 3 & 4

Min {566+500-985=81 (1-5), 500+640-510=630 (5-6), 640+361-762=239 (6-2), 361+566-608=319 (2-1)} = 81 (1-5)

Min {860+500-985=375 (1-5), 500+361-510=351 (5-6), 361+412-762=11 (2-6), 412+860-608=644 (2-1)} = 11 (2-6)*

Point 3

Min {81 (1-5), 630 (5-6), 595 (6-4), 265 (4-2), 319 (2-1)} = 81 (1-5)

Tour length = 2957

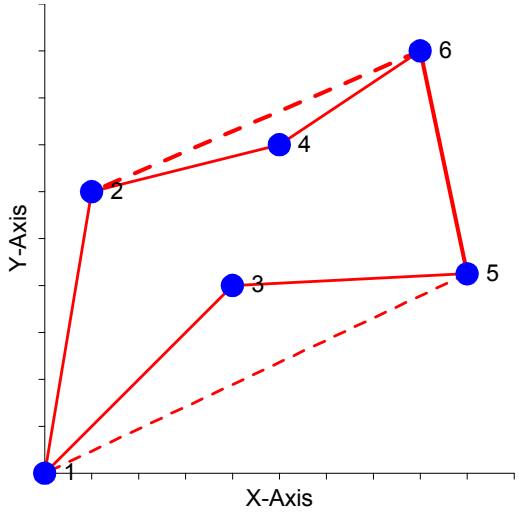


Figure 4.58. TSP Cheapest Insertion

Priciest Insertion

Point 3 & 4

Min {566+500-985=81 (1-5), 500+640-510=630 (5-6), 640+361-762=239 (6-2), 361+566-608=319 (2-1)} = 81 (1-5) *

Min {860+500-985=375 (1-5), 500+361-510=351 (5-6), 361+412-762=11 (6-2), 412+860-608=644 (2-1)} = 11 (2-6)

Max {81 (1-5), 11 (2-6)} = 81 (1-5)

Point 4

Min {610 (1-3), 316 (3-5), 351 (5-6), 11 (6-2), 319 (2-1)} = 11 (6-2)

Clarke and Wright Savings

Tour Primitive (1-6-1) = 2408

Max {1204+608-762=1050 (2), 1204+566-640=1130 (3), 1204+860-361=1703 (4), 1204+985-510=1679 (5)} = 1703 (6-4-1)

Tour Primitive (1-6-4-1)

Max {1050 (6-2-1), 1130 (6-3-1), 1679 (6-5-1), 860+608-412=1056 (4-2-1), 860+566-316=1110 (4-3-1), 860+985-500=1345 (4-5-1)} = 1679 (1-5-6)

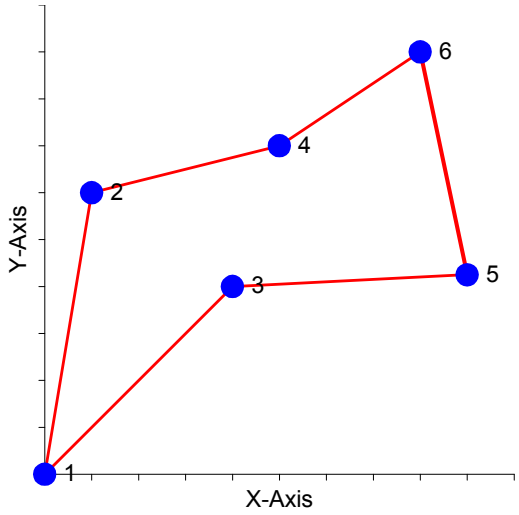


Figure 4.59. TSP Clarke and Wright Savings Tour

Spacefilling Curve Construction

Tour length = 3195

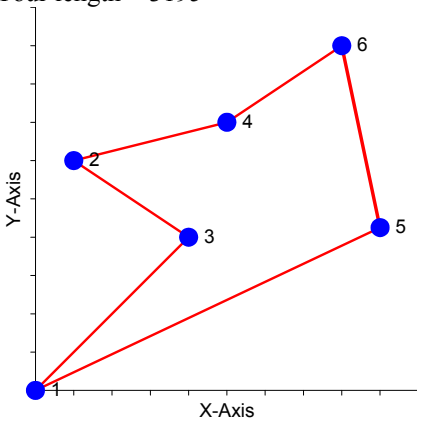


Figure 4.60. TSP Example Spacefilling Curve Tour

Minimum Spanning Tree and Matching Heuristic

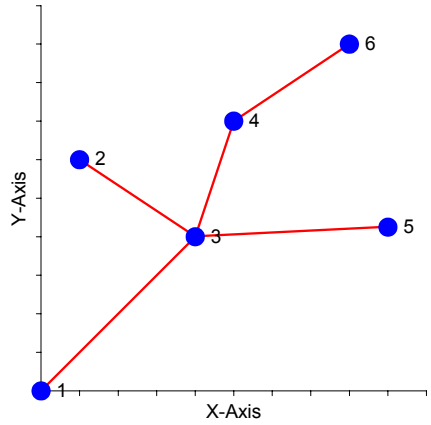


Figure 4.61. TSP Example Minimum Spanning Tree

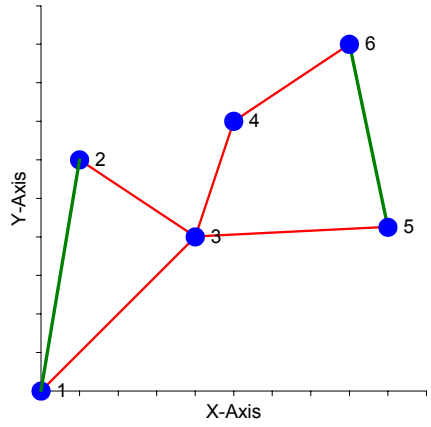


Figure 4.62. TSP Example Minimum Spanning Tree Plus Minimum Matching

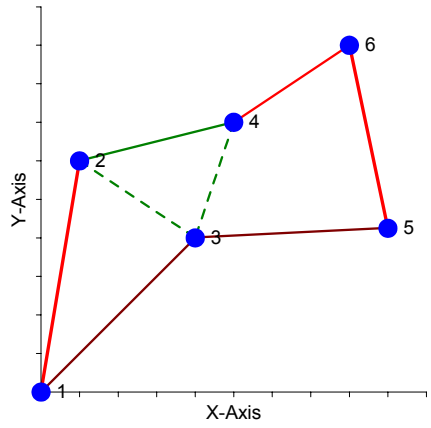


Figure 4.63. TSP Example Minimum Spanning Tree and Maximum Savings Shortcut

Lagrangean Relaxation and 1-Tree Heuristic

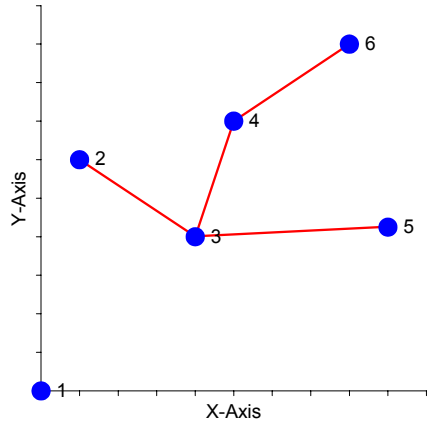


Figure 4.64. TSP Example Minimum Spanning Tree Excluding Base Node 1

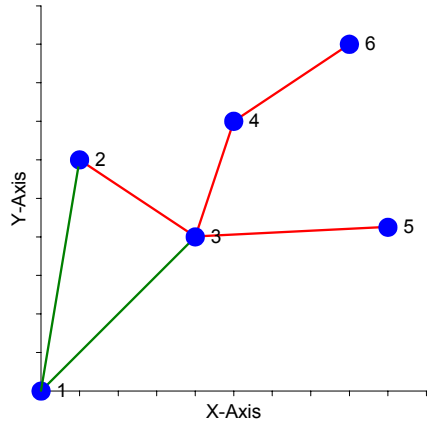


Figure 4.65. TSP Example First 1-Tree Based on Base Node 1

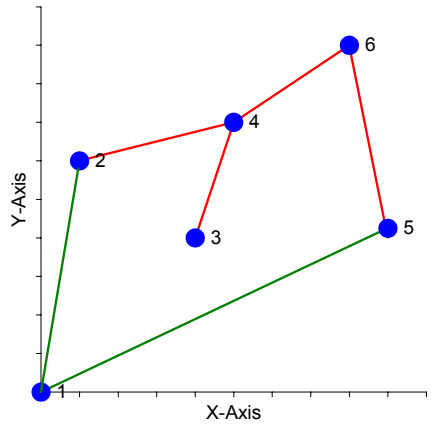


Figure 4.66. TSP Example Second 1-Tree based on Base Node 1

Tours Software Illustration

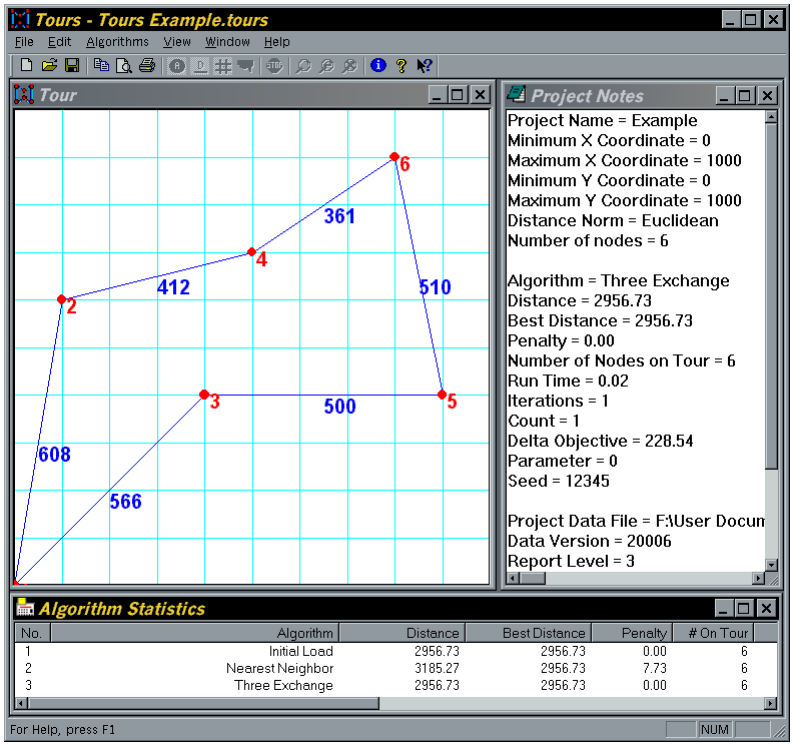


Figure 4.67. Tours TSP Example

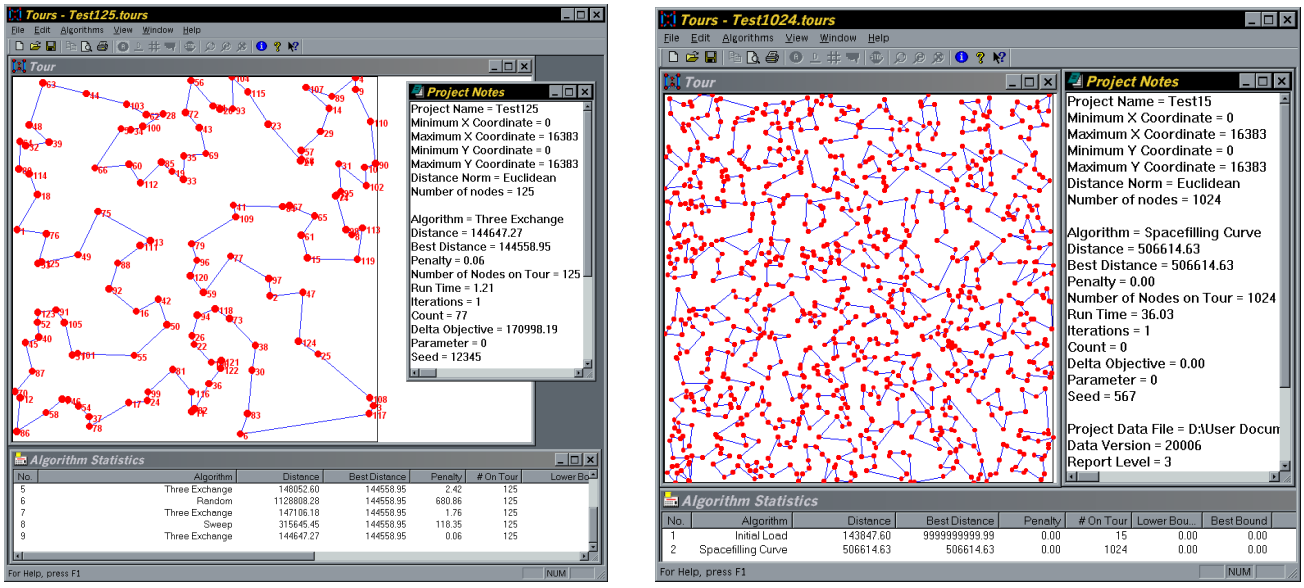


Figure 4.68. Tours Illustrations

4.5. Multiple Vehicle Roundtrip Vehicle Routing

Vehicle Routing Problem

Goal Is to Efficiently Use a Fleet of Vehicles

Given a Number of Stops to Pick Up or Deliver Passenger or Goods

Under a Variety of Constraints

- Vehicle Capacity
- Delivery Time Restrictions
- Precedence Constraints

Vehicle Routing Decisions
Which Customers Served by What Vehicle

Sequence of Stops for Each Vehicle

Number of Vehicles (Fleet Planning)

Vehicle Routing Variants

Traveling Salesman Problem

Pure Vehicle Routing

Linehaul-Backhaul

Vehicle Routing with Time Windows

Vehicle Routing and Scheduling

Mixed Pickup and Delivery

Vehicle Routing Algorithms Classification

Algorithm Classification by Required Input

Route Generating

- Large Variety
- Generalized Assignment Algorithm
- Tightly capacitated problems

Route Selecting

- Set Partitioning Algorithm
- Complex costs and constraints

Algorithm Classification by Basic Methodology

Problem Specific Heuristics

Optimization Based Heuristics

Artificial Intelligence Based Heuristics

Optimal Algorithms

Pure Vehicle Routing (VRP)

VRP Problem Definition

Single depot

N Customers (x_i, y_i, dem_i)

- known location and demand

K Vehicles (cap_k)

- known and equal size

Minimum travel cost objective

Travel distance norm

Route Generation Algorithms

Route Construction

- Nearest Neighbor
- Clarke and Wright Savings

Route Improvement

- Christofides and Eilon (K-TSP)
- Two, Three, and Or-Exchange Improvement

Two Phase Algorithms

Cluster First, Route Second

- Sweep A, Fisher and Jaikumar (GAP)

Route First, Cluster Second

- Sweep B, Great Tour

Nearest Neighbor

The nearest neighbor algorithm starts a new route at the depot and visits next the closest unvisited customer. If adding that customer to the route would violate the truck capacity, the route is terminated and the truck returns to the depot to start a new route if the maximum number of routes has not been reached. Otherwise the customer is added to the route and the vehicle travels next to the closest unvisited customer.

Cluster First, Route Second Sweep

The first variant of the Sweep algorithm rotates a ray with origin at the distribution center and the points are assigned to the current group or cluster when they are traversed by the ray. The current cluster is closed if the next point would violate the truck capacity. After all the clusters have been determined, TSP construction routes must be used to construct the actual route for each cluster.

Generalized Assignment Formulation (GAP)

General Nonlinear Formulation

$$
\begin{aligned}
\min \quad & \sum_{k}^{N} f(y_{ik}) \\
s.t. \quad & \sum_{i=1} dem_i y_{ik} \le cap_k && \forall k \\
& \sum_{k} y_{0k} = K \\
& \sum_{k} y_{ik} = 1 && i = 1..N \\
& y_{ik} \in \{0,1\}
\end{aligned}
\tag{4.30}
$$

Routing Formulation

$$
\begin{aligned}
f(y_{ik}) = \min \quad & \sum_{i}\sum_{j} c_{ij} x_{ijk} \\
s.t. \quad & \sum_{i} x_{ijk} = y_{jk} && \forall jk \\
& \sum_{j} x_{ijk} = y_{ik} && \forall ik \\
& \sum_{i \in S}\sum_{j \in S} x_{ijk} \le |S| - 1 && S \subseteq N(y_{ik}) \\
& x_{ijk} \in \{0,1\}
\end{aligned}
\tag{4.31}
$$

Linearized Clustering Cost

Heuristic Approximation

s_k Seed Customer for Route k

Many Variants for d_{ik} and Seed Selection

$$f(y_{ik}) = \sum_{i} d_{ik} y_{ik} \tag{4.32}$$

$$d_{ik} = c_{0i} + c_{is_k} - c_{s_k 0} \tag{4.33}$$

Sweep (Route First, Cluster Second Sweep)

This variant of the sweep algorithm first sequences all the customers based on their location into a single route. It ignores the customer demand requirements during the routing phase. The algorithm then creates routes that observe customer demand and satisfy truck capacity during the clustering phase.

This variant of the sweep algorithm sequences all customers to be visited on a single route by rotating a line segment or ray around the distribution center. Customers are added to the sequence when they are traversed by the rotating line. The starting angle and the rotational direction of the line are algorithm parameters. At the end of the first phase a single tour consisting of all customers has been created. After all customers have been sequenced, a customer is selected to start the first route. The route visits each customer according to the sequence determined above until appending the next customer would violate the truck capacity. That next customer starts a new route. The starting customer and the direction in which the sequence is traversed are algorithm parameters. After the routes have been determined, they can be improved by TSP improvement routines such as steepest descent exchanges.

Great Tour

The great tour algorithm creates the shortest length traveling salesman tour of all the customers, while ignoring their demand requirements, during the routing phase. After all customers have been sequenced, a customer is selected to start the first route. The route visits each customer according to the sequence until the next customer would violate the truck capacity. That next customer starts a new route. The starting customer and the direction in which the sequence is traversed are algorithm parameters.

Clarke and Wright Savings

Maximum number of routes = K

Serial variant

One route at-a-time

Simpler implementation

Parallel variant

No more than K routes at-a-time

More complex programming

Improvement Algorithms

Intra-route Improvements (TSP)

•Always feasible

•2 exchange, Or exchange, 3 exchange

Inter-route improvements

•Test and make only feasible exchanges

•Move (one point to another route)

•Swap (exchange two points between two routes)**Vehicle Routing (VRP) Example**

This example is based on the example data from Cullen et al. (1981).

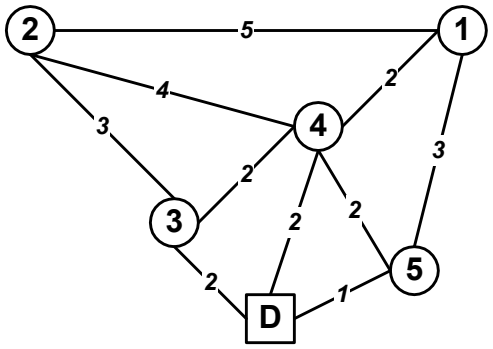


Figure 4.69. Vehicle Routing Distance Data

In addition, the truck capacity is assumed to be 15 and the customer demands are given in the following table.

Table 4.4. Vehicle Routing Customer Demand Data

Customer	Demand
1	3
2	6
3	4
4	7
5	6

The maximum number of routes is typically an input parameter specified by the user. The first quantity to be determined is the minimum number of routes required to service all customer demands. This minimum number must less than or equal to the maximum number for the problem to have a feasible solution. The maximum number of routes in this example is equal to three.

$$\min \#routes = \left\lceil \frac{\sum_{i=1}^{N} q_i}{Cap} \right\rceil = \left\lceil \frac{26}{15} \right\rceil = \lceil 1.73 \rceil = 2$$

Nearest Neighbor

The closest customer is customer 5, which has a distance of 1 unit. The truckload after visiting customer 5 is equal to 6. The closest unvisited customer to customer 5 is customer 4 with a distance of 2. The demand of customer 4 is 7 units so the total demand would be 13 and not violate the truck capacity of 15. Customer 4 is added to the route. The closest unvisited customer to customer 4 is customer 1 with a distance of 2. However, the demand of customer 1 is 3 units and the total demand would be 16, which would violate the truck capacity of 15. Hence, customer 1 is not added to the route. The vehicle returns to the depot after customer 4 and the length of this route equals 5. Since there is currently only one route out of a maximum of three possible routes a new route is started.

The closest unvisited customer to the depot is customer 3, which has a distance of 2 units. The truckload after visiting customer 3 is equal to 4. The closest unvisited customer to customer 3 is customer 2 with a distance of 3. The demand of customer 3 is 6 units so the total demand would be 10 and not violate the truck capacity of 15. Customer 3 is added to the route. The closest unvisited customer to customer 3 is customer 1 with a distance of 5. The demand of customer 1 is 3 units so the total demand would be 13 and not violate the truck capacity of 15. Customer 1 is added to the route. No unvisited customers remain, so the vehicle returns to the depot and terminates the route. The length of this route is 14 and the total route length equals 19.

Cluster-First Sweep (Variant A)

The starting direction of the ray is a user-specified algorithm parameter. We will compare the results for starting with a ray point due east and north. Note that the groupings are only clusters without any route information. A Traveling Salesman algorithm then needs to be used to sequence the points in each cluster. For this particular example, there are never more than three points in a cluster and the distances are symmetrical, so the problem of determining the optimal point sequence is trivial.

The computations for a starting ray with due east direction are:

$$D-5[6]-1[9]-D \qquad L_1 = 8$$
$$D-4[7]-2[13]-D \qquad L_2 = 11$$
$$D-3[4]-D \qquad L_3 = 4$$
$$L = \sum_r L_r = 8+11+4 = 23$$

The computations for a starting ray with due north direction are

$D-2[6]-3[10]-D \qquad L_1 = 10$

$D-5[6]-1[9]-D \qquad L_2 = 8$

$D-4[7]-D \qquad L_3 = 4$

$L = \sum_r L_r = 10+8+4 = 22$

Route-First Sweep (Variant B)

The computations for a starting ray with due east direction are

$D-5-1-4-2-3-D \qquad (TSP)$

$D-5[6]-1[9]-D \qquad L_1 = 8$

$D-4[7]-2[13]-D \qquad L_2 = 11$

$D-3[4]-D \qquad L_3 = 4$

$L = \sum_r L_r = 8+11+4 = 23$

Serial Clarke and Wright

The first step is to compute the savings for every feasible combination of two points. The computations for two pairs are shown next.

$q_1 + q_2 = 3+6 = 9 \le 15$

$s_{12} = d_{10} + d_{02} - d_{12} = 4+5-5 = 4$

$q_1 + q_4 = 3+7 = 10 \le 15$

$s_{14} = d_{40} + d_{01} - d_{14} = 4+2-2 = 4$

The savings can be summarized in a two-dimensional table. In all the following calculations for the example, savings do not need to be recomputed but can be copied from this initial savings table.

Table 4.5. Pair-wise Savings for the VRP Example

	1	2	3	4	5
1		4	2	4	2
2	4		4	3	0
3	2	4		2	0
4	4	3	2		1
5	2	0	0	1	

The pair with the largest savings that forms a feasible partial tour is selected. In this example, several pairs have savings equal to the maximum savings of four. Selecting the pair with maximum savings by increasing indices arbitrarily breaks the ties. The partial tour (1-2) of nodes one followed by two is

created. The savings matrix is reduced by eliminating the row of the origin (point 1) and the column of the destination (point 2) of the newly formed pair.

In the serial variant of the Clarke and Wright algorithm only one tour at the time is constructed, so only savings of combining other points with the current tour are relevant. The savings matrix for the second iteration is shown in the next table. Point four cannot be combined with partial tour (1-2) because it would violate the vehicle capacity; hence the corresponding savings are eliminated. Vehicle load for either partial tour (4-1-2) or (1-2-4) would have been 3+6+7=16, which is larger than the truck capacity of 15. The largest savings is generated by appending point three to the partial tour (1-2).**Table 4.6. Pair-wise Serial Savings for the VRP Example (Iteration Two)**

	(1-2)	3	4	5
(1-2)		4		0
3	2			
4				
5	2			

The column corresponding to point 3 and the route corresponding to partial route (1-2) are eliminated.

Table 4.7. Pair-wise Serial Savings for the VRP Example (Iteration Three)

	(1-2-3)	4	5
(1-2-3)			
4			
5			

There are no remaining points that can be combined with the partial tour without violating truck capacity, hence the current route is archived and a new route is started.

Table 4.8. Pair-wise Serial Savings for the VRP Example (Iteration Three)

	4	5
4		1
5	1	

The largest savings correspond to combining points 4 and 5. At that time all points have been included on a route and the algorithm terminates. The combined length of the two routes is computed as follows.

$$L = L_{(1-2-3)} + L_{(4-5)} = 14 + 5 = 19$$

Parallel Clarke and Wright

Table 4.9. Pair-wise Parallel Savings for the VRP Example (Iteration Two)

	(1-2)	3	4	5
(1-2)		4		0
3	2		2	0
4		2		1
5	2	0	1	

Table 4.10. Pair-wise Parallel Savings for the VRP Example (Iteration Two)

	(1-2-3)	4	5
(1-2-3)			
4			1
5		1	

Again, the largest savings correspond to combining points 4 and 5. At that time all points have been included on a route and the algorithm terminates. The combined length of the two routes is computed as follows.

$$L = L_{(1-2-3)} + L_{(4-5)} = 14 + 5 = 19$$

Set Partitioning Formulation

The Set Partitioning Problem (SPP) formulation belongs to the class of Alternative Selecting algorithms. This model is very powerful in the sense that many realistic route constraints and route cost functions can be incorporated during the route generation process. The most obvious disadvantage of the model is extremely large number of routes that may be generated.

Formulation

The problem is formulated based on the cover matrix ***A***. The columns of the matrix correspond to feasible alternatives and actions. There are N columns, which are indexed by *j*. The terms columns, feasible alternatives, and covers are used interchangeably. The rows of the matrix correspond to the service requests. There are M rows, which indexed by *i*. The terms rows and service requests will be used interchangeable. The objective is to minimize the overall cost for servicing all the requests with a subset of the feasible alternatives. The cost of executing a particular alternative is denoted by c_j. The elements a_{ij} of the cover matrix ***A*** are equal to one if the alternative *j* covers or serves service request *i*, and are equal to zero otherwise. The variables x_j correspond to the binary decision to execute a feasible

alternative or not. The decision variables can only assume the values zero or one. Finally, the estimated cost of servicing request *i* with the best combination of alternatives found so far is denoted by p_i.

In the vehicle routing framework, the service requests correspond to the customers to be visited. The alternative actions correspond to the various tours. The cost of the alternatives can represent the tour length or the tour cost.

The Set Partitioning formulation is given next.

Formulation 4.4. Set Partitioning Problem

$$\begin{aligned} \min \quad & \sum_{j=1}^{N} c_j x_j \\ s.t. \quad & \sum_{j=1}^{N} a_{ij} x_j = 1 \qquad i = 1..M \\ & x_j \in \{0,1\} \end{aligned} \tag{4.34}$$

The Set Partitioning Problem belongs to the class of binary integer programming problems and, as such, is difficult to solve to optimality for large problem instances.

A feasible partition, denoted by J^+, is a set of feasible alternatives that services all requests. The variables x_j of all the alternatives in a feasible partition are equal to one.

Based on a feasible partition, the nonnegative row prices can be determined. The prices must satisfy the condition that the cost of each of the alternatives in the feasible partition is exactly equal to the sum of the prices of the service requests it covers, i.e.,

$$c_j^+ = \sum_{i=1}^{M} a_{ij} p_i \qquad \forall j \in J^+ \tag{4.35}$$

In the vehicle routing framework, two important factors in the cost of servicing a customer are its distance to the distribution center, denoted by d_{0i}, and its demand, denoted by dem_i. One possible way to compute the row prices is to allocate them based on the product of distance to the depot and the demand, i.e.,

$$p_i = \frac{dem_i \cdot d_{0i}}{\sum_{k \in J_i} dem_k \cdot d_{0k}} c_j^+ = \frac{dem_i \cdot d_{0i}}{\sum_{k} a_{kj} \cdot dem_k \cdot d_{0k}} c_j^+ \tag{4.36}$$

Other price allocation schemes might be based solely on distance to the depot or demand.

$$p_i = \frac{d_{0i}}{\sum_{i \in J_i} d_{0i}} c_j^+ \tag{4.37}$$

A new feasible alternative or column can never be economically desirable if its total cost is more than the current cost for the rows it covers. In other words a new column need only to be considered if its savings are positive, where the savings are computed as

$$s_j = \sum_{i=1}^{M} a_{ij} p_i - c_j \tag{4.38}$$

Row Price Heuristic

This heuristic was originally proposed in Cullen et al. (1981). The heuristic is based on the principle that only feasible alternatives that can improve the current best solution are added to the set partitioning problem. This possible improvement is based on the computation of the estimated savings for each feasible alternative. The set partitioning problem is also called the master problem and the function to generate additional feasible alternatives is called the subproblem.

Algorithm 4.3. Row Price Heuristic for the Set Partitioning Problem

1. Start with a feasible partition J^+ and construct a master SPP problem with this feasible partition.
2. Determine the new row prices by allocating the column prices "equitable" among the service requests covered by it. The prices must satisfy condition (35).
3. Generate the next feasible alternative and compute its estimated savings s_j with $s_j = \sum_{i=1}^{M} a_{ij} p_i - c_j$
4. If the savings are positive (or at least nonnegative), add this alternative to the master Set Partitioning problem, otherwise discard this alternative
5. If enough new alternatives have been added to the master SPP formulation or if all possible alternatives have been evaluated, solve the current master SPP problem and continue with step 5. Otherwise return to step 3.
6. If the solution of the master SPP is within the desired tolerance or if all possible alternatives have been evaluated, then stop, else go to step 2.

Vehicle Routing (VRP) Example

We will use the same example that is based on the example data from Cullen et al. (1981).

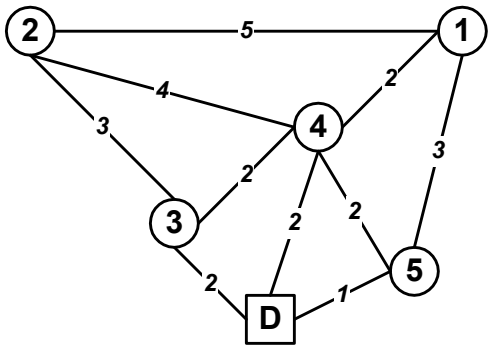


Figure 4.70. Vehicle Routing Distance Data

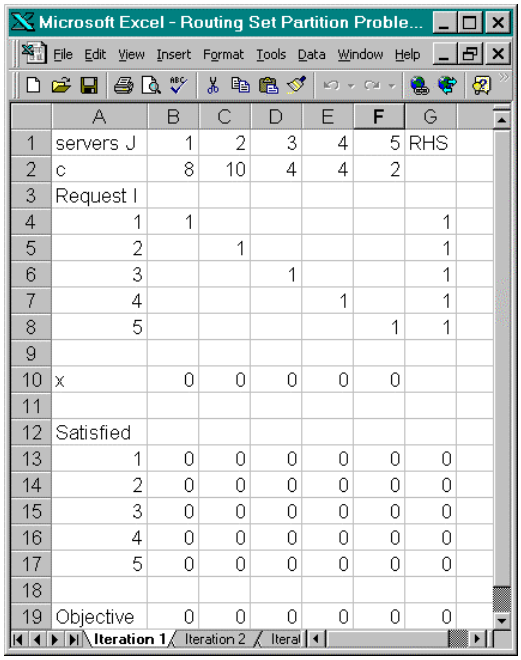
In addition, the truck capacity is assumed to be 15 and the customer demands are given in the following table.

Table 4.11. Vehicle Routing Customer Data

Customer	Demand	Distance to Depot	Distance * Demand
1	3	4	12
2	6	5	30
3	4	2	8
4	7	2	14
5	6	1	6

Assume that we start the algorithm with a feasible partition consisting of routes, each of which visits one customer **and** returns to the depot. The route costs are then equal to 8, 10, 4, 4, and 2, respectively. The total route cost is 28. Since each route visits only one customer the row prices are then also equal to 8, 10, 4, 4, and 2, respectively.

To illustrate the structure and growth of the Set Partitioning master problem, we will solve this trivial problem using the solver included in the Excel spreadsheet. The data for this iteration, the solver parameters, the solver options, and the solution of this iteration are shown in Figures 4.71, 4.72, 4.73, and 4.74, respectively. For the Set Partitioning problem the data area grows when we add columns to the problem and the problem formulation in the solver has to be extended as well.



	A	B	C	D	E	F	G
1	servers J	1	2	3	4	5	RHS
2	c	8	10	4	4	2	
3	Request I						
4	1	1					1
5	2		1				1
6	3			1			1
7	4				1		1
8	5					1	1
9							
10	x	0	0	0	0	0	
11							
12	Satisfied						
13	1	0	0	0	0	0	0
14	2	0	0	0	0	0	0
15	3	0	0	0	0	0	0
16	4	0	0	0	0	0	0
17	5	0	0	0	0	0	0
18							
19	Objective	0	0	0	0	0	0

Figure 4.71. VRP Excel Data for One Customer per Route

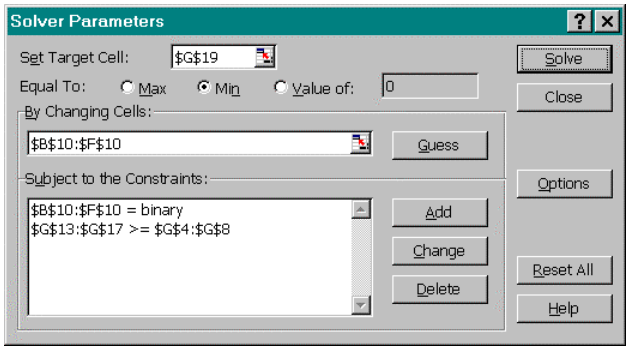


Figure 4.72. VRP Excel Solver for One Customer per Route

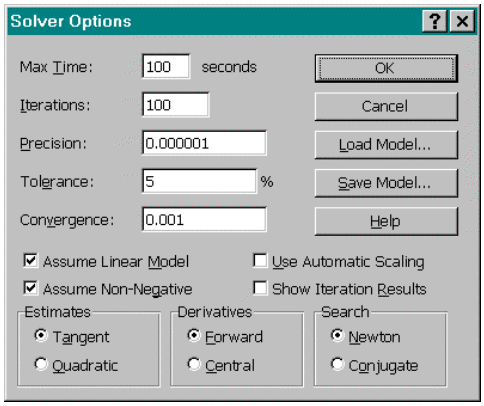
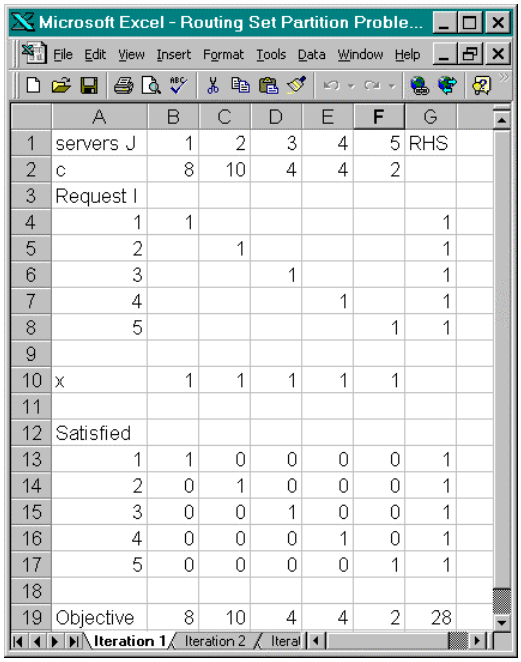


Figure 4.73. VRP Excel Solver Options for One Customer per Route



	A	B	C	D	E	F	G
1	servers J	1	2	3	4	5	RHS
2	c	8	10	4	4	2	
3	Request I						
4	1	1					1
5	2		1				1
6	3			1			1
7	4				1		1
8	5					1	1
9							
10	x	1	1	1	1	1	
11							
12	Satisfied						
13	1	1	0	0	0	0	1
14	2	0	1	0	0	0	1
15	3	0	0	1	0	0	1
16	4	0	0	0	1	0	1
17	5	0	0	0	0	1	1
18							
19	Objective	8	10	4	4	2	28

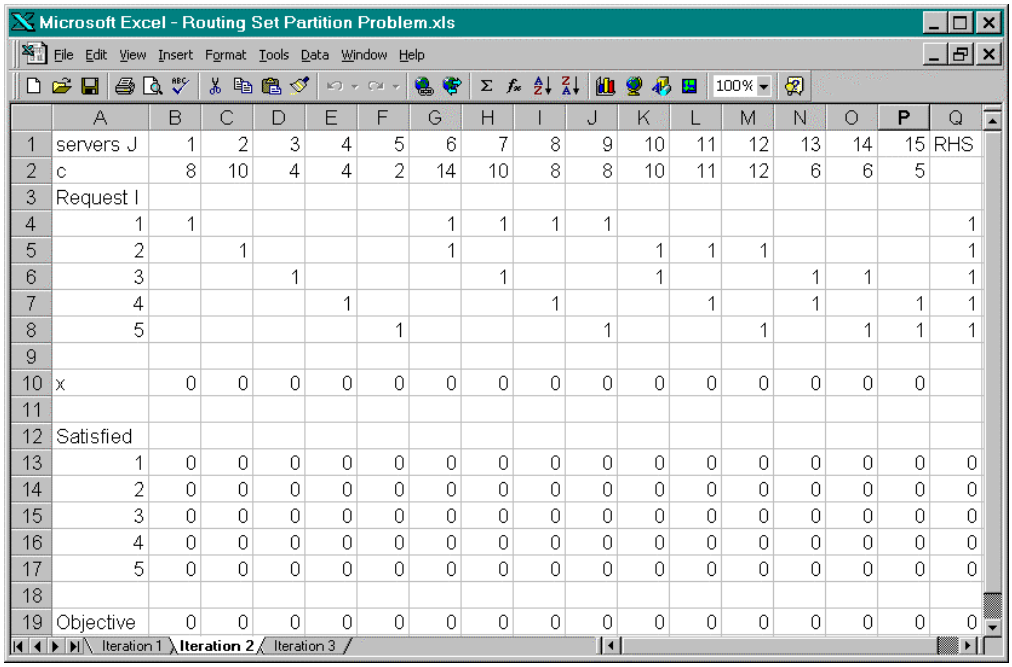
Figure 4.74. VRP Excel Solution for One Customer per Route

In the next pass, we will generate all the alternatives corresponding to routes that visit two customers and then return to the depot. Observe that any route with two customers is feasible with respect to the truck capacity. The computed route costs and savings for each of these alternatives are given in the next table.

Table 4.12. Two Customer Column Evaluations

Route	Customer One	Customer Two	Length	Savings
6	1	2	14	4
7	1	3	10	2
8	1	4	8	4
9	1	5	8	2
10	2	3	10	4
11	2	4	11	3
12	2	5	12	0
13	3	4	6	2
14	3	5	6	0
15	4	5	5	1

Since all these routes have nonnegative savings, they all will be added to the current master set partitioning problem. The data for this iteration, the solver parameters, and the solution of this iteration are shown in Figures 4.75, 4.76, and 4.77, respectively. Solving the master set partitioning problem, we obtain a new best feasible partition, consisting of routes {5, 8, 10}. The total route cost is 20.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1	servers J	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	RHS
2	c	8	10	4	4	2	14	10	8	8	10	11	12	6	6	5	
3	Request I																
4	1	1					1	1	1	1							1
5	2		1				1				1	1	1				1
6	3			1				1			1			1	1		1
7	4				1				1			1		1		1	1
8	5					1				1			1		1	1	1
9																	
10	x	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
11																	
12	Satisfied																
13	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
17	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
18																	
19	Objective	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 4.75. VRP Excel Data for Two Customer per Route

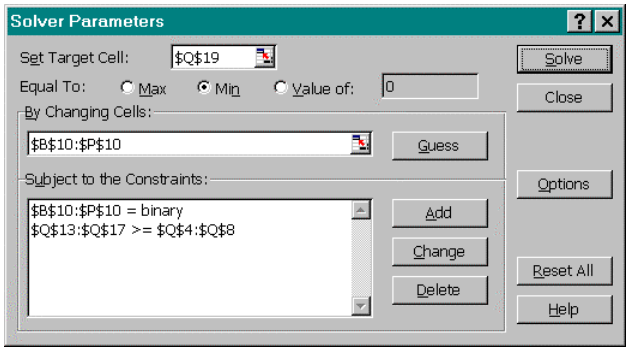
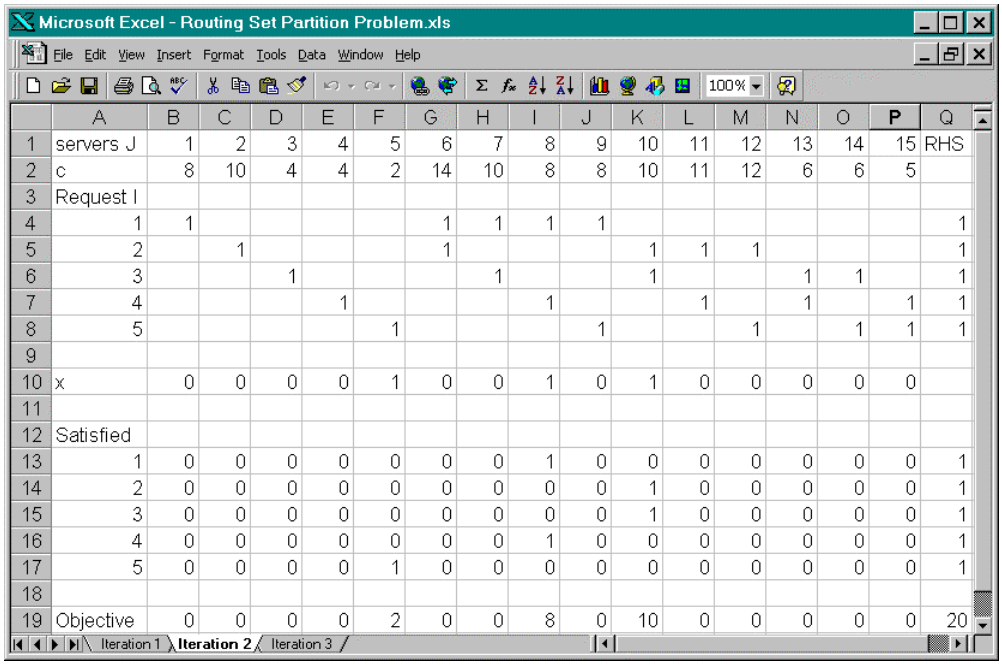


Figure 4.76. VRP Excel Solver for Two Customers per Route



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1	servers J	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	RHS
2	c	8	10	4	4	2	14	10	8	8	10	11	12	6	6	5	
3	Request I																
4	1	1					1	1	1	1							1
5	2		1				1				1	1	1				1
6	3			1				1			1			1	1		1
7	4				1				1			1		1		1	1
8	5					1				1			1		1	1	1
9																	
10	x	0	0	0	0	1	0	0	1	0	1	0	0	0	0	0	
11																	
12	Satisfied																
13	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1
14	2	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1
15	3	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1
16	4	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1
17	5	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1
18																	
19	Objective	0	0	0	0	2	0	0	8	0	10	0	0	0	0	0	20

Figure 4.77. VRP Excel Solution for Two Customer per Route

Since customer 5 is still the only customer service by this route, its row price remains the equal to the cost of the route, i.e., equal to 2. The row prices for customers one through four now have to be computed based on some allocation scheme. We will use the allocation scheme based on the product of distance to the depot and demand as given by Equation (4.36). The computations can be summarized as follows:

$$j = 8 \qquad I_8 = \{1,4\}$$
$$c_j = 8 \qquad p_1 + p_4 = 8$$
$$p_1 = \frac{12}{26} \cdot 8 = 0.46 \cdot 8 = 3.69$$
$$p_4 = \frac{14}{26} \cdot 8 = 0.54 \cdot 8 = 4.31$$

$$j = 10 \qquad I_{10} = \{2,3\}$$
$$c_j = 10 \qquad p_2 + p_3 = 10$$
$$p_2 = \frac{30}{38} \cdot 10 = 0.79 \cdot 10 = 7.89$$
$$p_3 = \frac{8}{38} \cdot 10 = 0.21 \cdot 10 = 2.11$$

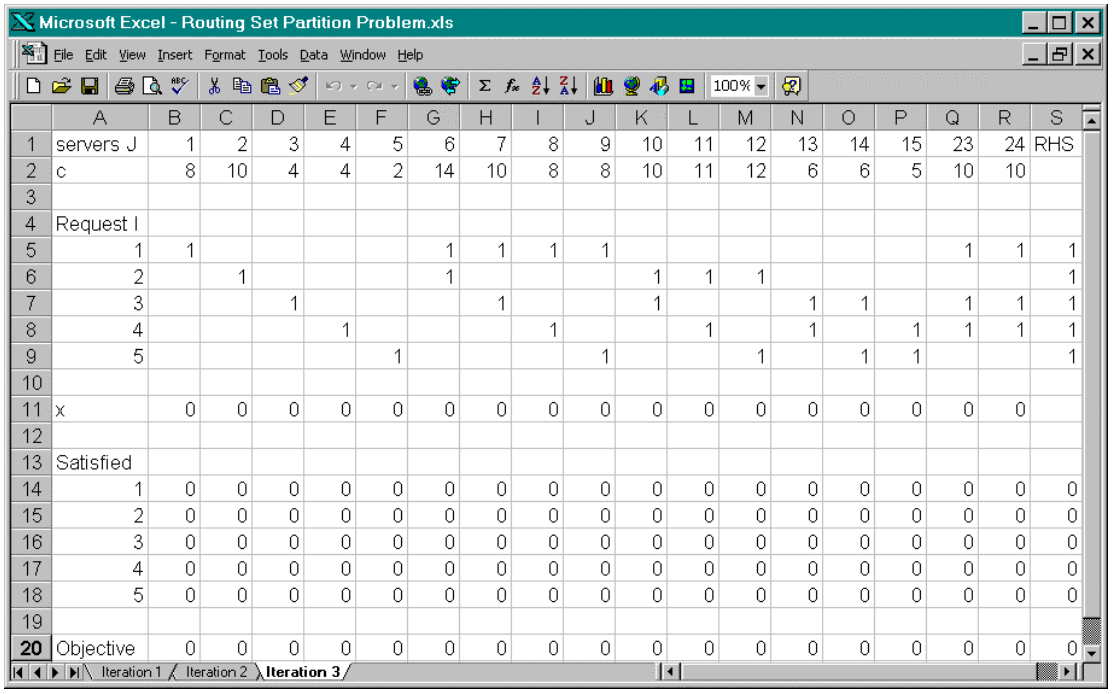
In the next pass, we will generated the feasible alternatives corresponding to routes that visit three customers and then return to the depot, provided the demand of the three customers does not violate truck capacity. Observe that every triplet of route candidates consists of the permutations of the

sequence of three customers on this route, so they have the same route demand. Also observe that since the routes can be executed in the reverse sequence at the same cost, only three different permutations are possible for each set of three customers on a route. The computed route cost and savings for each of these alternatives are given in the next table.

Table 4.13. Three Customer Route Evaluations

Route	Customer One	Customer Two	Customer Three	Route Demand	Route Length	Savings
16	1	2	3	13	14	-0.31
17	1	3	2	13	16	-2.31
18	2	1	3	13	16	-2.31
19	1	2	5	14	16	-2.42
20	1	5	2	14	18	-4.42
21	2	1	5	14	14	-0.42
22	1	3	4	14	12	-1.89
23	1	4	3	14	10	0.11
24	3	1	4	14	10	0.11
25	1	3	5	13	12	-4.20
26	1	5	3	13	12	-4.20
27	5	1	3	13	10	-2.20

Only routes 23 and 24 will be added to the set partitioning problem. The data for this iteration, the solver parameters, and the solution of this iteration are shown in Figures 4.78, 4.79, and 4.80, respectively. Solving the set partitioning problem, we obtain the same best feasible partition, again consisting of routes {5, 8, 10}. The total route cost is 20. Since no feasible routes with more than three customers on a route can be generated, the heuristic terminates. The last set partitioning problem to be solved contained 17 feasible alternatives.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1	servers J	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	23	24	RHS
2	c	8	10	4	4	2	14	10	8	8	10	11	12	6	6	5	10	10	
3																			
4	Request I																		
5	1	1					1	1	1	1							1	1	1
6	2		1				1				1	1	1						1
7	3			1				1			1			1	1		1	1	1
8	4				1				1			1		1		1	1	1	1
9	5					1				1			1		1	1			1
10																			
11	x	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
12																			
13	Satisfied																		
14	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
17	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
18	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
19																			
20	Objective	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 4.78. VRP Excel Data for Three Customer per Route

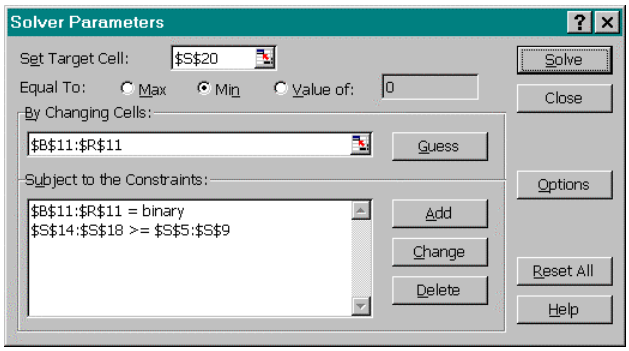
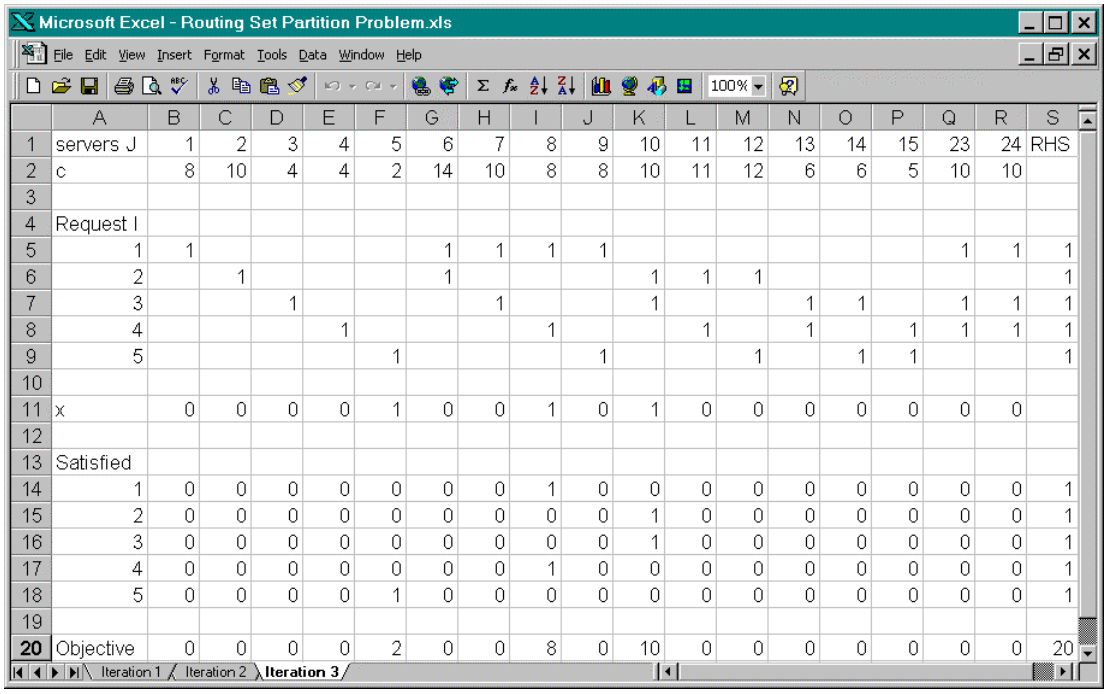


Figure 4.79. VRP Excel Solver for Three Customer per Route



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1	servers J	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	23	24	RHS
2	c	8	10	4	4	2	14	10	8	8	10	11	12	6	6	5	10	10	
3																			
4	Request I																		
5	1	1					1	1	1	1							1	1	1
6	2		1				1				1	1	1						1
7	3			1				1			1			1	1		1	1	1
8	4				1				1			1		1		1	1	1	1
9	5					1				1			1		1	1			1
10																			
11	x	0	0	0	0	1	0	0	1	0	1	0	0	0	0	0	0	0	
12																			
13	Satisfied																		
14	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1
15	2	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1
16	3	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1
17	4	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1
18	5	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1
19																			
20	Objective	0	0	0	0	2	0	0	8	0	10	0	0	0	0	0	0	0	20

Figure 4.80. VRP Excel Solution for Three Customer per Route

This algorithm is a heuristic because the row prices have been determined in a heuristic manner, i.e., in this example we used equation (4.36) to allocate the row prices. The row prices in turn determined if a feasible alternative entered the set partitioning problem or not. If no feasible alternatives could be added for any set of row prices satisfying equation (4.35), then the current feasible partition would be an optimal solution. If we had added all 27 feasible alternatives to the set partitioning problem, its solution would also have been optimal. But this strategy is obviously not practical for large scale problem instances.

Vehicle Routing Example

Consider the problem of finding the routes that minimize the total travel distance for the distribution system with six customers. The location and demand of the customers are specified in Table 4.14. All trucks are based at the depot and have the same capacity of 20 pallets. All routes start and terminate at the depot.

Table 4.14. Vehicle Routing Data

Customer	x	y	demand
(depot) 0			
1			
2			
3			
4			
5			
6			

Travel distances can be approximated with the Euclidean distance norm rounded to the nearest integer value. The distances are shown in upper triangular format in Table 4.15.

Table 4.15. Vehicle Routing Integer Euclidean Distance Matrix

	0	1	2	3	4	5	6
0							
1							
2							
3							
4							
5							
6							

Vehicle routes are computed with the Nearest Neighbor, Sweep Cluster-First-Route-Second, Sweep Great Tour, Clarke and Wright savings, and Generalized Assignment heuristic by Fisher and Jaikumer.

Vehicle Routing with Backhauling (VRPB)

Introduction

#1 Savings Technique in Routing

Problem Definition

Single Depot

NL Customers (x_i, y_i, dem_i) and NB Suppliers (x_i, y_i, sup_i)

M Equal Size Vehicles (cap_j)

Rear Loaded Vehicles

- all customers before any supplier

Minimize Total Travel Distance

Travel Distance Norm

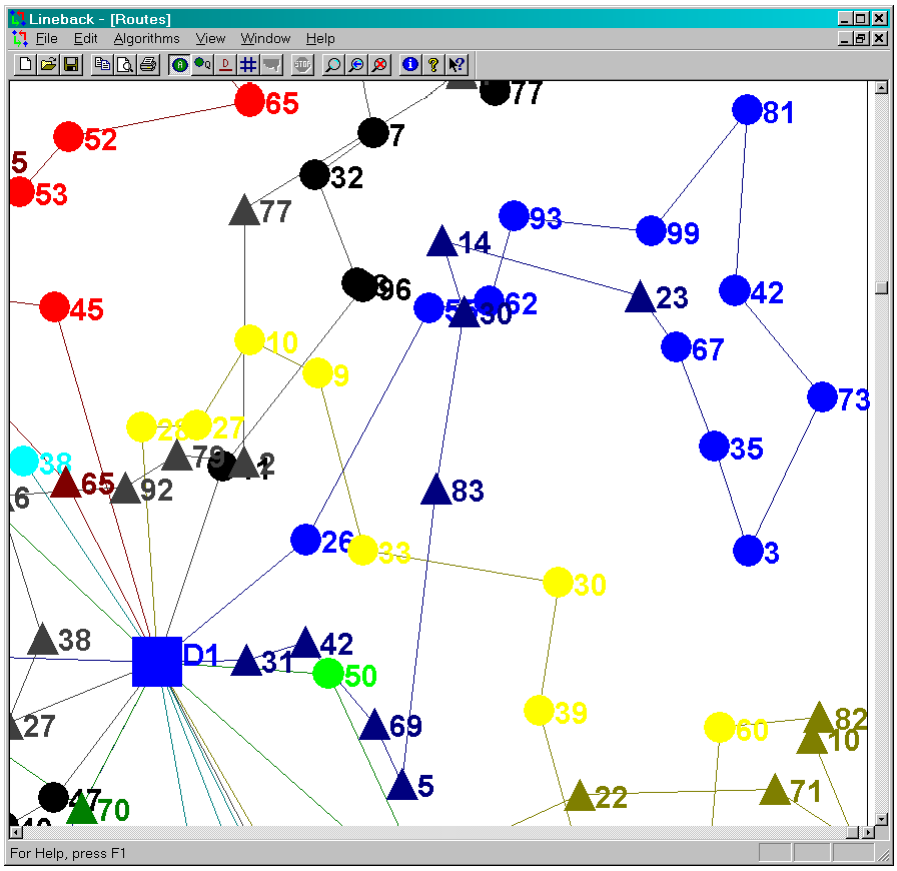
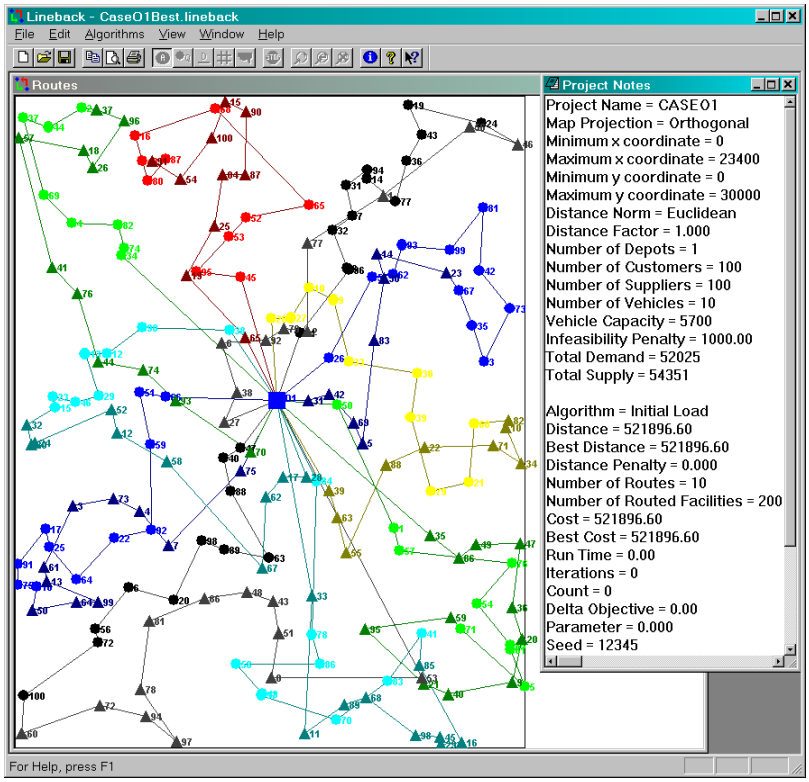


Figure 4.81. Vehicle Routing with Backhauling Tour Illustration


Lineback - CaseO1Best.lineback
File Edit Algorithms View Window Help
Routes
Project Notes
Project Name = CASEO1
Map Projection = Orthogonal
Minimum x coordinate = 0
Maximum x coordinate = 23400
Minimum y coordinate = 0
Maximum y coordinate = 30000
Distance Norm = Euclidean
Distance Factor = 1.000
Number of Depots = 1
Number of Customers = 100
Number of Suppliers = 100
Number of Vehicles = 10
Vehicle Capacity = 5700
Infeasibility Penalty = 1000.00
Total Demand = 52025
Total Supply = 54351
Algorithm = Initial Load
Distance = 521896.60
Best Distance = 521896.60
Distance Penalty = 0.000
Number of Routes = 10
Number of Routed Facilities = 200
Cost = 521896.60
Best Cost = 521896.60
Run Time = 0.00
Iterations = 0
Count = 0
Delta Objective = 0.00
Parameter = 0.000
Seed = 12345
For Help, press F1

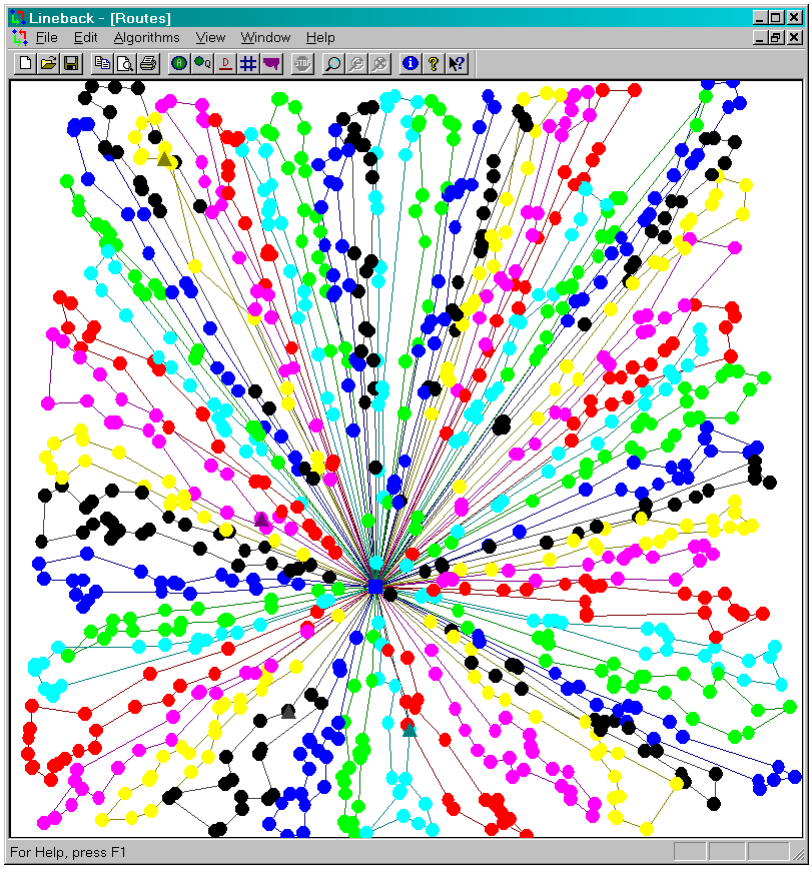

Lineback - [Routes]
File Edit Algorithms View Window Help
For Help, press F1

Figure 4.82. Lineback Illustrations

Vehicle Routing with Backhauling Algorithms

Nearest Neighbor

The Nearest Neighbor algorithm starts each route at the depot facility and then iteratively appends the nearest unvisited customer to the route. When appending the next customer would violate the truck capacity, the algorithm switches to iteratively appending the nearest unvisited supplier to the route. When appending the next supplier would violate truck capacity, then the route returns to the depot. If unvisited customers or suppliers remain and the maximum number of routes has not been reached, then a new route is started.

This algorithm was originally described by Rosenkrantz et al. (1977) for the Traveling Salesman Problem and is extended in a straightforward manner to the case of linehaul-backhaul vehicle routing.

Sweep

The Sweep algorithm rotates a ray around the depot. The starting angle and the rotational direction of the ray are algorithm parameters. When starting a new route, the first customer traversed by the ray becomes the first customer on the route, and the first supplier traversed by the ray becomes the last supplier on this route. When adding a facility to the route, it is inserted at one of the two endpoints of the empty vehicle travel link of the current tour and thus one of the two interface points of the empty vehicle travel link will change. Facilities are no longer being added to the current route, when adding the next facility would either violate the linehaul or the backhaul quantity on the vehicle. A new route is then started as long as the number of routes is less than or equal to the number of vehicles.

This algorithm was originally described by Gillett and Miller (1974) for the vehicle routing problem and is extended in a straightforward manner to the case of linehaul-backhaul vehicle routing.

Savings

Deif and Bodin (1984) adapted the original savings algorithm of Clarke and Wright (1964) to the case of the linehaul-backhaul vehicle routing. They observed that once the crossover link, corresponding to the empty vehicle travel, is added to a tour the tour becomes asymmetric and the possibilities for adding further facilities is cut in half. To avoid this phenomenon, they reduced the savings of the crossover link to delay the creation of this crossover link. The savings are then computed as:

$$s_{ij} = \begin{cases} d_{0j} + d_{i0} - d_{ij} - \alpha S & \text{if } i \in \mathbf{C}, j \in \mathbf{S} \\ d_{0j} + d_{i0} - d_{ij} & \text{if } i, j \in \mathbf{C} \text{ or } i, j \in \mathbf{S} \end{cases}$$

$$S = \max_{i,j} \left\{ d_{0j} + d_{i0} - d_{ij} \right\}$$

They found that a good value for α is around 0.2.

Vehicle Routing Problem with Backhauling Example

Consider the single origin vehicle routing problem with backhauling (VRPB). The distances between the various facilities are given in the network shown in Figure 4.83. The trucks can only drive over the given road network. The depot is indicated by a square. All routes start and terminate at the single depot. All suppliers are indicated by triangles and all customers are indicated by circles. The coordinates of the supplier and customer facilities and the supplies and demands are given in Table 4.44. All trucks have a capacity of 10 units. Additional constraints require that a truck can visit at most three facilities on a route aside from the origin and destination depot, because of the long load/unload times. We will solve this vehicle routing problem with several algorithms. If necessary, we will break any ties by selecting the facility with the smallest index.

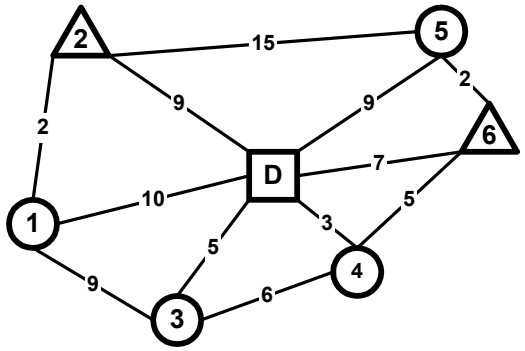


Figure 4.83. VRPB Example Network and Distance Data

Table 4.16. VRPB Example Facility Data

Facility	X-Coord	Y-Coord	Quantity
D	6	4	
1	1	3	4
2	2	7	6
3	4	1	7
4	8	2	5
5	10	7	8
6	11	5	9

We will first compute the shortest distance between every pair of facilities using Dijkstra's shortest path algorithm, since the distances on all network links are positive. The results are shown in the upper diagonal section of the distance matrix shown in Table 4.44.

Table 4.17. VRPB Example Interfacility Distances

	D	1	2	3	4	5	6
D		10	9	5	3	9	7
1			2	9	13	17	17
2				11	12	15	16
3					6	13	11
4						7	5
5							2
6							

Nearest Neighbor

We will first compute the routes with the Nearest Neighbor algorithm. Every time we add a facility to a route, we show the added or append facility, the anchor facility on the route to which this newly added facility is connected, the append distance to the new facility, and the linehaul and backhaul quantity on the route so far. When we start a new route, the anchor facility is the depot. The results are summarized in Table 4.18.

The closest unvisited customer to the depot is customer 4 with a distance equal to 3. Since this is the first customer on the route its demand will never exceed the truck capacity, so customer 4 is added to the route and the total demand or linehaul quantity on the route so far is equal to 5 units and the remaining linehaul capacity is also 5 units.

The unvisited customer closest to customer 4 is customer 3 with a distance equal to 6. The demand of customer 3 is 7 units and the remaining linehaul capacity on this route is 5 units, so customer 3 cannot be appended to this route. The route now switches from delivery to pickup operations. The unvisited supplier closest to customer 4 is supplier 6 with a distance equal to 5. Since this is the first supplier on the route its supply will never exceed the truck capacity, so supplier 6 is appended to the route. The backhaul quantity so far is equal to 9 units and the remaining backhaul capacity is 1 unit.

The unvisited supplier closest to supplier 6 is supplier 2 with a distance equal to 16. The supply quantity of supplier 2 is 6 units and the remaining backhaul capacity on the route is equal to 1 unit, so supplier 2 cannot be appended to this route. The route is closed and a new route is started.

The unvisited customer closest to the depot is customer 3 with a distance equal to 5. Since this is the first customer on the route its demand will never exceed the truck capacity, so customer 3 is appended to the route and the linehaul quantity on the route so far is equal to 7 units and the remaining linehaul capacity on the route is 3 units.

The unvisited customer closest to customer 3 is customer 1 with a distance equal to 9. The demand of customer 1 is 4 units and the remaining linehaul capacity on this route is 3 units, so customer 1 cannot be appended to this route. The route now switches from delivery to pickup operations. The unvisited supplier closest to customer 3 is supplier 2 with a distance equal to 11. Since this is the first supplier on the route its supply will never exceed the truck capacity, so supplier 2 is appended to the route. The backhaul quantity so far is equal to 6 units and the remaining backhaul capacity is 4 units.

Since there are no remaining unvisited suppliers, this route is closed and a new route is started. The unvisited customer closest to the depot is customer 5 with a distance equal to 9. Since this is the first customer on the route its demand will never exceed the truck capacity, so customer 5 is appended to the route and the linehaul quantity on the route so far is equal to 8 units and the remaining linehaul capacity is 2 units.

The unvisited customer closest to customer 5 is customer 1 with a distance equal to 17. The demand of customer 1 is 4 units and the remaining linehaul capacity on this route is 2 units, so customer 1 cannot be appended to this route. Since there are no remaining unvisited suppliers, this route is closed and a new route is started.

The unvisited customer closest to the depot is customer 1 with a distance equal to 10. Since this is the first customer on the route its demand will never exceed the truck capacity, so customer 1 is appended to the route and the linehaul quantity on the route so far is equal to 4 units and the remaining linehaul capacity is 6 units.

There are no remaining unvisited customers or suppliers, so this route is closed and the Nearest Neighbor algorithm terminates after having created four routes.

Table 4.18. VRPB Example Nearest Neighbor Facility Additions

Index	Append Facility	Anchor Facility	Append Distance	Linehaul Quantity	Backhaul Quantity
1	4	D	3	5	0
2	6	4	5	5	9
3	3	D	5	7	0
4	2	3	11	7	6
5	5	D	9	8	0
6	1	D	10	4	0

We list the sequence of all the facilities on the routes that we have created, using one route per row in the matrix shown in Table 4.19. Each route starts and ends with the depot. We compute the route length of each individual route and the total route length of all the routes created by this Nearest Neighbor algorithm. Since the Nearest Neighbor algorithm only created four routes, not all the rows in the following matrix are used.

Table 4.19. VRPB Example Nearest Neighbor Routes

Route	Facility 1	2	3	4	5	6	7	8	Length
1	D	4	6	D					15
2	D	3	2	D					25
3	D	5	D						18
4	D	1	D						20
5	D								
6	D								
Total									78

Sweep

There exist many variants of the sweep algorithm. We will execute the variant of the sweep algorithm where a route is terminated or closed as soon as adding either the next customer or next supplier facility would violate the truck capacity. We will start the rotating ray in the due north direction and turn the ray counterclockwise. Each time we add a facility to a route, we will show the added or append facility, the anchor facility on the route to which this newly added facility is connected, the append distance to the new facility, and the linehaul and backhaul quantity on the truck on that route so far. When we start a new route, the anchor facility is the depot for both the linehaul and the backhaul section of the route. Until a route is closed, it consists of a linehaul and a backhaul segment that have not yet been connected.

The first encountered facility is supplier 2. Since this is the first supplier on the route its supply will never exceed the truck capacity, so supplier 2 is included in the route. The total backhaul quantity on this route is now 6 units and the remaining backhaul capacity is 4 units.

The next facility traversed by the ray is customer 1. Since this is the first customer on the route its demand will never exceed the truck capacity, so customer 1 is included in the route and the linehaul quantity on the route so far is equal to 4 units and the remaining linehaul capacity is 6 units.

The next facility traversed by the ray is customer 3. The demand of customer 3 is 7 units and the remaining linehaul capacity on the route is 6 units, so customer 3 cannot be inserted in the route. So this route is closed by connecting customer 1 to supplier 2 and a new route is started.

The next facility traversed by the ray is customer 3. Since this is the first customer on the route its demand will never exceed the truck capacity, so customer 3 is included in the route and the linehaul quantity on the route so far is equal to 7 units and the remaining linehaul capacity is 3 units.

The next facility traversed by the ray is customer 4. The demand of customer 4 is 5 units and the remaining linehaul capacity on the route is 3 units, so customer 3 cannot be inserted in the route. So this route is closed and a new route is started.

The next facility traversed by the ray is customer 4. Since this is the first customer on the route its demand will never exceed the truck capacity, so customer 4 is included in the route and the linehaul quantity on the route so far is equal to 5 units and the remaining linehaul capacity is 5 units.

The next facility traversed by the ray is supplier 6. Since this is the first supplier on the route its supply will never exceed the truck capacity, so supplier 6 is included in the route and the total backhaul quantity on the route so far is equal to 9 units and the remaining backhaul capacity is 1 unit.

The next facility traversed by the ray is customer 5. The demand of customer 5 is 8 units and the remaining linehaul capacity on the route is 5 units, so customer 5 cannot be inserted in the route. So this route is closed by connecting customer 4 to supplier 6 and a new route is started.

The next facility traversed by the ray is customer 5. Since this is the first customer on the route its demand will never exceed the truck capacity, so customer 5 is included in the route and the linehaul quantity on the route so far is equal to 8 units and the remaining linehaul capacity is 2 units.

Table 4.20. VRPB Example Sweep Facility Additions

Index	Append Facility	Anchor Facility	Append Distance	Linehaul Quantity	Backhaul Quantity
1	2	D	9	0	6
2	1	D	10	4	6
3	3	D	5	7	0
4	4	D	3	5	0
5	6	D	7	5	9
6	5	D	9	8	0

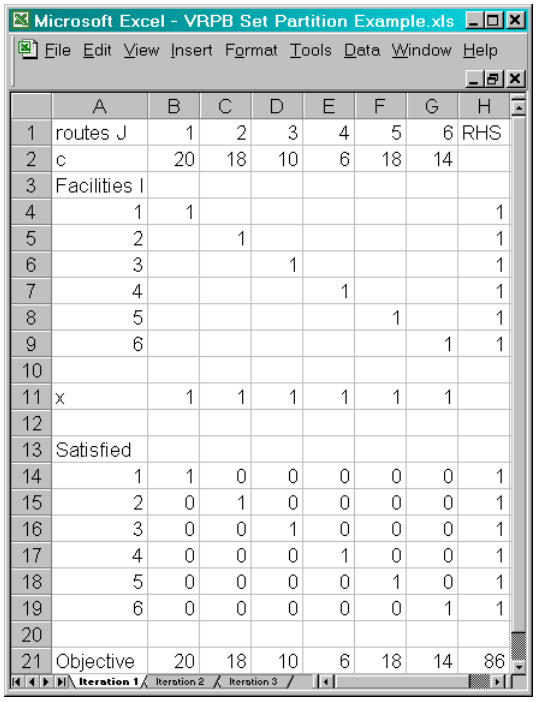
We will list the sequence of all the facilities on the routes that we have created, with one route per row in the following matrix. Each route starts and ends with the depot. We compute the route length of each individual route and compute the total route length of all the routes created by this Sweep algorithm. Since the Sweep algorithm created only four routes, not all rows or columns in the matrix are used.

Table 4.21. VRPB Example Sweep Routes

Route	Facility 1	2	3	4	5	6	7	8	Length
1	D	1	2	D					21
2	D	3	D						10
3	D	4	6	D					15
4	D	5	D						18
5	D								
6	D								
Total									64

Set Partitioning Algorithm

Next we will create the linehaul-backhaul routes for this problem with the Set Partitioning heuristic. We start of with a feasible partition for which each facility is on an individual route.



Microsoft Excel - VRPB Set Partition Example.xls

File Edit View Insert Format Tools Data Window Help

	A	B	C	D	E	F	G	H
1	routes J	1	2	3	4	5	6	RHS
2	c	20	18	10	6	18	14	
3	Facilities I							
4	1	1						1
5	2		1					1
6	3			1				1
7	4				1			1
8	5					1		1
9	6						1	1
10								
11	x	1	1	1	1	1	1	
12								
13	Satisfied							
14	1	1	0	0	0	0	0	1
15	2	0	1	0	0	0	0	1
16	3	0	0	1	0	0	0	1
17	4	0	0	0	1	0	0	1
18	5	0	0	0	0	1	0	1
19	6	0	0	0	0	0	1	1
20								
21	Objective	20	18	10	6	18	14	86

Iteration 1 | Iteration 2 | Iteration 3

Figure 4.84. VRPB Example Set Partitioning Single Facility Routes

We compute the total partition cost and then compute the row prices based on this partition. In general, we will allocate the row prices proportional to the product of the quantity and the distance of the facility to the depot. However, for the initial feasible partition, the row prices are equal to the round trip distances since each route contains a single facility.

$$T = 2 \cdot (10 + 9 + 5 + 3 + 9 + 7) = 2 \cdot 43 = 86$$

$$P_1 = 20 \qquad P_2 = 18 \qquad P_3 = 10$$

$$P_4 = 6 \qquad P_5 = 18 \qquad P_6 = 14$$

In the next column generation step, we consider only routes that have at most two facilities on each route (besides the depot). The formula used to compute the savings for each of these routes is

$$s_j = -c_j + \sum_i a_{ij} \cdot p_j$$

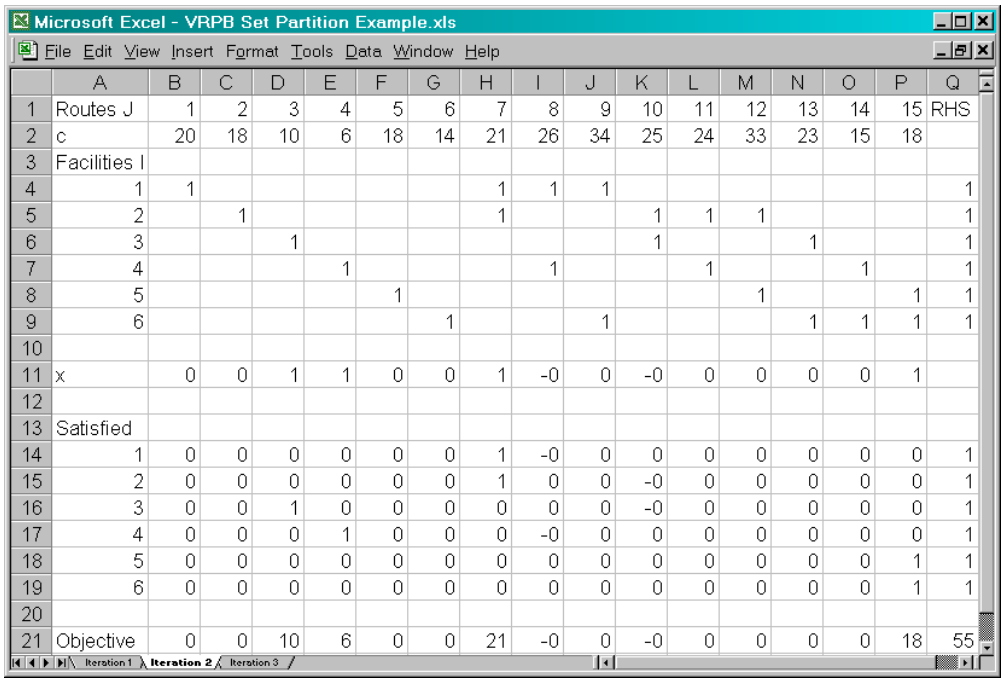
We compute the savings for each route considered and we indicate which routes get added to the master set partitioning problem. For each route, the route index, the sequence of facilities on that route, the route length, the potential route savings, and indication if this route is added to the set partitioning problem or not are shown in the next table. The start and end depot facility is included in the facility sequence of each route. We start indexing the routes with number 7, since there are already six out and

back routes included in the set partitioning problem. There are only nine feasible routes with two facilities on the route.

Table 4.22. VRPB Example Two Facility Routes

Index	Facility Sequence	Length	Savings	Added (Y/N)
7	D-1-2-D	21	17	Y
8	D-1-4-D	26	0	N
9	D-1-6-D	34	0	N
10	D-3-2-D	25	3	Y
11	D-4-2-D	24	0	N
12	D-5-2-D	33	3	Y
13	D-3-6-D	23	1	Y
14	D-4-6-D	15	5	Y
15	D-5-6-D	18	14	Y

Finally, we solve the resulting set partitioning problem, show the used routes and compute the total partition cost. Each route starts and ends with the depot. We compute the route length of each individual route and the total route length of the routes created by the Set Partitioning algorithm so far. Not all rows or columns have to be used.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1	Routes J	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	RHS
2	c	20	18	10	6	18	14	21	26	34	25	24	33	23	15	18	
3	Facilities I																
4	1	1						1	1	1							1
5	2		1					1			1	1	1				1
6	3			1							1			1			1
7	4				1				1			1			1		1
8	5					1							1			1	1
9	6						1			1				1	1	1	1
10																	
11	x	0	0	1	1	0	0	1	-0	0	-0	0	0	0	0	1	
12																	
13	Satisfied																
14	1	0	0	0	0	0	0	1	-0	0	0	0	0	0	0	0	1
15	2	0	0	0	0	0	0	1	0	0	-0	0	0	0	0	0	1
16	3	0	0	1	0	0	0	0	0	0	-0	0	0	0	0	0	1
17	4	0	0	0	1	0	0	0	-0	0	0	0	0	0	0	0	1
18	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
19	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
20																	
21	Objective	0	0	10	6	0	0	21	-0	0	-0	0	0	0	0	18	55

Figure 4.85. VRPB Example Set Partitioning Two-Facility Routes

Table 4.23. VRPB Set Partitioning Routes (Two Facilities Maximum)

Route	Facility 1	2	3	4	5	6	7	8	Length
1	D	1	2	D					21
2	D	3	D						10
3	D	4	D						6
4	D	5	6	D					18
5	D								
6	D								
Total									55

We compute the row prices based on this partition, where prices are allocated proportional to the product of the quantity and the distance of the facility to the depot. The row price formula following the notation used in class is thus.

$$p_i = \frac{d_{0i} \cdot q_i}{\sum_i a_{ij} \cdot d_{0i} \cdot q_i} c_j$$

$$p_1 = \frac{10 \cdot 4}{10 \cdot 4 + 9 \cdot 6} 21 = \frac{40}{94} 21 = 0.43 \cdot 21 = 8.94$$
$$p_2 = \frac{9 \cdot 6}{10 \cdot 4 + 9 \cdot 6} 21 = \frac{54}{94} 21 = 0.57 \cdot 21 = 12.06$$
$$p_3 = 10$$
$$p_4 = 6$$
$$p_5 = \frac{9 \cdot 8}{9 \cdot 8 + 7 \cdot 9} 18 = \frac{72}{135} 18 = 0.53 \cdot 18 = 9.60$$
$$p_6 = \frac{7 \cdot 9}{9 \cdot 8 + 7 \cdot 9} 18 = \frac{63}{135} 18 = 0.47 \cdot 18 = 8.40$$

In the next column generation step, we consider only routes that have three facilities on each route (besides the depot). We compute the savings for each route considered and indicate which routes get added to the set partitioning problem. We show for each route considered the route index, the sequence of facilities on that route, the route length, the potential route savings, and indication if this route is added to the set partitioning problem or not. The start and end depot facility are included in the facility sequence of each route. The computations for the savings of the route 24 are as follows

$$s_j = -c_j + \sum_i a_{ij} \cdot p_j =$$
$$s_{24} = -c_{24} + (p_1 + p_4 + p_2) = -44 + (8.94 + 6 + 12.06) = -44 + 27 = -17$$

Observe that route 23 has a savings potential with respect to the current row prices that is equal to zero. Including this route by itself will not improve the overall solution, but it gives additional flexibility

because alternative route sets, that have the same cost, can now be generated by the Set Partitioning algorithm.

Table 4.24. VRPB Three Facility Routes

Index	Facility Sequence	Length	Savings	Added (Y/N)
23	D-4-1-2-D	27	0	YorN
24	D-1-4-2-D	44	-17	N
25	D-4-1-6-D	40	-16.66	N
26	D-1-4-6-D	35	-11.66	N

Finally, we solve the resulting set partitioning problem, we show the used routes and compute the total partition cost. Each route starts and ends with the depot. We compute the route length of each individual route and the total route length of the routes created by the Set Partitioning algorithm. Observe that the algorithm selected the alternative configuration of routes, but with the same cost as the previous incumbent solution.

Microsoft Excel - VRPB Set Partition Example.xls

File Edit View Insert Format Tools Data Window Help

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1	Routes J	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	23	RHS
2	c	20	18	10	6	18	14	21	26	34	25	24	33	23	15	18	27	
3	Facilities I																	
4	1	1						1	1	1							1	1
5	2		1					1			1	1	1				1	1
6	3			1							1			1				1
7	4				1				1			1			1		1	1
8	5					1							1			1		1
9	6						1			1				1	1	1		1
10																		
11	x	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	1	
12																		
13	Satisfied																	
14	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
15	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
16	3	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1
17	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
18	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1
19	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1
20																		
21	Objective	0	0	10	0	0	0	0	0	0	0	0	0	0	0	18	27	55

Iteration 1 | Iteration 2 | Iteration 3

Figure 4.86. VRPB Example Set Partitioning Three-Facility Routes

Table 4.25. VRPB Set Partitioning Routes (Three Facilities Maximum)

Route	Facility 1	2	3	4	5	6	7	8	Length
1	D	4	1	2	D				27
2	D	3	D						10
3	D	5	6	D					18
4	D								
5	D								
6	D								
Total									55

The Set Partitioning algorithm as executed above generates a heuristic solution, without guarantee of optimality. However, in this small example case we generated all possible feasible routes since a route could contain at most three facilities. Solving the Set Partitioning problem with all the feasible routes included would generate the optimal solution. We can also solve the linear relaxation of the Set Partitioning formulation with all the feasible routes included. If the optimal linear programming solution value is not lower than the current best integer feasible solution value or incumbent, then the incumbent is optimal. Implementation of this last step requires only the solution of a linear programming formulation and not of an integer (binary) set partitioning formulation. However, for most real life cases, generating all feasible routes is not possible.

If we solve the linear relaxation of the final formulation, which contains 16 continuous variables, the optimal solution is identical to the solution of the binary Set Partitioning formulation. The optimal linear programming solution provides the following dual row prices

$$p_1 = 20, p_2 = 1, p_3 = 10, p_4 = 6, p_5 = 18, p_6 = 0$$

They represent a different cost allocation of the route cost to the facilities on the route. Observe that these costs satisfy the sum of the row prices condition expressed in equation (4.35). A similar strategy could have been followed for each iteration, i.e., the row prices are determined at each iteration by the optimal linear programming solution. If the integer and linear programming solution values are the same and no additional feasible route exists with positive savings, then the incumbent solution is optimal. Again, for most real life cases, it is impossible to establish that no additional feasible route exists with positive savings since it is impossible to generate all additional feasible routes.

4.6. Exercises

True-False Questions

Maximum flow networks are primarily used in the design of public sector and government networks, (T/F)_____(1).

The 2-Opt procedure by Lin to improve an existing TSP tour takes a sequence of nodes out of the tour and inserts it into another place in the tour to form a tour of shorter length, (TF) _____(2).

The use of computerized methods becomes more important in single vehicle roundtrip routing if the spatial relationships between the delivery points does not represent their true travel time, travel distance, or travel cost, (TF) _____(3).

Consider the classical Traveling Salesman Problem (TSP). The problem is to construct the single shortest cycle that visits all points exactly once, (T/F)_____(4).

If all the points of a TSP fall on the boundary of the convex hull of these points then this boundary of the convex hull is the shortest TSP tour, (T/F)_____(5).

If all possible columns, when evaluated with the current row prices, yield non-positive savings or equivalently non-negative reduced cost, then the current feasible partition on which the row prices are based is optimal, (T/F)_____(6).

On of the main advantage of set partitioning based solution methods is the fact that they can incorporate many complex feasibility constraints, (T/F)_____(7).

The largest problem instances of the Shortest Path Problem that can be solved in a reasonable amount of computer time contain about 2400 nodes, (T/F)_____(8).

The successive shortest path algorithm is currently the most efficient method for solving minimum cost network flow problems, (T/F)_____(9).

If the two and three exchange improvement procedures by Lin and Kernighan can no longer find any improvements in a traveling salesman tour then this tour is optimal, i.e., has the shortest length, (T/F)_____(10).

The subtour elimination constraints in the traveling salesman formulations make the problem so hard to solve because of the large coefficients in the right hand side of the constraints, (T/F)_____(11).

The vehicle routing problem attempts to minimize the cost of vehicles and distance traveled necessary to deliver goods to a number of customers with a fleet of vehicles, (T/F)_____(12).

The row price determination in the set partitioning algorithm can assign prices to the customers served on a particular route without any other conditions, (T/F)_____(13).

Consider the set partitioning problem solved with the row pricing algorithm as presented in class. If the master problem in the set partitioning algorithm is solved to optimality, then the routes selected by the master problem are optimal, (T/F)_____(14).

The network used to model emergency high-rise building evacuations belongs to the class of maximum flow networks, (T/F) _____(15).

The network used to model the assignment of workers to shifts in a 24-hour service operation belongs to the class of maximum flow networks, (T/F) _____(16).

The computational complexity of solving network formulations limits the size of the networks that can be solved to optimality to 10,000 arcs, (T/F) _____(17).

The standard network flow formulation can incorporate capacity restrictions on the nodes without any changes to the network structure, (T/F) _____(18).

A necessary requirement to convert the operator scheduling problem to a network flow formulation is that any shift or work tour covers an uninterrupted number of time periods, (T/F) _____(19).

The successive shortest path algorithm belongs to the class of dual algorithms, (T/F) _____(20).

Shortest Path Exercise One

Find the shortest duration path from node A (Amarillo, TX) to Node J (Fort Worth, TX) in the network below. The numbers on the edges indicate the approximate driving time between nodes, and the nodes indicate road junctions.

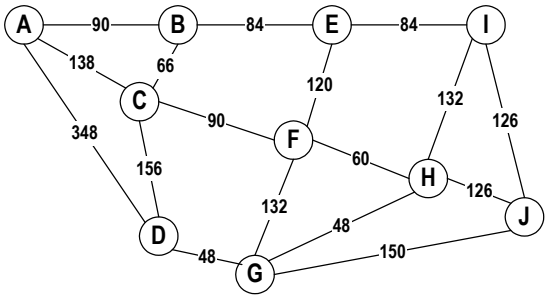


Figure 4.87. Simplified Highway Network for the Shortest Path Exercise

Shortest Path Exercise Two

Consider the network given in Figure 4.88. The length of each arcs is shown on the arc, with arrows indicated the arc length in a particular direction. Initial labels are shown above or below the nodes. Find manually the shortest path from node 1 to node 7 with Dijkstra's algorithm.

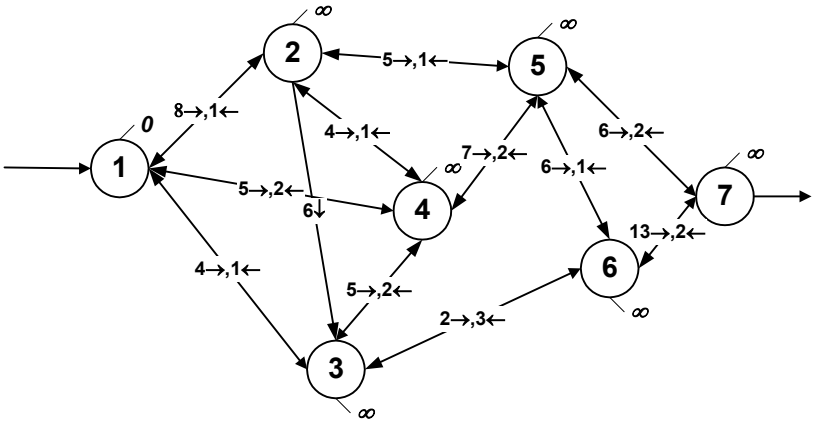


Figure 4.88. Shortest Path Exercise

Directed Graph Data Structures

Show first the network in the three data structures presented above for the shortest path network exercise 2 shown in Figure 4.88. Then show the three data structures for the network after the deletion of the arc from node 5 to node 4 and the insertion of a new arc from node 3 to node 2 with distance of 8.

Pseudo Code for Visiting All Direct Predecessors of a Node

For the three data structures presented above for the representation of directed graphs, give the pseudo code for implementing the two functions *find_first_predecessor* and *find_next_predecessor* used in performing an action on all direct or immediate predecessor nodes of node *k*. Use the notation developed above such as ***head*** and ***arc_array*** and ***pred*** and ***succ***. This pseudo code should consider and

incorporate the actual data structure used. The sequence of predecessor nodes is determined by the data structure used and is not relevant as long as all predecessor nodes get identified by these two functions.

The pseudo code for performing an action on all predecessor nodes of node *k* that is independent of the data structures used to represent the directed graph may look as follows:

Algorithm 4.4. All Predecessor Nodes Loop

```
p = find_first_predecessor (k)
while (p <> null) {
        perform_action (p)
        p = find_next_predecessor (k, p)
        }
```

where *p* and *k* are node indices and *perform_action* is the function that actually performs the required action and is not further discussed here

find_first_predessor (k) returns the first predecessor node of node *k* if node *k* has a predecessor or a *null* value indicating a non-existing node otherwise.

Find_next_predecessor (k, p) returns the next predecessor node of node *k* after the predecessor node *p* if the next predecessor node exists or a *null* value indicating a non-existing node otherwise.

This illustrates the principle where a different data structure implementation may be used to represent a directed graph and each data structure has different memory and execution time characteristics, but the higher level code would remain unchanged and does not need to be debugged again. Further information can be found in the references on the use of such abstract data types.

Pseudo Code for Listing All Ancestors of a Node

Develop an algorithm that will list all the ancestors of a node *k* in a network. In this exercise ancestor indicates any node from which the node *k* can be reached. The algorithm should be independent of the actual data structure implementing the directed graph structure. Show the pseudo code for this algorithm. The requested function is called *List_all_ancestors (k)*. The actual listing function is called *List_node (p)* and its implementation is not relevant. A node may be listed more than once as an ancestor of node *k*.

Modify the algorithm that you have developed above to incorporate the additional constraint that a node may be listed at most once as an ancestor of node k.

Ballou Ch7-4 Network Exercise

Compute the minimum cost network flows in the following network. Show the residual network after each iteration.

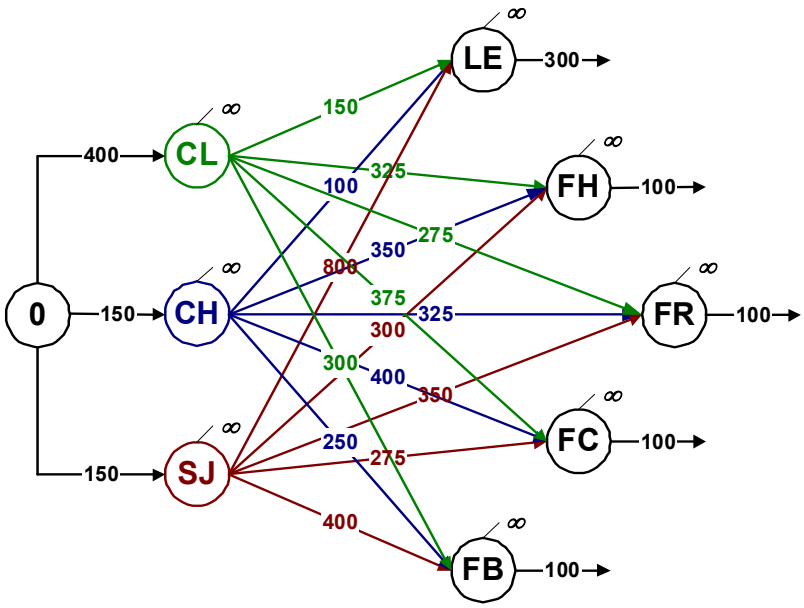


Figure 4.89. Network Exercise

Minimum Cost Network Exercise

Compute the minimum cost network flows in the following network. Show the residual network after each iteration. Interrupt the successive shortest path algorithm after two flow changes have been made. Compute the total cost of the flow solution determined so far. Is the current flow a feasible solution to the original problem? What would you be willing to pay for one extra unit of capacity on arc (1, 7)? Justify your answer in a single sentence. What would you be willing to pay for a reduction of a single unit in the required outgoing flow out of node 6 at this particular stage in the algorithm? Justify your answer in a single sentence.

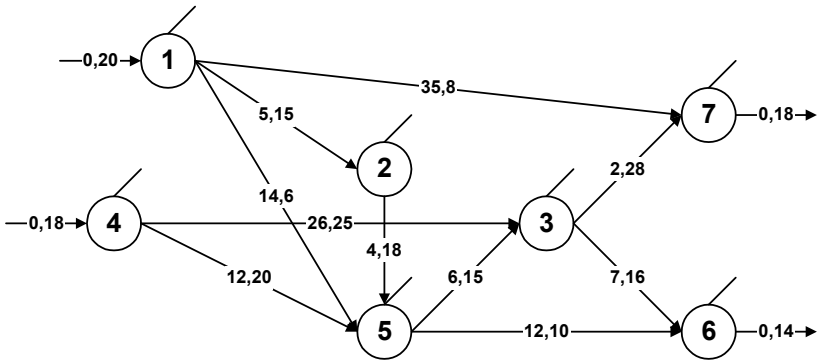


Figure 4.90. Network Exercise 3

TSP Exercise 1

Consider the problem of finding the shortest distance cycle through the following eight delivery points, given that the cycle visits each delivery point exactly once. The coordinates of the eight delivery points are given in Table 4.26.

Table 4.26. Point Coordinates

Point	x	y
1	5	1
2	8	9
3	3	5
4	7	6
5	1	9
6	2	1
7	4	8
8	8	2

The travel distances between each pair of points can be approximated with the Euclidean distance norm rounded to two digits behind the decimal point. First show the formula for the distance norm using the notation of Table 4.26 and point indices *i* and *j*. Next compute and show the distances in upper triangular format in Table 4.27.

Table 4.27. Delivery Points Distance Table

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

Compute heuristic delivery sequence with the Clarke and Wright (CW) savings algorithm. For the Clarke and Wright savings algorithm assume that the base or anchor point is point 2. First give the formula for the computation of the savings using the notation of Table 4.26 and observing that point 2 is the base point. For the Clarke and Wright algorithm show the pairwise savings with respect to the anchor or base point in Table 4.28. Show the savings based on points i and j ($j>i$) in matrix element [i,j]. Not all rows and columns in the table have to be computed or filled in!

Table 4.28. Clarke and Wright Savings

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

Write the formula that determines which points first form a shortcut on the tour, observing that point 2 is the base point. Assume that all points that not yet have been part of a shortcut and are not the base point are collected in a set F. Assume that the end points of the tour segment that has been rearranged with the shortcuts have indices p and q, respectively. Further assume that the base point has index b. Write the formula that determines which point will next form a shortcut on the tour.

For the Clarke and Wright savings algorithm, show chronologically the tour that you construct. First, show the length of the tour before any shortcut is made in row 1 of Table 4.29. Then compute and show for each iteration: the point for which you found the shortcut, the savings for the shortcut, the length of the tour after the shortcut, and the sequence of all the points on the tour that have been so far part of a

shortcut. In other words, you do not have to list the points that are by themselves on the out and back segments. For row 2, and row 2 only, two points are part of the first shortcut. Give both these points of the first shortcut in the column with "Shortcut Points".

Table 4.29. Clarke and Wright Tours

Route	Shortcut Points	Savings	Length	Points								
1	2			2								
2												
3												
4												
5												
6												
7												

TSP Exercise 2

Consider the problem of finding the shortest distance cycle through the following six delivery points, given that the cycle visits each delivery point exactly once. The coordinates of the six delivery points are given in Table 4.30.

Table 4.30. Delivery Point Coordinates for Exercise 2

#	x	y
1	200	900
2	900	200
3	400	300
4	100	100
5	300	600
6	800	700

The travel distances between each pair of delivery points can be approximated with the Euclidean distance norm rounded to the nearest integer value. Compute and show the distances in upper triangular format in Table 4.31.

Table 4.31. Delivery Points Distance Table for Exercise 2

	1	2	3	4	5	6
1						
2						
3						
4						
5						
6						

Compute heuristic delivery sequences with the Nearest Neighbor (NN), Sweep, Convex Hull followed by Priciest Insertion (CH+FI), and Clark and Wright (CW) savings algorithms. For the Nearest Neighbor algorithm the starting point is point 3. For the Sweep algorithm assume that the rotation center is located at coordinates (500, 500), the starting angle for the rotation ray is due east, and the rotation direction is clockwise. For the Clarke and Wright savings algorithm assume that the base or anchor point is point 6.

For the Convex Hull and Priciest Insertion algorithm show the insertion penalties for all points not on the convex hull after the convex hull has been determined in Table 32. Each row corresponds to an unvisited point and each column corresponds to an edge on the partial tour. Separate the insertion penalties for each iteration of the Priciest Insertion algorithm by a blank row from the lines for the next iteration. Clearly label your rows and columns. Not all rows and columns in the table have to be used.

Table 4.32. Insertion Penalties on the Convex Hull

For the Clarke and Wright algorithm show the pairwise savings with respect to the anchor point in Table 33. Show the savings based on points *i* and *j* (*j*>*i*) in matrix element [i,j]. Not all rows and columns in the table have to be computed.

Table 4.33. Clarke and Wright Savings

	1	2	3	4	5	6
1						
2						
3						
4						
5						
6						

For each algorithm give the chronological sequence in which you inserted or appended the points (starting with the base point or initial tour if applicable) and the delivery sequence starting with point 1. The chronological sequence is the sequence in which the points are added to the tour, i.e., which point is added first, second, third, etc. The delivery sequence is the sequence in which the points are visited by the tour. For example, is you inserted a point *k* between points *i* and *j* already on the tour, in the

chronological sequence point *k* would be the last point so far in the delivery sequence there would be a segment of points *i*, then *k*, and then *j*. For each heuristic compute the tour length and the ratio of the tour length divided by the length of the best tour you have found. Summarize your answer in the Table 4.34.

Table 4.34. Heuristics Summary Statistics

Heuristic	Chronological Sequence	Delivery Sequence	Length	Ratio
NN				
Sweep				
CH+FI				
CW				

Is the tour generated by the Convex Hull followed by Priciest Insertion algorithm guaranteed to be optimal? Base your answer only on the results of executing the Convex Hull and Priciest Insertion (CH+FI) algorithm. Give your answer in a few succinct sentences.

TSP Exercise 3

You are asked to route the crane in a person-aboard order picking system so that the distance traveled by the crane to pick up seven line items for the current customer order is minimized. The cabin with the order picker can move simultaneously up and down the mast of the crane while the crane moves back and forth in the order picking aisle. Therefore, the Chebyshev travel norm is judged an acceptable approximation to compute the travel distance of the crane. The crane starts and ends its picking route at the pickup and deposit (PD) station at lower front corner of the rack. The coordinates of the PD station and the seven bins storing the line items are given in Table 4.35, where the index of the PD station is zero.

Table 4.35. Point Coordinates

Point	x	y
0	0	0
1	9	9
2	3	5
3	4	3
4	1	9
5	2	1
6	5	6
7	9	2

First show the formula for the distance norm using the notation of Table 4.35 and point indices ***i*** and ***j***. Then compute and show the distances in upper triangular format in Table 4.36.

Table 4.36. Order Picking Bins Distance Table

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

Next, compute heuristic picking sequences with the Nearest Neighbor (NN), Sweep (SW), Convex Hull followed by Priciest Insertion (CH+PI), and Clark and Wright (CW) savings algorithms. For the Nearest Neighbor algorithm, the starting point is the PD station. For the Sweep algorithm, assume that the rotation center is located at PD station, the starting angle for the rotation ray is due east, and the rotation direction is counter-clockwise. Break any point selection or sequencing ties by selecting the point with the lowest index and any link selection ties by selecting the link first encountered on the tour.

Table 4.37. Nearest Neighbor Route Construction

Index	Anchor Point	Append Point	Append Distance
1			
2			
3			
4			
5			
6			
7			
8			
Total			

Table 4.38. Sweep Route Construction

Index	Anchor Point	Append Point	Append Distance
1			
2			
3			
4			
5			
6			
7			
8			
Total			

For the Convex Hull, use the point sequence that follows a counter-clockwise direction. For the Convex Hull and Priciest Insertion algorithm show the insertion penalties for all points not on the convex hull after the convex hull has been determined in Table 4.40. Each row corresponds to an unvisited point and each column corresponds to an edge on the partial tour. Separate the insertion penalties for each iteration of the Priciest Insertion algorithm by a blank row from the lines for the next iteration. Clearly label your rows and columns. Not all rows and columns in the table have to be used! An example of the table structure is shown in the next table, where the tour consists of points {a, b, c, d}, the remaining free points are {e, f, g} and in the first iteration point {f} is inserted on the link (b-c). This table shows the structure only, is not complete, and you should not modify this table in any way.

Table 4.39. Insertion Penalties Table Structure

Point	Tour Edges					
	(a-b)	(b-c)	(c-d)	(d-a)		
e						
f						
g						
	(a-b)	(b-f)	(f-c)	(c-d)	(d-a)	
e						
g						

Table 4.40. Insertion Penalties for Priciest Insertion

Point	Tour Edges							

Compute heuristic delivery sequence with the Clarke and Wright (CW) savings algorithm. For the Clarke and Wright savings algorithm assume that the base or anchor point is PD station. First, give the formula for the computation of the savings using the notation of Table 4.35 and observing that the PD station is the base point. For the Clarke and Wright algorithm show the pair-wise savings with respect to the anchor or base point in Table 4.41. Show the savings based on points i and j ($j>i$) in matrix element [i,j]. Not all rows and columns in the table have to be computed or filled in!

Table 4.41. Clarke and Wright Savings

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

Write the formula that determines which points first form a shortcut on the tour, observing that PD station is the base point. Assume that all points that not yet have been part of a shortcut and are not the base point are collected in a set ***F*** (indicating Free points). Assume that the end points of the tour segment, that has been rearranged with the shortcuts, have indices ***p*** and ***q***, respectively. Further assume that the base point has index ***b***. Write the formula that determines which point will next form a shortcut on the tour. For the Clarke and Wright savings algorithm, show chronologically the tour that you construct. First, show the length of the tour before any shortcut is made in row 1 of Table 4.42. Then compute and show for each iteration the point for which you found the shortcut, the savings for the shortcut, the length of the tour after the shortcut, and the sequence of all the points on the tour that so far have been part of a shortcut. In other words, you do not have to list the points that are by themselves on the out and back segments. For row 2, and row 2 only, two points are part of the first shortcut. Give both these points of the first shortcut in the column with title "Shortcut Points."

Table 4.42. Clarke and Wright Tours

Route	Shortcut Points	Savings	Length	Points							
1	0			0							
2											
3											
4											
5											
6											
7											

Finally, for each algorithm, give the chronological sequence in which you inserted or appended the points (starting with the base point or initial tour if applicable) and the delivery sequence starting with the PD station. The chronological sequence is the sequence in which the points are added to the tour, i.e., which point is added first, second, third, etc. The delivery sequence is the sequence in which the points are visited by the tour. For example, is you inserted a point *k* between points *i* and *j* already on the tour, in the chronological sequence point *k* would be the last point so far in the delivery sequence there would be a segment of points *i*, then *k*, and then *j*. For each heuristic compute the tour length and the ratio of the tour length divided by the length of the best tour you have found. Summarize your answer in the Table 4.43.

Table 4.43. Heuristics Summary Statistics

Heuristic	Chronological Sequence	Delivery Sequence	Length	Ratio
NN				
Sweep				
CH+PI				
CW				

Vehicle Routing with Backhauls (VRPB) Exercise 1

Consider the single origin vehicle routing problem with backhauls (VRPB) illustrated in Figure 4.91. All routes start and terminate at the single depot. The depot is indicated by a square. All suppliers are indicated by triangles and all customers are indicated by circles. The distances between the various facilities are also shown in Figure 4.91. All travel occurs over the network links. The supplies and demands are expressed in fractions of truck capacity and are shown in Table 4.44. Due to the long load/unload times, there is only enough time to visit three facilities on a single truck route. Hence, an additional constraint requires that there are at most three facilities on a route, aside from the origin and destination depot.

Table 4.44. VRPB Example Facility Data

Customer or Supplier	Demand or Supply	Distance to Depot
1	0.10	9
2	0.20	9
3	0.30	7
4	0.40	3
5	0.50	5
6	0.45	10

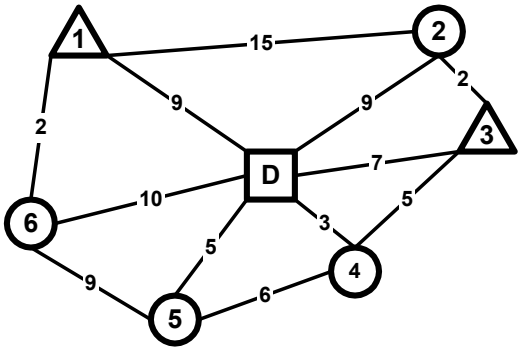


Figure 4.91. VRPB Exercise 1 Transportation Network

Table 4.45. VRPB Exercise 1 Interfacility Distances

	D	1	2	3	4	5	6
D							
1							
2							
3							
4							
5							
6							

Nearest Neighbor

First construct the routes with the Nearest Neighbor algorithm. Every time you add a facility to a route, show the added or append facility, the anchor facility on the route this newly added facility is connected to, the append distance to the new facility, and the linehaul and backhaul quantity on the truck on that route so far. When a new route is started, the anchor facility is the depot.

Table 4.46. VRPB Exercise 1 Nearest Neighbor Algorithm Steps

Index	Append Facility	Anchor Facility	Append Distance	Linehaul Quantity	Backhaul Quantity
1					
2					
3					
4					
5					
6					

List the sequence of all the facilities on the routes that you have created, with one route per line or row. Each route must start and end with the depot. Compute the route length of each individual route and the total route length of all the routes created by this algorithm. Not all rows or columns in the table have to be used.

Table 4. 47. VRPB Exercise 1 Nearest Neighbor Routes

Route	Facility 1	2	3	4	5	6	7	8	Length
1	D								
2	D								
3	D								
4	D								
5	D								
6	D								
Total									

Sweep

There exist many variants of the sweep algorithm. Execute the sweep algorithm for the VRPB problem where a route is terminated or closed as soon as adding either the next customer or supplier facility would violate the truck capacity and where the facilities are sequenced on the route by the rotating ray. Start the rotating ray in the due east direction and turn the ray clockwise. For every time you add a facility to a route, show the added or append facility, the anchor facility on the route this newly added facility is connected to, the append distance to the new facility, and the linehaul and backhaul quantity on the truck on that route so far. When you start a new route, the anchor facility is the depot.

Table 4.48. VRPB Exercise 1 Sweep Algorithm Steps

Index	Append Facility	Anchor Facility	Append Distance	Linehaul Quantity	Backhaul Quantity
1					
2					
3					
4					
5					
6					

List the sequence of all the facilities on the routes that you have created, with one route per line. Each route must start and end with the depot. Compute the route length of each individual route and compute the total route length of all the routes created by this algorithm. Not all rows or columns in the table have to be used.

Table 4.49. VRBP Exercise 1 Sweep Routes

Route	Facility 1	2	3	4	5	6	7	8	Length
1	D								
2	D								
3	D								
4	D								
5	D								
6	D								
Total									

Set Partitioning Algorithm

In the next column generation step, we consider only routes that have at most two facilities on each route (besides the depot). Observe that these routes have to satisfy the standard vehicle routing with backhauls condition that all customers on a route are visited before any supplier can be visited. We use the standard formula to compute the savings for each route candidate.

Table 4.50. VRBP Exercise 1 One-Facility Routes

Route	Facility 1	2	3	4	5	6	7	8	Length
1	D								
2	D								
3	D								
4	D								
5	D								
6	D								
Total									

Table 4.51. VRPB Exercise 1 Two-Facility Route Evaluations

Index	Facility Sequence	Length	Savings	Added (Y/N)

Table 4.52. VRBP Exercise 1 Two-Facility Routes

Route	Facility 1	2	3	4	5	6	7	8	Length
1	D								
2	D								
3	D								
4	D								
5	D								
6	D								
Total									

Table 4.53. VRPB Exercise 1 Three-Facility Route Evaluations

Index	Facility Sequence	Length	Savings	Added (Y/N)

Table 4.54. VRBP Exercise 1 Three-Facility Routes

Route	Facility 1	2	3	4	5	6	7	8	Length
1	D								
2	D								
3	D								
4	D								
5	D								
6	D								
Total									

Vehicle Routing Problem with Backhauling Exercise 2

Find the routes that minimize the total distance traveled to supply a number of customers from a single depot and to pickup from a number of suppliers and return to the depot. The trailers are assumed to be rear loaded, so on every trip all deliveries have to be made before any pickup can be made. All trucks are assumed to be the same size. There are four vehicles, each with a capacity of 160. Due to loading and time constraints, each vehicle can only execute a single route. There are a total of 34 linehaul customers and 16 backhaul suppliers. The location of the customers and suppliers and their respective

demand and supply are given in the table below. Use the Euclidean distance norm without any adjustment factor to determine the interfacility distances.

You can use any computer algorithm and program that you desire to determine the routes. You need to show the routes on a graph and provide the following summary statistics for each route and for the total problem: route distance, number of linehaul customers, total linehaul demand, number of backhaul suppliers, total backhaul supply. Report which computer program and what computer hardware you used. If you use a commercial software package you are encouraged to report the purchase price of the software. Describe clearly and succinctly the algorithmic steps you or the computer program used to generate the routes. Record and the report the time it took to enter the data, to find the shortest distance routes, and to decode the solution of the computer program and to copy it to the solution graph. So you need to report three different times. Part of the grade will be based on the total length of the routes you generated, i.e., shorter routes will get you a better grade.

Table 4.55. VRPB Exercise 2 Facility Locations

Label	X	Y	Demand/Supply	
Depot				
D1	10000	10000	0	
Linehaul Customers				
1	10070	10120	7	
2	10190	10090	30	
3	9900	9860	9	
4	10100	9900	21	
5	9870	10230	19	
6	10010	10220	23	
7	10210	9810	5	
8	10120	10010	19	
9	9750	9850	23	
10	9820	10020	21	
11	10220	10010	15	
12	9970	9830	3	
13	9830	9730	9	
14	10270	10180	28	
15	10120	10170	8	
16	9860	10170	16	
17	9770	9980	28	
18	9970	10280	7	
19	10130	10270	14	
20	10280	10080	6	
21	10070	10290	11	
22	10080	10060	12	
23	10310	9930	26	
24	10320	10230	17	
25	10020	9820	9	
26	10150	9950	15	
27	9750	9660	7	
28	9800	9770	27	
29	9750	10240	11	
30	10000	9750	16	
31	10020	9990	5	
32	9950	9920	25	
33	10180	9880	18	
34	10260	9970	10	

Backhaul Suppliers				
1	10220	10240	16	
2	9910	10070	15	
3	10220	9930	11	
4	10010	9920	29	
5	10060	9760	10	
6	9870	9930	41	
7	10320	10020	8	
8	9780	10120	10	
9	10000	10080	15	
10	10280	9870	19	
11	10160	9700	23	
12	10330	10290	6	
13	10290	9750	14	
14	9910	9700	13	
15	10090	9700	10	
16	9950	10150	17	
Note: Use Euclidean Distance Norm (no adjustments)				

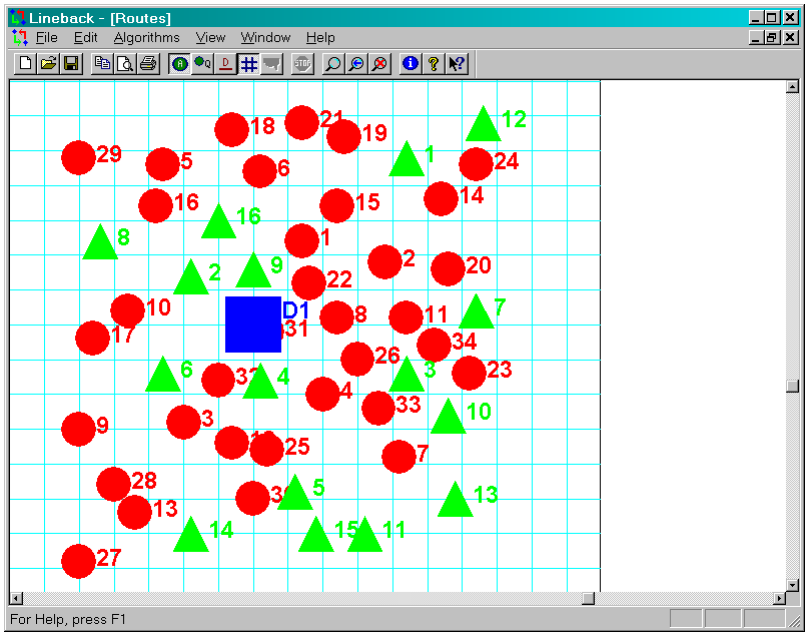


Figure 4.92. VRPB Exercise 2 Facility Locations

4.7. References

Publications

1. Aho, A., J. Hopcroft, and J. Ullman, (1983**).** **Data Structures and Algorithms**. Addison-Welsey, Reading, Massachusetts.

2. Ahuja, R., T. Magnanti, and J. Orlin, (1993). **Network Flows**. Prentice Hall, Englewood Cliffs, New Jersey.

3. Akl S. G. and G. T. Toussaint, (1978). "A Fast Convex Hull Algorithm". ***Information Processing Letters***, Vol. 7, No. 5, pp. 219-222.

4. Allison D. C. and Noga, M. T., (1984), "The L_1 Traveling Salesman Problem." ***Information Processing Letters***, Vol. 18, No. 4, pp. 195-199.

5. Balas E. and Padberg M., (1976). "Set Partitioning: A Survey." ***SIAM Review***, Vol. 18, No. 4, pp. 710-760.

6. Ball, M. O., T. L. Magnanti, C. L. Monma, G. L. Nemhauser (eds.), (1995). **Network Routing**. Elsevier Science, Amsterdam, The Netherlands.

7. Bellmore, M. and Nemhauser, G. L., (1968), "The Traveling Salesman Problem: A Survey," ***Operations Research***, Vol. 16, No. 3, pp. 538-558.

8. Boyd S. C., W. R. Pulleyblank and G. Cornuejols, (1987). "TRAVEL - An Interactive Traveling Salesman Problem Package for the IBM Personal Computer," ***Operations Research Letters***, Vol. 6, No. 3, pp. 141-143.

9. Christofides N. and Eilon S., (1972), "Algorithms For Large-Scale Traveling Salesman Problems," ***Operations Research Quarterly***, Vol. 23, pp. 511-518.

10. Christofides, N., (1975). **Graph Theory: An Algorithmic Approach**. Academic Press, New York.

11. Cullen F., Jarvis J. and Ratliff H., (1981). "Set Partitioning Based Heuristics for Interactive Routing." ***Networks***, Vol. 11, No. 2, pp. 125-143.

12 Clarke, G. and J. Wright, (1964). "Scheduling of Vehicles From a Central Depot to A Number of Delivery Points." ***Operations Research***, Vol. 12, pp 568-581.

13. Deif, I. and L. D. Bodin, (1984). "Extension of the Clarke-Wright Algorithm for Solving the Vehicle Routing Problem With Backhauling". Proceedings of the Babson Conference on Software Uses in Transportation and Logistics Management, A. E. Kidder, Ed., Babson Park, MA, pp. 75-96.

14. Duhamel, C., J.-Y. Potvin, and J.-M. Rousseau, (1997). "A Tabu Search Heuristic for the Vehicle Routing Problem with Backhauls and Time Windows." ***Transportation Science***, Vol. 31, No. 1, pp. 49-59.

15. Evans, J. and E. Minieka, (2nd Ed.) (1992). **Optimization Algorithms for Networks and Graphs**. Marcel Dekker, New York, New York.

16. Fishetti, M., P. Toth, and D. Vigo, (1994). "A Branch-and-Bound Algorithm for the Capacitated Vehicle Routing Problem on Directed Graphs." ***Operations Research***, Vol. 42, No. 5, pp. 846-859.

17. Gendreau, M., G. Laporte and A. Hertz, (1997). "An Approximation Algorithm for the Traveling Salesman Problem with Backhauls." ***Operations Research***, Vol. 45, No. 4, pp. 639-641.

18 Gillett, B. and L. Miller, (1974), "A Heuristic Algorithm For the Vehicle Dispatch Problem," ***Operations Research***, Vol. 22, pp 340-349.

19. Golden, B. and A. Assad, (Eds.), (1988). **Vehicle Routing: Methods and Studies**. North Holland, Amsterdam.

20. Golden, B. L., Bodin, L., Doyle, T., and Stewart, W. Jr., (1980), "Approximate Traveling Salesman Algorithms," ***Operations Research***, Vol. 28, No. 3, pp. 694-711.

21. Halse, K. (1992). Modeling and Solving Complex Vehicle Routing Problems. Ph.D. Dissertation, Technical University of Denmark.

22. Helbig Hansen, K. and Krarup, J., (1974), "Improvements on the Held-Karp Algorithm for the Symmetric Traveling Salesman Problem," ***Mathematical Programming***, Vol. 7, pp. 87-96.

23. Held, M. and Karp, R. M., (1970), "The Traveling-Salesman Problem and Minimum Spanning Trees," ***Operations Research***, Vol. 18, No. 6, pp. 1138-1162.

24. Held, M. and Karp, R. M., (1971), "The Traveling-Salesman Problem and Minimum Spanning Trees: Part II," ***Mathematical Programming***, Vol. 1, pp. 6-25.

25. Horowitz, E., and S. Shani, (1984**).** **Fundamentals of Computer Algorithms**. Computer Science Press, Rockville, Maryland.

26. Kennington, J. and R. Helgason, (1980). **Algorithms for Network Programming**. John Wiles & Sons, New York, New York.

27. Kirkpatrick, S., C. Gelat, and M. Vechi, (1983). ***Science***, Vol. 220, pp 671-680.

28. Laporte, G., (1992), "The Traveling Salesman Problem: An Overview of Exact and Approximate Algorithms," ***European Journal of Operational Research***, Vol. 59, pp. 231-247.

29. Lawler E. L., Lenstra J. K., Rinnooy Kan A. H. G., and Schmoys, D. B., (1985), **The Traveling Salesman Problem**, John Wiley & Sons, Chichester, Great Britain.

30. Lin S., and B. Kernighan, (1973), "An Effective Heuristic Algorithm for the Traveling Salesman Problem," ***Operations Research***, Vol. 21, pp. 498-516.

31. Lin, S., (1965), "Computer Solutions of the Traveling Salesman Problem," ***Bell System Technical Journal***, Vol. 44, pp. 2245-2269.

32. Little J. D., Murty K. G., Sweeney D. W. and Karel C., (1963), "An Algorithm for the Traveling Salesman Problem," ***Operations Research***, Vol. 11, No. 6, pp. 972-989.

33. Or, I., (1976), "Traveling Salesman-Type Combinatorial Problems and their Relation to the Logistics of Regional Blood Banking," Unpublished Ph.D. Dissertation, Northwestern University, Evanston, Illinois.

34. Parker R, G. and Rardin R. L., (1983). "The Traveling Salesman Problem: An Update of Research," ***Naval Research Logistics Quarterly***, Vol. 30, pp. 69-99.

35. Platzman, Loren K. and John J. Bartholdi, III, (1984), "Spacefilling Curves and the Planar Traveling Salesman Problem," PDRC Report Series 83-02, School of Industrial and Systems Engineering, Georgia Institute of Technology.

36. Rosenkrantz, D. J., R. E. Stearns, and P. M. Lewis, (1977), "An Analysis of Several Heuristics for the Traveling Salesman Problem," ***SIAM Journal of Computing***, Vol. 6, pp. 563-581.

37. Savelsbergh, M. and M. Goetschalckx, (1994). "A Comparison of the Efficiency of Fixed versus Variable Vehicle Routes." ***Journal of Business Logistics***, Vol. 16, No. 1, pp. 163-188.

38. Sedgewick, R. (1983). **Algorithms**. Addison-Wesley, Reading, Massachusetts.

39. Smith T. H. and Thompson G. L., (1977), "A LIFO Implicit Enumeration Search Algorithm for the Symmetric Traveling Salesman Problem using Held and Karp's 1-Tree Relaxation," ***Annals of Discrete Mathematics***, Vol. 1, pp. 479-493.

40. Toth, P. and D. Vigo, (1996). "A Heuristic Algorithm for the Vehicle Routing Problem with Backhauls", in **Advanced Methods in Transportation Analysis**, L. Bianco and P Toth (eds.), Springer-Verlag, Berlin, pp. 585-608.

41. Toth, P. and D. Vigo, (1997). "An Exact Algorithm for the Vehicle Routing Problem with Backhauls." ***Transportation Science***, Vol. 31, No. 4, pp. 372-386.

42. Vechi, M. and S. Kirkpatrick, (1983). ***IEEE Transactions on Computer Aided Design***, Vol. CAD-2, pp. 215.

43. Volgenant, T. and R. Jonker, (1982), "A Branch and Bound Algorithm for the Symmetric Traveling Salesman Problem based on the 1-Tree Relaxation," ***European Journal of Operations Research***, Vol. 9, pp. 83-89.

Programs

44. **Tours**, (1993). Marc Goetschalckx

45. **Lineback** (1997). Marc Goetschalckx