**LOG820 Vehicle Routing**

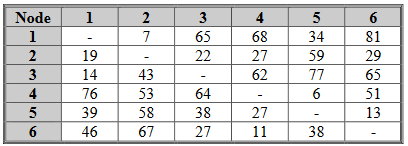
**Spring 2022**

**Home Exam 30%**

**Candidate number 28**

# Problem 1. Travelling Salesman Problem

a) Below you find a cost matrix for the asymmetric graph with 6 nodes.



Find the optimal Hamiltonian cycle by using the model for Asymmetric TSP with the Miller-Tacker-Zemlin (MTZ) sub-cycle prohibiting constraints;

**2) AMPL code**

File task1a.dat:

param n := 6;

param cost:

1 2 3 4 5 6:=

1 . 7 65 68 34 81

2 19 . 22 27 59 29

3 14 43 . 62 77 65

4 76 53 64 . 6 51

5 39 58 38 27 . 13

6 46 67 27 11 38 .;

File task1a.mod:

param n;

set ARCS := {i in 1..n, j in 1..n: i<>j};

param cost {ARCS} >=0;

var Route {ARCS} binary;

var Load {2..n} >=1, <= n-1;

minimize Total\_Cost:

sum {(i,j) in ARCS} cost[i,j] \* Route[i,j];

subject to Out\_Of {i in 1..n}:

sum {(i,j) in ARCS} Route[i,j] = 1;

subject to Route\_Continuity {i in 1..n}:

sum {(i,j) in ARCS} Route[i,j]= sum {(j,i) in ARCS} Route[j,i];

subject to Load\_Continuity{i in 2..n, j in 2..n: i<>j}:

Load[j] >= Load[i] + 1+ (n-1)\*(Route[i,j] - 1);

File task1a.run:

option solver cplex;

model task1a.mod;

data task1a.dat;

option show\_stats 1;

solve;

display Total\_Cost > task1a.sol;

display Route > task1a.sol;

File task1a.sol:

Total\_Cost = 94

Route [\*,\*]

: 1 2 3 4 5 6 :=

1 . 1 0 0 0 0

2 0 . 0 1 0 0

3 1 0 . 0 0 0

4 0 0 0 . 1 0

5 0 0 0 0 . 1

6 0 0 1 0 0 .

;

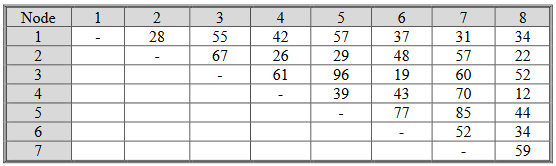
**3) Solution**

Optimal route should be 1-2-4-5-6-3-1.

Total cost in this case will be 94.

b) Saint-Martin company distributes fresh fishing products in Normandy, France. Last 7 June, the company received seven orders from sales points all located in northern Normandy. It was decided to serve the seven requests by means of a single vehicle sited in Betteville. The problem was formulated as an STSP on a complete graph, shown in the figure below, where vertices from 2 to 8 correspond to the seven sales points and vertex 1 is associated with the depot.

The shortest distances (in kilometres) between the vertices are reported in the following table:



Find the optimal Hamiltonian cycle by using the model for Symmetric TSP and iteratively eliminating generated sub-cycles (if any) by adding the corresponding sub-cycle eliminating constraints (SEC);

**2) AMPL code**

File task1b.dat:

param n:= 6;

param K := 2;

set SUBN[1] := 4 5 6;

set SUBN[2] := 1 2 3;

param cost:

1 2 3 4 5 6 7 8:=

1 . 28 55 42 57 37 31 34

2 . . 67 26 29 48 57 22

3 . . . 61 96 19 60 52

4 . . . . 39 43 70 12

5 . . . . . 77 85 44

6 . . . . . . 52 34

7 . . . . . . . 59;

File task1b.mod:

param n;

param K; # number of sub-cycles

set EDGES:= {i in 1..n,j in 1..n: i < j};

param cost {EDGES} >=0;

set SUBN {k in 1..K}; # set of nodes in sub-cycle Sk

set SUBE {k in 1..K}:= {i in SUBN[k], j in SUBN[k]: i < j}; # edges

var Route {EDGES} binary;

minimize Total\_Cost:

sum {(i,j) in EDGES} cost[i,j] \* Route[i,j];

subject to Degree\_Const {i in 1..n}:

sum {(i,j) in EDGES} Route[i,j] + sum {(j,i) in EDGES} Route[j,i] = 2;

subject to SEC {k in 1..K}: # SEC

sum {(i,j) in SUBE [k]} Route [i,j] <= card(SUBN [k]) - 1;

file task1b.run:

option solver cplex;

model task1b.mod;

data task1b.dat;

option show\_stats 1;

solve;

display Total\_Cost > task1b.sol;

display Route > task1b.sol;

file task1b.sol:

Total\_Cost = 205

Route :=

1 2 0

1 3 1

1 4 0

1 5 0

1 6 1

2 3 0

2 4 1

2 5 1

2 6 0

3 4 0

3 5 0

3 6 1

4 5 1

4 6 0

5 6 0

;

**3) Solution**

Optimal route should be 1-3-4-5-2-6-1.

Total cost in this case will be 205.

c) Find the shortest Hamiltonian path for the graph in a) using the Asymmetric TSP model on the extended graph with one added artificial node and corresponding arcs.

**2) AMPL code**

File task1c.dat:

param n:= 7;

param cost:

1 2 3 4 5 6 7:=

1 . 7 65 68 34 81 0

2 19 . 22 27 59 29 0

3 14 43 . 62 77 65 0

4 76 53 64 . 6 51 0

5 39 58 38 27 . 13 0

6 46 67 27 11 38 . 0

7 0 0 0 0 0 0 .;

File task1c.sol:

Total\_Cost = 67

Route [\*,\*]

: 1 2 3 4 5 6 7 :=

1 . 1 0 0 0 0 0

2 0 . 0 1 0 0 0

3 1 0 . 0 0 0 0

4 0 0 0 . 1 0 0

5 0 0 0 0 . 1 0

6 0 0 0 0 0 . 1

7 0 0 1 0 0 0 .

;

**3) Solution**

Optimal route should be 1-2-4-5-6-7-3-1.

Total cost in this case will be 67.

d) Find for the graph from b) the optimal solution to 2-TSP problem using the Symmetric TSP model where two travelling persons start their tours from node 1, and they visit at least three nodes each.

**2) AMPL code**

File task1d.dat:

param n:= 8;

param m:= 4;

param v:= 3;

param cost:

1 2 3 4 5 6 7 8:=

1 . 28 55 42 57 37 31 34

2 . . 67 26 29 48 57 22

3 . . . 61 96 19 60 52

4 . . . . 39 43 70 12

5 . . . . . 77 85 44

6 . . . . . . 52 34

7 . . . . . . . 59;

param start:= 1;

file task1d.mod:

param n;

param m;

param v;

set EDGES:= {i in 1..n,j in 1..n: i < j};

param cost {EDGES} >=0;

param start; #start

var Route {EDGES} binary;

minimize Total\_Cost:

sum {(i,j) in EDGES} cost[i,j] \* Route[i,j];

subject to Start: #Start from depot

sum {(start,j) in EDGES} Route[start,j] = m;

subject to Degree\_Const {i in 2..n}:

sum {(i,j) in EDGES} Route[i,j] + sum {(j,i) in EDGES} Route[j,i] = 2 ;

subject to Visit {i in 1..n}:

sum {(i,j) in EDGES} Route[i,j] >= v;

file task1d.run:

option solver cplex;

model task1d.mod;

data task1d.dat;

option show\_stats 1;

solve;

display Total\_Cost > task1d.sol;

display Route > task1d.sol;

file task1d.sol:

Total\_Cost = 283

Route [\*,\*]

: 2 3 4 5 6 7 8 :=

1 1 0 0 0 1 1 1

2 . 0 0 1 0 0 0

3 . . 0 0 1 1 0

4 . . . 1 0 0 1

5 . . . . 0 0 0

6 . . . . . 0 0

7 . . . . . . 0

;

**3) Solution**

There should be 2 optimal routes: 1-2-5-4-8-1 and 1-6-3-7-1.

Total cost in this case will be 283.

e) Assume that in the graph from b) the start node is 1 and the incomes collected when visiting nodes are: node 2 is 30, node 3 is 15, node 4 is 20, node 5 is 40, node 6 is 55, node 7 is 45, and node 8 is 45. Using the corresponding mathematical models for TSP with Profits

i) Find the optimal tour on this graph that maximizes the profit, and specify the income collected and the cost of travel;

**2) AMPL code**

File task1e.dat:

param n := 8;

param cost:

1 2 3 4 5 6 7 8:=

1 . 28 55 42 57 37 31 34

2 . . 67 26 29 48 57 22

3 . . . 61 96 19 60 52

4 . . . . 39 43 70 12

5 . . . . . 77 85 44

6 . . . . . . 52 34

7 . . . . . . . 59;

param income := 2 30 3 15 4 20 5 40 6 55 7 45 8 45;

File task1e.mod:

param n;

set ARCS := {i in 1..n, j in 1..n: i<j};

param cost {ARCS} >= 0;

param income {2..n};

var x {ARCS} >= 0 binary;

var y {2..n} >= 0 binary;

maximize Total\_Profit:

sum {i in 2..n} income[i]\*y[i] - sum {(i,j) in ARCS} cost[i,j]\*x[i,j];

subject to Start\_Depot:

sum {(1,j) in ARCS } x[1,j] = 2;

subject to Linking {i in 2..n}:

sum {(i,j) in ARCS } x[i,j] + sum {(j,i) in ARCS } x[j,i]= 2\*y[i];

File task1e.run:

option solver cplex;

model task1e.mod;

data task1e.dat;

option omit\_zero\_rows 1;

solve;

display Profit > task1e.sol;

display x > task1e.sol;

display sum {i in 2..n} income[i]\*y[i] > task1e.sol;

display sum {(i,j) in ARCS} cost[i,j]\*x[i,j] > task1e.sol;

File task1e.sol:

Profit = 19

x :=

1 2 1

1 6 1

2 4 1

4 8 1

6 8 1

;

sum{i in 2 .. n} income[i]\*y[i] = 150

sum{(i,j) in ARCS} cost[i,j]\*x[i,j] = 131

**3) Solution**

Optimal route should be 1-2-4-8-6-1.

Profit in this case will be 150 - 131 = 19.

ii) Find the optimal tour that maximises the collected income within the travel budget Cmax = 120, and specify the budget used.

**2) AMPL code**

File task1e.dat:

param c\_max = 120;

File task1e.mod:

param c\_max;

subject to Travel\_Budget:

sum {(i,j) in ARCS} cost[i,j]\*x[i,j] <= c\_max;

File task1e.sol:

Income = 130

x :=

1 2 1

1 6 1

2 8 1

6 8 1

;

sum{i in 2 .. n} income[i]\*y[i] = 130

sum{(i,j) in ARCS} cost[i,j]\*x[i,j] = 115

**3) Solution**

Optimal route should be 1-2-8-6-1.

The budget used is 115, travel income is 130.

iii) Find the minimum cost tour that satisfies the requirement on the minimal collected income Pmin = 200, and specify the income collected.

**2) AMPL code**

File task1e.dat:

param p\_min = 200;

File task1e.mod:

param p\_min;

subject to Minimal\_Income:

sum {i in 2..n} income[i]\*y[i] >= p\_min;

File task1e.sol:

Total\_Cost = 204

x :=

1 2 1

1 7 1

2 4 1

3 6 1

3 7 1

4 8 1

6 8 1

;

sum{i in 2 .. n} income[i]\*y[i] = 210

**3) Solution**

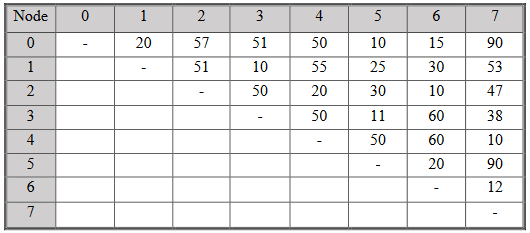
Optimal route should be 1-2-4-8-6-3-7-1.

Total income will then be 210.

Total cost in this case will be 204.

# Problem 2. Vehicle Routing Problem

Below you will find a symmetric graph together with the corresponding cost matrix:



Node 0 is a depot node. Each customer has a delivery demand that should be delivered from the depot. Demands are given in the following table:



We have a uniform vehicle fleet with a vehicle capacity of 80 units.

a) Find an optimal solution to the delivery problem above using the two-index asymmetric VRP model for homogeneous fleet with the load continuity (MTZ type) constraints.

**2) AMPL code**

File task2a.dat:

param N:=7;

param K:=4;

param demand:= 0 0 1 46 2 55 3 33 4 30 5 24 6 75 7 30;

param capacity:= 80;

param cost: 0 1 2 3 4 5 6 7 :=

0 . 20 57 51 50 10 15 90

1 20 . 51 10 55 25 30 53

2 57 51 . 50 20 30 10 47

3 51 10 50 . 50 11 60 38

4 50 55 20 50 . 50 60 10

5 10 25 30 11 50 . 20 90

6 15 30 10 60 60 20 . 12

7 90 53 47 38 10 90 12 .;

File task2a.mod:

param N >= 0; # number of nodes

param K >= 0; # number of vehicles

set A := {i in 0..N, j in 0..N: i<>j}; # set of arcs

param cost {A} >= 0; # travel costs

param demand {1..N} >= 0; # customer demands

param capacity >= 0; # vehicle capacity

var Route {A} binary; # 1 if arc (i,j) is used

var Load {i in 0..N} >=0, <= capacity; # vehicle’s load

minimize Total\_Cost:

sum {(i,j) in A} cost[i,j] \* Route[i,j];

subject to Vehicles:

sum {(0,j) in A} Route[0,j] = K;

subject to Out\_Of {i in 1..N}:

sum {(i,j) in A} Route[i,j] = 1;

subject to Route\_Continuity {i in 0..N}:

sum {(i,j) in A} Route[i,j] = sum {(j,i) in A} Route[j,i];

subject to Load\_Continuity {i in 0..N, j in 1..N: i<>j}:

Load[j] <= Load[i] - demand[j] + capacity \* (1 - Route[i,j]);

File task2a.run:

option solver cplex;

model task2a.mod;

data task2a.dat;

option show\_stats 1;

solve;

display Total\_Cost > task2a.sol;

display Route > task2a.sol;

display Load > task2a.sol;

File task2a.sol:

Total\_Cost = 358

Route [\*,\*]

: 0 1 2 3 4 5 6 7 :=

0 . 1 1 0 1 0 1 0

1 0 . 0 1 0 0 0 0

2 0 0 . 0 0 1 0 0

3 1 0 0 . 0 0 0 0

4 0 0 0 0 . 0 0 1

5 1 0 0 0 0 . 0 0

6 1 0 0 0 0 0 . 0

7 1 0 0 0 0 0 0 .

;

Load [\*] :=

0 80

1 34

2 25

3 0

4 50

5 0

6 5

7 0

;

**3) Solution**

The optimal routes for the vehicles are:

* Vehicle 1: 0-1-3-0;
* Vehicle 2: 0-2-5-0;
* Vehicle 3: 0-4-7-0;
* Vehicle 4: 0-6-0.

Total cost in this case will be 358.

b) Find a heuristic solution to the problem Route Generation heuristic generating a set of some feasible routes at the first stage and solving a set-partitioning model at the second stage;

Some of the feasible routes are:

0-1-0, load = 46, cost = 40

0-2-0, load = 55, cost = 114

0-3-0, load = 33, cost = 102

0-4-0, load = 30, cost = 100

0-5-0, load = 24, cost = 20

0-6-0, load = 75, cost = 30

0-7-0, load = 30, cost = 180

0-1-3-0, load = 79, cost = 81

0-1-4-0, load = 76, cost = 125

0-1-5-0, load = 70, cost = 55

0-1-7-0, load = 76, cost = 163

0-2-5-0, load = 79, cost = 97

0-3-4-0, load = 63, cost = 151

0-3-5-0, load = 57, cost = 72

0-3-7-0, load = 63, cost = 179

0-4-5-0, load = 64, cost = 110

0-4-7-0, load = 60, cost = 150

0-5-7-0, load = 54, cost = 190

**2) AMPL code**

File task2b.dat:

param M:= 7;

param K:= 18;

param routes: 1 2 3 4 5 6 7 :=

1 1 0 0 0 0 0 0

2 0 1 0 0 0 0 0

3 0 0 1 0 0 0 0

4 0 0 0 1 0 0 0

5 0 0 0 0 1 0 0

6 0 0 0 0 0 1 0

7 0 0 0 0 0 0 1

8 1 0 1 0 0 0 0

9 1 0 0 1 0 0 0

10 1 0 0 0 1 0 0

11 1 0 0 0 0 0 1

12 0 1 0 0 1 0 0

13 0 0 1 1 0 0 0

14 0 0 1 0 1 0 0

15 0 0 1 0 0 0 1

16 0 0 0 1 1 0 0

17 0 0 0 1 0 0 1

18 0 0 0 0 1 0 1;

param cost:= 1 40 2 114 3 102 4 100 5 20 6 30 7 180 8 81 9 125 10 55 11 163 12 97 13 151 14 72 15 179 16 110 17 150 18 190;

file task2b.mod:

param M;

param K;

set R:= 1..K;

set N:= 1..M;

param routes {R,N};

param cost {R};

var Route {R} binary;

minimize Total\_Cost:

sum {i in R} cost[i] \* Route[i];

subject to Route\_Continuity {i in N} :

sum {j in R} Route[j] \* routes[j,i]=1;

file task2b.run:

option solver cplex;

model task2b.mod;

data task2b.dat;

solve;

display Total\_Cost > task2b.sol;

display Route > task2b.sol;

display routes > task2b.sol;

file task2b.sol:

Total\_Cost = 358

Route [\*] :=

1 0

2 0

3 0

4 0

5 0

6 1

7 0

8 1

9 0

10 0

11 0

12 1

13 0

14 0

15 0

16 0

17 1

18 0

;

routes [\*,\*]

: 1 2 3 4 5 6 7 :=

1 1 0 0 0 0 0 0

2 0 1 0 0 0 0 0

3 0 0 1 0 0 0 0

4 0 0 0 1 0 0 0

5 0 0 0 0 1 0 0

6 0 0 0 0 0 1 0

7 0 0 0 0 0 0 1

8 1 0 1 0 0 0 0

9 1 0 0 1 0 0 0

10 1 0 0 0 1 0 0

11 1 0 0 0 0 0 1

12 0 1 0 0 1 0 0

13 0 0 1 1 0 0 0

14 0 0 1 0 1 0 0

15 0 0 1 0 0 0 1

16 0 0 0 1 1 0 0

17 0 0 0 1 0 0 1

18 0 0 0 0 1 0 1

;

**3) Solution**

The optimal routes for the vehicles are the same as in point a):

* Vehicle 1: 0-1-3-0;
* Vehicle 2: 0-2-5-0;
* Vehicle 3: 0-4-7-0;
* Vehicle 4: 0-6-0.

Total cost is 358.

c) Find a heuristic solution using Fisher&Jaikumar model with your own choice of seed nodes for clustering at the first stage and generating routes at the second stage.

**2) AMPL code**

File task2c.dat:

param N:=7;

param K:=4;

param demand:= 1 46 2 55 3 33 4 30 5 24 6 75 7 30;

param capacity:= 1 80 2 80 3 80 4 80;

param added\_cost: 1 2 3 4 5 6 7 :=

1 35 52 96 95 15 0 87

2 0 88 41 85 15 25 123

3 35 77 52 90 0 25 170

4 25 27 51 0 10 25 50;

File task2c.mod:

param N;

param K;

set nodes:= 1..N;

set seed\_nodes:= 1..K;

param added\_cost {seed\_nodes, nodes};

param demand {nodes};

param capacity {seed\_nodes};

var Route {seed\_nodes, nodes} binary;

minimize Total\_Cost:

sum {i in seed\_nodes, j in nodes} added\_cost[i,j] \* Route[i,j];

subject to Out\_Of {j in nodes}:

sum {i in seed\_nodes} Route[i,j] = 1;

subject to Route\_Continuity {k in seed\_nodes}:

sum {j in nodes} Route[k,j] \* demand[j] <= capacity[k];

File task2c.run:

option solver cplex;

model task2c.mod;

data task2c.dat;

solve;

display Total\_Cost > task2c.sol;

display Route > task2c.sol;

File task2c.sol:

Total\_Cost = 168

Route [\*,\*] (tr)

: 1 2 3 4 :=

1 0 1 0 0

2 0 0 1 0

3 0 1 0 0

4 0 0 0 1

5 0 0 1 0

6 1 0 0 0

7 0 0 0 1

;

**3) Solution**

The optimal routes for the vehicles are:

* Vehicle 1: 0-6-0;
* Vehicle 2: 0-1-3-0;
* Vehicle 3: 0-2-5-0;
* Vehicle 4: 0-4-7-0.

Total cost in this case will be 168, but it doesn’t account for the depot travel costs (getting both in and out). Adding these costs increases the total cost up to 358.

d) Assume that customers and depot have Time Windows for service (in minutes)



showing the earliest and the latest start of service. Travel times at arcs are calculated as 2.5\*cost, and service time at customers is 50 minutes. Find an optimal solution to the VRPTW and show the optimal routes together with times of arrival, waiting times, start times for service and departure times at customers.

**2) AMPL code**

File task2d.dat:

param n:= 7;

param demand:= 0 0 1 46 2 55 3 33 4 30 5 24 6 75 7 30 8 0;

param capacity:= 80;

param service\_time:= 0 0 1 50 2 50 3 50 4 50 5 50 6 50 7 50 8 0;

param M:= 10000;

param earliest:= 0 420 1 840 2 930 3 810 4 720 5 900 6 540 7 720 8 420;

param latest:= 0 1140 1 900 2 990 3 870 4 810 5 950 6 630 7 840 8 1140;

param cost: 0 1 2 3 4 5 6 7 8:=

0 . 20 57 51 50 10 15 90 0

1 20 . 51 10 55 25 30 53 20

2 57 51 . 50 20 30 10 47 57

3 51 10 50 . 50 11 60 38 51

4 50 55 20 50 . 50 60 10 50

5 10 25 30 11 50 . 20 90 10

6 15 30 10 60 60 20 . 12 15

7 90 53 47 38 10 90 12 . 90

8 0 20 57 51 50 10 15 90 .;

param time: 0 1 2 3 4 5 6 7 8 :=

0 . 50 142 127 125 25 37 225 0

1 50 . 127 25 137 62 75 132 50

2 142 127 . 125 50 75 25 117 142

3 127 25 125 . 125 27 150 95 127

4 125 137 50 125 . 125 150 25 125

5 25 62 75 27 125 . 50 225 25

6 37 75 25 150 150 50 . 30 37

7 225 132 117 95 25 225 30 . 225

8 0 50 142 127 125 25 37 225 .;

File task2d.mod:

param n;

set nodes:= 0..n+1;

set vehicles:= 1..n;

set A:= {i in nodes,j in nodes: i<>j};

param demand {nodes};

param capacity;

param service\_time {nodes};

param earliest {nodes};

param latest {nodes};

param M;

param cost {A};

param time {A};

var start\_time {nodes, vehicles};

var Route {nodes, nodes, vehicles} binary;

minimize Total\_Cost:

sum {k in vehicles, (i,j) in A} cost[i,j] \* Route[i,j,k];

# (2)

subject to Zero\_Link {k in vehicles}:

sum {(0,j) in A} Route[0,j,k] <= 1;

# (3)

subject to Single\_Route {i in 1..n}:

sum {k in vehicles, (i,j) in A} Route[i,j,k] = 1;

# (4)

subject to Load\_Balance {k in vehicles, i in 1..n}:

sum {(i,j) in A} Route[i,j,k] = sum {(j,i) in A} Route[j,i,k];

# (5)

subject to Single\_Link {k in vehicles}:

sum {(i,n+1) in A} Route[i,n+1,k] = 1;

# (6)

subject to Time\_Balance {k in vehicles, (i,j) in A}:

start\_time[i,k] + service\_time[i] + time[i,j] - start\_time[j,k] <= (1 - Route[i,j,k]) \* M;

# (7)

subject to Time\_Window {k in vehicles, i in nodes}:

earliest[i] <= start\_time[i,k] <= latest[i];

# (8)

subject to Capacity {k in vehicles}:

sum {(i,j) in A} demand[i] \* Route[i,j,k] <= capacity;

File task2d.run:

option solver cplex;

model task2d.mod;

data task2d.dat;

solve;

display Total\_Cost > task2d.sol;

display Route > task2d.sol;

display start\_time > task2d.sol;

display {i in nodes, k in vehicles} start\_time[i, k] \* sum{(i,j) in A} Route[i,j,k] > task2d.sol;

display {i in nodes, k in vehicles} start\_time[i, k] \* sum{(j,i) in A} Route[j,i,k] > task2d.sol;

File task2d.sol:

Total\_Cost = 395

Route [\*,\*,1]

: 0 1 2 3 4 5 6 7 8 :=

0 0 0 0 0 0 1 0 0 0

1 0 0 0 0 0 0 0 0 0

2 0 0 0 0 0 0 0 0 0

3 0 0 0 0 0 0 0 0 0

4 0 0 0 0 0 0 0 0 0

5 0 0 0 0 0 0 0 0 1

6 0 0 0 0 0 0 0 0 0

7 0 0 0 0 0 0 0 0 0

8 0 0 0 0 0 0 0 0 0

[\*,\*,2]

: 0 1 2 3 4 5 6 7 8 :=

0 0 0 0 0 0 0 0 0 1

1 0 0 0 0 0 0 0 0 0

2 0 0 0 0 0 0 0 0 0

3 0 0 0 0 0 0 0 0 0

4 0 0 0 0 0 0 0 0 0

5 0 0 0 0 0 0 0 0 0

6 0 0 0 0 0 0 0 0 0

7 0 0 0 0 0 0 0 0 0

8 0 0 0 0 0 0 0 0 0

[\*,\*,3]

: 0 1 2 3 4 5 6 7 8 :=

0 0 0 1 0 0 0 0 0 0

1 0 0 0 0 0 0 0 0 0

2 0 0 0 0 0 0 0 0 1

3 0 0 0 0 0 0 0 0 0

4 0 0 0 0 0 0 0 0 0

5 0 0 0 0 0 0 0 0 0

6 0 0 0 0 0 0 0 0 0

7 0 0 0 0 0 0 0 0 0

8 0 0 0 0 0 0 0 0 0

[\*,\*,4]

: 0 1 2 3 4 5 6 7 8 :=

0 0 0 0 1 0 0 0 0 0

1 0 0 0 0 0 0 0 0 1

2 0 0 0 0 0 0 0 0 0

3 0 1 0 0 0 0 0 0 0

4 0 0 0 0 0 0 0 0 0

5 0 0 0 0 0 0 0 0 0

6 0 0 0 0 0 0 0 0 0

7 0 0 0 0 0 0 0 0 0

8 0 0 0 0 0 0 0 0 0

[\*,\*,5]

: 0 1 2 3 4 5 6 7 8 :=

0 0 0 0 0 0 0 1 0 0

1 0 0 0 0 0 0 0 0 0

2 0 0 0 0 0 0 0 0 0

3 0 0 0 0 0 0 0 0 0

4 0 0 0 0 0 0 0 0 0

5 0 0 0 0 0 0 0 0 0

6 0 0 0 0 0 0 0 0 1

7 0 0 0 0 0 0 0 0 0

8 0 0 0 0 0 0 0 0 0

[\*,\*,6]

: 0 1 2 3 4 5 6 7 8 :=

0 0 0 0 0 1 0 0 0 0

1 0 0 0 0 0 0 0 0 0

2 0 0 0 0 0 0 0 0 0

3 0 0 0 0 0 0 0 0 0

4 0 0 0 0 0 0 0 1 0

5 0 0 0 0 0 0 0 0 0

6 0 0 0 0 0 0 0 0 0

7 0 0 0 0 0 0 0 0 1

8 0 0 0 0 0 0 0 0 0

[\*,\*,7]

: 0 1 2 3 4 5 6 7 8 :=

0 0 0 0 0 0 0 0 0 1

1 0 0 0 0 0 0 0 0 0

2 0 0 0 0 0 0 0 0 0

3 0 0 0 0 0 0 0 0 0

4 0 0 0 0 0 0 0 0 0

5 0 0 0 0 0 0 0 0 0

6 0 0 0 0 0 0 0 0 0

7 0 0 0 0 0 0 0 0 0

8 0 0 0 0 0 0 0 0 0

;

start\_time [\*,\*]

: 1 2 3 4 5 6 7 :=

0 420 420 420 420 420 420 420

1 840 840 840 885 840 840 840

2 930 930 948 930 930 930 930

3 810 810 810 810 810 810 810

4 720 720 720 720 720 765 720

5 900 900 900 900 900 900 900

6 540 540 540 540 540 540 540

7 720 720 720 720 720 840 720

8 1140 1140 1140 1140 1140 1140 1140

;

start\_time[i,k]\*(sum{(i,j) in A} Route[i,j,k]) [\*,\*]

: 1 2 3 4 5 6 7 :=

0 420 420 420 420 420 420 420

1 0 0 0 885 0 0 0

2 0 0 948 0 0 0 0

3 0 0 0 810 0 0 0

4 0 0 0 0 0 765 0

5 900 0 0 0 0 0 0

6 0 0 0 0 540 0 0

7 0 0 0 0 0 840 0

8 0 0 0 0 0 0 0

;

start\_time[i,k]\*(sum{(j,i) in A} Route[j,i,k]) [\*,\*]

: 1 2 3 4 5 6 7 :=

0 0 0 0 0 0 0 0

1 0 0 0 885 0 0 0

2 0 0 948 0 0 0 0

3 0 0 0 810 0 0 0

4 0 0 0 0 0 765 0

5 900 0 0 0 0 0 0

6 0 0 0 0 540 0 0

7 0 0 0 0 0 840 0

8 1140 1140 1140 1140 1140 1140 1140

;

**3) Solution**

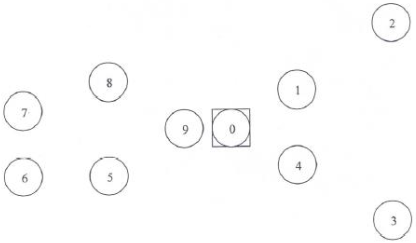
Adding time windows increases the amount of vehicles we need from 4 to 5:

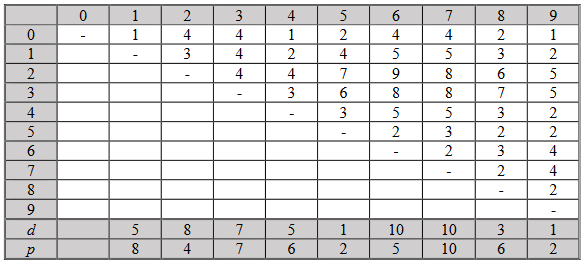
* Vehicle 1: 0-5-0;
* Vehicle 2: 0-2-0;
* Vehicle 3: 0-1-3-0;
* Vehicle 4: 0-6-0;
* Vehicle 5: 0-4-7-0.

Total cost in this case will be 395.

# Problem 3. Vehicle Routing Problem with Pickups and Deliveries

Oskar Sylte beverage company located in Molde delivers soft drinks to 9 retail outlets located outside the town. Figure below depicts the geographical positions of the outlets, where vertex 0 represents the depot, while the shortest distances and respective delivery and pickup demands (in number of boxes) are reported in the table. For each outlet delivery demand should be delivered from the depot and the pickup demand should be collected and brought back to the depot using vehicles with capacity of 25 boxes.





a) Find with a model an optimal solution where each outlet is visited exactly once.

**2) AMPL code**

File task3a.dat:

param N:= 9;

param K:= 3;

param capacity:= 1 25 2 25 3 25;

param to\_deliver:= 0 0 1 5 2 8 3 7 4 5 5 1 6 10 7 10 8 3 9 1;

param to\_pickup:= 0 0 1 8 2 4 3 7 4 6 5 2 6 5 7 10 8 6 9 2;

param cost: 0 1 2 3 4 5 6 7 8 9 :=

0 . 1 4 4 1 2 4 4 2 1

1 1 . 3 4 2 4 5 5 3 2

2 4 3 . 4 4 7 9 8 6 5

3 4 4 4 . 3 6 8 8 7 5

4 1 2 4 3 . 3 5 5 3 2

5 2 4 7 6 3 . 2 3 2 2

6 4 5 9 8 5 2 . 2 3 4

7 4 5 8 8 5 3 2 . 2 4

8 2 3 6 7 3 2 3 2 . 2

9 1 2 5 5 2 2 4 4 2 .;

File task3a.mod:

param N;

param K;

set EDGES:= {i in 0..N, j in 0..N: i<>j};

set NODES:= {1..N};

set VEHICLES:= {m in 1..K};

set ALL\_NODES:= {i in 0..N};

param cost {EDGES};

param capacity {VEHICLES};

param to\_deliver {ALL\_NODES};

param to\_pickup {ALL\_NODES};

var x {EDGES, VEHICLES} binary;

var delivery {ALL\_NODES, VEHICLES};

var pickup {ALL\_NODES, VEHICLES};

minimize Total\_Cost:

sum{(i,j) in EDGES} sum {k in VEHICLES} cost[i,j] \* x[i,j,k];

subject to Left\_Depot{k in VEHICLES}:

sum{(0,j) in EDGES} x[0,j,k] <= 1;

subject to Visit\_Once {i in NODES}:

sum{(i,j) in EDGES} sum {k in VEHICLES} x[i,j,k]=1;

subject to Route\_Continuity {i in ALL\_NODES, k in VEHICLES}:

sum {(i,j) in EDGES} x[i,j,k] = sum {(j,i) in EDGES} x[j,i,k];

subject to Zero\_Delivery {k in VEHICLES}:

delivery[0,k] = sum {(i,j) in EDGES} to\_deliver[i] \* x[i,j,k];

subject to Flow {k in VEHICLES, i in ALL\_NODES, j in NODES: i<>j}:

delivery[j,k] >= delivery[i,k] - to\_deliver[j] - (1 - x[i,j,k]) \* capacity[k];

subject to Zero\_Pickup {k in VEHICLES}:

pickup[0,k] = 0;

subject to Pickup\_Operation {k in VEHICLES, i in ALL\_NODES, j in NODES: i<>j}:

pickup[j,k] >= pickup[i,k] + to\_pickup[j] - (1 - x[i,j,k]) \* capacity[k];

subject to Capacity {i in NODES, k in VEHICLES}:

delivery[i,k] + pickup[i,k] <= capacity[k];

File task3a.run:

option solver cplex;

model task3a.mod;

data task3a.dat;

option show\_stats 1;

solve;

display Total\_Cost > task3a.sol;

display x > task3a.sol;

display delivery > task3a.sol;

display pickup > task3a.sol;

File task3a.sol:

Total\_Cost = 26

x [\*,\*,1]

: 0 1 2 3 4 5 6 7 8 9 :=

0 . 0 0 0 0 1 0 0 0 0

1 0 . 0 0 0 0 0 0 0 0

2 0 0 . 0 0 0 0 0 0 0

3 0 0 0 . 0 0 0 0 0 0

4 0 0 0 0 . 0 0 0 0 0

5 0 0 0 0 0 . 1 0 0 0

6 0 0 0 0 0 0 . 1 0 0

7 0 0 0 0 0 0 0 . 1 0

8 1 0 0 0 0 0 0 0 . 0

9 0 0 0 0 0 0 0 0 0 .

[\*,\*,2]

: 0 1 2 3 4 5 6 7 8 9 :=

0 . 1 0 0 0 0 0 0 0 0

1 1 . 0 0 0 0 0 0 0 0

2 0 0 . 0 0 0 0 0 0 0

3 0 0 0 . 0 0 0 0 0 0

4 0 0 0 0 . 0 0 0 0 0

5 0 0 0 0 0 . 0 0 0 0

6 0 0 0 0 0 0 . 0 0 0

7 0 0 0 0 0 0 0 . 0 0

8 0 0 0 0 0 0 0 0 . 0

9 0 0 0 0 0 0 0 0 0 .

[\*,\*,3]

: 0 1 2 3 4 5 6 7 8 9 :=

0 . 0 0 0 1 0 0 0 0 0

1 0 . 0 0 0 0 0 0 0 0

2 0 0 . 0 0 0 0 0 0 1

3 0 0 1 . 0 0 0 0 0 0

4 0 0 0 1 . 0 0 0 0 0

5 0 0 0 0 0 . 0 0 0 0

6 0 0 0 0 0 0 . 0 0 0

7 0 0 0 0 0 0 0 . 0 0

8 0 0 0 0 0 0 0 0 . 0

9 1 0 0 0 0 0 0 0 0 .

;

delivery [\*,\*]

: 1 2 3 :=

0 24 5 21

1 19 0 -9

2 23 -28 1

3 20 -27 9

4 21 -25 16

5 23 -21 -5

6 15 -30 -14

7 5 -30 -14

8 2 -23 -7

9 -2 -21 0

;

pickup [\*,\*]

: 1 2 3 :=

0 0 0 0

1 6 8 2

2 2 -13 17

3 5 -10 13

4 4 -11 6

5 2 -15 -4

6 7 -12 -1

7 17 -7 4

8 23 -11 0

9 0 -15 19

;

**3) Solution**

We need to use 3 vehicles. The routes for them are:

* Vehicle 1: 0-5-6-7-8-0, load = 24;
* Vehicle 2: 0-1-0, load = 5;
* Vehicle 3: 0-4-3-2-9-0, load = 21.

Total cost in this case will be 26.

b) Find with a model an optimal route serving outlets 5, 6, 7, 8, 9 where customers can be visited twice.

**2) AMPL code**

File task3b.mod:

subject to Visit\_Once {i in NODES: i<=4}:

sum{(i,j) in EDGES} sum {k in VEHICLES} x[i,j,k]=0;

subject to Visit\_Once\_Sub1 {i in NODES: i>=5 and i<=9}:

sum{(i,j) in EDGES} sum {k in VEHICLES} x[i,j,k]>=1;

subject to Visit\_Once\_Sub2 {i in NODES: i>=5 and i<=9}:

sum{(i,j) in EDGES} sum {k in VEHICLES} x[i,j,k]<=2;

file task3b.sol:

Total\_Cost = 12

x [\*,\*,1]

: 0 1 2 3 4 5 6 7 8 9 :=

0 . 0 0 0 0 0 0 0 0 1

1 0 . 0 0 0 0 0 0 0 0

2 0 0 . 0 0 0 0 0 0 0

3 0 0 0 . 0 0 0 0 0 0

4 0 0 0 0 . 0 0 0 0 0

5 0 0 0 0 0 . 0 0 0 0

6 0 0 0 0 0 0 . 0 0 0

7 0 0 0 0 0 0 0 . 0 0

8 0 0 0 0 0 0 0 0 . 0

9 1 0 0 0 0 0 0 0 0 .

[\*,\*,2]

: 0 1 2 3 4 5 6 7 8 9 :=

0 . 0 0 0 0 0 0 0 0 0

1 0 . 0 0 0 0 0 0 0 0

2 0 0 . 0 0 0 0 0 0 0

3 0 0 0 . 0 0 0 0 0 0

4 0 0 0 0 . 0 0 0 0 0

5 0 0 0 0 0 . 0 0 0 0

6 0 0 0 0 0 0 . 0 0 0

7 0 0 0 0 0 0 0 . 0 0

8 0 0 0 0 0 0 0 0 . 0

9 0 0 0 0 0 0 0 0 0 .

[\*,\*,3]

: 0 1 2 3 4 5 6 7 8 9 :=

0 . 0 0 0 0 1 0 0 0 0

1 0 . 0 0 0 0 0 0 0 0

2 0 0 . 0 0 0 0 0 0 0

3 0 0 0 . 0 0 0 0 0 0

4 0 0 0 0 . 0 0 0 0 0

5 0 0 0 0 0 . 1 0 0 0

6 0 0 0 0 0 0 . 1 0 0

7 0 0 0 0 0 0 0 . 1 0

8 1 0 0 0 0 0 0 0 . 0

9 0 0 0 0 0 0 0 0 0 .

;

delivery [\*,\*]

: 1 2 3 :=

0 1 0 24

1 -29 -30 19

2 -32 -33 23

3 -31 -32 20

4 -29 -30 21

5 -25 -26 23

6 -34 -35 15

7 -34 -35 5

8 -27 -28 2

9 0 -26 -2

;

pickup [\*,\*]

: 1 2 3 :=

0 0 0 0

1 -15 -17 6

2 -19 -21 2

3 -16 -18 5

4 -17 -19 4

5 -21 -23 2

6 -18 -20 7

7 -13 -15 17

8 -17 -19 23

9 2 -23 0

;

**3) Solution**

Now we only need 2 vehicles with the following routes:

* Vehicle 1: 0-9-0, load = 1;
* Vehicle 2: 0-5-6-7-8-0, load = 24.

Total cost in this case will be 12.

# Problem 4. Periodic Vehicle Routing Problem

Consider three customers M, N and K supplied several times during the planning horizon of 3 days with the vehicle capacity of 10 units from supplier P. Customers' total 3-days demands and visit frequencies are given in the table below.

Table

Description automatically generated

A road network with travel costs shown on edges is depicted in figure below.

Diagram

Description automatically generated

Solve the Periodic Vehicle Routing Problem above using a set-covering model with pre-generation of routes for two cases:

a) When all possible visit schedules for customers are valid;

**1) Mathematical model**

|  |  |
| --- | --- |
| **Mathematical model** | **AMPL names:** |
| **Formulation:** |  |
| 1) | Total\_Cost |
| St |  |
| 2) | One\_Schedule {i in Customers} |
| 3) | Day\_Schedule {i in Customers, t in 1..T} |
| 4) | x {r in Routes, t in 1..T} |
| 5) | y {i in Customers, s in 1..S} |
| **Notation:** |  |
| **Sets:** |  |
| – set of all feasible daily routes | Routes |
| – subset of routes in set that contain customer | r |
| **Parameters:** |  |
| – binary «coverage» parameter equal to 1 if customer is visited by route | a {i in Customers, r in Routes} |
| – cost of route | cost {r in Routes} |
| **Variables:** |  |
| – binary variable equal to 1 if route is performed on day | x {r in Routes, t in 1..T} |
| – binary variable equal to 1 if customer is visited on schedule | y {i in Customers, s in 1..S} |
| **Description** |  |
| Objective function (1) minimizes the total cost of all routes during the planning horizon. Constraints (2) ensure that exactly one feasible schedule is chosen for each customer. Constraints (3) guarantee that each customer is visited only on those days that are specified by the selected visit schedule. Constraints (4) and (5) ensure that variables and are binary. | |
| **Problem size** |  |
| The resulting model has following dimensions: | |
| 2 variables | |
| 4 constraints | |

**2) AMPL code**

File task4.dat:

param T := 3;

set Customers := M N K;

param S := 4;

#set Schedules := A B C D;

param b: 1 2 3 4:=

1 1 0 1 1

2 1 1 0 1

3 0 1 1 1;

set Schedule[M] := 4;

set Schedule[N] := 1 2 3;

set Schedule[K] := 1 2 3;

set Routes := a b c d e f g;

param a: a b c d e f g:=

M 1 0 0 1 0 1 1

N 0 1 0 1 1 0 1

K 0 0 1 0 1 1 1;

param cost := a 40, b 80, c 40, d 90, e 85, f 85, g 95;

File task4.mod:

param T >=0;

set Customers;

param S >=0;

#set Schedules {s in 1..S};

param b {t in 1..T, s in 1..S} binary;

set Routes;

param cost {r in Routes};

param a {i in Customers, r in Routes} binary;

set Schedule {i in Customers} within {1..S};

var x {r in Routes, t in 1..T} binary;

var y {i in Customers, s in 1..S} binary;

minimize Total\_Cost:

sum {t in 1..T, r in Routes} cost[r]\*x[r,t];

subject to One\_Schedule {i in Customers}:

sum{s in Schedule[i]} y[i,s] = 1;

subject to Day\_Schedule {i in Customers, t in 1..T}:

sum {r in Routes} a[i,r]\*x[r,t] = sum {s in Schedule[i]} b[t,s]\*y[i,s];

File task4.run

option solver cplex;

#option cplex\_options 'sensitivity';

model task4.mod;

data task4.dat;

solve;

#option omit\_zero\_rows 1;

#option show\_stats 1;

display Total\_Cost > task4.sol;

display x > task4.sol;

display y > task4.sol;

File task4.sol

Total\_Cost = 230

x [\*,\*]

: 1 2 3 :=

a 0 1 0

b 0 0 0

c 0 0 0

d 0 0 0

e 0 0 0

f 0 0 0

g 1 0 1

;

y :=

K 1 0

K 2 0

K 3 1

K 4 0

M 1 0

M 2 0

M 3 0

M 4 1

N 1 0

N 2 0

N 3 1

N 4 0

;

**3) Solution**

To minimize the total cost, the following schedule should be implemented: customer M is visited every day, customers N and K – on the first and the third day. The chosen routes are:

* On day 1 – P-M-N-K-P;
* On day 2 – P-M-P;
* On day 3 – P-M-N-K-P.

Total cost in this case will be 230.

b) When customers N and K cannot be visited in two consecutive days.

**2) AMPL code**

File task4.dat:

set Schedule[M] := 4;

set Schedule[N] := 3;

set Schedule[K] := 3;

File task2a.sol

Total\_Cost = 230

x [\*,\*]

: 1 2 3 :=

a 0 1 0

b 0 0 0

c 0 0 0

d 0 0 0

e 0 0 0

f 0 0 0

g 1 0 1

;

y :=

K 1 0

K 2 0

K 3 1

K 4 0

M 1 0

M 2 0

M 3 0

M 4 1

N 1 0

N 2 0

N 3 1

N 4 0

;

**3) Solution**

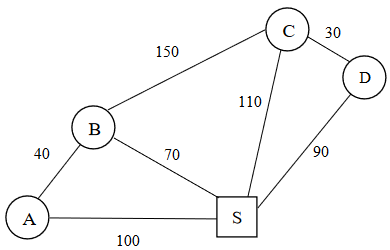
The solution from point a) already included visiting customers N and K every other day, so nothing changes here. To minimize the total cost, the following schedule should be implemented: customer M is visited every day, customers N and K – on the first and the third day. The chosen routes are:

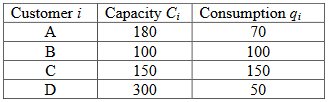
* On day 1 – P-M-N-K-P;
* On day 2 – P-M-P;
* On day 3 – P-M-N-K-P.

Total cost is still 230.

# Problem 5. Inventory Routing Problem

Consider a network with a single supplier S and four customers A, B, C and D depicted on the graph below with travel costs shown on edges, and customers’ daily consumption rates and storage capacities in units given in the following table.





The planning horizon is 4 days and there are unlimited number of vehicles with capacity Q = 250 units. Assume that we consider the direct routes to a customer and the routes that serve adjacent customers, and it’s assumed that each route can be performed at most once per day. Assume for the beginning that there are no initial inventories at customers, deliveries take place before consumption, and inventories are measured at the end of the day.

Find a cheapest distribution policy (show routes for each day and how much will be delivered/stored at customers) for the Inventory Routing Problem for each of the following cases:

a) when only travel costs are minimized;

**2) AMPL code**

File task5a.dat:

param K:=4;

param T:=4;

param veh\_cap:= 250;

param consumption:= 1 70 2 100 3 150 4 50;

param inv\_cap:= 1 180 2 100 3 150 4 300;

set Routes := 1 2 3 4 12 23 34;

param cost:= 1 200 2 140 3 220 4 180 12 210 23 330 34 230;

set Route[1]:= 1 12;

set Route[2]:= 2 12 23;

set Route[3]:= 3 23 34;

set Route[4]:= 4 34;

File task5a.mod:

param K;

param T;

set Customers:= {i in 1..K};

set Routes;

param veh\_cap;

param consumption {Customers};

param inv\_cap {Customers};

param cost {Routes};

set Route {i in Customers} within Routes;

var x {k in Routes, t in 1..T} binary;

var amount {i in Customers, k in Routes, t in 1..T} >= 0;

var inv {i in Customers, t in 0..T};

minimize Total\_Cost:

1 / T \* (sum{t in 1..T} sum {k in Routes} cost[k] \* x[k,t]);

subject to Vehicle\_Capacity{t in 1..T, i in Customers, k in Route[i]}:

sum {p in Customers} amount[p,k,t] <= veh\_cap \* x[k,t];

subject to Inventory\_Level{i in Customers, t in 1..T}:

inv[i,t] = inv[i,t-1] + sum{k in Route[i]} amount[i,k,t] - consumption[i];

subject to Inventory\_Capacity {i in Customers, t in 1..T}:

inv[i,t] + consumption[i] <= inv\_cap[i];

subject to Initial\_Inventory {i in Customers}:

inv[i,0] = 0;

subject to No\_Stockouts {i in Customers, t in 0..T}:

inv[i,t] >= 0;

File task5a.run:

option solver cplex;

model task5.mod;

data task5.dat;

option show\_stats 1;

solve;

display Total\_Cost > task5.sol;

display x > task5.sol;

display amount > task5.sol;

display inv > task5.sol;

File task5a.sol:

Total\_Cost = 385

x [\*,\*]

: 1 2 3 4 :=

1 0 0 0 0

2 0 0 0 0

3 0 0 0 0

4 0 0 0 0

12 1 0 1 0

23 0 1 0 1

34 1 0 1 0

;

amount [1,\*,\*]

: 1 2 3 4 :=

1 0 0 0 0

2 0 0 0 0

3 0 0 0 0

4 0 0 0 0

12 140 0 140 0

23 0 0 0 0

34 0 0 0 0

[2,\*,\*]

: 1 2 3 4 :=

1 0 0 0 0

2 0 0 0 0

3 0 0 0 0

4 0 0 0 0

12 100 0 100 0

23 0 100 0 100

34 0 0 0 0

[3,\*,\*]

: 1 2 3 4 :=

1 0 0 0 0

2 0 0 0 0

3 0 0 0 0

4 0 0 0 0

12 0 0 0 0

23 0 150 0 150

34 150 0 150 0

[4,\*,\*]

: 1 2 3 4 :=

1 0 0 0 0

2 0 0 0 0

3 0 0 0 0

4 0 0 0 0

12 0 0 0 0

23 0 0 0 0

34 100 0 100 0

;

inv :=

1 0 0

1 1 70

1 2 0

1 3 70

1 4 0

2 0 0

2 1 0

2 2 0

2 3 0

2 4 0

3 0 0

3 1 0

3 2 0

3 3 0

3 4 0

4 0 0

4 1 50

4 2 0

4 3 50

4 4 0

;

**3) Solution**

Routes for each day are the following:

* Day 1: S-A-B-S, S-C-D-S;
* Day 2: S-B-C-S;
* Day 3: S-A-B-S, S-C-D-S;
* Day 4: S-B-C-S.

Average daily cost in this case will be 385.

b) when both travel costs and inventory holding costs at customers are minimized (unit holding costs are 1);

**2) AMPL code**

File task5b.dat:

param hold\_cost:= 1;

File task5b.mod:

param hold\_cost;

minimize Total\_Cost:

1 / T \* (sum{t in 1..T} sum {k in Routes} cost[k] \* x[k,t] + sum{t in 1..T} sum{i in Customers} inv[i,t] \* hold\_cost);

file task5b.sol:

Total\_Cost = 440

x [\*,\*]

: 1 2 3 4 :=

1 0 0 0 0

2 0 0 0 0

3 0 0 0 0

4 0 0 0 0

12 1 1 1 1

23 0 0 0 0

34 1 1 1 1

;

amount [1,\*,\*]

: 1 2 3 4 :=

1 0 0 0 0

2 0 0 0 0

3 0 0 0 0

4 0 0 0 0

12 70 70 70 70

23 0 0 0 0

34 0 0 0 0

[2,\*,\*]

: 1 2 3 4 :=

1 0 0 0 0

2 0 0 0 0

3 0 0 0 0

4 0 0 0 0

12 100 100 100 100

23 0 0 0 0

34 0 0 0 0

[3,\*,\*]

: 1 2 3 4 :=

1 0 0 0 0

2 0 0 0 0

3 0 0 0 0

4 0 0 0 0

12 0 0 0 0

23 0 0 0 0

34 150 150 150 150

[4,\*,\*]

: 1 2 3 4 :=

1 0 0 0 0

2 0 0 0 0

3 0 0 0 0

4 0 0 0 0

12 0 0 0 0

23 0 0 0 0

34 50 50 50 50

;

inv :=

1 0 0

1 1 0

1 2 0

1 3 0

1 4 0

2 0 0

2 1 0

2 2 0

2 3 0

2 4 0

3 0 0

3 1 0

3 2 0

3 3 0

3 4 0

4 0 0

4 1 0

4 2 0

4 3 0

4 4 0

;

**3) Solution**

In this case the routes for each day will be the same: S-A-B-S and S-C-D-S.

Average daily cost of this solution is 440.

c) when travel costs and inventory holding costs at customers are minimized and initial inventory levels at customers are found by optimization (unit holding costs are 0.5).

**2) AMPL code**

File task5c.dat:

param hold\_cost:= 0.5;

File task5c.mod:

var start\_inv {i in Customers} >=0;

minimize Total\_Cost:

1 / T \* (sum{t in 1..T} sum {k in Routes} cost[k] \* x[k,t]

+ sum{t in 1..T} sum{i in Customers} inv[i,t] \* hold\_cost

+ sum{i in Customers} start\_inv[i] \* hold\_cost);

File task5c.sol:

Total\_Cost = 415

start\_inv [\*] :=

1 0

2 0

3 0

4 0

;

x [\*,\*]

: 1 2 3 4 :=

1 0 0 0 0

2 0 0 0 0

3 0 0 0 0

4 0 0 0 0

12 1 0 1 0

23 0 1 0 1

34 1 0 1 0

;

amount [1,\*,\*]

: 1 2 3 4 :=

1 0 0 0 0

2 0 0 0 0

3 0 0 0 0

4 0 0 0 0

12 140 0 140 0

23 0 0 0 0

34 0 0 0 0

[2,\*,\*]

: 1 2 3 4 :=

1 0 0 0 0

2 0 0 0 0

3 0 0 0 0

4 0 0 0 0

12 100 0 100 0

23 0 100 0 100

34 0 0 0 0

[3,\*,\*]

: 1 2 3 4 :=

1 0 0 0 0

2 0 0 0 0

3 0 0 0 0

4 0 0 0 0

12 0 0 0 0

23 0 150 0 150

34 150 0 150 0

[4,\*,\*]

: 1 2 3 4 :=

1 0 0 0 0

2 0 0 0 0

3 0 0 0 0

4 0 0 0 0

12 0 0 0 0

23 0 0 0 0

34 100 0 100 0

;

inv :=

1 0 0

1 1 70

1 2 0

1 3 70

1 4 0

2 0 0

2 1 0

2 2 0

2 3 0

2 4 0

3 0 0

3 1 0

3 2 0

3 3 0

3 4 0

4 0 0

4 1 50

4 2 0

4 3 50

4 4 0

;

**3) Solution**

Routes for each day are the same as in a):

* Day 1: S-A-B-S, S-C-D-S;
* Day 2: S-B-C-S;
* Day 3: S-A-B-S, S-C-D-S;
* Day 4: S-B-C-S.

Optimal inventory level is 0.

Average daily cost of this solution is 415.

Compare costs of solutions for cases a)-c), and show the impact of adding holding costs at customers in b) compared to a), and the impact of adding optimal initial inventory levels in c) compared to b).