Problem 1. Vehicle Routing Problem

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

Node	0	1	2	3	4	5	6	7
0	-	20	57	51	50	10	15	90
1		-	51	10	55	25	30	53
2			-	50	20	30	10	47
3				-	50	11	60	38
4					-	50	60	10
5						-	20	90
6							-	12
7								-

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

Customer	1	2	3	4	5	6	7
d_i	46	55	33	30	24	75	30

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)

Node	0	1	2	3	4	5	6	7
TW	[420,1140]	[840,900]	[930,990]	[810,870]	[720,810]	[900,950]	[540,630]	[720,840]

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

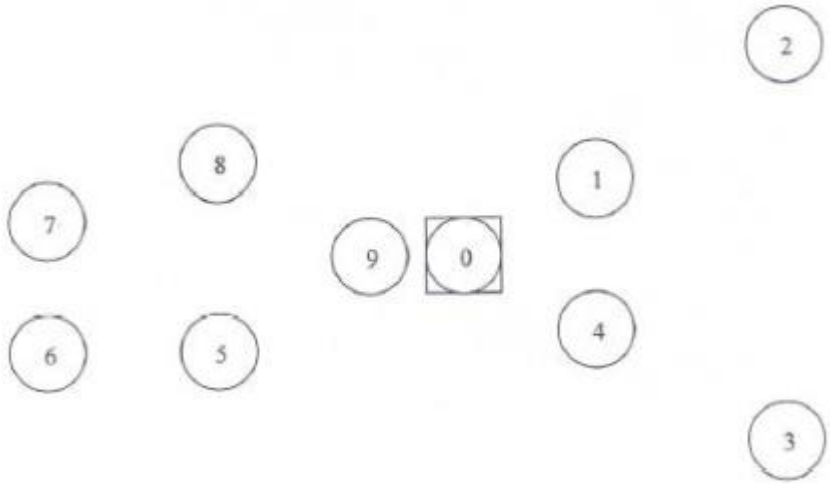
Adding time windows increases the amount of vehicles we need from 4 to 5:

- 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 2. 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.



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

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.