Car Rental

Domain: Ground Transportation

Niko Kaso
niko.kaso@optilogic.com

Problem Description

A car rental company has a fleet of 94 vehicles distributed among its 10 agencies. The distance between every two agencies is given. The required number of cars needed at each agency and the number of present cars each day are given. Also, a fixed cost per kilometer for transporting a car is given. The problem is to determine the movements of cars so as to satisfy the requirements of each agency and minimizing the total cost of transportation.

All the data are given in the data file Car-rental.dat.

Question 1

How to distribute cars to agencies so as the demands per agency are satisfied and the transportation cost is minimized?

Keywords: Integer program, ground transportation, abstract model, car-rental, transportation cost, routing.

1 Mathematical formulation

In order to build the mathematical formulation of the above Car Rental problem [1], one has to first define the decision variables and the sets under which the constructed variables will be indexed on. Let I be the set of all agencies, J_1 be the set of all agencies with excess in the number of cars and J_2 be the set of agencies with need in the number of cars.

We are interested to know the flow from agencies with excess to agencies in need. Therefore, integer variables $x_{j_1 j_2},\ \forall\ j_1 \in J_1,\ j_2 \in J_2$ are constructed, (3).

The total number of cars in the company is denoted by the parameter n. Moreover, the number of cars present at each agency $i \in I$ is p_i, whereas the number of cars required next day at each agency is given by r_i. The distance between any agency from the set J_1 to any agency from the set J_2 is represented by $d_{j_1 j_2}$. Lastly, the cost per km of transporting a car is denoted by c.

- Number of cars going from each excess agency to all agencies in need is equal to the difference between number of cars present with the number of cars required. Thus for each excess agency $j_1 \in J_1$, constraints (1) formulate the above condition.

- Number of cars going from each agency in need to all excess agencies is equal to the difference between number of cars required with the number of cars present. Thus for each agency in need $j_2 \in J_2$, constraints (2) formulate the above condition.
- The objective is to minimize the total transportation cost. Therefore, the objective function is the sum of products of the transportation cost per km, c, the distance between agencies with excess to agencies in need, $d_{j_1 j_2}$ and the integer variables $x_{j_1 j_2}$.

1.1 Mixed Integer Linear Programming Formulation

$$\begin{aligned}
\text{minimize} \quad & \sum_{j_1 \in J_1} \sum_{j_2 \in J_2} c \cdot d_{j_1 j_2} \cdot x_{j_1 j_2} \\
\text{subject to} \quad & \sum_{j_2 \in J_2} x_{j_1 j_2} = p_{j_1} - r_{j_1}, \quad \forall\, j_1 \in J_1 && (1)\\
& \sum_{j_1 \in J_1} x_{j_1 j_2} = r_{j_2} - p_{j_2}, \quad \forall\, j_2 \in J_2 && (2)\\
& x_{j_1 j_2} \in \mathbb{Z}^+, \quad \forall\, j_1 \in J_1,\ j_2 \in J_2 && (3)
\end{aligned}$$

2 Pyomo Implementation

The following is the implementation of the described abstract model.

```python
from pyomo.environ import *
import math

model = AbstractModel("Car rental")

#Sets and parameters

#set of agencies
model.agencies = Set()

#set of agencies with excess
model.excess_agencies = Set()

#set of agencies with need
model.need_agencies = Set()

#number of vehicles in the company
model.vehicles = Param(within = NonNegativeIntegers)

#number of cars present for each company
model.present_cars = Param(model.agencies, within = NonNegativeIntegers)

#number of cars required for each company
model.required_cars = Param(model.agencies, within = NonNegativeIntegers)

#X-coordinate of each agency
model.X = Param(model.agencies, within = NonNegativeReals)

#Y-coordinate of each agency
model.Y = Param(model.agencies, within = NonNegativeReals)

#road distance between each pair of agencies
def dist(model, i, j):
    return 1.3*math.sqrt((model.X[i] - model.X[j])**2 + (model.Y[i] - model.Y[j])**2)
model.road_distance = Param(model.excess_agencies, model.need_agencies, within =
                                            NonNegativeReals, initialize = dist)

#cost per km of transporting a car
model.cost_car = Param(within = NonNegativeReals)

#Variable
```

```
#variable indicating the flow from agencies with excess to agencies in need
model.move_ab = Var(model.excess_agencies, model.need_agencies, within =
                                              NonNegativeIntegers)

#Objective

#minimize the total transportation cost
def min_cost(model):
    return sum(sum(model.cost_car*model.road_distance[a,b]*model.move_ab[a,b] for b in
                                                  model.need_agencies) for a in model.
                                                  excess_agencies)
model.min_cost = Objective(rule = min_cost)

#Constraints

#Number of cars going from each excess agency to all agencies in need
def excess_release(model, a):
    return sum(model.move_ab[a,b] for b in model.need_agencies) == (model.present_cars[a]
                                                   - model.required_cars[a])
model.excess_release = Constraint(model.excess_agencies, rule = excess_release)

#Number of cars going in each agency in need from all excess agencies
def need_satisfy(model, b):
    return sum(model.move_ab[a,b] for a in model.excess_agencies) == (model.required_cars
                                                  [b] - model.present_cars[b])
model.need_satisfy = Constraint(model.need_agencies, rule = need_satisfy)
```

The following is the given data written in a .dat file for this specific model:

```
Car-rental.dat

#set of agencies
set agencies := 1 2 3 4 5 6 7 8 9 10;

#number of vehicles in the company
param vehicles := 94;

#number of cars present for each company
param present_cars := 1 8 2 13 3 4 4 8 5 12 6 2 7 14 8 11 9 15 10 7;

#number of cars required for each company
param required_cars := 1 10 2 6 3 8 4 11 5 9 6 7 7 15 8 7 9 9 10 12;

#X-coordinate of each agency
param X := 1 0 2 20 3 18 4 30 5 35 6 33 7 5 8 5 9 11 10 2;

#Y-coordinate of each agency
param Y := 1 0 2 20 3 10 4 12 5 0 6 25 7 27 8 10 9 0 10 15;

#cost per km of transporting a car
param cost_car := 0.5;

#set of agencies with excess
set excess_agencies := 2 5 8 9;

#set of agencies with need
set need_agencies := 1 3 4 6 7 10;
```

Since the above model was built as an abstract model where the data is separated from the model itself, then the following python script creates an instance of the model, chooses the solver and applies the chosen solver to solve the instance:

```
#linking the data with the abstract model
instance = model.create_instance("Car-rental.dat")

#creating the solver
solver = SolverFactory('OptiSolver')

#solving the created instance
results = solver.solve(instance)
```

The code below displays nicely the solution on the command line:

```
for i in instance.excess_agencies:
    for j in instance.need_agencies:
        if value(instance.move_ab[i,j]) > 0:
            print(f'Number of cars needed to move from agency {i} to agency {j} is {value
                                                            (instance.move_ab[i,j])}')
print(f'The minimum total transportation cost is {value(instance.min_cost)}')
```

While checking the termination condition of the solver (optimal, infeasible or unbounded) one can use similar code for writing the solution in a file as follows:

```
if results.solver.termination_condition == TerminationCondition.infeasible:
    print('The model is infeasible: No solution available')
elif results.solver.termination_condition == TerminationCondition.unbounded:
    print('The model has an unbounded solution')
elif results.solver.termination_condition == TerminationCondition.optimal:
    output = open('results.txt', 'w')
    for i in instance.excess_agencies:
        for j in instance.need_agencies:
            if value(instance.move_ab[i,j]) > 0:
                output.write(f'Number of cars needed to move from agency {i} to agency {j
                                                            } is {value(instance.move_ab
                                                            [i,j])}\n\n')
    output.write(f'The minimum total transportation cost is {value(instance.min_cost)}')
    output.close()
```

3 Results

The model has a total of 24 integer variables and 10 constraints. OptiSolver takes almost 0 seconds to solve the instance created by the given data. The Greedy Equality heuristic achieved the optimal solution. Below is displayed the optimal solution:

- Number of cars needed to move from agency 2 to agency 3 is 1
- Number of cars needed to move from agency 2 to agency 6 is 5
- Number of cars needed to move from agency 2 to agency 7 is 1
- Number of cars needed to move from agency 5 to agency 4 is 3
- Number of cars needed to move from agency 8 to agency 10 is 4
- Number of cars needed to move from agency 9 to agency 1 is 2
- Number of cars needed to move from agency 9 to agency 3 is 3
- Number of cars needed to move from agency 9 to agency 10 is 1

The minimum total transportation cost is 152.63901632295628

References

[1] Christelle Guéret, Christian Prins, and Marc Sevaux. Applications of optimization with xpress-mp. *contract*, page 00034, 1999.