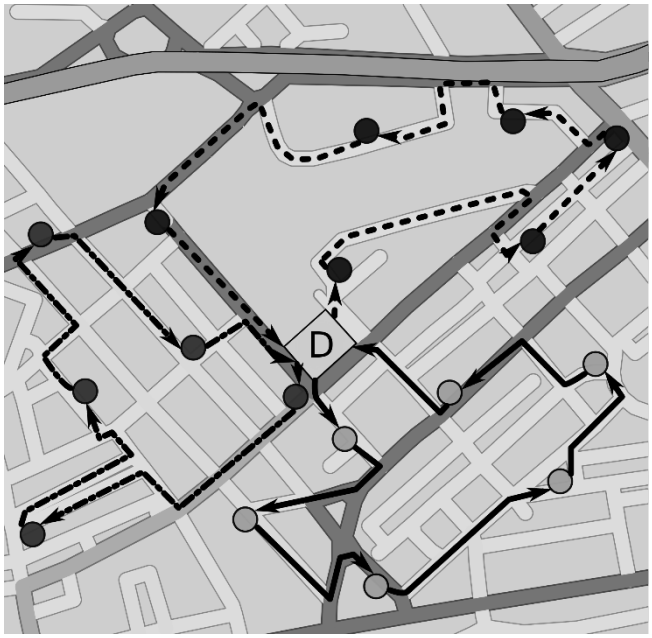
EB5104: Decision Making and Optimization

Tour Optimization



Submitted to:

A/Prof Poh Kim Leng
Dept. of Industrial Systems Eng. & Mgmt.
National University of Singapore

Submitted by:

Project Group No. 14

Apurv Garg	A0178205E
Daksh Gupta	A0178466M
Dibyajyoti Panda	A0178271Y
Duc Tran	A0178186N
Gopesh Dwivedi	A0178338R
Indu Arya	A0178360B

24 August 2018

Table of Contents

Background and Literature

Travelling experience has great impact on the tourist satisfaction, and therefore, tourists pay more attention to the experience utility of the tour. The present problem is how to plan the tour route to maximize tourism experience utility considering tourists' preference of attraction, time, and cost budgets. The utility function for the tourism experience, consisting of utilities of tourism activities and travel, was proposed. An optimization model for tour route planning was established with the objective function of the tourism experience utility. Then, the computational method to obtain the optimal solution was given, and the feasibility of the method was validated by an example of a tourism transportation network. The results showed that the tourists' preference of attraction, degree of attention to travel time, and travel cost had great influence on the tour route planning. The tourists with high value of time tend to choose transportation mode with shorter travel times, and the experience utility of the tourists with high value of time was higher than that of the tourists with low value of time.

Business Objective

The main objective of our project is categorised and sub-divided into three individual optimization problems, as given below:

- **Route optimization**: Optimisation of the route for a tourist to ensure that the shortest path is proposed before the journey to facilitate smoothened travelling experience

  Minimise $Z_1 = \sum_{i,j=1}^{i,j=n} C_{i,j} X_{i,j}$

  All defined arcs from node 'i' to node 'j' and C

- **Number of vehicles optimization**: Inventory Management to ensure the number of vehicle (Pre-owned and Rented) are in sync with the demands

  Minimise $Z_2 = l * c + m * d$

  We assumed that the running cost of vehicle/day of c type is $ 'l' and d type is $ 'm'

- **Cost optimization**: Vehicle Management to ensure the distribution of vehicles to different routes based on the capacity and budget requirements of the travellers

  Minimize $Z_3 = \sum_{1}^{n} (p_i c_i * q_i d_i)$

Methodology

The methodology used takes a hint from the **TRAVELLING SALESMAN PROBLEM** for route optimisation and using dynamic programming to ensure a faster response rate for the model.

Dynamic optimization is a method for solving a complex problem by breaking it down into a collection of simpler sub-problems, solving each of those sub-problems just once, and storing their solutions. Dynamic programming problems can be solved in 'stages'. Here decisions made in one stage will affect the decisions in the subsequent stages. In this scenario, backward recursion methodology is applied where decisions are made from the end stages than from the starting stages. The key characteristics of dynamic programming are as below

1. The problem can be divided into stages, with a policy decision required at each stage
2. Each stage has a number of states associated with it
3. The effect of the policy decision at each stage is to transform the current state into a state associated with the next stage according to the probability distribution
4. Given the current state, an optimal policy for the remaining stages is independent of the policy adopted in previous stages.
5. The solution procedure begins by finding the optimal policy for each state of last stage
6. A recursive relationship that identifies optimal policy for each state at stage n, given the optimal policy for each state at stage (n+1), is available

Mathematical Model

There are a total of three models designed in the problem statement solving three different objectives.

Model 1: Route Optimization

Objective Function

$$Z_1 = d_{12} + d_{21} + d_{23} + d_{32} + d_{34} + d_{43} + d_{45} + d_{54} + 2 * (d_{13} + d_{34} + d_{24} + d_{42} + d_{35} + d_{53}) + 3 * (d_{14} + d_{41} + d_{25} + d_{52}) + 4 * (d_{15} + d_{51})$$

Assumptions

1. 'd' is the binary variable declared for every route in both the direction.
2. There is 1 unit distance assumed between the two consecutive locations.
3. The objective is to minimize the route in terms of distance covered.

Constraints

The following constraints ensure that one location is visited by a vehicle only once.

1. $d_{21} + d_{31} + d_{41} + d_{51} == 1$
2. $d_{12} + d_{32} + d_{42} + d_{52} == 1$
3. $d_{13} + d_{23} + d_{43} + d_{53} == 1$
4. $d_{14} + d_{24} + d_{34} + d_{54} == 1$
5. $d_{15} + d_{25} + d_{35} + d_{45} == 1$

These constraints ensure that only one vehicle leaves each location.

1. $d_{12} + d_{13} + d_{14} + d_{15} == 1$
2. $d_{21} + d_{23} + d_{24} + d_{25} == 1$
3. $d_{31} + d_{32} + d_{34} + d_{35} == 1$
4. $d_{41} + d_{42} + d_{43} + d_{45} == 1$
5. $d_{51} + d_{52} + d_{53} + d_{54} == 1$

Sub-routing of the vehicles should be avoided, hence the following constraints also come into picture.

1. $u_2 - u_3 + 5 * d_{23} \leq 4$
2. $u_2 - u_4 + 5 * d_{24} \leq 4$
3. $u_2 - u_5 + 5 * d_{25} \leq 4$
4. $u_3 - u_2 + 5 * d_{32} \leq 4$
5. $u_3 - u_4 + 5 * d_{34} \leq 4$
6. $u_3 - u_5 + 5 * d_{35} \leq 4$
7. $u_4 - u_2 + 5 * d_{42} \leq 4$
8. $u_4 - u_3 + 5 * d_{43} \leq 4$
9. $u_4 - u_5 + 5 * d_{45} \leq 4$
10. $u_5 - u_2 + 5 * d_{52} \leq 4$
11. $u_5 - u_3 + 5 * d_{53} \leq 4$
12. $u_5 - u_4 + 5 * d_{54} \leq 4$

After following all of these constraints the model will set those binary variables (for every route) to 1 which minimizes the total distance travelled, rest all 0.

Model 2: No. of Vehicles optimization

Objective Function

$$Z_2 = 40x + 20y$$

Constraint

$$19x + 9y \geq 100$$

Assumption

1. Cost for type c vehicle is 40$ per day & cost for type ‘d’ vehicle is 20$ per day.
2. Average demand of 100 travellers per day in groups of different sizes.
3. Capacity of type c vehicle is 19 travellers $ type d is 9 passengers.

The constraint ensures that the total capacity of vehicles is sufficient enough to satisfy the demand. At the the same time we minimize the expenditure cost incurred by the traveller.

Model 3: Cost Optimization

Objective Function

$$Z_3 = 4 * c_1 + 2 * d_1 + 8 * c_2 + 4 * d_2 + 12 * c_3 + 6 * d_3 + 16 * c_4 + 8 * d_4 + 20 * c_5 + 10 * d_5$$

Assumption

1. Cost per traveller, travelling in c type vehicle (self-owned) for destination 1 is 4 units and for d type vehicle is 2 units and so on.
2. C_1 quantifies the number of travellers going in C type vehicle for destination 1, same for the rest
3. There are a total of 5 locations assumed [1→5] with 100, 200,300,400,500 unit distance from the company location respectively.

Constraints

The following are the constraint on the demands for every location:

1. $c_1 + d_1 \geq 2$
2. $c_2 + d_2 \geq 7$
3. $c_3 + d_3 \geq 23$
4. $c_4 + d_4 \geq 19$
5. $c_5 + d_5 \geq 13$

In this case a traveller has two options to travel to every location, either by vehicle c or by vehicle d. The numbers 2, 7, 23, 19, 13 are the different groups of travellers (demand) required to go to destination1, 2, 3, 4, 5 respectively.

The following are the constraints on the total capacity of the c type and d type vehicles, assuming an upper bound of 5 on the number of each type of vehicle owned by the company.

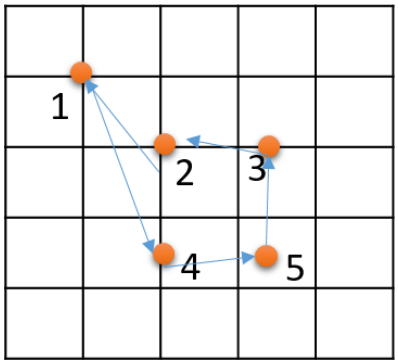
1. $c_1 + c_2 + c_3 + c_4 + c_5 \leq 95$ (19*5)
2. $d_1 + d_2 + d_3 + d_4 + d_5 \leq 45$ (9*5)

19 & 9 are the per vehicle capacities of the c type and d type vehicles respectively.

Optimal Solution

Model 1:

Minimum distance travelled after covering all the locations is = 8.0 unit distance
Arc selected for the Optimal distance: d_{14} = 1.0
Arc selected for the Optimal distance: d_{21} = 1.0
Arc selected for the Optimal distance: d_{32} = 1.0
Arc selected for the Optimal distance: d_{45} = 1.0
Arc selected for the Optimal distance: d_{53} = 1.0
Arc selected for the Optimal distance: u_5 = 1.0
Hence, the optimal route comes out to be: 1 → 4 → 5 → 3 → 2 → 1



Model 2:

Minimum investment by the company to buy vehicles and satisfy the daily demand is
= 220.0 $
Number of vehicles x = 5.0 (c-type)
Number of vehicles y = 1.0 (d-type)

Model 3:

Group carried by vehicle c_3 = 14.0
Group carried by vehicle c_4 = 10.0
Group carried by vehicle c_5 = 4.0
Group carried by vehicle d_1 = 2.0
Group carried by vehicle d_2 = 7.0
Group carried by vehicle d_3 = 9.0
Group carried by vehicle d_4 = 9.0
Group carried by vehicle d_5 = 9.0

So, there were 5 groups of 2,7,23,19,13 for locations 1,2,3,4,5 respectively:
1st group has been taken to their location by d-type vehicle
2nd group has been taken by d- type vehicle
3rd group has been taken by both c-type and d –type (14 by c type & 9 by d type)
4th group has again been taken by both types (10 by c-type & 9 by d-type)
5th group again by both types (4 by c-type & 9 by d-type)

Minimum Cost to Company = 656.0 $

Application and Future Scope

The application of the proposed solution runs through every sector which involves managing the demand of vehicles along with the optimisation of routes as per the demand to reduce cost and time.

- Demonstrate how supermarkets can make use of ridesharing vehicles to augment their delivery fleet and to develop a model that routes both the supermarkets vehicles and the ridesharing vehicles to satisfy customer demands
- Optimise the travel route for a fashion retailer's delivery unit based on daily orders , location of orders and availability of carrier vehicles
- Airline Crew scheduling where have crew stationed at different cities. They need to allocate crew people to flights such that at the end of their duty (ideally), every crew member is back at their home airport. Thus, assigning crew to flights can be completely mapped to the formulations stated

As part of the future enhancements –

- Google API's can be integrated to show live cost and distance forecasts for the destinations selected
- Along with route optimisation, the time window constraint can be added making it more flexible to demands

Appendix

Code (Python)

a) Objective - 1 (Route optimization)

```
import pulp
model1 = pulp.LpProblem("Cost minimization", pulp.LpMinimize)

# Declaring Binary variable for every route in every direction, assuming there are 5 location
d21 = pulp.LpVariable('d21', cat = 'Binary')
d31 = pulp.LpVariable('d31',   cat = 'Binary')
d41= pulp.LpVariable('d41',   cat = 'Binary')
d51 = pulp.LpVariable('d51',   cat = 'Binary')
d13 = pulp.LpVariable('d13',   cat = 'Binary')
d23= pulp.LpVariable('d23',   cat = 'Binary')
d43= pulp.LpVariable('d43',   cat = 'Binary')
d53 = pulp.LpVariable('d53',   cat = 'Binary')
d15= pulp.LpVariable('d15',   cat = 'Binary')
d25= pulp.LpVariable('d25',   cat = 'Binary')
d35= pulp.LpVariable('d35',   cat = 'Binary')
d45 = pulp.LpVariable('d45',   cat = 'Binary')
d12 = pulp.LpVariable('d12',   cat = 'Binary')
d32 = pulp.LpVariable('d32',   cat = 'Binary')
d42= pulp.LpVariable('d42',   cat = 'Binary')
d52  = pulp.LpVariable('d52',   cat = 'Binary')
d14 = pulp.LpVariable('d14',   cat = 'Binary')
d24 = pulp.LpVariable('d24',   cat = 'Binary')
d34 = pulp.LpVariable('d34',   cat = 'Binary')
d54= pulp.LpVariable('d54',   cat = 'Binary')

u2 = pulp.LpVariable('u2', lowBound = 0, cat='Integer')
u3 = pulp.LpVariable('u3', lowBound = 0, cat='Integer')
u4 = pulp.LpVariable('u4', lowBound = 0, cat='Integer')
u5 = pulp.LpVariable('u5',  lowBound = 0, cat = 'Integer')

# defining the Objective function based on the distance between 2 points [1 unit distance
assumed between consecutive locations]
model1+=d12 + d21 + d23 + d32 + d34 + d43 + d45 + d54 + 2*( d13 + d31 + d24 + d42 +
d35 + d53) +3* (d14 + d41 + d25 + d52 ) + 4* (d15 + d51) , "Z"

model1+= d21 + d31 + d41 + d51 == 1 #...
model1+= d12 + d32 + d42 + d52 == 1
model1+= d13 + d23 + d43 + d53 == 1
model1+= d14 + d24 + d34 + d54 == 1
model1+= d15 + d25 + d35 + d45 == 1# One location is visited by a vehicle only once

model1+= d12 + d13 + d14 + d15 == 1#...
```

```
model1+= d21 + d23 + d24 + d25 == 1
model1+= d31 + d32 + d34 + d35 == 1
model1+= d41 + d42 + d43 + d45 == 1
model1+= d51 + d52 + d53 + d54 == 1# each location is left by a vehicle only once

# To avoid subrouting of vehicles
model1+= u2 - u3 + 5*d23 <= 4
model1+= u2 - u4 + 5*d24 <= 4
model1+= u2 - u5 + 5*d25 <= 4
model1+= u3 - u2 + 5*d32 <= 4
model1+= u3 - u4 + 5*d34 <= 4
model1+= u3 - u5 + 5*d35 <= 4
model1+= u4 - u2 + 5*d42 <= 4
model1+= u4 - u3 + 5*d43 <= 4
model1+= u4 - u5 + 5*d45 <= 4
model1+= u5 - u2 + 5*d52 <= 4
model1+= u5 - u3 + 5*d53 <= 4
model1+= u5 - u4 + 5*d54 <= 4
model1
model1.solve()
pulp.LpStatus[model1.status]
value = pulp.value(model1.objective)
print ("Minimum distance travelled after covering all the locations is = {} unit
distance".format(value))

for variable in model1.variables():  #prints just the variables with value 1, signifying that the
vehicle goes on that route.
    if(variable.varValue ==1):
        print ("Arc selected for the Optimal distance: {} = {}".format(variable.name,
variable.varValue))
```

```
Output:
Minimum distance travelled after covering all the locations is =     8.0 unit distance
Arc selected for the Optimal distance: d14 = 1.0
Arc selected for the Optimal distance: d21 = 1.0
Arc selected for the Optimal distance: d32 = 1.0
Arc selected for the Optimal distance: d45 = 1.0
Arc selected for the Optimal distance: d53 = 1.0
Arc selected for the Optimal distance: u5 = 1.0
```

b) Objective - 2 (No of vehicle optimization):

```
import pulp
model = pulp.LpProblem("Cost minimization", pulp.LpMinimize)

# x is the number of vehicle of c type(self-owned) and y for d type(rented)
x = pulp.LpVariable('x', lowBound = 0, cat='Integer')
```

```
y = pulp.LpVariable('y', lowBound = 0, cat='Integer')

model+= 40*x + 20*y, 'Z' # We assume that the running cost of vehicle/day of c type is 40$ and d type is 20$
model+= 19*x +9*y>=100   # Assuming an average demand of 100 tourist/day in different groups; 19 is capacity/vehicle for c type, 9 is capacity/vehicle for d type

model
model.solve()
pulp.LpStatus[model.status]

value = pulp.value(model.objective)
print ("Minimum investment by the company to buy vehicles and satisfy the daily demand is = {} $".format(value))

for variable in model.variables():
    print ("Number of vehicles {} = {}".format(variable.name, variable.varValue))
```

```
Output:
Minimum investment by the company to buy vehicles and satisfy the daily demand is
 = 220.0 $
Number of vehicles x = 5.0
Number of vehicles y = 1.0
```

c) Objective - 3 (Cost optimization):

```
import pulp
model1 = pulp.LpProblem("Cost minimization", pulp.LpMinimize)

# c is the self owned vehicle type, 5 [upper bound] in number [There are 5 locations assumed as demand, therefore c1 to c5]
c1 = pulp.LpVariable('c1', lowBound = 0, upBound = 19, cat='Integer')
c2 = pulp.LpVariable('c2', lowBound = 0, upBound = 19, cat='Integer')
c3 = pulp.LpVariable('c3', lowBound = 0, upBound =  19, cat='Integer')
c4 = pulp.LpVariable('c4', lowBound = 0, upBound = 19,cat ='Integer')
c5 = pulp.LpVariable('c5',  lowBound = 0, upBound = 19, cat = 'Integer')

# d is the rented type vehicle, 5 in number [upper bound] [There are 5 locations assumed as demand, therefore d1 to d5]
d1 = pulp.LpVariable('d1', lowBound = 0, upBound = 9, cat='Integer')
d2 = pulp.LpVariable('d2', lowBound = 0, upBound = 9, cat='Integer')
d3 = pulp.LpVariable('d3', lowBound = 0, upBound = 9, cat='Integer')
d4 = pulp.LpVariable('d4', lowBound = 0, upBound = 9, cat='Integer')
d5 = pulp.LpVariable('d5',  lowBound = 0, upBound = 9, cat = 'Integer')

#5 locations at 100, 200, 300, 400 & 500 unit distnaces apart from the company location with different costs/passenger attached to it - Assumption
```

```
model1+=4*c1 + 2*d1 + 8*c2 + 4*d2 + 12*c3 + 6*d3 + 16*c4 + 8*d4 + 20*c5 + 10*d5, "Z" # Objective Function

# Let's say 5 groups approach the company in sizes of 2, 7,23,19,13 for each location.
model1+=c1 + d1 >=2 # People going in c type - c1 in d type - d1
model1+=c2 + d2 >=7
model1+=c3 + d3 >=23
model1+=c4 + d4 >=19
model1+=c5 + d5 >=13
model1+=c1 + c2 +c3 +c4 +c5 <=95 # capacity constraint
model1+=d1 + d2 +d3 +d4 +d5 <=45

model1

model1.solve()
pulp.LpStatus[model1.status]


for variable in model1.variables():

    print ("Group carried by vehicle {} = {}".format(variable.name, variable.varValue))


value = pulp.value(model1.objective)
print ("Minimum Cost to Company = {} $".format(value))
```

Output:

```
Group carried by vehicle c1 = 0.0
Group carried by vehicle c2 = 0.0
Group carried by vehicle c3 = 14.0
Group carried by vehicle c4 = 10.0
Group carried by vehicle c5 = 4.0
Group carried by vehicle d1 = 2.0
Group carried by vehicle d2 = 7.0
Group carried by vehicle d3 = 9.0
Group carried by vehicle d4 = 9.0
Group carried by vehicle d5 = 9.0
Minimum Cost to Company = 656.0 $
```

References

- https://neos-guide.org/content/woodmans-delivery#description
- https://developers.google.com/optimization/
- Pulp Documentation : https://pythonhosted.org/PuLP/
- https://en.wikipedia.org/wiki/Vehicle_routing_problem
- http://staffnew.uny.ac.id/upload/198504142009122003/penelitian/complete-paperiicmagp.pdf