A Van on the Mission: A Traveling Salesman Problem

Peter Xu
University of Washington, Seattle

August 16, 2021

Abstract

In this report an asymmetric traveling salesman problem is solved using Miller-Tucker-Zemlin formulation. Specifically, a linear model is built to minimize the transportation cost for the commercial practice called "road show". A road show is the practice of one company sending a few number of staffs to a city, where they will travel in a van to smaller cities around the starting city to hold small scale business conferences. This is commonly used among small businesses in main land China. Moreover, the model built in this report is designed to provide a solution to satisfy the requirements of one company alone, which means that this model may not be ideal for others. In this report, we are only considering the transportation costs imposed by distances between cities, human costs from traveling between the cities, and rental fee for the vehicle. The money spent within every city (i.e. hotel cost and everything related to the conferences) is considered as fixed cost for the company and is irrelevant of the transportation cost.

1 Introduction

In modern society, advertisements targeting consumers can directly be sent to cellphones through many medias. And international corporations are able to hold large scale online conferences easily to get attentions from potential business partners all over the globe. However, small companies do not have such resource to have any of those mentioned above. So, these small companies in main land China must rely on road shows to search for business partners in small cities. Then, there rise the problem of planning, as the company is small and every optimization can help the business to survive. In what order should the staff visit the cities such that the transportation cost can be minimized?

1.1 Brief Background

Here is how a road show progress. A small business located in a big city wants to expand its business in smaller cities with existing commercial agents. These commercial agents have the duty to sell products and provide communication services for the company on local level. However, most of the agents do not have the resource to hold conferences, to

attract potential local partners, on their own. Thus, the company must send staff to aid them. Again, due to the fact that these small cities are small, using public transportation to move necessary equipment for the team can be costly. Thus renting a van for the team is the chosen method.

1.2 Description

Now that we are given the background, we will state the problem that we are solving. Given a list of cities, we are to **find a visiting order of the cities such that our transportation cost is minimized**.

2 Assumptions

For many practical reasons, there are many assumptions made in the development of this model.

1. Assume that the team will not encounter any human or natural disaster. The planning of road shows can happen more than a month earlier, thus worrying about these unpredictable elements is unnecessary.
2. There is always going to be one vehicle for the team. Considering the small scale nature of the conferences, sending a large number of staff is wasteful. Also, a single rental van among the vans available on Chinese market is enough to transport the team and equipment legally and safely.
3. The fuel cost and human cost are assumed to be constants. Since the exact fuel cost may vary among different makes of the vehicle and different conditions of the roads, specifying these elements in our model will be unnecessarily complicate. Human cost consists of two parts: hotel and bonus salary. Each staff is given a room in a hotel for every night and an amount of bonus salary for every day. The cost for a room in hotels can vary significantly, but the company is providing hotels with basic functions which tend to not have a great variation in prices. As for the bonus salary, it is a constant for all staff to begin with.
4. The team always depart a city in the morning. No matter what time the conference ends the day before, the team will stay at a hotel in the city for the night. Also, the team will not hold any conference the day the arrive at a city, regardless the arriving. This arrangement is made basing on the considerations for the employees. Giving them enough rest and preparation time can ensures their health and working efficiency during the conferences.
5. All local commercial agents will be available whenever the date appointed for the conference is. We can make such assumption because the conferences do not need many staff and the conferences last only for a few hours. Thus, the agents will have their arrangements to fit the company's schedule.
6. Based on 5, we can assume that the team will visit every city only once.

7. All cities are connected by roads for vehicles. Villages unable to reach by cars hold no business value for the company.
8. The team must rest for 30 minutes after 2 hours of consecutive driving. Excluding the resting time, the team cannot drive more than 8 hours a day. If the path chosen by the model requires a driving more than 8 hours. The team will find a hotel in the middle and rest for the night.
9. Assume the furthest distance between two cities will not take more than 16 hours of driving, which translate into 3 days by assumption 8. Typically, in mainland China, such distance means different provinces. The company almost always restrict a road show to one province. Thus, we assume the maximum driving distance between two cities is 16 hours worth.
10. Based on assumption 4 and 8, we can sort all hotel costs in the cities as a part of the fixed costs. The team has to stay in their hotel for the night before they depart and the night after they arrive at a city. As long as there is a city that need road show, the hotel cost of these two nights cannot be optimized.

3 Mathematical Model

Given the assumptions above, we can safely use linear programming with the Miller-Tucker-Zemlin formulation to find the cost minimizing route. And below is the list of mathematical representations of the constants and variables used in solving the problem:

1. Suppose n as the number of staff on the trip.
2. Let g be the fuel cost of our van, with the unit of RMB/km. c_h as the hotel room cost for one person per night. c_s as the salary cost for one person per day. c_v as the cost for renting a van per day.
3. Let D be the distance matrix. Each entry (d_{ij}) is the most recommended route suggested by Baidu Map. Note that D is not necessarily symmetric as the map may suggest different routes when starting city and destination is changed.
4. Let V be the set of indices that represent all the cities.
5. Suppose binary variable x_{ij} be 1 if route from city i to j is used, 0 otherwise. And $i, j \in V$.
6. Suppose matrix T to have its ij-entry (t_{ij}) as binary variables indicating whether the estimated time using route from city i to city j exceed 8 hours.
7. Also, create u_i for $i \in V$ where $u_i =$ the order of visit to city i.

Using the above constants and variables, we write our linear programming problem as:

$$
\begin{array}{lll}
\text{minimize} & \sum_{i,j}[t_{ij} \cdot n \cdot (c_s + c_h) + n \cdot c_s + g \cdot d_{ij} + (t_{ij} + 1) \cdot c_v] x_{ij} & \\
\text{subject to} & \sum_{j, i \neq j} x_{ij} = 1 & \forall i \\
& \sum_{i, i \neq j} x_{ij} = 1 & \forall j \\
& u_1 = 1 & \\
& 2 \leq u_i \leq n & \forall 2 \leq i \leq n \\
& u_i - u_j + n x_{ij} \leq n - 1 & \forall 2 \leq i, j \leq n
\end{array}
$$

The objective function consists of two parts: constant $[t_{ij} \cdot n \cdot (c_s + c_h) + g \cdot d_{ij} + (t_{ij} + 1) \cdot c_v]$ and variable x_{ij}. If the model chooses to use the path from city i to city j, then $x_{ij} = 1$ and the cost of path ij is summed into the transportation cost. If the model chooses to not use the path, then $x_{ij} = 1$ and the cost from city i to city j is not considered. As for the constant part, we can further separate them into human cost $t_{ij} \cdot n \cdot (c_s + c_h) + n \cdot c_s$ and vehicle cost $(g \cdot d_{ij} + (t_{ij} + 1) \cdot c_v)$. These are all user input so we regard them as constants in our model. For the path from city i to j, each person in a team consists of n people will cost an extra $c_h + c_s$ RMB if the path takes more than 8 hours to drive through, because they have to stay in a hotel for a night. If the path takes less than 8 hours, then each person only costs c_s RMB for his bonus salary of the day, the hotel cost for the day is categorized as fixed cost and not considered by our model. Taking the path from city i to j, the van will consume g RMB worth of fuel per kilometer and a rental fee of c_v RMB. Then the path ij will contribute $g \cdot d_{ij} + c_v$ RMB to the transportation cost. If the path ij requires 2 days, then an extra c_v rental cost will be added. The first constraint makes sure that we are leaving one city for only one next city. The second constraint ensures that we arrive at a city only from one city. The third and fourth constraints are the bounds on variable u_i for all i. As for the fifth constraint, it prevents the result of having more than one loop among the cities. This formulation is taken from Lecture 11 of MATH 381 of Summer 2021, with a slight edit on the bounds of variable u_i.

3.1 Solution of the Mathematical Model

The solution of the model above will give the minimized cost of our route in the city. But in the implementation we will also extract a list of city names and their order of visit as it is an essential information for the company.

4 Simulation and Solution

As an example, we will be finding the cost minimizing route for visiting the following ten cities from Shandong province in mainland China: Jinan, Jiyangqu, Yucheng, Gaotangxian, Liaocheng, Yangguxian, Taian, Xintai, Zibo, Binzhou. Below is the data entry for the model:

[2]:
```
n = 2.0
g = 0.8
ch = 180.0
```

```
cs = 120.0
cv = 780.0
Cities = {1:'Jinan', 2:'Jiyangqu', 3:'Yucheng', 4:'Gaotangxian', 5:
 ↪'Liaocheng', 6:'Yangguxian', 7:'Taian',
          8:'Xintai', 9:'Zibo', 10:'Binzhou'}
D = {1: [0, 46, 66.7, 102.6, 125.6, 176.8, 74.8, 132.6, 111, 153.3],
     2: [45.5, 0, 74.1, 107.6, 135, 192.4, 126.4, 155.3, 110.2, 109],
     3: [67.6, 71.3, 0, 41.2, 97.8, 155.2, 109.7, 189.9, 152.1, 176.4],
     4: [108.7, 104.9, 41.2, 0, 54.2, 112.1, 146.1, 230.7, 191, 215.4],
     5: [125.5, 134.4, 93.5, 54, 0, 45.9, 130.5, 201.4, 215.2, 239.5],
     6: [183, 191.8, 155, 111.8, 45.9, 0, 153.2, 224, 270.1, 296.9],
     7: [75, 122.8, 119, 172, 130.8, 153.2, 0, 82.8, 156.9, 230],
     8: [126.8, 161.8, 189.4, 229.9, 201.7, 223.4, 81.7, 0, 130, 182.3],
     9: [112, 119.7, 152.3, 191.2, 211.1, 268.6, 159.2, 126.8, 0, 85.3],
     10: [160.5, 99.4, 170.9, 212, 237.8, 295.3, 217.3, 184.3, 80.7, 0]}
T = {1: [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
     2: [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
     3: [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
     4: [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
     5: [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
     6: [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
     7: [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
     8: [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
     9: [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
     10: [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]}
```

Due to the fact that Google Map cannot be accessed in mainland China, the distance matrix D and time matrix T are determined using Baidu Map, the most commonly used navigation application in mainland China. The distance and time are both taken from the most recommended route from Baidu Map. Using python, we write two functions, *buildVanModel* and *reportOnVan*, to build, solve, and report the linear programming model. *buildVanModel* will build the linear programming model using the input data and *reportOnVan* will call builVanModel and solve the model. Below is how we use reportOnVan to solve the model and report the result.

[4]:
```
reportOnVan(n, g, ch, cs, cv, D, T)
```

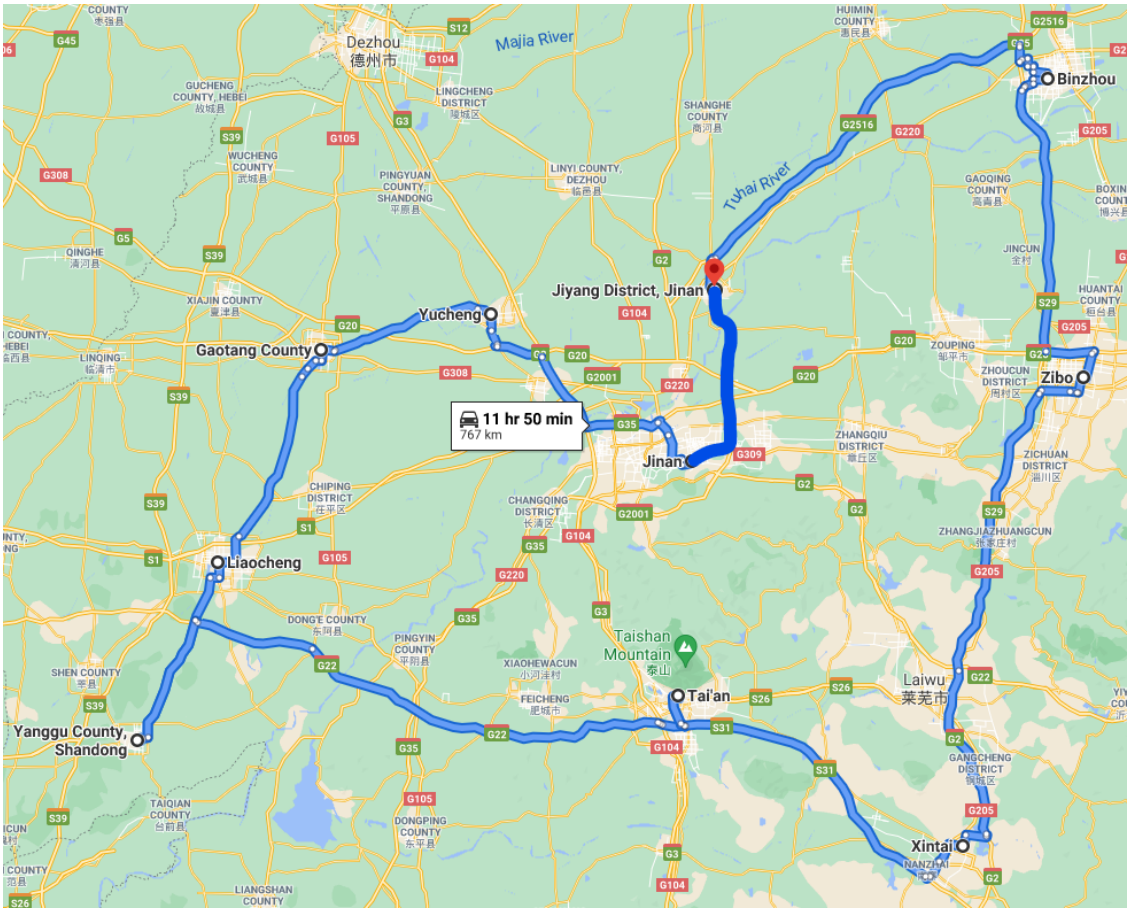
```
Minimized cost of transportation: 10843.36 RMB

The cities and their visiting order:
Jinan: 1.0
Jiyangqu: 10.0
Yucheng: 2.0
Gaotangxian: 3.0
Liaocheng: 4.0
```

```
Yangguxian: 5.0
Taian: 6.0
Xintai: 7.0
Zibo: 8.0
Binzhou: 9.0
```

5 Results and Analysis

As we can see, the model gives us a minimized cost of transportation at 10843.36 RMB. The integers after the cities' names are the order which they will be visited. One means the first, two means the second, and ten means the last. Then, our team will return to the first city after visiting the tenth city. Below is the route highlighted using Google Map instead of Baidu Map, since Baidu only allows 6 destinations at most while Google Map allows 9. Based on the order list provided by the model, we will start in Jinan and go counter clockwise along the highlighted path till we return to Jinan. The path between Jiyangqu and Jinan is highlighted by hand following the suggested route from Baidu, the reason is that Google Map can highlight at most 9 paths.



This result makes sense and is also practical in reality. If the team travel through the cities

following the reversed order of the order calculated by the model, the new transportation cost will be 10844.88 RMB. The difference here is tiny, but the model has caught it and found the optimal solution.

6 Improvements/Future Work

The current model requires a very specific type of data input for the matrices D and T. And these matrices are also manually typed into the program. Thus, a possible improve can be a automatized import of distance matrix from Baidu Map Route Matrix API. However, due to the time sensitivity, learning the automation can cause distraction from the most important learning subject of this project. Therefore, the manual import of matrices is kept. Also, the only reason it is useful to the company is due to the fact that it is at the high end of the command chain. If the company has to respond to the local commercial agents' requests on schedule, this model will not give a satisfying result. Hence, a more general version of this model can be made by considering time slots for the agents.

7 Conclusions

In this report, my own work in homework 2 to homework 4 of MATH 381 are used as reminder of many functions of PySCIPOpt. And the MTZ formulation of the asymmetric traveling salesman problem provided in Lecture 14 of MATH 381 Summer 2021 forms the core of the model built and used in this report. In the process of writing the code for the model, I made some design choices such as how should I represent the matrices with dictionary. And writing the report itself requires a lot of work as organizing thoughts into a professional report for a math project is a first to me.

8 Acknowledgments

I want to thank my father and his company, Uni-Colour International Inc., for providing the valuable problem and data to make this report possible. Also, my instructor Junaid Hasan for the feedback on many parts of this work.

References

[Hasan] *Lecture-11-Traveling Salesman Problem*, Junaid Hasan 2021.

[PySCIPOpt] *PySCIPOpt: Mathematical Programming in Python with the SCIP Optimization Suite*, Stephen J. Maher, Matthias Miltenberger, João Pedro Pedroso, Daniel Rehfeldt, Robert Schwarz, Felipe Serrano, 2016.

[1]:
```
def buildVanModel(n, g, ch, cs, cv, D, T):
    from pyscipopt import Model, quicksum
    modelVan = Model("Travelling Van")

    N = len(D)
    x = {}
    u = {}

    for i in range(1, N+1):
        u[i] = modelVan.addVar(lb = 1, ub = N, vtype = "I", name =␣
↪"u(%s)"%i)
        for j in range(1, N+1):
            if i != j:
                x[i,j] = modelVan.addVar(vtype = "B", name = "x(%s,␣
↪%s)"%(i, j))

    for i in range(1, N+1):
        modelVan.addCons(quicksum(x[i,j] for j in range(1, N+1) if i !=␣
↪j) == 1, "Only 1 out at %s"%i)
        modelVan.addCons(quicksum(x[j,i] for j in range(1, N+1) if i !=␣
↪j) == 1, "Only 1 in at %s"%i)

    modelVan.addCons(u[1] == 1, "u(1) = 1")
    for i in range(2, N+1):
        modelVan.addCons(2 <= (u[i] <= N), "Bound for u(%s)"%i)

    for i in range(2, N+1):
        for j in range(2, N+1):
            if i != j:
                modelVan.addCons(u[i] - u[j] + N*x[i,j] <= N-1, "Only one␣
↪loop in graph")

    modelVan.setObjective(quicksum((T[i][j-1]*n*(cs+ch) + n*cs +␣
↪g*D[i][j-1] + (T[i][j-1]+1)*cv)*x[i,j] for (i,j) in x),
                          "minimize")
    modelVan.data = x, u

    return modelVan
```

[3]:
```
def reportOnVan(n, g, ch, cs, cv, D, T):
    modelVan = buildVanModel(n, g, ch, cs, cv, D, T)
    modelVan.optimize()
    x, u = modelVan.data
```

```
    print("Minimized cost of transportation: " + str(modelVan.
↪getObjVal()) + " RMB")
    print()

    print("The cities and their visiting order: ")
    for i in range(1, len(D)+1):
        print(Cities[i] + ": " + str(modelVan.getVal(u[i])))
```