Operation Research Final Project

Team name: Youbike3.0
Team member: 詹承諺、蔡承耘、張永毅、歐逸宣

Introduction

Since 2020, NTU(National Taiwan University) has started introducing the bicycle-sharing system, Welcome to YouBike 2.0, to reduce the number of self-used bikes on campus. After 3 years of operation, more and more stations and bikes were joined, but the two core problems, (1) no bicycle for renting, and (2) no space to park, are still bothering most users in peak hours. Therefore, an optimized replenishment strategy is in dire need. In this paper, we propose a method with the optimization tool TSP(Traveling Salesman's Problem) to deal with the challenge.

Literature Review

To propose an outstanding performance, the paper: “Replenishment Strategies for the YouBike System in Taipei” is referred to. In that paper, two performance indexes are considered, one focused on no bike to rent situation, and another focused on no space to park the bikes situation. The result shows that the proposed method: “If the parking space is 70 % full, 20 % of the station’s maximal parking bikes are moved to another station which is nearest” increases 210.82% average available bikes, and decreases 97.61% average lack of parking space than the lower bound. Compared to other methods, we can see that lowering the threshold to move the bikes and increasing the number of bikes to be moved is the key to better results. Although the target of our paper is to find the minimum path to replenish bikes rather than the most efficient replenishment strategy, we can still consult the preliminary, experiment setting and the concluded knowledge from the paper, and the details will be shown in the following package.

Methodology

1. Model Introduction
    a. Objective:
        i. We hope to optimize the best route for YouBike Company to replenish bikes at stations near National Taiwan University through operational research. Our goal is to minimize the time it takes to replenish vehicles, where we take into account comparing the demand to empty ratio at each site. Detailed implementation details will be explained in subsequent chapters.

b. Assumption:
    i. During the vehicle replacement process, no other vehicles will be loaned or replaced.
    ii. Only one work vehicle is undergoing repair operations.
    iii. The work vehicle that is used for vehicle replacement needs to return to the starting station.
    iv. The work vehicle that is used for vehicle replacement needs to traverse every NTU station, and after replenishing vehicles at each station, they will not pass through again.
    v. Each station can be reached in a straight line.
    vi. The work vehicle that is used for vehicle replacement has enough bicycle capacity to meet the needs of all sites.
c. The main challenge in the real world:
    There will still be vehicles entering and exiting each station during the vehicle replacement process, and it is not possible to reach each station through a straight line between stations.

All the code is available on Github: https://github.com/JimmyTsai-Yun/112-1_OR_FinalProject

2. Data Collection

We obtained YouBike-related information through YouBike2.0 Taipei City public bicycle real-time information provided by the Taipei City Government, including site name, site longitude and latitude, site volume, and site empty volume. This data is updated every minute, but in our method, we collect NTU YouBike2.0 site data every ten minutes between 6:00 and 7:00 on weekdays. We average the number of empty carriages at each station collected at each time point to determine the number of empty carriages in the final model. Since we have the absolute location of each station, we use Geopy to compute the distance between each station which are represented as a n*n matrix, n is the number of total stations we consider as near National Taiwan University.

3. The Model:
   a. Initial Version of TSP (Traveling salesman problem):
      To solve the problem of replenishment strategy for Youbike in NTU, we use the method of Traveling salesman problem to get the shortest path for replenishment. The initial version of TSP is as follows.

$$\min \sum_{i \in V} \sum_{j \in V} c_{ij} x_{ij}$$

s.t.

$$\sum_{i \in \{V \setminus j\}} x_{ij} = 1, \forall j \in V$$

$$\sum_{j \in \{V \setminus i\}} x_{ij} = 1, \forall i \in V$$

$$u_i - u_j + N \cdot x_{ij} \leq N - 1, \forall i, j \in \{V \setminus N\} \text{ and } i \neq j$$

$$u_i \geq 0, \forall i \in \{V \setminus N\}$$

$$x_{ij} \text{ binary}, \forall i, j \in V$$

   b. Revised Version of TSP:
      To let the station with fewer bikes be replenished earlier, we tried to add some weights so that the model could consider both the proportion of empty bikes and the distance between every station. Specifically, we use the proportion of remaining bikes (between 0 and 1) as the weight (ω_{ij}) to be multiplied by the distance around that station. The revised version of TSP is as follows.

$$\min \sum_{i \in V} \sum_{j \in V} \omega_j * c_{ij} * x_{ij}$$

s.t.

$$\sum_{i \in \{V \setminus j\}} x_{ij} = 1, \forall j \in V$$

$$\sum_{j \in \{V \setminus i\}} x_{ij} = 1, \forall i \in V$$

$$u_i - u_j + N \cdot x_{ij} \leq N - 1, \forall i, j \in \{V \setminus N\} \text{ and } i \neq j$$

$$u_i \geq 0, \forall i \in \{V \setminus N\}$$

$$x_{ij} \text{ binary}, \forall i, j \in V$$

4. Gurobi Code

We implement two types of TSP as mentioned above: Initial Version of TSP (TSP_no_weight.py) and Revised Version of TSP (TSP_with_weight.py).

a. Initial Version of TSP (TSP_no_weight.py):

1. Define nodes and distance

```
nodes = []
for i in range(len(distance_matrix)):
    nodes.append(str(i+1))
distance = {}
for i in range(distance_matrix.shape[0]):
    for j in range(distance_matrix.shape[1]):
        distance[(str(i+1),str(j+1))] = distance_matrix.iloc[i,j]
arcs, distance = multidict(distance)
```

2. Create the model and variables

```
# Create optimization model
m = Model('TSP')
# Create variables
x = {}
u = {}
for i,j in arcs:
    x[i,j] = m.addVar(obj=distance[i,j], vtype = 'B',
    name='x_%s%s' % (i, j))
N = len(nodes)
for i in nodes:
    if i != nodes[N-1]:
        u[i] = m.addVar(obj=0, name='u_%s' % i)
m.update()
```

3. Add constraints and compute the optimal solution

```
# Constraint for sum of incoming links to j
for j in nodes:
    m.addConstr(quicksum(x[i,j]
            for i in nodes if i != j) == 1,
                'incom_%s' % (j))
# Constraint for sum of outgoing links from i
for i in nodes:
    m.addConstr(quicksum(x[i,j]
            for j in nodes if i != j) == 1,
                'outgo_%s' % (i))
# Subtour elimination constraints
for i,j in arcs:
    if i != nodes[N-1] and j != nodes[N-1]:
        m.addConstr(u[i] - u[j] + N*x[i,j] <= N-1,
                'subtour_%s_%s' % (i, j))
# Compute optimal solution
m.optimize()
```

4. Print and save the solution to further draw the route

```
# Print solution
route = []
if m.status == GRB.Status.OPTIMAL:
    print('objective: %f' % m.ObjVal)
    solution = m.getAttr('x', x)
    for i,j in arcs:
        if solution[i,j] > 0:
                print('%s -> %s: %g' % (i, j, solution[i,j]))
                route.append([i,j])
print(route)
route = pd.DataFrame(route)
route.to_csv('/Users/jasper/Operation Research/Final_Project/route_no_weight.csv', index=False, header=False)
```

b. Revised Version of TSP (TSP_with_weight.py):
Use the average number of empty bikes for every station and the total docks to calculate the average proportion of available bikes.

```
with open('/Users/jasper/Operation Research/Final_Project/schoolStationList.json') as station_file:
    station_info = json.load(station_file)
print(station_info[0]["bemp"])
bike_ava = np.empty((70,1), dtype=float)
for i in range(len(station_info)):
    total = station_info[i]["tot"]
    bemp = station_info[i]["bemp"]
    bike_ava[i,0] = 1-(bemp/total)
print(bike_ava)
```

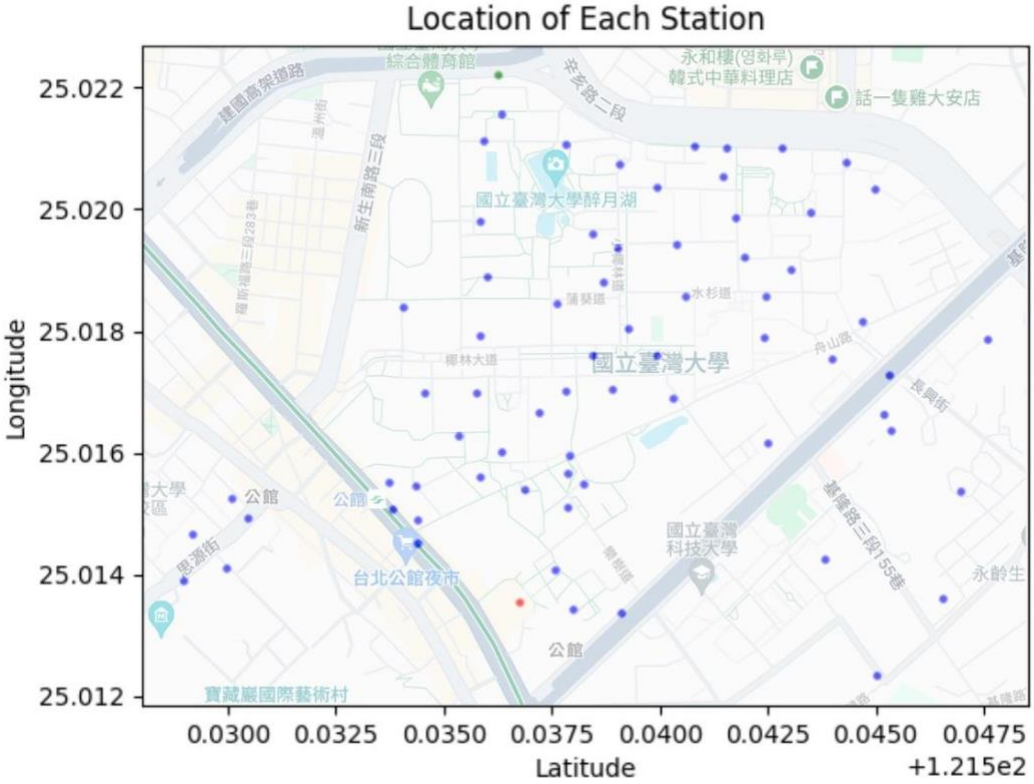
Afterward, we let the average proportion of available bikes be the weight to be multiplied by distances.

```
distance = {}
for i in range(distance_matrix.shape[0]):
    for j in range(distance_matrix.shape[1]):
        distance[(str(i+1),str(j+1))] = distance_matrix.iloc[i,j]*bike_ava.transpose[0,j]
```

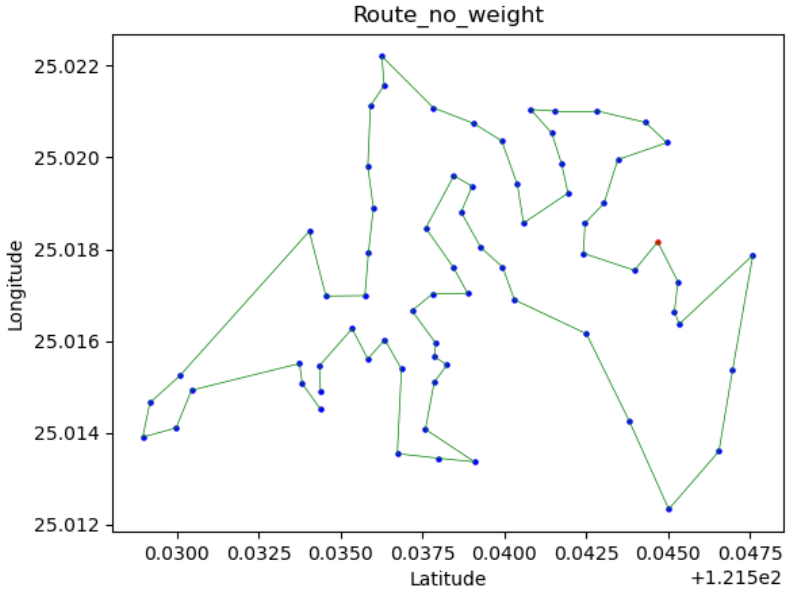
The remaining part of “TSP_with_weight.py” is the same as “TSP_no_weight.py”.

Results

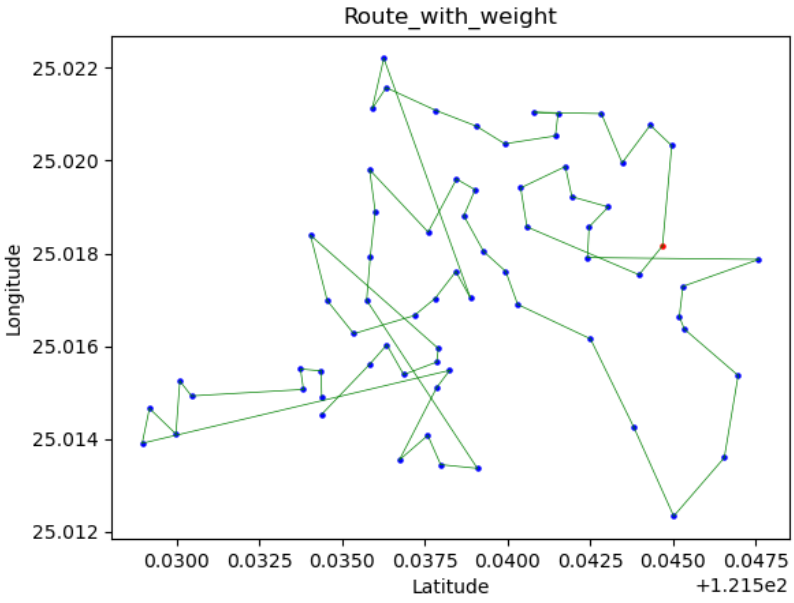
The following picture shows a total of 70 stations set up within and near National Taiwan University that we have collected this time.



The following figure shows the optimal route designed by the model without putting weights in. As can be seen from the figure, without the influence of weights on distance, the route will simply become a loop formed by each station because it will automatically select the nearest station that has not yet been supplemented with Youbikes.



The following figure is the result after adding the weight. It can be seen that if we put the weights in the model, the distance between the stations in the model is changed, so although the nearest station will be automatically selected as the next target, it will not simply select the adjacent station, and the route will become more complicated.



Considering the way Youbike replenishes vehicles in reality, I think this model may only be used as a reference, because in reality, when replenishing vehicles, all stations will not be replenished at once like in the model. Additionally, we assumed that only one truck is responsible for all stations within NTU. This is obviously not in line with the actual operation model, and the weights in our model are fixed values obtained by calculating the empty bike rates which should actually be values that will fluctuate over time, so I think this model may only be used as a reference, rather than a tool that can be fully applied to reality. If the empty rate can be used to determine the stations that the route needs to pass, or add a method that can dynamically calculate the empty rates, the model may be more realistic.

Personal Thought

Because I’m responsible for programming, what I’ve learned is how to deal with large amounts of data including collecting, preprocessing, and applying to our project. The hardest part of this project is to define the weights of every station to indicate which one should be replenished first. For this project, we only use the average proportion of the available bikes as the weight multiplied by the distance around the station. This method might not be comprehensive because we only use one-day data which may be different every day.

Work Distribution

詹承諺 30%、蔡承耘 30%、張永毅 20%、歐逸宣 20%