Feasibility Validation of Launching LA Metro Bike Relocation Service using Stochastic Programming

University of Southern California
Jae Yul Woo

Data Configuration

- Metro Bike share open-data consists of 14 columns (with each data type):
  - Trip ID (int)
  - Duration (float)
  - Start Time (object)
  - End Time (object)
  - Starting Station ID (int)
  - Starting Station Latitude (float)
  - Starting Station Longitude (float)
  - Ending Station ID (int)
  - Ending Station Latitude (float)
  - Ending Station Longitude (float)
  - Bike ID (int)
  - Plan Duration (float)
  - Trip Route Category (string)
  - Passholder Type (string)

- Trip data from July 2016 to June 2017
- Trip Duration (in seconds)
- 64 Stations
  - Station Latitude and Longitude
- Bike ID
- Plan Duration :
  - 1 day, 30 days, 365 days
- Trip Route Category :
  - One Way, Round Trip
- Passholder Type :
  - Walk-up, Monthly Pass, Flex Pass, Staff Annual
- 112427 rows

trip_id	duration	start_time	end_time	start_station	start_lat	start_lon	end_station	end_lat	end_lon	bike_id	plan_duration	trip_route_cat	passholder_type
32815764	18	2017.6.25 19:53	2017.6.25 20:11	3047	34.039982	-118.2664	3005	34.0485	-118.25854	4727	30	One Way	Monthly Pass
32821341	25	2017.6.25 20:35	2017.6.25 21:00	3005	34.0485	-118.25854	3020	34.031052	-118.26709	4727	30	One Way	Monthly Pass
31652471	10	2017.6.16 8:41	2017.6.16 8:51	3023	34.050911	-118.24097	3005	34.0485	-118.25854	4727	30	One Way	Monthly Pass
31700167	4	2017.6.16 17:39	2017.6.16 17:43	3005	34.0485	-118.25854	3051	34.045422	-118.25352	4727	30	One Way	Monthly Pass
31717085	17	2017.6.16 20:17	2017.6.16 20:34	3051	34.045422	-118.25352	3005	34.0485	-118.25854	4727	0	One Way	Walk-up

Data Preprocessing

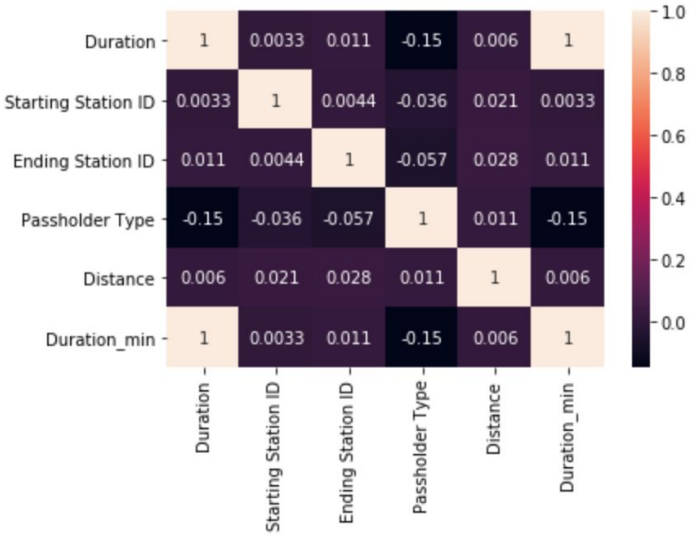
- Excluded... (112427 rows → 88266 rows)
  - 11177 rides with any NaN value
  - 3225 rides by LA Metro ‘Staff Annual’
  - 10459 rides of ‘Round-trips’
  - Station 4108 (Outlier)
- Category 0 (‘Walk-up’’) : 24241, Category 1 (‘Monthly Pass’ + ‘Flex Pass’) : 64025
- Remove non-relevant 9 columns
- Calculate the Manhattan distances in miles between starting stations and ending stations for each trip.
  - For cost estimation after the network optimization.

	Duration	Start Time	Starting Station ID	Ending Station ID	Passholder Type	Distance
5	780	2016-07-07T12:51:00	3021.0	3054.0	1	0.441488
6	600	2016-07-07T12:54:00	3022.0	3014.0	1	0.763951
7	600	2016-07-07T12:59:00	3076.0	3005.0	1	0.624177
9	960	2016-07-07T13:01:00	3031.0	3078.0	1	1.556036
10	960	2016-07-07T13:02:00	3031.0	3047.0	1	0.864494

Data Preprocessing

- Add Duration_min feature to the data for calculating the price for each trip.
- The current price policy is in the table below.
- The correlation between each factor are not significant.

Pass Type	Price		Category
1 Ride	$1.75/ 30min	every 30 min.	0
Monthly	$1.75/ 30min	first 30 min. free	1
Annual	$1.75/ 30min	first 30 min. free	1



EDA

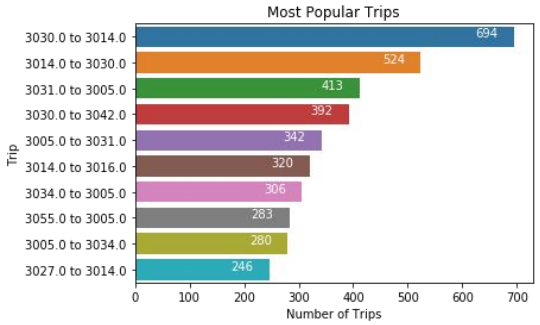
- Passholder Types according to each factor.

- Mean :

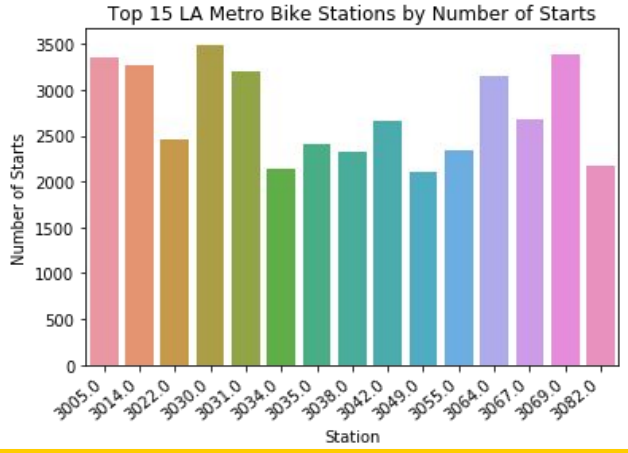
Passholder Type	Duration	Duration_min	Distance
0	2166.942783	36.115713	1.431571
1	771.725107	12.862085	5.347379

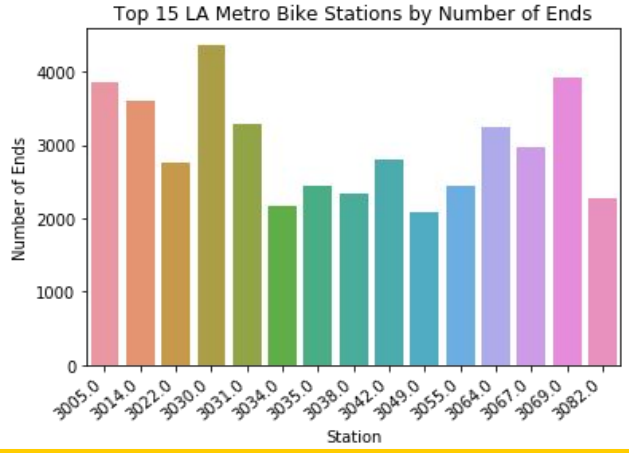
- Median :

Passholder Type	Duration	Duration_min	Distance
0	960	16.0	0.730822
1	480	8.0	0.612904



- The most frequent route : station 3030 to station 3014
- We can see there are imbalances in both starting stations and ending stations





Network Formulation - Supply and Demand

- For the relocation service, stations defined as
  - Supplying Stations : Incoming(Ending Trips) bikes > Outgoing(Starting Trips) bikes
  - Demanding Stations : Incoming bikes < Outgoing bikes

- Number of Supplying Stations and Demanding Stations

```
# Number of Supply stations and Demand stations
print(len(s_data), len(d_data))
```

```
36 27
```

- Ex) Supplying Stations (total 36) Demanding Stations (total 27)

Ending Station ID	Net
0	1012
1	143
3	107
5	191
6	587

Starting Station ID	Net
2	488
4	39
7	21
14	731
16	230

- Store the data into dictionaries. (key: Station ID, value: amount)
  - s_data_constraint = {0: 1012, 1: 143, 3: 107, …}
  - d_data_constraint = {2: 488, 4: 39, 7: 21, …}

Network Formulation - Route Price

- Assume we discount the full price of ‘Walk-up’ trips and reward the full price to ‘Passholders’
  - We only consider the mean duration time to get the ‘full Price’ for each path.

	Ending Station ID	Starting Station ID	Duration	Duration_min	Distance	Price
0	0	1	525.974026	8.766234	0.248361	1.75
1	0	2	1255.636364	20.927273	0.288265	1.75
2	0	3	1032.000000	17.200000	0.249927	1.75
3	0	4	1710.000000	28.500000	0.828233	1.75
4	0	5	2896.800000	48.280000	0.724004	3.50

- Store the full discount price into a dictionary. (key: route, value: price)
  - example : cost_constraint = {(0,1) : 1.75, (0,2) : 1.75, … }
- We need 972 (36*27) pairs but we only have 838 pairs ← 124 pairs with no trips in the original dataset.
  - For these pairs, we use big M method on the constraints to get 0 moves.
- Store the missing pairs into a dictionary. (key: route, value: big M=200)
  - missing_pairs = {(1,60) : 200, (3,7) : 200, (3,38) : 200, …}
- After deep-copy cost_constraint and update it with missing_pairs, we get 972 pairs.

Optimization Formulation

For each pair, parameters $Price[s, d]$ denote the `maximum discount dollar-amount` from the current price policy. $R_{discount} \in [0, 1]$ is defined as the `discount rate` for the 'Walk-ups' and $Price[s, d] \times R_{discount}$ is defined as the `reward dollar-amount` for 'Passholders.' Then, this becomes the company's cost for relocation service. The model solver will determine `amount of bikes to be relocated` over each pair, which will be represented as non-negative integer decision variables $x[s, d]$.

- The discount price for 'walk-ups' and the reward for 'passholders' are $Price[s, d] \times R_{discount}$
- By observing trade-off between discount rates and demands people will use Metro Bike, the realized demand rate is set as a function of the discount rate. Hence, the demand will become $Demand[d] \times f_{realized}(R_{discount})$
- The difference between realized demand and maximum demand (when discount rate is 1) is $(Demand[d] - x[s, d])$. This becomes cost as well.

Optimization Formulation

The problem objective is to minimize the total cost to all demanding stations from all supplying stations.

$$Minimize \sum_{s \in Supply} \sum_{d \in Demand} Price[s,d] \times R_{discount} \times x[s,d] + (price\_per\_mile) \times Distance[s,d] \times (Demand[d] - x[s,d])$$

$$subject\ to \sum_{d \in Demand} x[s,d] \leq \text{Supply}[s] \qquad \forall s \in Supply$$

$$\sum_{s \in Supply} x[s,d] = \text{Demand}[d] \times f_{realized}(R_{discount}) \qquad \forall d \in Demand$$

$$x[s,d]\ is\ integer.$$

> Relocation demands from all sources can not exceed the supplying capacity. Relocations to each station must satisfy their demand with respect to the discount rate.

Optimization Objective

- Find the optimal discount rate for the launch of relocation service.
- Find the routes (Starting Station → Ending Station) of relocation services and the number of discounted trips for each route.
- Find the range of operation cost which makes the relocation service profitable.
    - After the launch of relocation service, the operation cost can be fluctuate.

Network Assumption

- The discount dollar amount for non-passholders and reward dollar amount for passholders are the same. (for modeling simplicity)
- The optimal moves will always be met when the price is free to users (when the discount rate is 1).
- The discount rate for prices and the relocation demand percentage for each trip is the same.
    - When the price is discounted by 10%, 10% of people on each route will use the relocation service.

Network Optimization - Basic Scenario

- There are 62 paths when relocation service is free.
  - Discount rate = 1, all the demands are met.

- Each 62 paths are…

	S	D	Optimal Moves
4	0	16	71.0
6	0	18	271.0
13	0	35	337.0
14	0	36	109.0
16	0	38	1.0
18	0	40	206.0
19	0	44	17.0
32	1	17	95.0
35	1	20	32.0
37	1	29	16.0
58	3	16	107.0
81	5	2	191.0
131	6	52	587.0

etc ...

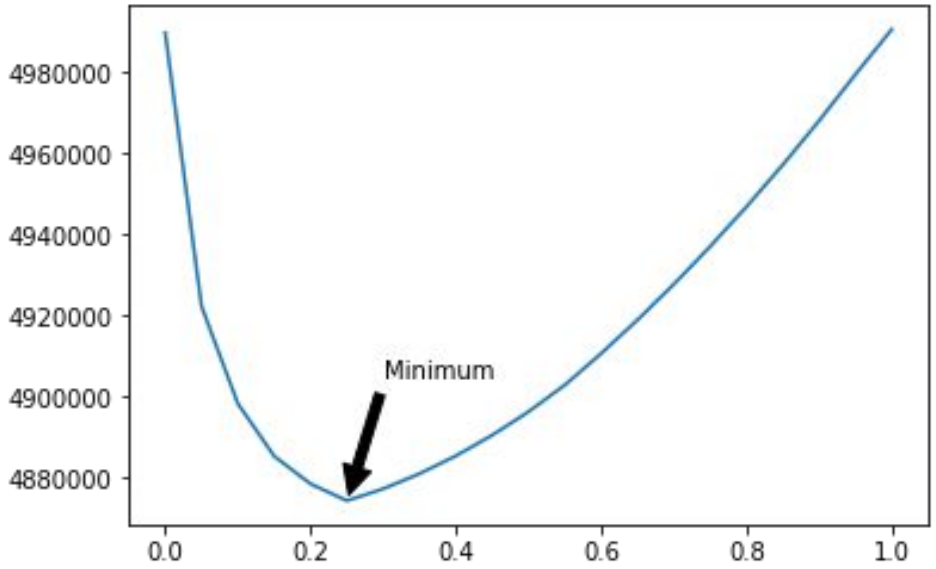
Network Optimization (1)

- **Optimal discount rate to minimize the total cost.**

➔ Used Pyomo for network optimization and solve the problem with gurobi solver.

➔ Found the realistic *price_per_mile* is $1/mile in the LA area.

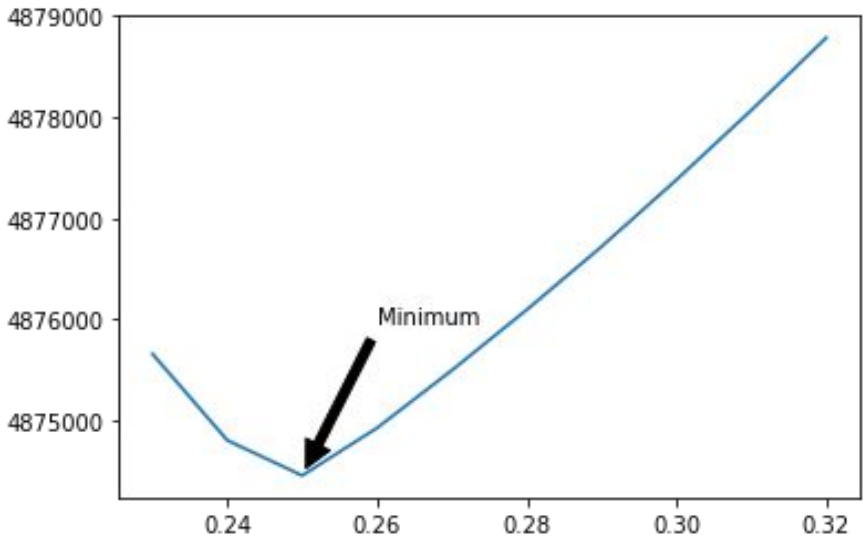
➔ Discount rate from 0% to 100%.



Network Optimization (1) - Detailed

- **Optimal discount rate to minimize the total cost.**

➔ Discount rate from 22% to 32%.



➔ We can conclude that the optimal discount price for minimizing the total cost is **25%**.

Network Optimization (1)

- **Optimal discount rate to minimize the total cost.**

➔ Set discount rate = 0.25

➔ We get the optimal solution of total cost, $4,061,567.47

➔ Total relocation movements are 7663.

➔ The optimal moves from supplying station to demanding station are stored in ‘optimal_sol.csv’

```
Solver:
- Status: ok
  Message: Gurobi 8.1.0\x3a optimal solution; objective 4061567.471694626
4; 93 simplex iterations
  Termination condition: optimal
  Id: 0
  Error rc: 0
  Time: 0.05458498001098633
"
```

```
# Store optimal solution into dataframe
optsol = []
for s in S_list:
    for d in D_list:
        optsol.append((s, d, model.x[s,d]()))
cols=['S','D','Optimal Moves']
optsol = pd.DataFrame(optsol,columns=cols)
optsol = optsol[optsol['Optimal Moves']!=0]
opts_list = optsol['S'].unique().tolist()
optd_list = optsol['D'].unique().tolist()
# optsol.to_csv('optimal_sol.csv')
optsol['Optimal Moves'].sum()
# Total movements for every route pairs.
```

```
7663.0
```

	S	D	Optimal Moves
3	0	14	103.0
5	0	17	117.0
7	0	19	15.0
8	0	20	131.0

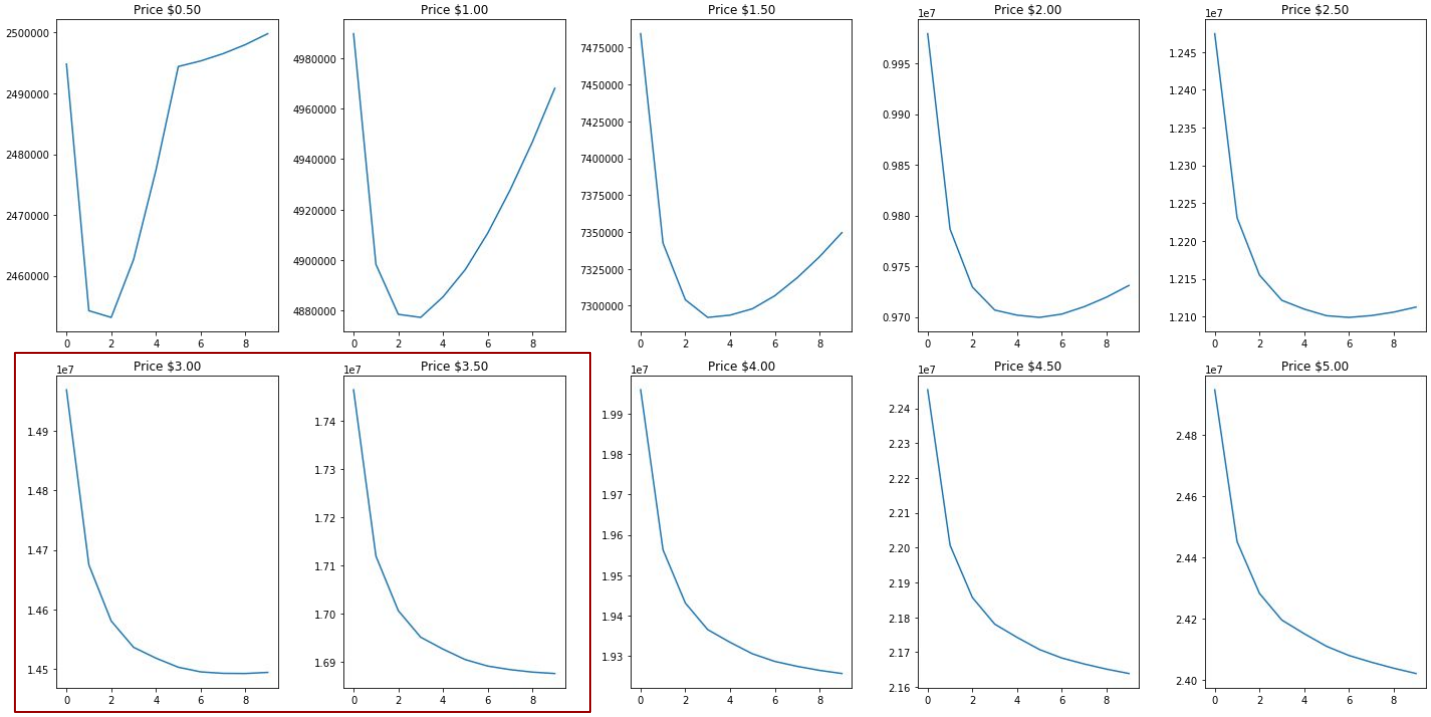
etc ...

Network Optimization (2)

- **Estimation of operation cost to determine relocation service feasibility. <Criteria on the need of providing relocation service at free price.>**

➔ Used Pyomo for network optimization and solve the problem with gurobi solver.

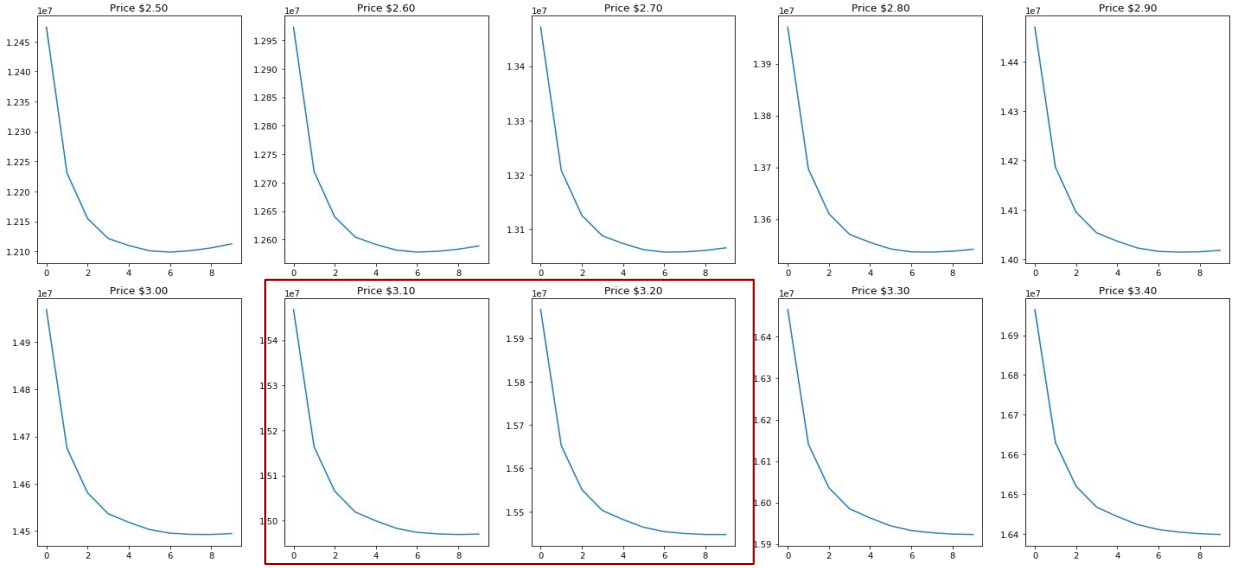
➔ Changed *price_per_mile* from $0.50 to $5.00



Network Optimization (2) - Detailed

- **Estimation of operation cost to determine relocation service feasibility. <Criteria on the need of providing relocation service at free price.>**

➔ Changed *price_per_mile* from $2.50 to $3.40



➔ Total cost decreases monotonically when *price_per_mile* at $3.20 and increases at last when the price is $3.10.

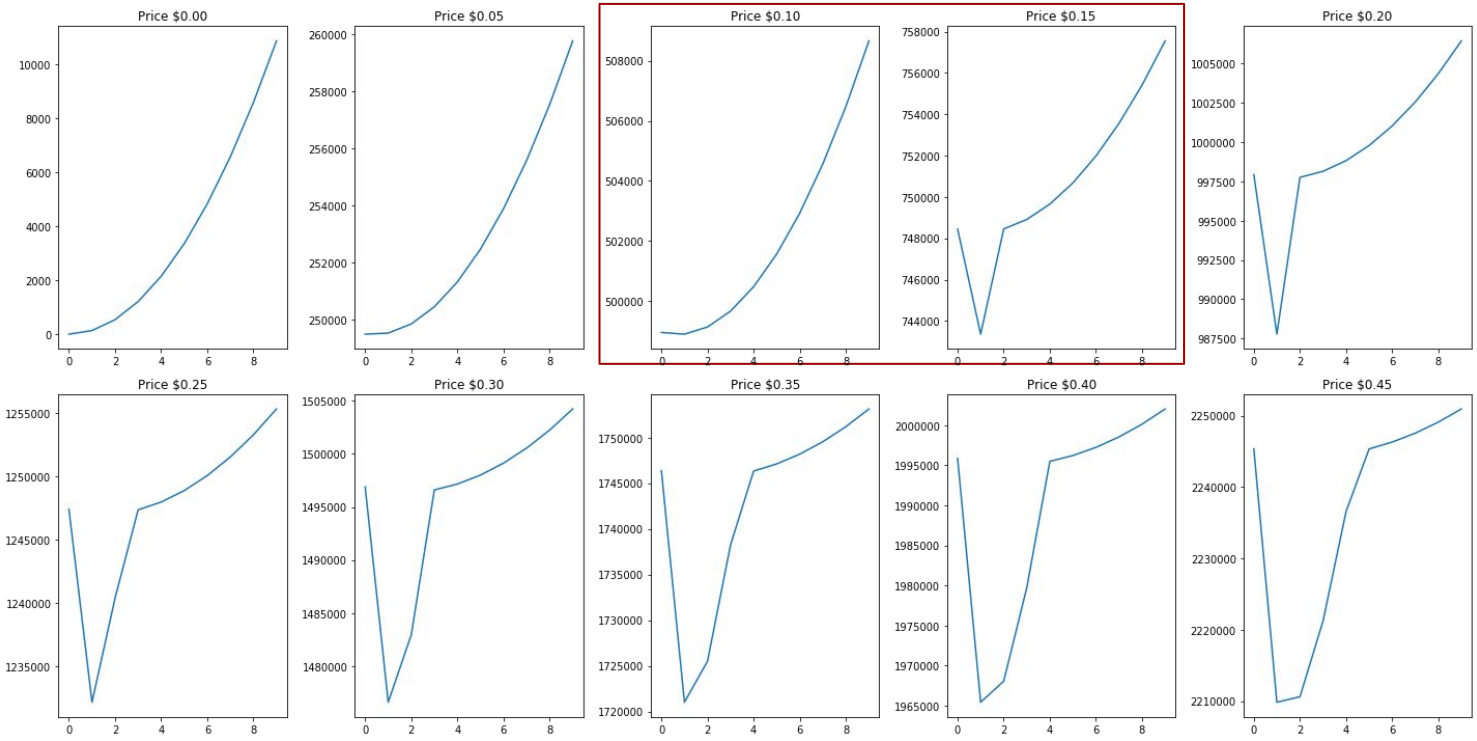
➔ We can observe : when *price_per_mile* is more than $3.10, it is better to launch the relocation service at free price.

Network Optimization (2)

- **Estimation of operation cost to determine relocation service feasibility. <Criteria on no need of relocation service.>**

➔ Used Pyomo for network optimization and solve the problem with gurobi solver.

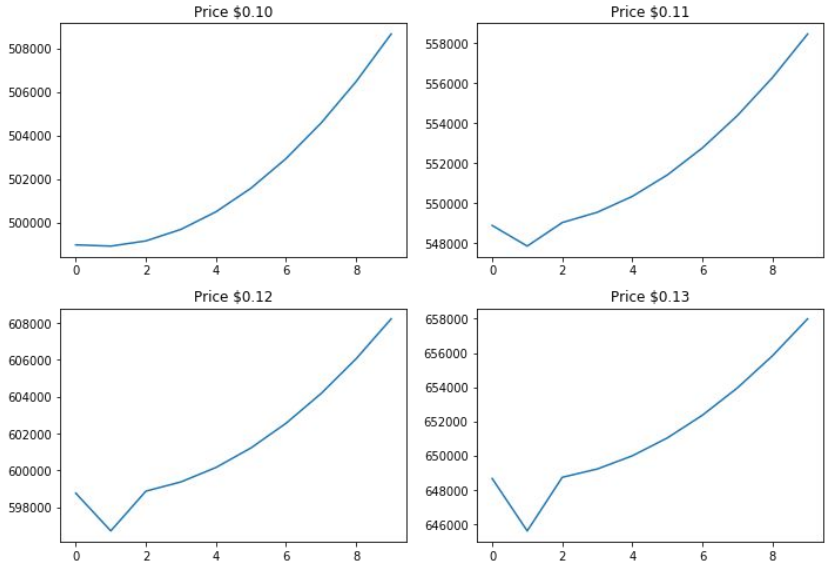
➔ Changed *price_per_mile* from $0.00 to $0.45



Network Optimization (2) - Detailed

- **Estimation of operation cost to determine relocation service feasibility. <Criteria on no need of relocation service.>**

➔ Changed *price_per_mile* from $0.10 to $0.15



➔ We can observe : when *price_per_mile* is less than $0.11, it is better not to launch the relocation service.

Conclusion

- The optimal discount rate for relocation service is **25%.**
- The optimal routes of relocation services and the number of discounted trips are stored in ‘optimal_sol.csv’.
- The relocation service is profitable when the operation cost is **between $0.11 and $3.10 per mile.**
  - If the operation cost is less than $0.11/mile, it is better not to launch relocation service.
  - If the operation cost is more than $3.10/mile, it is better to provide relocation service at free.

Detailed solution and codes :
https://github.com/yyul10/LA_metro_bike_relocation