Taxi scheduling system

Project report– ISYE 6669

Anqi Wei 903519110

Part 1 Problem description

Currently new mobility services like ride sharing have received widespread popularity to solve the last-mile problem. Since ridesharing has the ability to provide door-to-door on-demand services and tends to increase vehicle occupancy rate, its introduction into the transportation network could help alleviate the traffic congestions as well as reduce the energy consumptions and emissions. This project aims to design a vehicle scheduling system for a taxi company that performs trip selection and vehicle assignment to maximize the company's profits based on the ride sharing idea. It's assumed that the taxis owned by the company will strictly follow the assignment instructions sent from the scheduling system.

1.1 Roadway network

For a taxi company with K vehicles in operation, the taxi scheduling system will be employed to assign each taxi to passengers based on their travel demand. To solve the last-mile problem, the ridesharing service will serve as a feeder service to the public transit system, which means that these taxis will only provide trips for travelers to enter or exit the subway stations. Thus, the network will be designed as an undirected graph $G(V, E)$, where V is a set of n vertices representing the train stations and residential areas, while E represents the links in the roadway network with distance D_{ij} between any two vertices (i, j). To simplify the problem, it's assumed that the road capacity is large enough so there will be no traffic congestion, and the traveling speed of taxis is $s = 60MPH$ regardless of the number of vehicles on the roads. In that case, the direct travel time between any two vertices in the network is constant.

1.2 Ridesharing trip requests

Since the ridesharing service provided by the taxi company is on-demand, so there will be no reservation. When passengers decide to call for a taxi service, they will be asked to input their origins o_r and destinations d_r and the total number of q_r people that plans to travel along. Then a trip request will be generated at the system at the beginning of the next scheduling.

For each request r appeared in the system at the beginning of scheduling, it will have an attribute P_r indicating the assignment status. $P_r = 1$ means request r has already been assigned a taxi from the previous scheduling while $P_r = 0$ means request r hasn't been assigned any taxi yet. Each request also has an attribute A_r representing the index of taxi assigned to it, and $A_r = 0$ if it hasn't been assigned any taxi. These two attributes will be updated at the end of scheduling based on the taxi assignments and used as input data for the next scheduling.

Besides, each request will have an attribute t_r indicating the accumulated wait time since it has been generated in the system. At the end of scheduling, the system will update the accumulated time t_r if it hasn't been assigned any taxi. Passengers of each request will have a maximum wait time of w_r minutes, if the accumulated wait time exceeds the maximum, then passengers will be compensated by the company with a

discount on their payment. The discount ratio is α and it's also proportional to the exceeding time when the taxi arrives at its origin. Every request can be served at most once by only by one taxi, and once passengers has arrived at their destinations, the request will disappear from the system.

The pricing for the trip request consists of the fixed cost of f dollars and the traveling cost which is proportional to the direct travel distance between the passengers' origins and destinations, the unit cost is $g\ dollars/mile$.

1.3 Vehicle assignment process

The operating process of the scheduling system includes data collection and vehicle assignment. At the beginning of each time period, the system will first collect all the data associated with the travel requests appeared, including the basic trip information, the assignment status and the accumulated wait time. Then for the vehicle part, the system will collect the current location u_k, intended destination v_k of taxi k.

After inputting the required information, the system will assign available taxis to passengers based on their trip requests. Each taxi has a maximum capacity of Q_k passengers and for the assignment, available taxis are those that can pick up passengers, while satisfying the time constraints associated with the requests, and without exceeding their seat capacity. The and the number of passengers m_k onboard of taxi k is computed based on the previously assigned requests. Additionally, no request will be assigned to a taxi k if it hasn't arrived at the pickup point of the previously assigned requests. However, if it has been assigned with a new request, then the taxi should head towards the origin of the new assigned request to pick up the passengers first and then drive to the destination of pervious request. And each taxi will be assigned at most one new request at the beginning of scheduling and at most two requests in total at the same time. Suppose there is no time delay for passengers' pickup or drop-off at each origin or destination.

At the end of each scheduling, the intended destination will also be updated based on the chosen travel link of each taxi outside of the scheduling system. If the taxi k has no trip assignment, then its intended destination will be set to be the same as its current location. If taxi k hasn't reached to its intended destination outputted from last assignment, then v_k remain unchanged. Otherwise, it will be changed into the current or previously assigned requests' destinations, determined by a more specific rule, which will not be discussed in the project.

Considering the limited number of vehicles available, taxis don't have to respond to all the trip requests. However, if the request is not served within the acceptable wait time window, then the passengers will be compensated by the company and the amount is proportional to the product of the exceeding time and the original payment. For each taxi, the operating costs are proportional to the total distance it has traveled and the unit cost is $h\ dollars/mile$.

1.4 Objective of the scheduling system

The overall goal of the scheduling system is to assign travel requests to taxis and determine the service order of these requests to maximize the total profits, which is the offset between the passengers' payments and the sum of the operating costs plus a penalty associated with unserved requests.

Part 2 Model statement

2.1 Indices

$i, j = 1,2, \dots, n$ index the vertices in the roadway network
$r = 1,2, \dots, R$ index the trip requests
$k = 1,2, \dots, K$ index the taxis owned by the company

2.2 Data

(1) Network data

$D_{ij} =$ the direct travel distance between vertex i and vertex j
$T_{ij} =$ the direct travel time between vertex i and vertex j

(2) Trip request data

$o_r =$ the origin vertex of request r
$d_r =$ the destination vertex of request r
$P_r = \begin{cases} 0, & \text{if request } r \text{ hasn't been assigned to any taxi} \\ 1, & otherwise \end{cases}$
$A_r =$ the index of taxi assigned to request r and $A_r = 0$ if no taxi assigned
$t_r =$ the accumulated wait time of request r
$q_r =$ the total number of passengers traveling with of request r
$w_r =$ the maximum wait time since trip request r is processed in the scheduling
$f =$ the fixed cost of the trip
$g =$ the unit cost that passengers pay for their trip per mile
$\alpha =$ the discount ratio on the payment once the taxi has arrived at the origin exceeding the maximum wait time

(3) Vehicle data

$s = 60\ MPH = 1 mile/min$ the constant traveling speed of taxi
$u_k =$ the current location of taxi k
$v_k =$ the intended destination of taxi k
$h =$ the unit operating cost per mile taxi traveled
$Q_k =$ the maximum capacity of taxi k

2.3 Decision variables

$x_{rk} = \begin{cases} 1, & \text{if request } r \text{ is assigned to taxi } k \\ 0, & otherwise \end{cases}$

2.4 Auxiliary variables

c_r

$= \begin{cases} 1, & \text{if taxi arrives at the origin of request } r \text{ exceeding the maximum wait time} \\ 0, & otherwise \end{cases}$

$y_{ij}^k = \begin{cases} 1, & \text{if taxi } k \text{ departure from vertex } i \text{ and heads towards vertex } j \\ 0, & otherwise \end{cases}$

$B_{rk} =$

$\begin{cases} 0, & \text{if taxi } k \text{ hasn't arrived at the origin of the previously assigned request} \\ & \text{to pick up the passengers, which indicates } v_k - P_r \cdot o_r \cdot x_{rk} = 0 \\ 1, & otherwise \end{cases}$

$m_k =$ the number of passengers on board of taxi k at the beginning of scheduling

$z_{rk} = c_r \cdot x_{rk}$

$Pu_r =$ the updated assignment status of request r at the end of scheduling

$Au_r =$ the updated index of taxi assigned to request r at the end of scheduling

$tu_r =$ the updated accumulated wait time of request r at the end of scheduling

2.5 Objective function

The overall objective function consists of three parts:

(1) The total payment received from passengers:

$$\sum_{r=1}^{R} \sum_{k=1}^{K} (f + D_{o_r,d_r} \cdot g) \cdot x_{rk}$$

(2) The total operating costs:

$$\sum_{k=1}^{K} \sum_{i=1}^{n} \sum_{j=1}^{n} y_{ij}^k \cdot D_{ij} \cdot h$$

(3) The penalties related to the delayed requests:

$$\sum_{r=1}^{R} \alpha \cdot c_r \cdot (t_r + \sum_{k=1}^{K} T_{u_{k,}o_r} \cdot x_{rk} - w_r) \cdot (f + D_{o_r,d_r} \cdot g)$$

To linearize the $\sum_{k=1}^{K} T_{u_{k,}o_r} \cdot x_{rk} \cdot c_r$, add another auxiliary variable $z_{rk} = c_r \cdot x_{rk}$,

so the above equation is changed to $\alpha \cdot \sum_{r=1}^{R} c_r \cdot (t_r - w_r) \cdot (f + D_{o_r,d_r} \cdot g) + \alpha \cdot$

$\sum_{k=1}^{K} T_{u_{k,}o_r} \cdot z_{rk}$

To maximize the total profits:

$$\max \quad \sum_{r=1}^{R} \sum_{k=1}^{K} \left(f + D_{o_r,d_r} \cdot g\right) \cdot x_{rk} - \sum_{k=1}^{K} \sum_{i=1}^{n} \sum_{j=1}^{n} y_{ij}^k \cdot D_{ij} \cdot h$$
$$-\alpha \cdot (\sum_{r=1}^{R} c_r \cdot (t_r - w_r) \cdot \left(f + D_{o_r,d_r} \cdot g\right) + \sum_{k=1}^{K} T_{u_{k,}o_r} \cdot z_{rk})$$

2.6 Constraints

(1) At most one new request can be assigned to taxi k at each scheduling and each taxi will have at most 2 requests at the same time

$$\sum_{r=1}^{R}(x_{rk}\cdot(1-P_r))\le 1,\qquad \forall k\in\{1,2,\dots,K\}$$

$$\sum_{r=1}^{R}x_{rk}\le 2,\qquad \forall k\in\{1,2,\dots,K\}$$

(2) Each request r can be served at most once by only one taxi

$$\sum_{k=1}^{K}x_{rk}\le 1,\qquad \forall r\in\{1,2,\dots,R\}$$

(3) If a request is already been previously assigned with taxi k, then the corresponding $x_{rk}=1$

$$M_1\cdot x_{r,A_r}\ge A_r,\qquad \forall r\in\{1,2,\dots,R\}$$

Where M_1 is a large number and $M_1\ge k$

(4) The direct travel time between any two vertices (a,b) equals the direct distance divided by the traveling speed v

$$T_{ij}=D_{ij}/s,\qquad \forall(i,j)\in[n]\times[n]$$

(5) Compute the number of passengers on board of taxi k at the beginning of scheduling

$$m_k=\sum_{r=1}^{R}q_r\cdot P_r\cdot x_{rk},\quad \forall k\in\{1,2,\dots,K\}$$

(6) Only if the total of the passengers planning to take the taxi of request r and the passengers already on board of taxi k don't exceed the maximum capacity will the request r be assigned to the taxi

$$M_2\cdot(1-(1-P_r)\cdot x_{rk})\ge m_k+q_r-Q_k,\quad \forall r\in\{1,2,\dots,R\},k\in\{1,2,\dots,K\}$$

Where M_2 is a large number and $M_2\ge\max(m_k+q_r-Q_k)$

(7) If no new trip request has been assigned to taxi k, then the destination v_k remain unchanged and set $y^k_{u_k,v_k}=1$, else if there is a new trip request r has been assigned to taxi k, then set $y^k_{u_k,o_r}=1$.

For $\forall r\in\{1,2,\dots,R\},k\in\{1,2,\dots,K\}$:

$$1-y^k_{u_k,v_k}\le\sum_{r=1}^{R}(x_{rk}\cdot(1-P_r))$$

$$y^k_{u_k,o_r}\ge(1-P_r)\cdot x_{rk}$$

(8) Each taxi k only takes one route during scheduling

$$\sum_{i=1}^{n} y_{u_k,i}^{k} = 1, \qquad \forall k \in \{1,2,\dots,K\}$$

(9) Taxi k will not be assigned a new request if it hasn't arrived at the pickup point of the previously assigned requests

For $\forall r \in \{1,2,\dots,R\}, k \in \{1,2,\dots,K\}$:

$$(v_k - P_r \cdot o_r \cdot x_{rk}) + M_3 \cdot B_{rk} \geq \sum_{r=1}^{R} x_{rk} \cdot (1 - P_r)$$

$$-(v_k - P_r \cdot o_r \cdot x_{rk}) + M_3 \cdot (1 - B_{rk}) \geq \sum_{r=1}^{R} x_{rk} \cdot (1 - P_r)$$

Where M_3 is a large number and $M_3 \geq$ n

(10) The definition of B_{rk}

For $\forall r \in \{1,2,\dots,R\}, k \in \{1,2,\dots,K\}$:

$$0 \leq (v_k - P_r \cdot o_r \cdot x_{rk}) + M_4 \cdot B_{rk} \leq M_4$$

Where M_4 is a large number and $M_4 \geq$ n

(11) Update the assignment status of request r

$$Pu_r = \sum_{k=1}^{K} x_{rk}, \qquad \forall r \in \{1,2,\dots,R\}$$

(12) Update the index of taxi assigned to request r

$$Au_r = \sum_{k=1}^{K} (x_{rk} \cdot k), \qquad \forall r \in \{1,2,\dots,R\}$$

(13) Update the accumulated wait time of request r

$$tu_r = t_r + T_0 \cdot \left(1 - \sum_{k=1}^{K} x_{rk}\right), \qquad \forall r \in \{1,2,\dots,R\}$$

Where $T_0 = 2min$ is the interval of scheduling.

(14) If assigned taxi k arrives at the origin of request r exceeding its maximum wait time, then the request will receive a compensate and set $c_r = 1$

$$0 \leq -(t_r + \sum_{k=1}^{K} T_{u_{k,}o_r} \cdot x_{rk} - w_r) + M_5 \cdot c_r \leq M_5, \qquad \forall r \in \{1,2,\dots,R\}$$

Where$D_0 = \min(D_{ij})$ and M_5 is a large number and $M_5 \geq \max|t_r - w_r|$.

(15) Auxiliary variable z_{rk}

For $\forall r \in \{1,2,\dots,R\}, k \in \{1,2,\dots,K\}$:

$$z_{rk} \leq c_r$$

$$z_{rk} \leq x_{rk}$$

$$z_{rk} \geq x_{rk} + c_r - 1$$

(16) Variable constraints

For $\forall r \in \{1,2,\dots,R\}, k \in \{1,2,\dots,K\}$:

$$x_{rk} \text{ binary variable}$$
$$c_r \text{ binary variable}$$
$$B_{rk} \text{ binary variable}$$
$$z_{rk} \text{ binary variable}$$
$$Pu_r \text{ binary variable}$$
$$Au_r \geq 0,\ tu_r \geq 0,\ m_k \geq 0, \text{ integer variable}$$

For $\forall (i,j) \in [n] \times [n], k \in [K]$:

$$y_{ij}^k \text{ binary variable}$$

Part 3 Model validation

To validate the model constructed in part 2, some numerical cases were tested to see if the optimal solutions satisfy all the constraints. A basic small-scale case was first presented in 3.1, along with some modifications of the parameters to see their influence on the optimal solutions. Then for the relatively large-scale cases, some parameters were set to their extreme values and tested in 3.2. Additionally, the situations with infeasible solution were also discussed in 3.3. Finally, the relationship between the number of taxis in operation and the overall profits were analyzed in 3.4 to verify the correctness of the model. All the test code can be found in the attached python files.

3.1 Basic small-scale case

Construct a roadway network G with 4 vertices and 6 links, as is demonstrated in figure 1. Suppose the company owns two taxis, and the unit operating cost $h = \$1/mile$. For each trip request, passengers have to pay a fixed cost $f = \$3$ and the unit cost $g = \$2/mile$. The ratio of the penalty for unserved trip requests $\alpha = 0.5$. At the beginning of scheduling, three trip requests appeared in the system. The collected information associated with the trip requests and taxis from the last scheduling is shown in table 1,2.

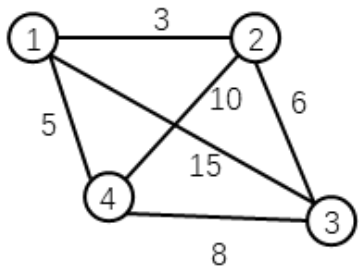

Figure 1 Constructed roadway network

Table 1 Trip requests' attribute data

Trip request r	Origin o_r	Destination d_r	Number of passengers q_r	Assignment status P_r	Assigned taxi A_r	Accumulated wait time t_r	Maximum wait time w_r
1	1	3	1	1	1	1	10
2	2	4	1	0	0	1	5
3	4	1	2	0	0	0	10

Table 2 Taxis' attribute data

Taxi k	Current location u_k	Intended destination v_k	Maximum capacity Q_k
1	2	3	4
2	3	3	4

Based on the given information, run the model in Gurobi to find the optimal solutions. The obtained assignment result is shown in table 3:

Table 3 Optimal solution obtained from Gurobi

x_{rk}	1	2	Maximum profits
1	1	0	\$171.5
2	0	1	
3	1	0	

Where $x_{rk} = 1$ indicating the trip request r is assigned to taxi k.

Based on the results, check whether all the constraints are satisfied:

- **Number of requests assigned:**

$x_{11} = x_{31} = x_{22} = 1$ shows that the apart from the previously assigned request 1, taxi 1 is assigned a new request 3 and taxi 2 is assigned a new request 2 at this time of scheduling. The assignment satisfies the constraints that at least one new request will be assigned to an available taxi and each request will be assigned at most once with only one taxi.

- **Taxi capacity:**

The basic model set the maximum capacity of each taxi to be 4, and though taxi 1 has already carried request 1 of 1 passenger, since request 2, 3 has 1, 2 passengers respectively, so the assignment won't break the capacity constraint.

If change the number of passengers of request 2 to be 5, then rerun the model and the obtained result $x_{11} = x_{31} = 1$ with the maximum profit being \$78.5. Or if change the taxi maximum capacity to 1, then the corresponding results have also changed into $x_{11} = x_{22} = 1$ with the maximum profit being \$110.

- **The request attribute c_r**

Check the time constraint for each request, as the accumulated wait time for each request is $t_1 = t_2 = 1, t_3 = 0$, and the traveling time for taxi 1 to reach request 3 and taxi 2 to reach request 2 is 6 and 10 minutes respectively, the arrival time has exceeded the maximum wait time of both the request 2 and 3. Thus, as indicated by the obtained result $c_1 = 0,\ c_2 = c_3 = 1$, the constraints are proven to be feasible.

- **No new request will be assigned to taxi if it hasn't picked up the passengers from previous assigned requests:**

As only request 1 has been previously assigned to taxi 1 and it has already picked up the passengers and were heading towards the destination of request 1, so as long as it could satisfy the capacity constraint, it will be available to be assigned a new request. However, if we change the current location of taxi 1 to 4 and the intended destination to $o_1 = 1$, which indicates that it's still traveling to pick up the passengers of request 1

assigned from the last scheduling, then after rerunning the model, the obtained results are $x_{11} = 1, x_{22} = 1$ with the maximum profit being \$111. Compared with the previous setting, request 3 is not assigned to any taxi at the current scheduling since taxi 1 is still heading to pick up request 1.

- **Route selection of taxi k**

The results $y_{24}^1 = 1$ and $y_{32}^2 = 1$ indicated that at the end of this scheduling, taxi 1 will be heading towards vertex 4 to pick up passengers of request 3 and taxi 2 will be heading for vertex 2, which is the origin of request 2. To further check the constraints related to y_{ij}^k, also change the current location of taxi 1 to 4 and the intended destination to $o_1 = 1$, and the obtained results $y_{32}^2 = 1$ and $y_{41}^1 = 1$ indicating that taxi 2 will be traveling to vertex 2 to service request 2 and taxi 1 will still be traveling to vertex 1 to pick up request 1.

From the tests obtained above, we could see that the obtained optimal solution as well as the model constraints are feasible.

While during the validation, a problem was noticed that the model at first cannot guarantee the uniformity of taxis assigned to each request. Since the current scheduling is based on the conditions from last assignment, so fail to ensure the previously assigned request still being serviced by the same taxi will lead to certain issues. To solve that problem, a new attribute of the trip request A_r was added indicating the index of its assigned taxi, and $A_r = 0$ if it hasn't been assigned any taxi. Then for the next scheduling, simply set the $x_{r,A_r} = 1$ for those previously served requests will ensure the uniformity of their assigned taxis. Since A_r could be 0, so there will be extra variables $x_{r,0}$ generated, but these variables have no practical meanings.

3.2 Extreme value cases

To test whether the model could work well with large-scale cases, construct a network with 100 vertices and the distance between each two vertices is generated randomly within the range of [3,75] miles. Suppose at the beginning of scheduling, there are 100 requests newly generated at the system with their origins and destinations randomly allocated in the network. Meanwhile, there is only one taxi in operation and it's located at vertex 50 with no assignment.

(1) Zero operating costs case

To test the model, it's assumed that all the trips haven't been assigned to any taxis, their accumulated wait time is 0 and their maximum wait time is set to be extremely large so there will be no penalty situations. For the trip requests, the fixed cost, unit traveling cost $f = \$3/mile, g = \$2/mile$ and set the unit operating cost $h = 0$. In that case, to maximize the objective function, the system will only be looking for the request with the longest direct traveling distance between its origin and destination, which is $\max D_{o_r,d_r}$.

Input the trip and taxi information to the model, the obtained results indicated that $x_{191} = 1$ while the other $x_{rk} = 0$ with the optimized objective value being \$151. The system assigned taxi to serve request 19 with its origin at vertex 15 and destination at vertex 92. To check validity, find the longest OD distance among all the requests and it appeared to be request 19 with $D_{15,92} = 74\,mile$. Therefore, the optimal solution obtained from the model was correct.

(2) Zero payment case

Again assume there is no previously assigned requests and no penalty for arriving late at the pickup locations of trip requests. For the trip requests, set both the fixed cost and unit travel cost $f = g = 0$ but the unit operating cost for taxi is $h = \$1/mile$. Then as both the income and penalty amount are constant, and there is no requirement that the requests generated at the system have to be serviced, so to maximize the total profits, we can deduce that the system will not assign the taxi to any request.

To validate the assumption, rerun the model and the obtained results indicated that $x_{rk} = 0$ for any request or taxi, which implied that the model is feasible.

The code file could be found in "p3-Extreme test case -1, 2.py" file.

3.3 No feasible solution cases

Since the input of the scheduling system depends on the output from last assignment, therefore while using the random data generator to define the requests and taxis, it can easily generate some input data with contradiction to each other, like the assignment status and assigned taxi index of each trip request, and cause no feasible solution situations. Besides, as the model use the output data from the last assignment as an input, so no feasible solution situation will occur only if there are no previously assigned requests in the system, because otherwise, $x_{r,A_r} = 1$ would be those feasible solutions. Thus, it's assumed that all trip requests generated at the system haven't been assigned to any taxi yet.

One situation where there will be no feasible solution is to simply set the number of passengers planning to travel along with for each request to be larger than the maximum capacity of taxis. While the model allows no trip assignment, so another constraint $\sum_{k=1}^{K}\sum_{r=1}^{R} x_{rk} \geq R$ that require all the requests be serviced will make the infeasible situation occurred.

3.4 Qualitative behavior of the solutions

In order to further validate the constructed model with feasible solutions, a network with 100 vertices was created. There are in total of 100 requests appeared in the system. To analyze the potential influence of the number of taxis in operation on the total profits gained, the models with a range from 5 to 90 taxis were tested. This could also help the company decide how many taxis are suitable to purchase to gain the maximum profits. Since in this model, the initial capital costs of taxis are not considered in the overall profits, so it can be deduced that as more taxis being put into operation, more trip requests will be matched an more profits will be obtained.

For the large-scale cases, I simplified the initial setting that assumes all requests are newly generated at the system without previous taxi assignment. Then run the model and check the potential influences of increasing the number of taxis in operation. The corresponding results were demonstrated in figure 2.

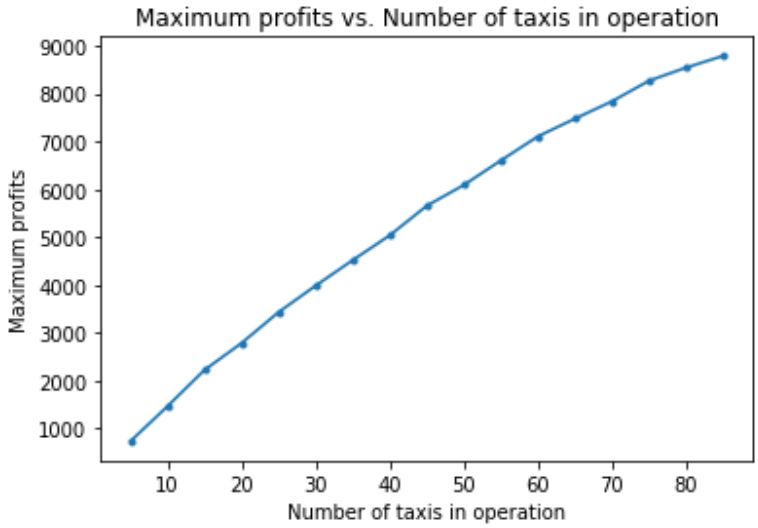

Figure 2 Objective value vs. number of taxis in operation

From the above scatter plot, it can be seen that as the number of taxis put into the market increases, the overall profits earned will increase accordingly, since more requests will be satisfied. The code could be found in “p3-qualitative test.py” file.

Part 4 Model improvement

This section will attempt to improve the runtime performance of a relatively large-scale model using different methods. A baseline case was provided in 4.1. Then the influences of the modification to the optimizer parameter values as well as the constraints on the model computation time are analyzed in 4.2 and 4.3. Moreover, an initial feasible solution based on a simple heuristic is given at the start of the model to see the potential improvement in the model runtime.

At first I tried constructing a network of 200 vertices with 100 taxis in operations. Then for the 800 trip requests, the original model could solve the optimization in 1496.57s, and through the modification in 4.3, the solution runtime was reduced to 1267.09s. However, when I tried parameters tuning, the tuning process kept indicating the optimization exhausted available memory. Thus, I had to change the initial setting from 800 trip requests to 500 and then try different improving methods.

4.1 Baseline case

For the baseline case, construct a network with 200 vertices each connected. The distance between any two vertices are randomly selected from 3 to 50 miles. 500 trip requests were generated at the beginning of the scheduling with the number of passengers varying from 1 to 4. For each trip request, the accumulated wait time also ranges from 0 to 3 minutes and the maximum wait time ranges from 5 to 15 minutes. There are 100 taxis traveling in the network, the maximum capacity is set to 4. In terms of the costs, set parameters $f = \$3,\ g = \$2/mile,\ h = \$1/mile$ and $\alpha = 0.5$.

Run the model and the output log is shown in figure 3. The optimal objective value is \$55,252, and the run time is 194.05s.

```
Root simplex log...

Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0    1.2186500e+04   1.520000e+02   2.643250e+08    189s
   17374    5.5252000e+04   0.000000e+00   0.000000e+00    190s
   17374    5.5252000e+04   0.000000e+00   0.000000e+00    190s
Concurrent spin time: 0.19s

Solved with primal simplex

Root relaxation: objective 5.525200e+04, 17374 iterations, 1.12 seconds

    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Time

*    0     0               0    55252.000000 55252.0000  0.00%     -  190s

Explored 0 nodes (17374 simplex iterations) in 194.05 seconds
Thread count was 8 (of 8 available processors)

Solution count 2: 55252 21963

Optimal solution found (tolerance 1.00e-04)
Best objective 5.525200000000e+04, best bound 5.525200000000e+04, gap 0.0000%
Optimal Objective Value 55252.0
```

Figure 3 logging of baseline case

4.2 Optimizer parameters tuning

To reduce the model solution time, first try modifying the default parameter settings of the optimizer. Use the parameter tuning tool in Gurobi to perform multiple runs for each parameter, and then find the settings with the best performance. The summary of tried parameter sets is shown in figure 4.

```
Tested 10 parameter sets in 5677.86s

Baseline parameter set: mean runtime 189.08s

Improved parameter set 1 (mean runtime 164.40s):

        Heuristics 0
        PrePasses 1
```

Figure 4 The summary of parameter tuning

For a MIP model as in this project, several baseline runs were performed and the mean runtime is 189.08s. Then based on the tuning result, the best parameter settings would be changing the value of parameter Heuristics and PrePasses to be 0 and 1 respectively, which means spending 0 time on feasibility heuristics to accelerate the progress in the best bound.

Additionally, the log file implies that presolve consumes most of the runtime (177.68s/ 194.05s), thus in order to improve the runtime performance, limit the number of passes presolve performs to be 1 to reduce the presolve runtime. The improved parameter setting will decrease the mean runtime by around 25s. The code could be found in 'p4-tuning.py' file and the corresponding parameter file could be found in "tune0.prm".

4.3 Constraint modification

Apart from modifying the optimizer parameter setting, another method is to consider modifying the constraints. During the process of stating the model, all the constraints and the objective function have been managed to be linearized. While previously the operation of ride sharing was designed in a way that if the taxi arrives at the origin of the assigned request while exceeds the requests' maximum wait time, then a penalty will be incurred. To tighten the constraint, change the rule to allow the taxi be assigned to a request only if its arrival time doesn't exceed the maximum wait time, and if after the maximum wait time the request is still not served, passengers will then cancel the request.
So based on the previous model, add the constraint below to ensure that only if the arrival time of taxi k at the origin of request r don't exceed the maximum wait time will request r be assigned to the taxi.

$$M_6 \cdot (1 - (1 - P_r) \cdot x_{rk}) \geq t_r + T_{u_k,o_r} - w_r, \quad \forall r \in \{1,2,\dots,R\}, k \in \{1,2,\dots,K\}$$

Where M_6 is a large number and $M_6 \geq \max(t_r + T_{u_k,o_r} - w_r)$

With the other setting remain unchanged, rerun the model and the output indicated that the runtime has been reduced to 120.52s, as is shown in figure 5 The code could be found in 'p4-constraint modification.py'.

```
Root simplex log...

Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0    1.3853350e+05   1.294887e+03   0.000000e+00    117s
    2963    5.1881669e+04   0.000000e+00   0.000000e+00    117s

Root relaxation: objective 5.188167e+04, 2963 iterations, 0.08 seconds

    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Time

     0     0 51881.6692    0   12 23762.0000 51881.6692   118%     -  117s
H    0     0                     51400.000000 51881.6692  0.94%     -  117s
H    0     0                     51791.000000 51881.6692  0.18%     -  117s
H    0     0                     51800.000000 51875.9084  0.15%     -  117s

Cutting planes:
  Cover: 2
  Implied bound: 2
  RLT: 4

Explored 1 nodes (3107 simplex iterations) in 120.52 seconds
Thread count was 8 (of 8 available processors)

Solution count 5: 51800 51791 51400 ... 13416

Optimal solution found (tolerance 1.00e-04)
Best objective 5.180000000000e+04, best bound 5.180000000000e+04, gap 0.0000%
Optimal Objective Value 51800.0
```

Figure 5 logging of constraint modification case

4.4 Feasible solution given at the start

As MIP solver could spend a lot of time finding an initial feasible solution, I tried using a simple heuristic to choose an initial feasible solution. Since the discount ratio for arriving late $\alpha = 0.5$, and the penalty is proportional to the exceeding time and the trip requests' distance. While the passenger payment is also proportional to the trip's distance, thus, for the requests with large accumulated wait time, it's highly possible that the company would lose profits. In this case, for the initial starting solution, it's assumed that if the accumulated wait time of the trip request is larger than 2 minutes, then it won't be assigned any taxi by the system. Run the model along with a solution, the results indicate that the run time is changed into 322.67s, which is even larger than the original one.

Then I changed a different heuristic that for each taxi, the system will assign them the nearest trip requests. The test results still indicated no benefits will be gained in terms of the runtime. The details could be found in 'p4-initial solution given.py'.

4.5 Reformulate the model

In this project, the assumed reschedule interval is 5 minutes. For a network with 200 stops and 100 taxis in operation, the system could handle 500 trip requests in 194.05s, which is applicable since it takes less than 5 minutes to obtain the optimal assignments. While in terms of 800 trip requests, the original model would take 1496.57s to compute the objective value, exceeding the 5-minute interval, and even the modified model would also take 1267.09s. In that case, some other formulations should be considered.

As the interval between each consecutive rescheduling is relatively short, so the taxi scheduling problem could be considered as a static optimization problem. The static share-a-ride problem has been investigated by many studies, and based on the idea proposed by Herbawi et al. (2012) [1], the original model could be reformulated.

Suppose for an undirected network, there are R trip requests and K taxis. For each trip request i, its origin and destination will be seen as vertex $i \in O_r$ and $i + R \in D_r$ respectively. And instead of using the actual network stops to represent taxis' current locations and intended destinations, taxis are considered to have the same origin and destination depots denoted as 0 and $2R + 1$, which are separated from requests' origins and destinations. Set $V = O_i \cup D_i \cup \{u\} \cup \{v\}$. In that case, the total number of vertices in the network will be $(2 * R + 2)$. Every origins and destinations of requests and taxis are seen as a unique vertex, even though they might be the same vertices in the real network.

Every vertex in the network is associated with a load q_i such that $q_i = -q_{i+R}$ and $q_0 = q_{2*R+1} = 0$. Let t_i^k represents the time that taxi k arrives at vertex i and w_i^k be the load of taxi k after servicing vertex i. Each request has a maximum wait time m_i and for the decision variable, the new model use binary variable $x_{ij}^k = 1$ representing that taxi k travels from vertex i directly to j. Based on these notations, the new formulations are shown below:

(1) At most one new request can be assigned to taxi k at each scheduling

$$\sum_{i \in O_r} (x_{0,i}^k \cdot (1 - P_i)) \leq 1, \qquad \forall k \in \{1,2, \dots, K\}$$

(2) Each request r can be served at most once by only one taxi

$$\sum_{k \in K} \sum_{j \in V} x_{i,j}^k \leq 1, \qquad \forall i \in O_r$$

$$\sum_{i \in V} x_{i,j}^k = \sum_{i \in V} x_{i,j+R}^k, \qquad \forall j \in O_r$$

(3) Define the origin and destination depots of each taxi

$$\sum_{i \in V} x_{0,i}^k = \sum_{i \in V} x_{i,2R+1}^k = 1, \qquad \forall k \in \{1,2, \dots, K\}$$

$$\sum_{i \in V} x_{i,0}^k = \sum_{i \in V} x_{2R+1,i}^k = 0, \qquad \forall k \in \{1,2, \dots, K\}$$

(4) The direct travel time between any two vertices (a, b) equals the direct distance divided by the traveling speed v

$$T_{ij} = \frac{D_{ij}}{s}, \qquad \forall (i,j) \in V \times V$$

(5) Every stop has a proceeding and succeeding vertex except vertices 0 and 2R+1

$$\sum_{j\in V} x_{i,j}^k = \sum_{j\in V} x_{j,i}^k, \qquad \forall k \in \{1,2,\dots,K\}, i \in O_r \cup D_r$$

(6) Compute the arrival time and loads of each taxi after servicing vertex i

$$t_j^k - t_i^k \geq t_{ij} + M_1 \cdot (x_{ij}^k - 1)$$

$$w_j^k - w_i^k \geq q_i + M_2 \cdot (x_{ij}^k - 1)$$

Where M_1, M_2 is a large number.

(7) Capacity constraints of each taxi

For $\forall i \in V, k \in \{1,2,\dots,K\}$:

$$w_i^k \geq 0, \qquad w_i^k \geq q_i$$
$$w_i^k \leq Q_k, \qquad w_i^k \leq Q_k + q_i$$

(8) If assigned taxi k arrives at the origin of request i exceeding its maximum wait time, then the request will receive a compensate and set $c_i = 1$

$$0 \leq -(\sum_{k\in K} t_i^k - w_i) + M_3 \cdot c_i \leq M_3, \qquad \forall i \in O_r$$

Where M_3 is a large number and $M_3 \geq \max \left| t_i^k - w_i \right|$.

(9) Variable constraints

For $\forall i \in V, k \in \{1,2,\dots,K\}$:

$$x_{i,j}^k \text{ binary variable}$$

$$c_i \text{ binary variable}$$
$$w_i^k \geq 0, \text{ integer variable}$$
$$t_i^k \geq 0$$

For $\forall i \in O_r$:

$$m_i \geq 0$$

Then run the model on the base case setting to see its potential of improving the runtime. However, the new model took extremely long time in the root relaxation process, even changing the parameter 'Method' to 2 or 3 won't bring any benefits to the solving time. The corresponding code could be found in 'p4-methods.py' file.

Part 5 Future applications in real data

The data inputted in the project is either fabricated or using random generator to create. For the real-world application, the data to run the model would consist of three parts, namely the road network data, the passengers' trip requests data and the taxis' vehicle data. The road network data can be easily acquired online using map tools like Google Map or ArcGIS. The passengers' trip request data including the origins, destinations and travel time could refer to the travel and activity survey conducted by each state's DOT. In addition, the trip data as well as the vehicle data could both be acquired from any ride sharing company.

Since the scale of real-world data could be extremely huge, especially when using the real-time data, thus in order to make the scheduling system model applicable, more improvements should be made to reduce the solution runtime and storage space. As the key to make the static ride sharing problem closer to the dynamic ones is to make the reschedule interval relatively small enough so that the input information is not outdated, which also asks for the optimization to be completed within a small amount of time.

References:

[1] Herbawi, Wesam, and Michael Weber. "The ridematching problem with time windows in dynamic ridesharing: A model and a genetic algorithm." 2012 IEEE Congress on Evolutionary Computation. IEEE, 2012.