COLUMBIA | ENGINEERING
The Fu Foundation School of Engineering and Applied Science

Understanding the Traffic Pattern in NYC and Taxi Trip Pairing

Group Members

Margaret Bellon (mmb2285)

Congying Qiu (cq2192)

Jiawei Ma (jm4431)

Jiaqi Zhou (jz2787)

Liutong Zhou (lz2484)

CIEN E4011 Infrastructure Systems Optimization

Department of Civil Engineering & Engineering Mechanics

Columbia University, Fall 2016

Table of Contents

1.Introduction

You are not the only one going your way. Uber and other ride sharing platforms have changed our lifestyle fundamentally. Absorbing the popular concept of "sharing economy", UberPOOL has become a smart, efficient, and affordable ride. Ridesharing allows one passenger to share the ride and split the cost of the trip with another rider headed in the same, or similar direction. Figure 1.1 shows a typical ride of UberPOOL. In the image, multiple passengers are picked up along a route, and dropped off at their requested destination. With UberPOOL's intelligent matching algorithm, one ride could save the passenger almost half of the fare, while adding only a few minutes to their trip. But how does this magic work? With the techniques we have learned in this class, we aim to approach the question and construct our matching algorithm based on NYC (New York City) taxi trip patterns.

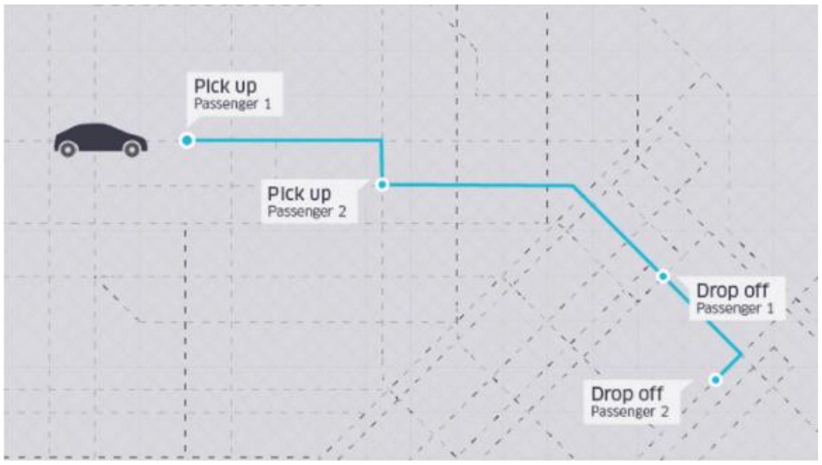


Figure 1.1: Ride Sharing Trip Example

The project report is organized as follows. First, a traffic pattern analysis is performed via a close scrutinization of each trip by distance, time and location to find out hotspots and peak hours. Then, a deeper analysis is done on our data with OD Matrix and probability tools. From this analysis, NYC traffic patterns were characterized in preparation for building a trip pairing model, and locating the focus area for this model. Then, this report will discuss the construction of our trip paring model, which is a pairing maximizing model with linear constraints. Model parameters, assumptions, objectives and constraints are included. After that, the results of the model are discussed with visualization and sensitivity analysis. Finally, this report concludes by commenting on the limitations of the model and our future work.

2. Problem Statement

2.1 Traffic Pattern Analysis

To understand NYC traffic pattern, pick-up location and pick-up time are focused on to find out the "hotspots" and peak hours by data visualization. Then focus is switched to the traffic pattern by OD matrix.

2.2 Pair Trip Problem

In this part, simplification of the pair trip problem is performed and formulated it into an integer program. Then, the filtered data is used to solve the problem in MATLAB and to obtain the results. Last, further analysis and results is discussed to come to a conclusion.

3. Taxi Data Analysis

To better understand the taxi trips that can be paired for ride sharing, historical taxi trips were analyzed from www.nyc.gov. This analysis focused on two main categories: finding the areas in New York City that had the highest volume of taxi trips, and finding the times of day that had the most frequent historical taxi trips.

First, before analyzing the data at a granular level, taxi trip data was looked at to get a high level understanding of taxi trips in New York City. It was discovered that New York City taxi trips are fairly short, on average, and also vary by season. Some of the high level results are shown in Table 1. (Taxi and Limosine Commission of New York City, 2015)

Table 1: High Level New York City Taxi Data Analysis

Average Trip Distance:	2.6 Miles
Percent of trips are less than 1 mile	20%
Percent of trips are less than 12 miles	99%
Season with the highest volume of taxi trips	Spring
Season with the lowest volume of taxi trips	Summer

3.1 Taxi Trip Pick-Up Locations

Next, when looking for the area of the highest volume of taxi trips, or "hot spots," the longitude and latitude locations of each taxi trip pick up location were first plotted. The initial image of this plot can be seen in Figure 3.1. Our initial prediction is that the pick-up locations might be close to the most popular tourist and transit locations in New York City. The results from this plot show that our prediction held true for June. The most popular locations for taxi trip pick-ups in June 2016 were by transportation hubs, including LaGuardia and John F. Kennedy International

Airport, and in Midtown. Specifically, in midtown, the hotspots were by Pennsylvania Station, Grand Central Station, and Times Square. A contour plot of the most popular origins is shown in Figure 3.2 with the hotspots indicated.

After analyzing the hot spots in June 2016, other days in the year were analyzed to compare this to. While the exact number of taxi trips changed from day-to-day in this area, the overall trend was that these areas were the most popular pick up locations. This was confirmed by the data released by New York City, as shown in Figure 3.3. The percent of taxi trip pick-up locations in Manhattan is over 90%.

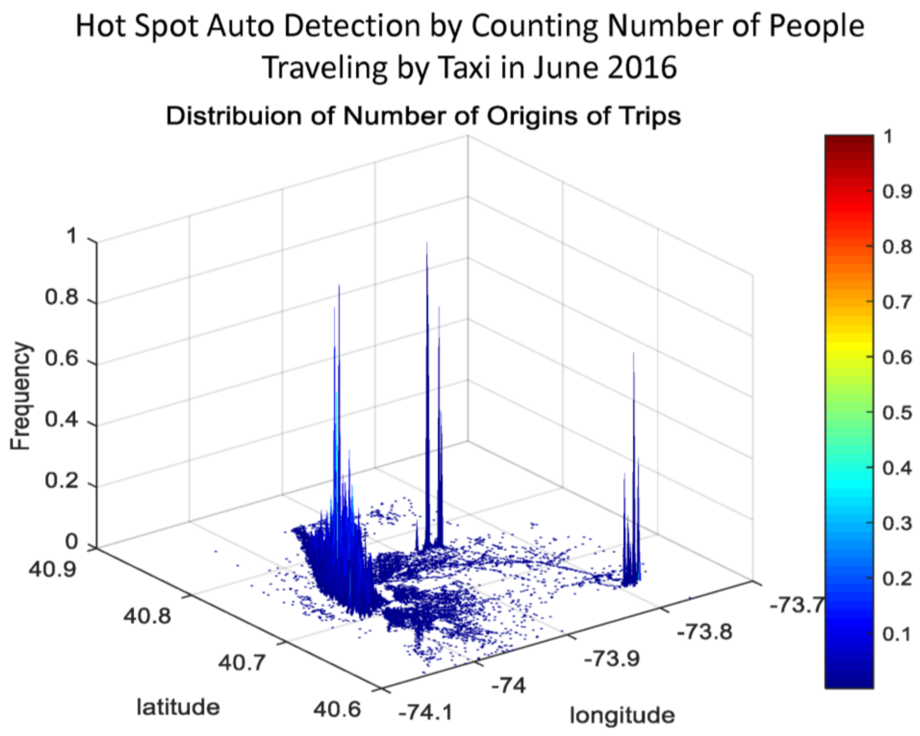


Figure 3.1: Taxi trip pick-up locations in June 2016

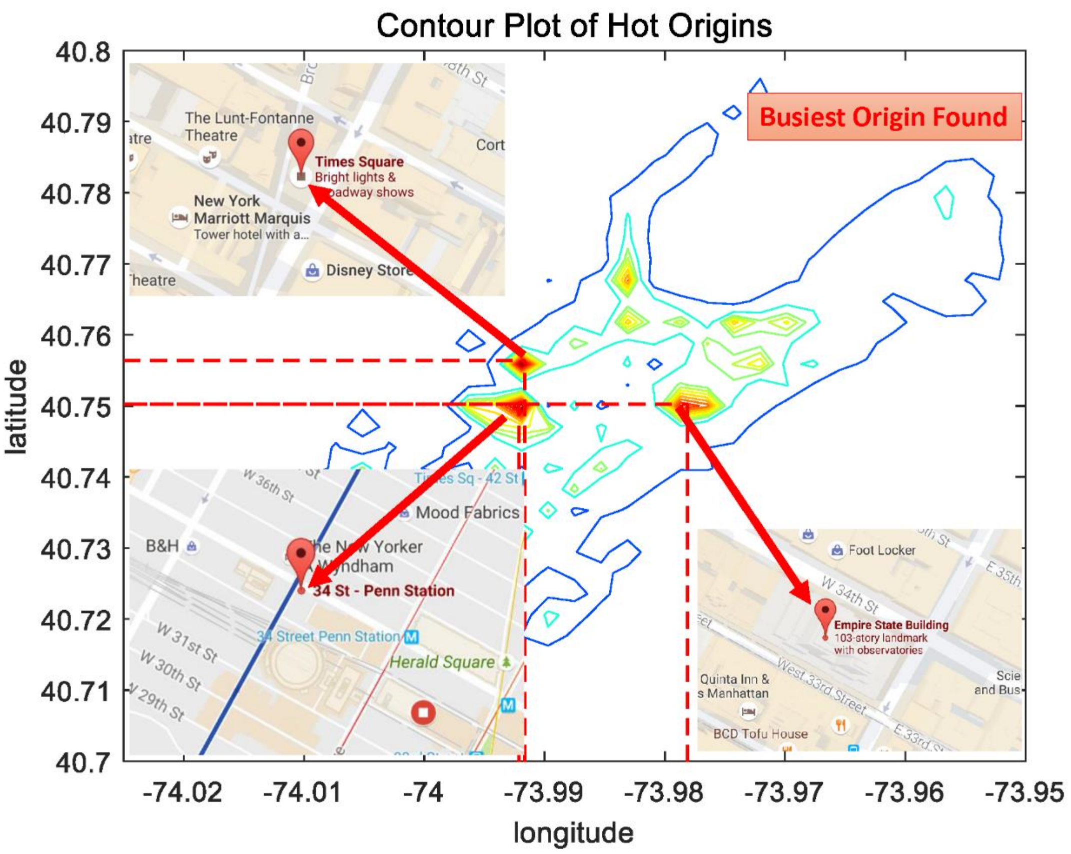
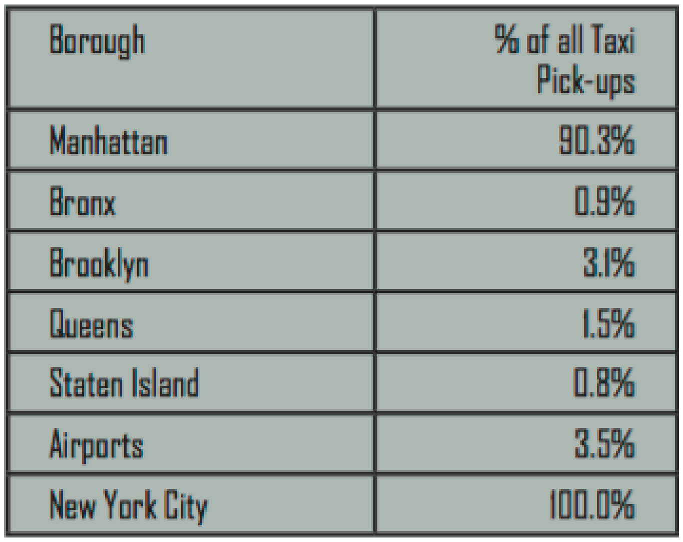


Figure 3.2: Taxi Trip Pick-Up Hot Spots in Manhattan



Borough	% of all Taxi Pick-ups
Manhattan	90.3%
Bronx	0.9%
Brooklyn	3.1%
Queens	1.5%
Staten Island	0.8%
Airports	3.5%
New York City	100.0%

Figure 3.3: Percent of taxi trip pick-ups split out by borough

3.2 Taxi Trip Pick-Up Times

The second main goal of the taxi data analysis portion of this report looks at hotspots that were discussed in the first portion to get a better understanding of taxi trip patterns by time of day and of the week. The area of the hotspots in Midtown were selected as the focus point for this analysis, between latitude and longitude of 40.74 °N to 40.77 °N and -74.00°W to -73.96°W.

First, daily traffic volume in May 2015 was plotted in a frequency histogram, as shown in Figure 3.4. From this graph, it is clear that there is not consistency in peak days of the week for taxi pick up time. Saturdays and Fridays, as one may expect, were not necessarily the peaks in May 2015. Instead, days around holidays and large events appeared to have the highest volume of taxi trips in a day.

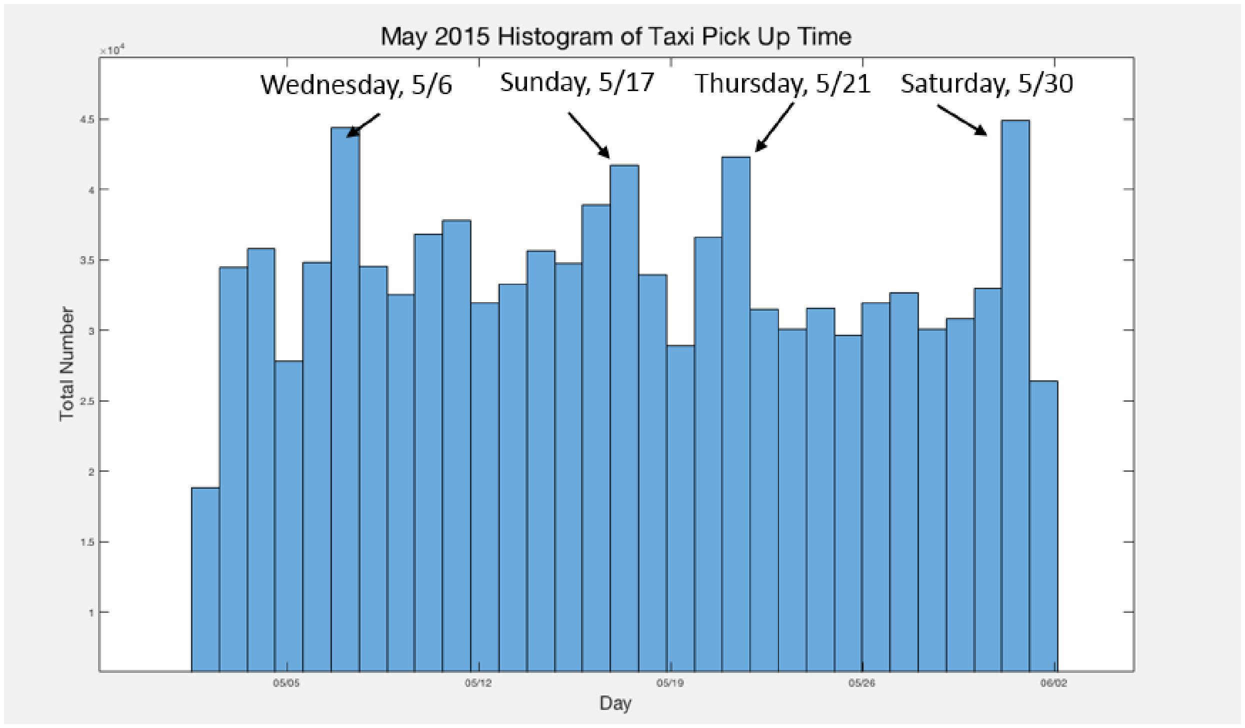


Figure 3.4: Traffic volume of New York City taxi trips per day in May 2015

Next, hours of the day were more closely analyzed for weekday and weekend patterns. First, weekday patterns were looked at for four weekdays in the beginning of May 2015. The number of taxi trips within every 30-minute interval are plotted in Figure 3.5. After looking at the differences in taxi trip patterns on these four days, it is not possible to draw any significant conclusions on the taxi patterns during the week. The only general conclusion that can be made is that it seems as though there are more trips in the afternoon and evening as opposed to the morning.

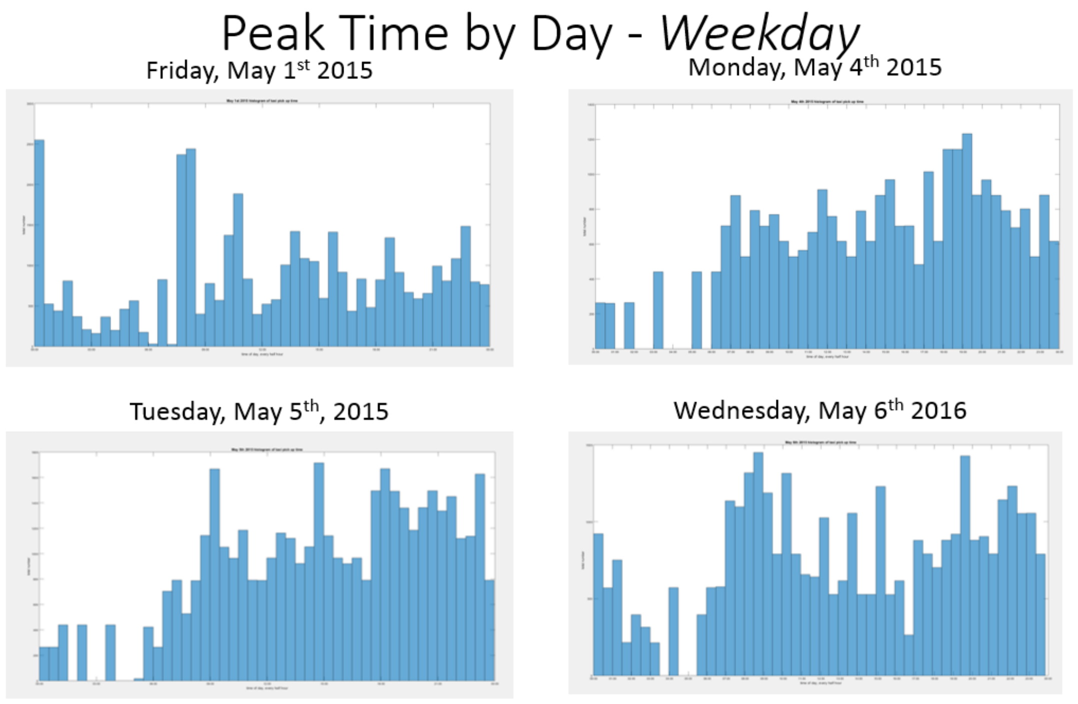


Figure 3.5: Traffic volume every half hour in New York City on four weekdays in May 2015

A similar analysis was conducted for a weekend in May 2015 as well, as shown in Figure 3.6. From looking at this image, a similar observation can be made. Saturday and Sunday seem to vary in their traffic pattern, but no conclusion on the traffic pattern on weekends can be made.

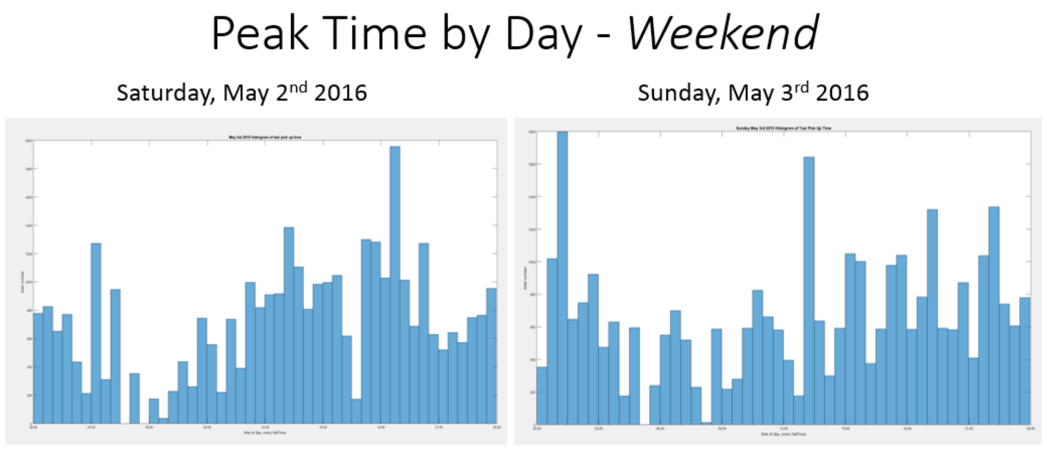
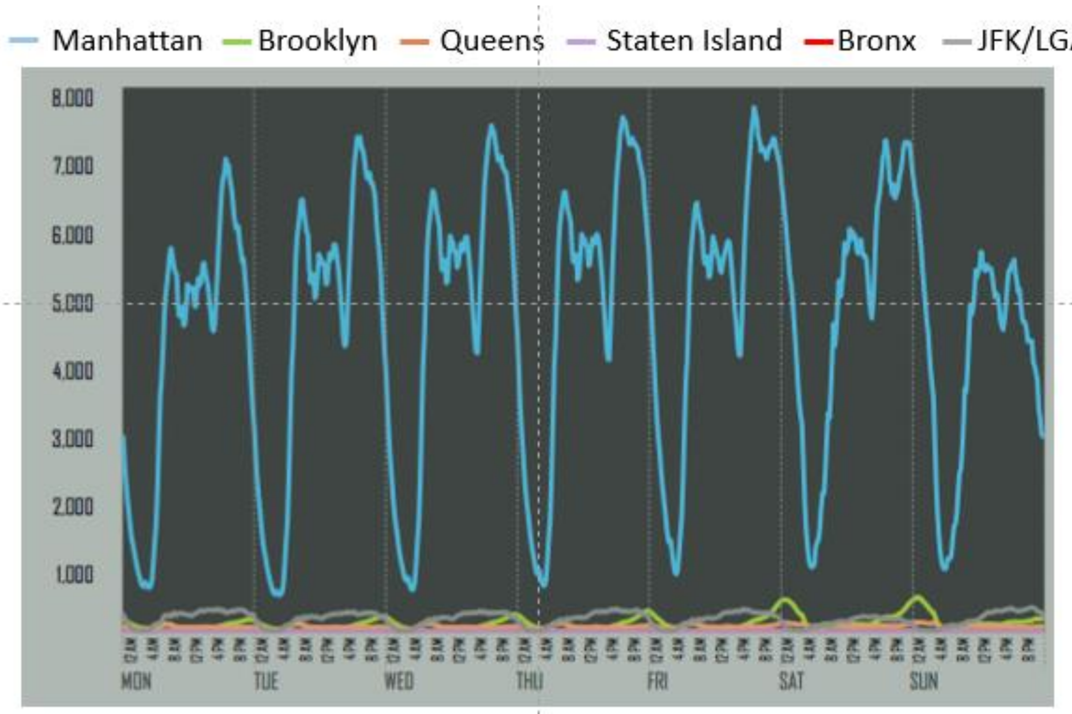


Figure 3.6: Traffic volume every half hour in New York City on a weekend in May 2015

However, to really understand the traffic pattern during the week, many more days need to be analyzed. In order to do so, the average hourly taxi trip volume during every day of the week was plotted over 5 years, as shown in Figure 3.7. With many more data points to notice a traffic pattern, it is clear that a traffic pattern during weekdays and weekend does exist.

Average Taxi Trips per Week



Source: http://www.nyc.gov/html/tlc/downloads/pdf/2014_taxicab_fact_book.pdf

Figure 3.7: Averaged Taxi Volume broken out by hours and week in New York City from 2008 to 2014 (Taxi and Limosine Commission of New York City, 2015)

After looking at the overall average of taxi trip hourly volume versus a snapshot within May 2015, some conclusions can be drawn. First, taxi trip volume has many variables that influence the frequency. This can include the weather, a holiday, the time of day, a special event that is occurring, traffic, among other reasons. Therefore, to truly understand the taxi volume, many data points need to be looked at and averaged. Once this data is averaged, as in Figure 3.7, there are clear patterns. On weekdays, the peaks are pretty similar, with a large spike in the evening. On Weekends the peaks are a little less dramatic, but are consistent with typical mealtimes in New York City.

Furthermore, from Figure 3.7, it is difficult to say if there is a rush hour consistent with the typical commuters' rush hour. The New York Times reported in 2010 that only 1% of commuters use taxis to get to work. (Roberts, 2010) If this is the case, then a deeper understanding of who taxi riders are will need to be conducted to understand their behavior and predict their patterns in the future.

4. OD Matrix Overview

In this section, an OD Matrix is used to get an initial understanding of the taxi pattern and potential trips that can be paired. It should be noted that this analysis is stricted to the Manhattan area for simplification purposes.

The taxi location data that is used in this analysis carries location information of every trip. The road network of New York City is seemingly rectangle-shape. However, as Figure 4.1 shows, it does not distribute exactly horizontally or vertically. This geographical characteristic will be a potential obstacle for our analysis.

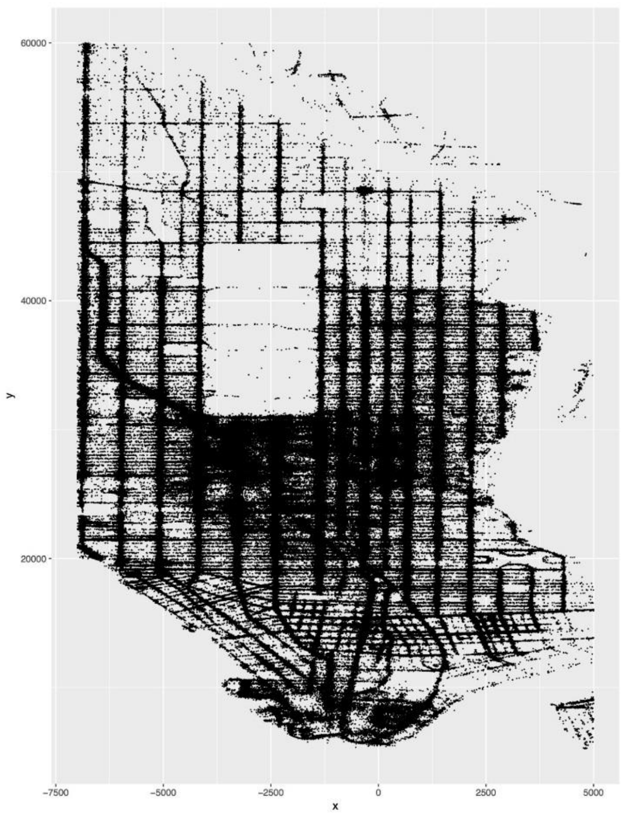


Figure 4.1: Pick up location distribution of New York City

4.1 Data Clean-Up Process

To begin the OD matrix and visualize the data, R programming was used. The dataset used is from the entire month of May 2016. The data size was roughly 11GB, containing 70 million rows and 12 columns.

4.1.1 Coordinate Rotation

For easier operations, longitude and latitude were converted to “px”, “py” “dx” and “dy” by rotating with a tilt angle, where “px” and “py” were derived from pick up locations, “dx” and “dy” from drop off locations. These variables were set as a series of integers that could easily divide the square-grid road distribution. “px” and “py” were then used to create x-axis and y-axis for our new coordinate.

4.1.2 Weekdays and weekends Analysis

As discussed earlier, transportation patterns may vary on weekdays and weekends. So for this portion of the analysis, two OD matrixes were created: one for weekend and one for weekdays. Therefore, to classify data points into two categories, the variable “isWeekdays” was added, “isWeekdays == 1” representing weekdays data while “isWeekdays == 0” representing weekends.

4.2 Initial Location Data Visualization

Determining the numbers and areas of sub-regions was the first step of creating an OD matrix. Since the density of Manhattan at the beginning of the data process was unkown, the a preliminarily visualization was completed to allow Manhattan’s area to be divided evenly. Pick-up locations were plotted with the May 2016 data. The result is the shown as Figure 4.2.

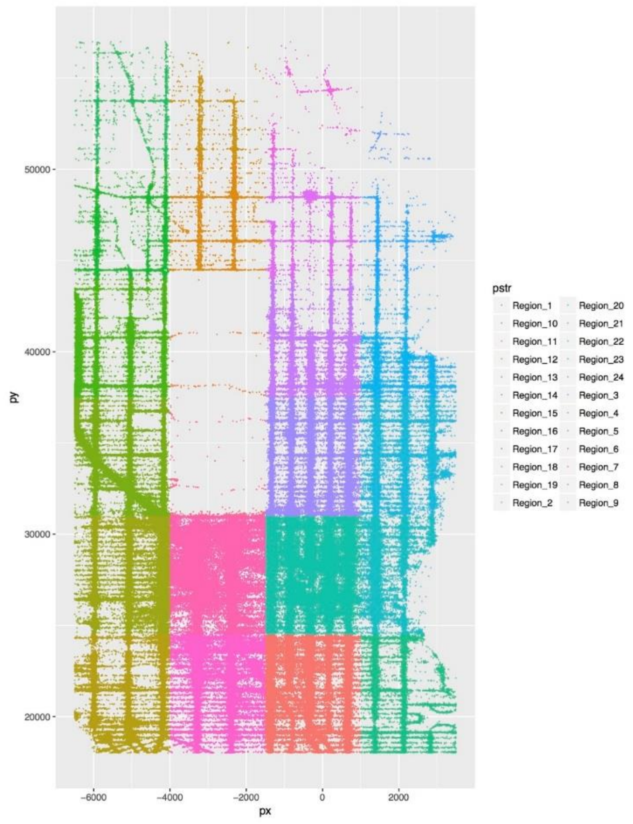


Figure 4.2: Manhattan region separations

The data points roughly outlined a road network, and at some point indicated the trip density. The graph shows that Midtown attracted most trips, and the avenues had relatively more trips than streets by several times. This analysis focuses on areas where -6500 < px & px < 3500 and 18000 < py & py < 57000.

By the characteristics of trip distribution and the square-grid road network, 24 regions were defined as the components of our OD Matrix. The matrix is expected to be 24 x 24, and will separate Manhattan area on two geographical directions, that is, East-West and North-South.

4.3 Results

To better understand the transportation patterns, trip distributions were looked at on weekdays and weekend by filtering variable “isWeekdays”.

From Figure 4.4 and 4.6, it can be seen that the trips that can be paired are most concentrated on the midtown area for both weekends and weekdays. This result looks reasonable because the majority of the city’s skyscrapers, major transportation hubs, and tourist attractions lie within it, as discussed in Section 3.

Figures 4.5 and 4.7 display partial quantitative result due to the large size of 24 x 24 matrix. The matrix format can be future improved by denoting “Region No.” to replace “(px, py) and “(dx, dy)”.

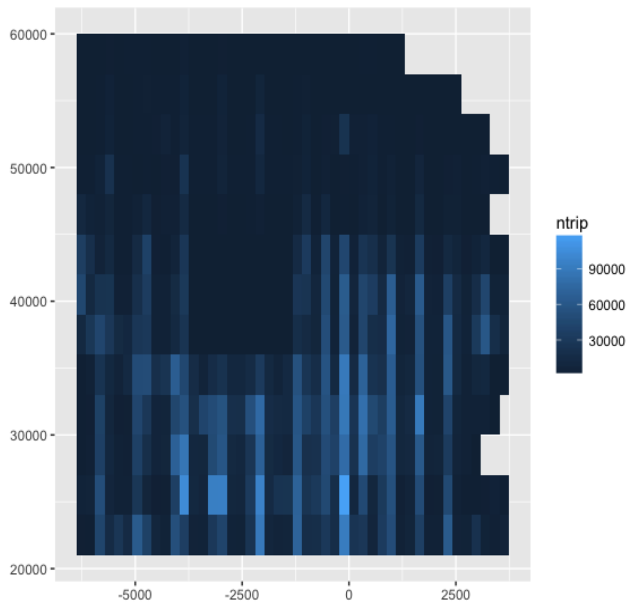
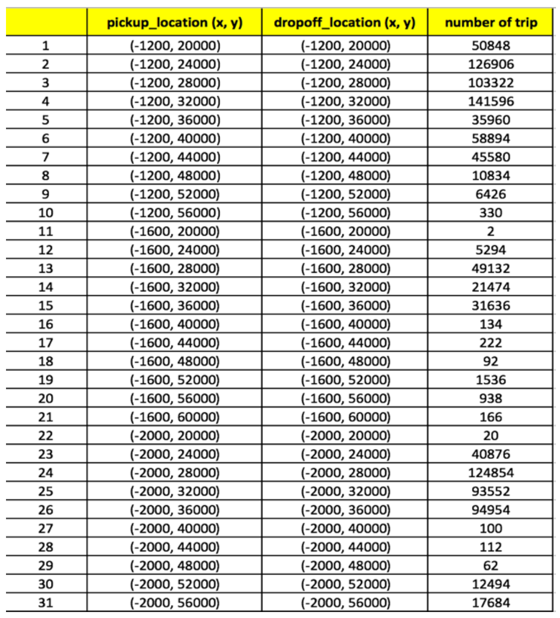


Figure 4.4: Trip distribution on weekdays



	pickup_location (x, y)	dropoff_location (x, y)	number of trip
1	(-1200, 20000)	(-1200, 20000)	50848
2	(-1200, 24000)	(-1200, 24000)	126906
3	(-1200, 28000)	(-1200, 28000)	103322
4	(-1200, 32000)	(-1200, 32000)	141596
5	(-1200, 36000)	(-1200, 36000)	35960
6	(-1200, 40000)	(-1200, 40000)	58894
7	(-1200, 44000)	(-1200, 44000)	45580
8	(-1200, 48000)	(-1200, 48000)	10834
9	(-1200, 52000)	(-1200, 52000)	6426
10	(-1200, 56000)	(-1200, 56000)	330
11	(-1600, 20000)	(-1600, 20000)	2
12	(-1600, 24000)	(-1600, 24000)	5294
13	(-1600, 28000)	(-1600, 28000)	49132
14	(-1600, 32000)	(-1600, 32000)	21474
15	(-1600, 36000)	(-1600, 36000)	31636
16	(-1600, 40000)	(-1600, 40000)	134
17	(-1600, 44000)	(-1600, 44000)	222
18	(-1600, 48000)	(-1600, 48000)	92
19	(-1600, 52000)	(-1600, 52000)	1536
20	(-1600, 56000)	(-1600, 56000)	938
21	(-1600, 60000)	(-1600, 60000)	166
22	(-2000, 20000)	(-2000, 20000)	20
23	(-2000, 24000)	(-2000, 24000)	40876
24	(-2000, 28000)	(-2000, 28000)	124854
25	(-2000, 32000)	(-2000, 32000)	93552
26	(-2000, 36000)	(-2000, 36000)	94954
27	(-2000, 40000)	(-2000, 40000)	100
28	(-2000, 44000)	(-2000, 44000)	112
29	(-2000, 48000)	(-2000, 48000)	62
30	(-2000, 52000)	(-2000, 52000)	12494
31	(-2000, 56000)	(-2000, 56000)	17684

Figure 4.5: Partial OD matrix on weekdays

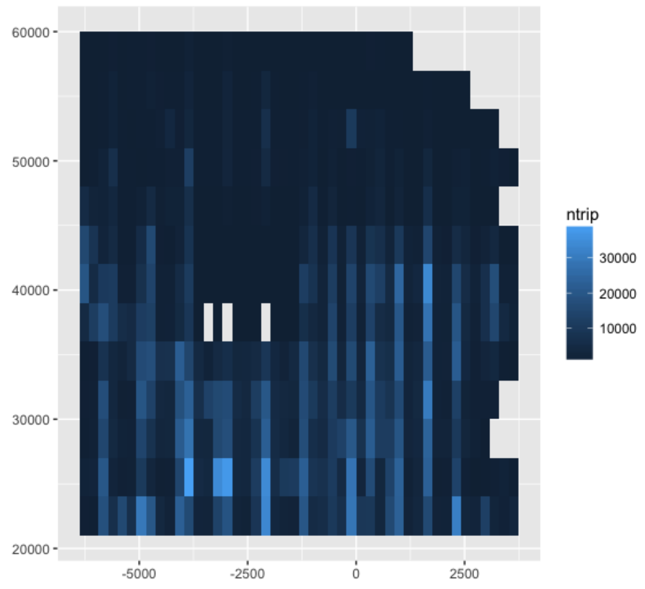
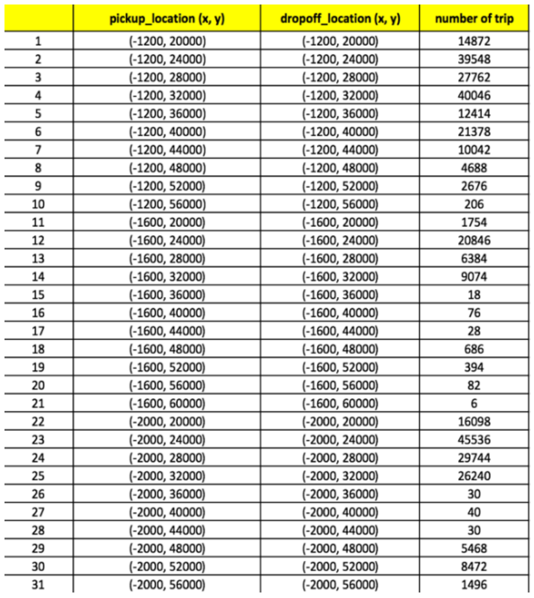


Figure 4.6: Trip distribution on weekends



	pickup_location (x, y)	dropoff_location (x, y)	number of trip
1	(-1200, 20000)	(-1200, 20000)	14872
2	(-1200, 24000)	(-1200, 24000)	39548
3	(-1200, 28000)	(-1200, 28000)	27762
4	(-1200, 32000)	(-1200, 32000)	40046
5	(-1200, 36000)	(-1200, 36000)	12414
6	(-1200, 40000)	(-1200, 40000)	21378
7	(-1200, 44000)	(-1200, 44000)	10042
8	(-1200, 48000)	(-1200, 48000)	4688
9	(-1200, 52000)	(-1200, 52000)	2676
10	(-1200, 56000)	(-1200, 56000)	206
11	(-1600, 20000)	(-1600, 20000)	1754
12	(-1600, 24000)	(-1600, 24000)	20846
13	(-1600, 28000)	(-1600, 28000)	6384
14	(-1600, 32000)	(-1600, 32000)	9074
15	(-1600, 36000)	(-1600, 36000)	18
16	(-1600, 40000)	(-1600, 40000)	76
17	(-1600, 44000)	(-1600, 44000)	28
18	(-1600, 48000)	(-1600, 48000)	686
19	(-1600, 52000)	(-1600, 52000)	394
20	(-1600, 56000)	(-1600, 56000)	82
21	(-1600, 60000)	(-1600, 60000)	6
22	(-2000, 20000)	(-2000, 20000)	16098
23	(-2000, 24000)	(-2000, 24000)	45536
24	(-2000, 28000)	(-2000, 28000)	29744
25	(-2000, 32000)	(-2000, 32000)	26240
26	(-2000, 36000)	(-2000, 36000)	30
27	(-2000, 40000)	(-2000, 40000)	40
28	(-2000, 44000)	(-2000, 44000)	30
29	(-2000, 48000)	(-2000, 48000)	5468
30	(-2000, 52000)	(-2000, 52000)	8472
31	(-2000, 56000)	(-2000, 56000)	1496

Figure 4.7: Partial OD matrix on weekends

4.4 GPS Noise Modeling

An interesting phenomenon can be observed in Figure 4.1; some data points are distributed at the places that they were not supposed to be, such as inside of the buildings or between blocks. The GPS distortion may be caused by sensor errors due to huge buildings, as shown in Figure 4.8, getting in the way of signals. To learn the probabilities of a taxi being at each point, a Gaussian Mixture Model (GMM). Understanding the probabilities could potentially help re-identify these locations and improve the accuracy of number of trips for each region.

Midtown was selected, as shown in Figure 4.9, to do a sample analysis because the GPS noise was largest in this area. Four trees were trained, which were corresponding with four avenues as below.

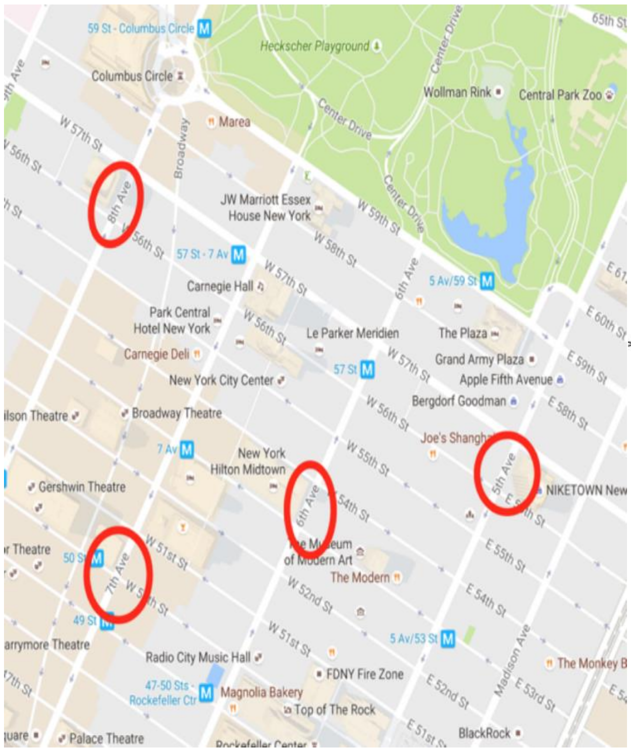
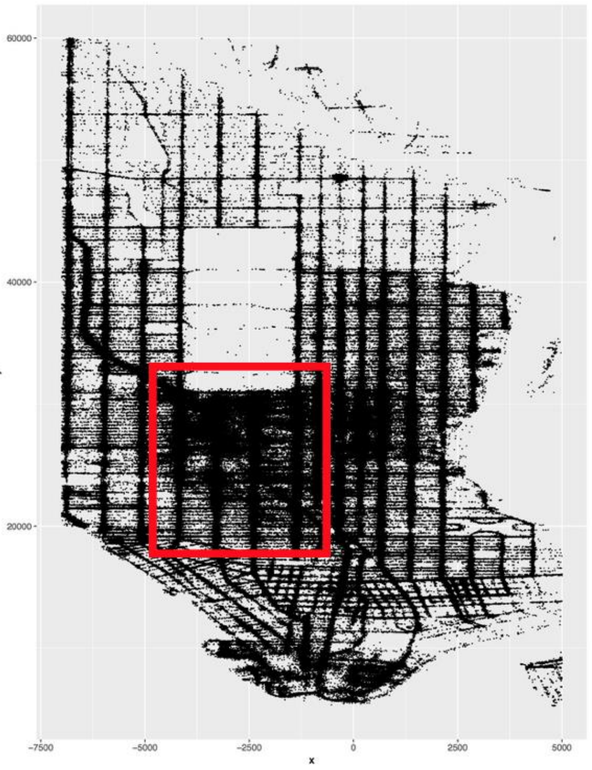
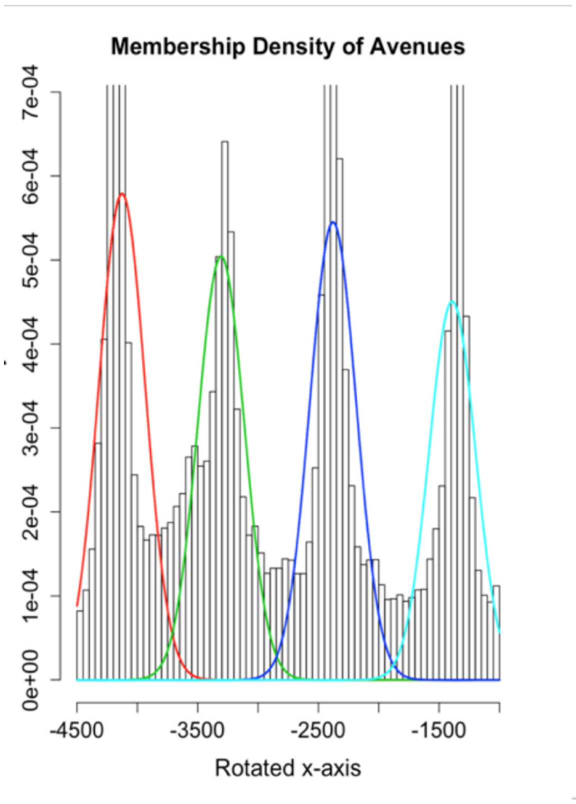



Figure 4.8: Sample selection **Figure 4.9:** Sample avenues selection

The R GMM package was applied to this area, and the initial conditions were set as the following:

1) lambda (weight number) are the same for all components,
2) mean as the x value for each avenue,
3) Sigma(variance) equals to 5.

The membership densities are well separated into four GMMs, shown in Figure 4.10. The histograms stand for the initial trip counts for each avenue, and the curves in Figure 4.11, represent the likelihood for each GMM.



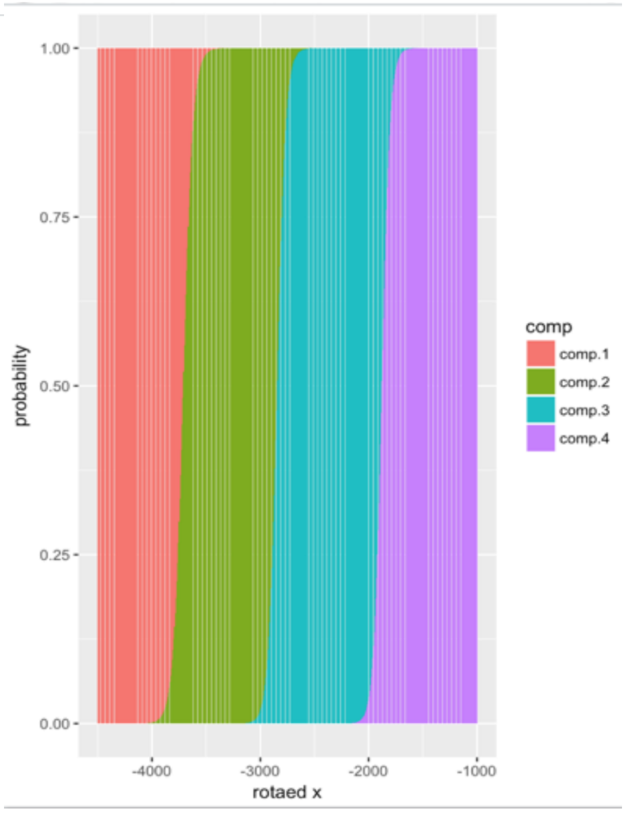


Figure 4.10: Membership density of avenues **Figure 4.11:** Probability for each component

5. Integer Programming Model

5.1 Model Setup

5.1.1 Assumptions

To formulate the problem into a general integer programming model, several assumptions are made in accordance with constraints in real life.

1) Each passenger has his own tolerance for waiting time
2) Each passenger has his own tolerance for distance between pick-up locations and that between drop-off locations
3) The total number of passengers in a paired trip cannot exceed the capacity of the car

5.1.2 Decision Variables

We use $x_{i,j}$ to judge whether trip i and trip j should be paired:

$x_{i,j} = 1$ if trip i and trip j can be paired together;

$x_{i,j} = 0$ if trip i and trip j cannot be paired together.

Here $x_{i,j}$=$x_{j,i}$should be satisfied. And we set $x_{i,i}$ =1 for the convenience of calculation.

5.1.3 Constraints and Objective

The constraints for this problem include distance between pick-up locations, distance between drop-off locations, difference between start time and number of travelers. To avoid the nonlinear formula of distance, a set for each trip was first put in place, which contains all the trips that satisfy passengers' tolerance of waiting time and distance.

For each trip i, set up a set {R} called ***Candidate*** such that

$\forall$i = 1,2,3…, n $\quad Candidate_i$ = {*indices*}

Where$\forall$k$\in$ $Candidate_i$ $\quad$ *Distance* $(\vec{O}i, \vec{O}k) \leq$ tolerance distance for origins

Distance $(\vec{D}i, \vec{D}k) \leq$ tolerance distance for destinations

$|t_i - t_k| \leq$ tolerance waiting time

Note that: Origin $\vec{O}i$= $(latitude_i, longitude_i)$ = (φ_i, λ_i)

Destination $\vec{D}i$= $(latitude_i, longitude_i)$ = (φ_i, λ_i)

Distance are calculated by Haversine formula:

$$d = 2r \arcsin\left(\sqrt{\operatorname{hav}(\varphi_2 - \varphi_1) + \cos(\varphi_1)\cos(\varphi_2)\operatorname{hav}(\lambda_2 - \lambda_1)}\right)$$
$$= 2r \arcsin\left(\sqrt{\sin^2\left(\frac{\varphi_2 - \varphi_1}{2}\right) + \cos(\varphi_1)\cos(\varphi_2)\sin^2\left(\frac{\lambda_2 - \lambda_1}{2}\right)}\right)$$

After setting up the Candidate set, the problem can be formulated into an integer program. And the set makes the constraints much simpler.

The objective is to pair as many trips as possible — maximize total number of the paired trips.

$$\max \quad Z = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} x_{i,j}$$

Subject to:

$\forall i, j$ = 1,2,3…, n $\quad x_{i,j} = x_{j,i}$

$x_{i,i} = 1$

$x_{i,j} \in \{0,1\}$

$\forall i$ = 1,2,3…,n $\quad \sum_{j=1}^{n} x_{i,j} p_j \leq$ Available seats in the car

$x_{i,j}$=0 if $j \notin Candidate_i$

5.2 A Problem-Specific Solver

To solve this integer programming model, a solver named NYC-Taxi-Trip-Carpooling-Assistant is specifically created. It has a simple user interface for easily manipulating parameters and for conveniently visualizing the results.

Here is a list of its features

1) Automatically parsing the input dataset. Users only need to specify the name of the input dataset
2) Efficient in memory space and running time. It can handle large sample, automatically use sparse matrix for large-scale model to save memory space, and is well tuned to run fast

without loss of accuracy. Parallel computing for machines with multiple cores will be automatically launched for solving large-scale problems.

3) Two approaches to solve the integer programming model. One uses the intlinprog solver provided by MATLAB Optimization toolbox, the other one makes use of a greedy search algorithm. The greedy search algorithm provides a close-to-optimality solution but runs much faster.
4) Visualization. Users can easily query the results. The results will be visualized and output in text form.

The latest version of the app is released at the project website.

Figure 5.1: NYC-Taxi-Trip-Carpooling-Assistant demonstrates some basic functions of the app.

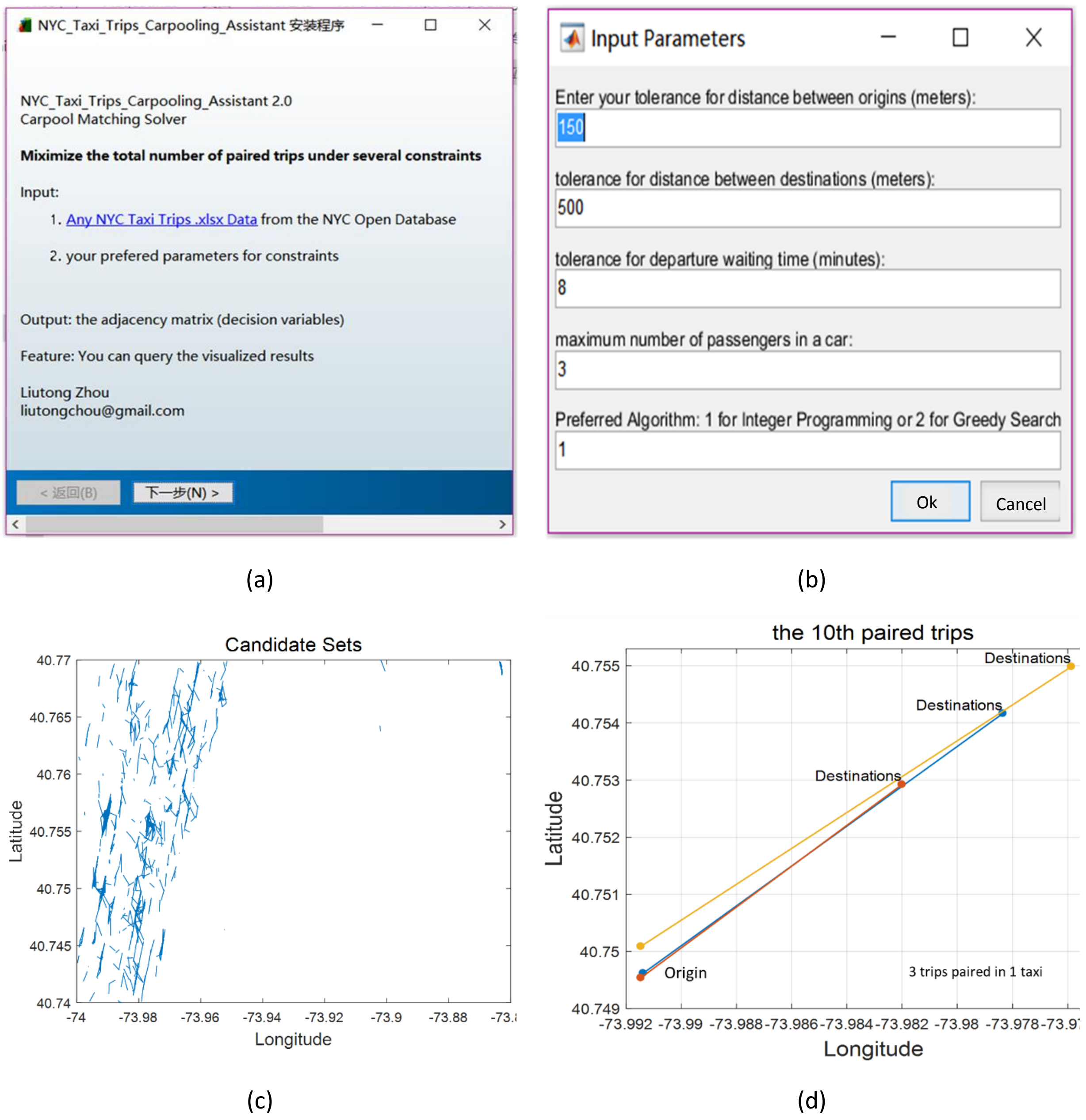
5.3 Results

To simplify the analysis, some particular examples under the following assumptions are further considered and analyzed.

1) All passengers share the same tolerance for waiting time— less than 10 minutes;
2) Assume the tolerance distance for pick up is 100m and for drop off is 1000m;
3) We only consider traveler number less than 4 people (1,2 or 3).

From the model, the initial results were that 257 trips can be paired out of 3898 trips from the data set of 7:30-11:00 a.m. on May 1st, 2015, which represents the pairing percentage is around 6.16%. Figure 5.2 is graph showing the initial solution plotted in Manhattan's longitude and latitude grid.

To understand our initial solution, and whether it is the best solution given our assumptions, it must be proved that we have seleceted good parameters (not eliminating the solutions scarely and not enlarging the solutions greatly). Since the parameters are selected based on assumptions of tolerance for wait time and distance between start trips, a sensitivity analysis discussed in the next section will give us more insight into the assumptions.


NYC_Taxi_Trips_Carpooling_Assistant 安装程序

NYC_Taxi_Trips_Carpooling_Assistant 2.0
Carpool Matching Solver

Miximize the total number of paired trips under several constraints

Input:

1. Any NYC Taxi Trips .xlsx Data from the NYC Open Database
2. your prefered parameters for constraints

Output: the adjacency matrix (decision variables)

Feature: You can query the visualized results

Liutong Zhou
liutongchou@gmail.com

< 返回(B) 下一步(N) >

Input Parameters

Enter your tolerance for distance between origins (meters):
150

tolerance for distance between destinations (meters):
500

tolerance for departure waiting time (minutes):
8

maximum number of passengers in a car:
3

Preferred Algorithm: 1 for Integer Programming or 2 for Greedy Search
1

Ok Cancel

Figure 5.1: NYC-Taxi-Trip-Carpooling-Assistant (a) Installation interface of the app (b) Input Parameters for constraints (c) Automatically visualize the carpooling candidates (d) Query the results, output and visualize the queried results.

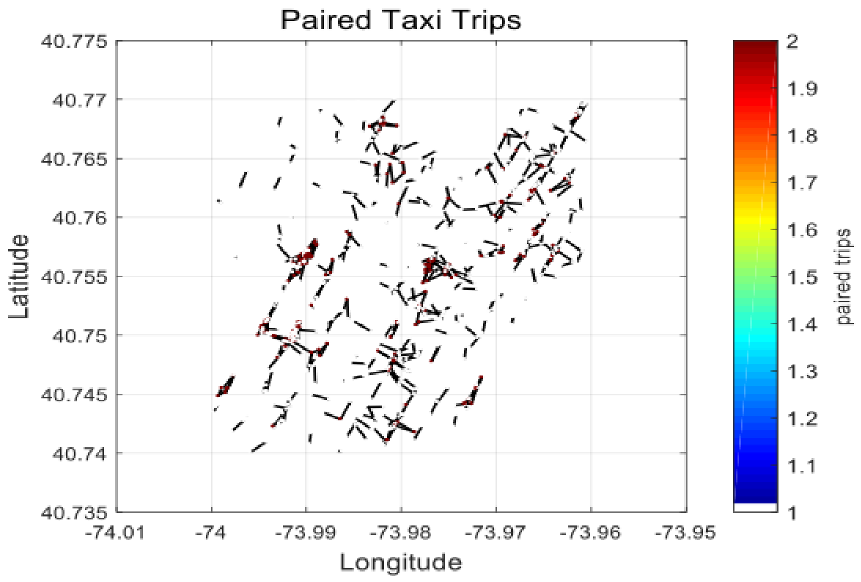


Figure 5.2 Initial Solution

5.4 Sensitivity Analysis

For this model, the objective function and constraints co-efficients have physical meanings. Therefore for sensitivity analysis, right-hand-side values of the constraints were only considered. These three values are the tolerance for departure origins, tolerance for destination distance and tolerance for waiting time.

When conducting the sensitivity analysis, one of the three values was changed while the other two were fixed. Then the results were compared. This process was then repeated for each variable to come up with the Tables 2, 3, and 4. These tables compare the influence of changing baseline values to the change in the paired percentage. In each of these tables the value of tolerance was altered and the effect the change had on the paired percentage is shown. If solution set contained large amount of data that was not feasible to process through the computer, "No answer" is left in that case.

Table 2: Changing the Tolerance for Departure Origins

Tolerance for departure origins (meters)	Tolerance for destination distance (meters)	Waiting tolerance (10 min)	#trips paired using IP	#trips paired using GS	total trips		Pairred Percentage
100	1000	10	240	253	3898	Baseline	6.16%
30	1000	10	102	104	3898	-70.00%	2.62%
50	1000	10	160	167	3898	-50.00%	4.10%
150	1000	10	322	337	3898	50.00%	8.26%
170	1000	10	342	359	3898	70.00%	8.77%
200	1000	10	370	389	3898	100.00%	9.49%
250	1000	10	No answer (Too many solutions)	No answer (Too many solutions)	3898		
300	1000	10	No answer (Too many solutions)	No answer (Too many solutions)	3898		

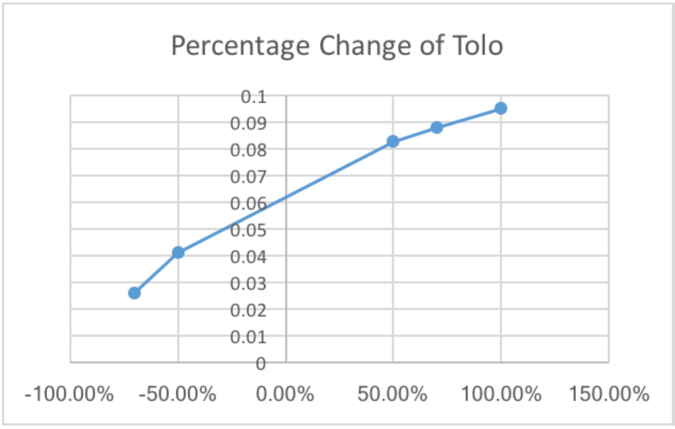
Table 3: Changing the Tolerance for Destination Distance

Tolerance for departure origins (meters)	Tolerance for destination distance (meters)	Waiting tolerance (10 min)	#trips paired using IP	#trips paired using GS	total trips		Pairred Percentage
100	1000	10	240	253	3898	Baseline	6.16%
100	100	10	33	32	3898	-90.00%	0.85%
100	300	10	80	80	3898	-70.00%	2.05%
100	500	10	147	152	3898	-50.00%	3.77%
100	700	10	179	185	3898	-30.00%	4.59%
100	1200	10	270	284	3898	20.00%	6.93%
100	1500	10	301	313	3898	50.00%	7.72%
100	1700	10	322	340	3898	70.00%	8.26%
100	2000	10	No answer (Too many solutions)	No answer (Too many solutions)	3898		

Table 4: Changing the Tolerance for Waiting Time

Tolerance for departure origins (meters)	Tolerance for destination distance (meters)	Waiting tolerance (10 min)	#trips paired using IP	#trips paired using GS	total trips		Pairred Percentage
100	1000	10	240	253	3898	Baseline	6.16%
100	1000	3	145	152	3898	-70.00%	3.72%
100	1000	5	172	183	3898	-50.00%	4.41%
100	1000	7	204	215	3898	-30.00%	5.23%
100	1000	13	249	263	3898	30.00%	6.39%
100	1000	15	261	272	3898	50.00%	6.70%
100	1000	20	No answer (Too many solutions)	No answer (Too many solutions)	3898		

Then, the relation between change of baseline to change of paired percentage for each parameter is observed. Figures 5.3, 5.4 and 5.5 show this. “Tolo” represents the tolerance for departure origin, “Told” represents the tolerance for destination distance and “Tolt” represents the tolerance for waiting time.)



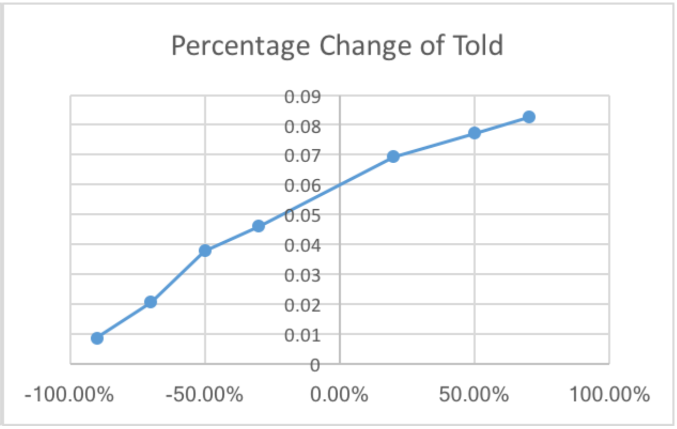


Figure 5.3: Percentage Change of Tolo **Figure 5.4:** Percentage Change of Told

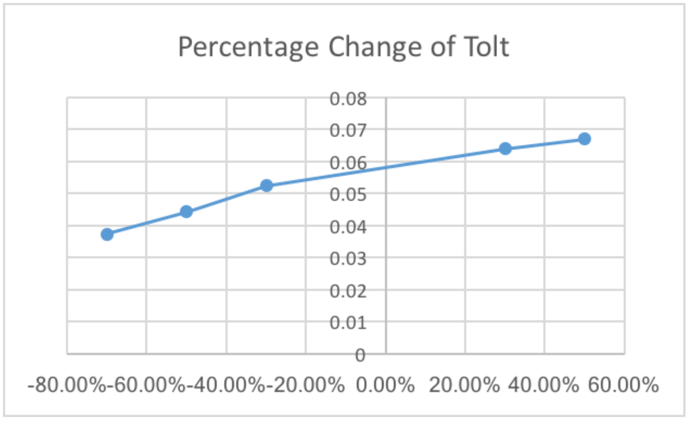


Figure 5.5: Percentage Change of Toldt

From these graphs, it can be observed that all parameters can only change within ±100%, which represents that the initial value for each parameter chosen worked quite well. It would be hard to determine which parameter is the most influential in this model, while it may hold true that if a certain percentage of variance is set, then one parameter may be more influential.

6. Conclusion

Based on the results of the model, New York City taxi ride sharing is very efficient and would be very successful, as UberPOOL is. This is due to the short trips that taxis make and the high demand of taxis in the New York area. While this model only focused on a small subset of data in a small area of New York City, the model can be applied to greater data sets to better understand taxi trip ride sharing in all New York City.

Our initial analysis in Section 3 of the taxi trip data concluded that the frequency and volume of taxi trips depends on many factors. Applying this observation to the integer programming model, where a subset of data in a small area was analyzed, presents a need for a larger modeling analysis to be done to truly understand the taxi trip pairing possibility in New York City. Running the model for multiple days in all of New York City would provide more plausible results of what trip pairing may be for the entire system. In addition, strong pairs can be found within the dataset, if two riders frequently share similar trip times and locations. These strong pairs can be utilized so that taxis can plan ahead for the trips, or coordinate their routes to reduce empty cab time.

Implementing a ride sharing program may change many taxi riders' behaviors. Some riders who do not normally take taxis may be encouraged to participate in this program, or riders may be discouraged from taking part in this program. This will need to be considered when ride sharing is implemented, and dynamic modeling will need to be done in order to pair trips as customer behavior changes.

The objective this project looked at was maximizing the number of paired trips. For the purpose of this report and analysis, this objective worked very well. To create a model that is more accurate for all of New York City, cost minimization, traffic, and route optimization per trip will also need to be considered in the objective. In addition, better understanding who rides taxis and what their needs are will benefit the trip pairing model and the future of taxi trip ride sharing.

Works Cited

Roberts, S. (2010, May 9). Take a Taxicab to Work? More New Yorkers Walk. *The New York Times*, p. 1.

Taxi and Limosine Commission of New York City. (2015). *2014 Taxicab Factbook.* New York: New York City Office of Policy and External Affairs.

Appendix I Codes

R source code

```
load("/Users/qiucongying/Desktop/4011/trips.20160501_0531.preped.RData")
dim(univ1month)
options(max.print = 200)
library(dplyr)
library(lubridate)
library(ggplot2)

univ1month <- tbl_df(univ1month)
class(univ1month)
str(univ1month)

#weekend_sample <- univ1month %>%
            # sample_frac(0.25) %>%
            # filter(isWeekday == 0) %>%
            # mutate(day = wday(pickup_datetime, label = TRUE)) %>%
            # arrange(pickup_datetime)
weekend <- univ1month %>%
 filter(isWeekday == 0) %>%
 mutate(day = wday(pickup_datetime, label = TRUE)) %>%
 arrange(pickup_datetime)

weekday <- univ1month %>%
 filter(isWeekday == 1) %>%
 mutate(day = wday(pickup_datetime, label = TRUE)) %>%
 arrange(pickup_datetime)

ggplot(weekend_sample, aes(px, py)) + geom_point(size = 0.001) + labs(x = "x", y = "y")

weekend_sample %>% filter(  -6500 < px & px <  3500 &  18000 < py & py <  57000) %>%
## px(-6500, -4000, -1500, 1000, 3500); py(18000, 24500, 31000, 37500, 44000, 50500, 57000)
```

```
 mutate( pxcat = cut(px, breaks=4),
         pycat = cut(py, breaks=6),
         pcat = as.factor(paste0(pxcat,'and',pycat)) %>% as.numeric,
         pstr = paste0('Region_', pcat) ) %>%     # dplyr::select(pxcat, pycat) %>%
 group_by(pxcat, pycat) %>% mutate( n = n() ) %>%
 ggplot() + geom_point(aes(px, py, color=pstr), size = 0.0001)
xg = 220
yg = 3000

tmp1 <-
weekend %>% filter(  -6500 < px & px <  3500 &  18000 < py & py <  57000 ) %>%
 mutate( npx = round(px,0) + (xg - round(px,0) %% xg), # x interval = 25
         ndx = round(dx,0) + (xg - round(dx,0) %% xg), # y interval = 90
         npy = round(py,0) + (yg - round(py,0) %% yg),
         ndy = round(dy,0) + (yg - round(dy,0) %% yg)
         )  %>%
 group_by(npx, npy) %>% mutate( pn = n() ) %>%
 group_by(ndx, ndy) %>% mutate( dn = n() ) %>%
 mutate( pxy = paste0(npx,',',npy),
         dxy = paste0(ndx,',',ndy)) %>%
 group_by(pxy,dxy) %>% mutate( ntrip = n() ) %>% dplyr::slice(1) #%>%
dplyr::select(pxy,dxy,ntrip)
 #%>% View
ggplot(tmp1) + geom_rect(aes(xmin = npx, xmax = npx + xg,
                             ymin = npy, ymax = npy + yg, fill=ntrip ))
tmp2 <-
 weekday %>% filter(  -6500 < px & px <  3500 &  18000 < py & py <  57000 ) %>%
 mutate( npx = round(px,0) + (xg - round(px,0) %% xg), # x interval = 25
         ndx = round(dx,0) + (xg - round(dx,0) %% xg), # y interval = 90
         npy = round(py,0) + (yg - round(py,0) %% yg),
         ndy = round(dy,0) + (yg - round(dy,0) %% yg)
 )  %>%
 group_by(npx, npy) %>% mutate( pn = n() ) %>%
 group_by(ndx, ndy) %>% mutate( dn = n() ) %>%
 mutate( pxy = paste0(npx,',',npy),
         dxy = paste0(ndx,',',ndy)) %>%
 group_by(pxy,dxy) %>% mutate( ntrip = n() ) %>% dplyr::slice(1) # %>%
dplyr::select(pxy,dxy,ntrip)
 # %>% View
ggplot(tmp2) + geom_rect(aes(xmin = npx, xmax = npx + xg,
                             ymin = npy, ymax = npy + yg, fill=ntrip ))
```