

AoCMM #692

October 11, 2017

Contents

| | | |
|----------|--|-----------|
| 1 | Summary | 3 |
| 1.1 | Interpretation and Methods of Problem 1 | 3 |
| 1.2 | Interpretation and Methods of Problem 2a | 4 |
| 1.3 | Interpretation and Methods of Problem 2b | 4 |
| 2 | Introduction | 5 |
| 2.1 | Background Information of Problem 1 | 5 |
| 2.2 | Background Information of Problem 2 | 5 |
| 3 | Problem 1 Solution | 6 |
| 3.1 | Introduction of Model | 6 |
| 3.2 | Assumption and justification | 6 |
| 3.3 | Variables and Parameters | 7 |
| 3.4 | Solution of Problem 1a – Category 2 | 8 |
| 3.4.1 | Approach 1a.1: Categorize words into groups and find the speed | 8 |
| 3.4.2 | Approach 1a.2: Determine left-hander, right-hander or undefined. | 8 |
| 3.4.3 | Approach 1a.3: Hungarian Algorithm | 10 |
| 3.5 | Solution of Problem 1b – Category 3 | 11 |
| 3.5.1 | Approach 1b Categorize the words and find the typing speed . | 11 |
| 3.6 | Case analysis of problem 1b | 12 |
| 3.7 | Sensitivity Analysis | 13 |
| 3.8 | Strengths and Weaknesses | 15 |
| 3.9 | Conclusion of Solution 1 | 15 |
| 4 | Problem 2 Solution | 16 |
| 4.1 | Introduction of Model | 16 |
| 4.2 | Assumptions and Justifications | 17 |
| 4.3 | Variables and Parameters | 18 |
| 4.4 | Solution of Problem 2a – Searching Case | 20 |
| 4.4.1 | Approach 2a.1: Finding the expected income of each zone . . | 20 |
| 4.4.2 | Approach 2a.2: Finding the expected value of each route . . . | 22 |
| 4.4.3 | Approach 2a.3: Determining whether stay or move | 23 |
| 4.5 | Solution of Problem 2a – Calling Case | 24 |
| 4.5.1 | Approach 2a.4: Dijkstras Algorithm | 24 |
| 4.5.2 | Approach 2a.5: Take it? Or leave it? | 25 |
| 4.6 | Solution of Problem 2b | 26 |
| 4.6.1 | Approach 2b.1: Minimize the total vacant time of all taxis . . | 26 |
| 4.6.2 | Approach 2b.2: Enlarge the market occupancy | 27 |
| 4.6.3 | Approach 2b.3: Communication system | 28 |
| 4.7 | Case Analysis | 29 |
| 4.8 | Sensitivity Analysis | 30 |
| 4.9 | Strengths and Weaknesses | 30 |
| 4.10 | Conclusion of Solution 2 | 31 |

#692 2

References **31**

Appendix **33**

1 Summary

1.1 Interpretation and Methods of Problem 1

We have to identify eleven officers. By analyzing the first category which includes eight quotes and three paragraphs, we may observe the typing patterns easily. Then we may identify and distinguish the officer by considering the second and third categories. The identifying person problem can be interpreted as a data matching problem. We divided the problem into approaches.

Before building our models, it is necessary to collect the data from the broken line graphs and the overview typing-pattern graphs of each officer. However, the data are pictures and graphics. It is difficult for us to analyze. So, we have to quantify them and transfer the typing speeds and accelerations into numbers. Such that we are able to analyze them by numerical methods.

The first approach is to categorize the words into different groups. This can be more accurate for us to find the typing patterns of each officer. By considering the average typing speed of each group of words, we are able to analyze the data by some calculations transforming the data into indexes representing the similarities and identify each officer.

The second approach is to determine if the officer is a left-hander or right-hander. It can be determined by considering the typing speeds of typing two consecutive letters from the same side. Then we can group the officers into three groups: left-hander, right-hander and undefined.

The third approach is to match the data by categories of officers by Hungarian Algorithm. It is a combinatorial optimization algorithm which can match the data to data. For the second category case, person K is said to have typed a different quote for test 1 from the rest of the ten other officers. We ignore that person as we only consider the speed of each category of words. For the third category case, we only collected from ten of the eleven officers. We solve this problem by using a special Hungarian Algorithm matching eleven officers to ten of them.

We can only test our model with only the speeds and accuracies as it was hard for us to transfer the graphical into numeric data. Through testing our model with randomly one person for each case, we were able to determine which officer he is. By knowing one of the officer, we can distinguish all 11 officers

1.2 Interpretation and Methods of Problem 2a

A taxi driver has a target to maximize his profit. When his taxi is vacant, he has basically build two decisions: searching around or accept a booking service. This can be divided into two problems. Both of the two cases can be interpreted as optimization problems where the objective is to maximize his profit. We divided the problem into two cases: i) Searching case and ii) Calling case.

For the Searching case, we divided the problem into three approaches. The first approach is to find the expected profit of travelling in that zone, including money cost, time cost and taxi fare. We obtained these values by analyzing the given data. The second approach is to find out the best route to travel with a destination zone, which is basically based on geometric distribution. We believed that there is a probability for a taxi driver to pick up a passenger at each zone. By adding the probabilities with expected profit, we can find the best route to travel. The third approach is to determine whether the driver should stay at a position or searching around in the zone.

For the Calling case, we divided the problem into two approaches. The first approach is to find the shortest path for the taxi to travel from the starting zone to the destination. As the position of destination is already known, the problem can be easily solved by Dijkstras Algorithm. The second approach is to determine the taxi drivers decision, whether he should take that job or not. We compared the expected searching time and the waiting time of the scheduled trip and consider the expected value of both cases.

1.3 Interpretation and Methods of Problem 2b

The taxi company has targets not only to maximize its profit, but to enlarge its market occupancy. These problems can also be interpreted as optimization problems.

We divided the problems into three approaches. The first approach is to distribute their taxis into different zones according to the normal distribution. The second approach is to enlarge the market occupancy by increasing the pick-up percentages of our taxis in the high pick-up density zones. This can be easily derived by simple calculations. The third approach is about the communicating system. An inseparably close communication system should be built between taxi drivers and the taxi company in order to obtain the immediate traffic and passengers conditions.

2 Introduction

2.1 Background Information of Problem 1

It is shown that the typing patterns can be used to identifying a person. This method is also widely used in companies nowadays. Keystrokes dynamic is one of the biometrics to identify person, which is distinguishing a person by the staying time and the flying time at a key.

In this section, we considered the speed and accuracy. We thought that typo is a problem cause by the state of the person including energy, environment etc. We excluded this factor in our consideration. We do not build he model by the whole paragraphs but by the categories of words and even the combinations of letters. We believed that the solution will be more accurate if we consider as precious as possible.

2.2 Background Information of Problem 2

The taxis of New York City are widely recognized icons of the city. Taxis painted yellow are medallion taxis and those painted apple green are Boro taxis. Yellow taxis are allowed to pick up passengers anywhere in New York City while green taxis are allowed to pick up passengers in five boroughs excluding LaGuardia Airport, John F. Kennedy International Airport and in Manhattan above East 96th and West 110th Streets.

Taxi in New York City is one of the most common transportation. We believed that our models and methods can effectively provide immediate advice for not only the taxi drivers but also taxi companies. The taxi system or even transportation system may become more comprehensive and all-round.

3 Problem 1 Solution

3.1 Introduction of Model

We quantified both the given data and text data, and compared the values to identify the officers. The accuracies of the officers typing quotes or letters can be affected by many factors, but not by their typing patterns. Everyone has his/her own typing habits. In order to distinguish the officers, we have to find out some unique characteristics of each officer.

Then, we analyze the data and transform them into indexes of each category of each officer. Moreover, we determine whether the officers are left-hander, right-hander or undefined by analyzing every two consecutive letters. By obtaining these indexes information, we are able to distinguish the officers by Hungarian Algorithm.

3.2 Assumption and justification

Assumption: All the officers type the letters and quotes seriously.

Justification: All the officers are obligated to type seriously in this test, without altering others' typing patterns. This will otherwise affect the result of the solution.

Assumption: All the officers type every letters and quotes with same accessories.

Justification: The qualities of typing accessories, for example, keyboards, may widely affect the typing speed. As there are keyboard companies claimed that people typing with their keyboards may increase the typing speed.

Assumption: The typo mistake is not a kind of typing patterns.

Justification: As there are many factors affecting the typing accuracy such as environment, state of the person etc. The typo mistake should be treated as case independent.

3.3 Variables and Parameters

| Variables | Explanation |
|-----------------|--|
| P_{a_i} | The i^{th} given person |
| P_{b_i} | The i^{th} unidentified person |
| m | The number of groups of words |
| k_i | The number of words in the i^{th} group |
| $AS_P^{i,l}$ | The corresponding speed of the l^{th} word in the i^{th} group |
| TSP_P^i | The typing speed of person P when typing words in the i^{th} group of words. |
| HSP_P^i | The typing speed of person P when typing the |
| $freq_P^{RR_i}$ | The number of times of RR appears in the i^{th} word |
| $freq_P^{RL_i}$ | The number of times of RL appears in the i^{th} word |
| $freq_P^{LR_i}$ | The number of times of LR appears in the i^{th} word |
| $freq_P^{LL_i}$ | The number of times of LL appears in the i^{th} word |
| spd_P^{RR} | The average speed when typing consecutive "RR" |
| spd_P^{RL} | The average speed when typing consecutive "RL" |
| spd_P^{LR} | The average speed when typing consecutive "LR" |
| spd_P^{LL} | The average speed when typing consecutive "LL" |
| $S_{i,j}$ | The similarity of P_{a_i} to P_{b_j} |

3.4 Solution of Problem 1a – Category 2

In this section, we first find out the speed of an officer of different type of words. Then determine the unique characteristics of each officer.

3.4.1 Approach 1a.1: Categorize words into groups and find the speed

First, we are going to find out the speed of each officer typing each word. According to the broken line graphs, if the slope is positive, it means that the typing speed of that officer is accelerating, vice versa. We can map the speed of the word in the passage to the corresponding point in the broken line graph, and hence find the speed of typing each word.

Then we categorized the words into different groups by commonalities and lengths, using a research [1] about the frequency of English letter. For each group of words, we recorded the speeds and frequencies of accelerations or decelerations occur. There are two main reasons for a deceleration: a typo occurs or the officer not familiar with that word. As we know, when a typo occurs, the officer's typing speed will significantly decrease. However, typo occurs due to the state of the person, which is case independent, and randomly occurs. Hence, we do not consider the data of the word with typo in our calculations of typing speed in each group. These data are used to find the typo habits of the officers.

We denote P_{a_n} as the given person n , P_{b_n} as the unidentified person n , and TSP^m as the typing speed of person P when typing words in the m^{th} group of words. There are k_m word in each group, and the corresponding speed of the i^{th} word in the m^{th} group is $AS_P^{m,i}$. We can find the average typing speed of each group by

$$TSP^m = \frac{\sum_{i=1}^{k_m} AS_P^{m,i}}{k_m} \quad (1)$$

3.4.2 Approach 1a.2: Determine left-hander, right-hander or undefined.

If we only consider the above situation, we would miss a crucial factor to identify a person: whether the officer is a left-hander or right-hander. So, we need to know whether he types faster with left or right hand. The speed ratio of the person typing with right hand to left hand reflects whether he is a left-hander or right-hander. At the same time, it determines how well his both hands can cooperate with each other, as an extra information. As long as he is a right-hander and types words faster with right hand than left hand most of the time, he would have this habit remained even when he is typing different passages. That's why we said this as a crucial factor to identify a person.

We represent alphabets that supposed to be typed with right hand by R, and left hand by L. With this rule, we could represent any words with a sequence that consists of L and R only. Consider every pair of consecutive alphabets, e.g. consecutive RR in the new generated sequence represents that the corresponding alphabets are both

typed with right hand. We would like to know the typing speed of a person when typing the consecutive right-hand alphabets, so do RL, LR, LL. By comparing the speed of typing RR and LL, we may know whether the person is a left-hander or right-hander.

Now, we are going to demonstrate the calculation of the speed of RR. For each word, we will first count the number of times that RR appears in the sequence. Repeat the procedure again on RL, LR, and LL. For example, considering the word question, u,i,o,n are supposed to be typed with right hand, while q,e,s,t are supposed to be typed with left hand. Hence, we represent "question" as LRLLLRRR. In this case, there are 2 RR, 1 RL, 1 LR and 2 LL.

Then, supposed that the person types the i^{th} word with a speed of HS_i WPM. We denote the number of times of RR, RL, LR, LL appear in the i^{th} word as $freq_P^{RR_i}$, $freq_P^{RL_i}$, $freq_P^{LR_i}$, $freq_P^{LL_i}$ respectively. We denote the average speeds of typing consecutive RR, RL, LR, LL as spd_P^{RR} , spd_P^{RL} , spd_P^{LR} , spd_P^{LL} respectively. So, spd_P^{RR} is equal to the weighted mean of the speed of each word that RR appear, as well as spd_P^{RL} , spd_P^{LR} , spd_P^{LL} . We denoted c as the number of words in the passage.

$$spd_P^{RR} = \frac{\sum_{i=1}^c freq_P^{RR_i} \times HS_i}{\sum_{i=1}^c freq_P^{RR_i}} \quad (2)$$

We are now able to determine whether the person types faster with right hand or left hand by comparing spd_P^{RR} with spd_P^{LL} . In addition, we may know whether the person's two hands can cooperate with each other properly by comparing the speed of spd_P^{RL} , spd_P^{LR} with spd_P^{RR} , spd_P^{LL} . The ratio of these four speeds can also be counted as a unique characteristic too.

By considering all the above data, excluding the typo habits, each person has an array of the values to represent them. We denote RR_P as the speeds of typing consecutive right-hand alphabets by person P , as well as RL_P , LR_P , LL_P . We can now generate an array of data for each person.

After calculating the values of RR_P , RL_P , LR_P , LL_P , we can try to determine whether person P is a left-hander or right-hander. We classify person P as a right-hander if $\frac{RR_P}{LL_P} > tr$, and classify person P as a left-hander if $\frac{LL_P}{RR_P} > tl$. The person P is said to be well cooperation with both hand if $\frac{RL_P+LR_P}{RR_P+LL_P} > tg$, the person P is said to be poor cooperation with both hand if $\frac{RR_P+LL_P+RL_P+LR_P}{RL_P+LR_P} > tb$. In order to find the threshold tr , tl , tg , tb , we examined them with some left-handers, right-handers, the people who are able to cooperate with two hands properly. Then we can categorize the people into 9 groups. If there are more than 1 people in a group, we will go on to the next approach to distinguish them.

3.4.3 Approach 1a.3: Hungarian Algorithm

We are going to find the similarity of the arrays between P_a and P_b . To find the similarity, we are going to compare each value of the arrays and record them as indexes. We denoted S as a $N \times N$ matrix. $S_{i,j}$ denoted the similarity of person P_{a_i} to person P_{b_j} can be written as the following

$$S_{i,j} = \prod_{l=1}^m \left(1 + \frac{|TS_{a_i}^l - TS_{b_j}^l|}{TS_{b_j}^l} \right) \quad (3)$$

By using Hungarian Algorithm, we are able to find the optimal pairing of a to b , which minimize the sum of $M_{i,j}$ of each pairing (i, j) .

Step 1: For each $i = 1..N$, find the minimum element of row i . Subtract each element of the row by that minimum element. That would create at least a zero in the row. (More than 1 zero will be created if there are more than one minimum element)

Step 2: Similarly, for each column $j = 1..N$, subtract each element of the column by the minimum element of that column.

Step 3: Use the least amount of lines to cover up every zero in the matrix. If the number of lines that used to cover is equal to N , jump to Step 5. Otherwise, move on to Step 4.

Step 4: Denote m as the minimum element not covered by line. Add m to every element intersected by two lines. Subtract m to every element not covered by any line. Repeat Step 3.

Step 5: An optimal pairing lies within the zeros of the matrix. Find the combination of N zeros that no two lie in the same row nor same column. If $M_{i,j}$ is one of the chosen zero, that means A_i is paired with B_j . This combination corresponds to the minimum sum of $M_{i,j}$ in the original matrix.

There might be more than one combination that yields that minimum sum though. We would choose the one that the standard deviation of the similarity is minimize.

After all these, we can map the given people to tested people.

3.5 Solution of Problem 1b – Category 3

In the case of letter, all participations type some random generated alphabets, and the data of speeds and accuracies are collected. We are again asked to map the give people to the tested people. We similarly used the broken line graph to find the typing speed of each word.

3.5.1 Approach 1b Categorize the words and find the typing speed

We divided the words to different groups with a different algorithm. As long as all words are not common to officer, we cut the words into the length of each word - 1 pairs of consecutive alphabets. Hence there will be 26×26 combinations of the consecutive alphabets. Considering the letter case, although some words are uncommon to the officers, there will still be some commonalities in between. According to some researches, they said that there are different commonalities for each two-letter sequence. Following to those researches, we divided the two-letter sequence into m groups by commonality.

To distinguish the people, we use a method similar to Typing Speed method. HS_P^j in this section also represent the j^{th} word. We denote $freq_P^{g,j}$ as the number of times the consecutive alphabets g appears in the j^{th} word in group. Denote spd_m as the average typing speed of group m which can be obtained by the following

$$spd_P^m = \frac{\sum_{i=1}^k freq_P^{m,j} \times HS_P^j}{\sum_{i=1}^k freq_P^{m,j}} \quad (4)$$

We are going to follow approach 1a.2 using the "Left-Right hand method" to categorize people into 9 groups.

Then we follow approach 1a.3 using the same way to calculate the similarity rates, generate a matrix to do Hungarian Algorithm, and can distinguish all the people.

3.6 Case analysis of problem 1b

Our algorithm depends largely on analyzing the overview typing-pattern graphs and the broken line graphs to get the typing speed of each words. Since the given graphs are too rough, we couldn't extract the required information effectively. We could not categorize the people into groups. So we supposed they are all in the same group.

By substituting the given average speed into the formula, we could yield the matrix M .

$$M_{1,1} = \prod_{l=1}^1 \left(1 + \frac{|83.325 - 102.75|}{102.75}\right)$$

$$M_{1,1} = 1.189$$

Keep doing this, and we can generate the below matrix.

```

1.189 1.260 1.005 1.503 1.243 1.476 1.174 1.402 1.068 1.478 1.027
1.423 1.104 1.285 1.069 1.462 1.050 1.413 1.003 1.240 1.051 1.269
1.413 1.089 1.273 1.088 1.453 1.068 1.403 1.014 1.227 1.069 1.257
1.019 1.583 1.264 1.889 1.049 1.855 1.038 1.762 1.343 1.858 1.291
1.001 1.554 1.241 1.855 1.066 1.821 1.019 1.730 1.318 1.824 1.268
1.013 1.534 1.224 1.830 1.079 1.797 1.005 1.706 1.301 1.799 1.251
1.190 1.259 1.005 1.502 1.244 1.475 1.175 1.401 1.068 1.477 1.027
1.413 1.089 1.273 1.088 1.453 1.068 1.403 1.014 1.227 1.069 1.257
1.415 1.092 1.275 1.084 1.455 1.064 1.405 1.010 1.230 1.065 1.259
1.379 1.036 1.230 1.151 1.421 1.130 1.368 1.073 1.182 1.131 1.213
1.269 1.135 1.094 1.354 1.318 1.330 1.256 1.263 1.037 1.332 1.074

```

Then we follow the procedure of the Hungarian Algorithm and generate the final matrix:

```

0.191 0.261 0.000 0.431 0.209 0.423 0.176 0.403 0.069 0.424 0.000
0.427 0.108 0.282 0.000 0.430 0.000 0.417 0.007 0.244 0.000 0.244
0.399 0.074 0.251 0.000 0.403 0.000 0.388 0.000 0.213 0.000 0.214
0.006 0.570 0.243 0.802 0.000 0.788 0.025 0.748 0.329 0.789 0.249
0.000 0.554 0.233 0.781 0.030 0.766 0.018 0.729 0.318 0.767 0.238
0.007 0.528 0.212 0.751 0.038 0.737 0.000 0.701 0.295 0.738 0.217
0.192 0.261 0.000 0.431 0.210 0.423 0.177 0.403 0.070 0.424 0.000
0.399 0.074 0.251 0.000 0.403 0.000 0.388 0.000 0.213 0.000 0.214
0.406 0.082 0.259 0.001 0.409 0.000 0.395 0.001 0.220 0.000 0.221
0.344 0.000 0.189 0.043 0.349 0.041 0.332 0.037 0.147 0.041 0.150
0.232 0.098 0.051 0.244 0.245 0.239 0.218 0.225 0.000 0.240 0.009

```

And hence by observing the location of 0.000, we can find out the mapping of

$$\begin{array}{lll}
 1 \rightarrow 3, 11 & 2 \rightarrow 4, 6, 10 & 3 \rightarrow 4, 6, 8, 10 \\
 4 \rightarrow 5 & 5 \rightarrow 1 & 6 \rightarrow 7 \\
 7 \rightarrow 3, 11 & 8 \rightarrow 4, 6, 8, 10 & 9 \rightarrow 6, 8 \\
 10 \rightarrow 2 & 11 \rightarrow 9 &
 \end{array}$$

with P_{a_i} being the given person and P_{b_i} being the unidentified person in $P_{a_i} \rightarrow P_{b_i}$.

3.7 Sensitivity Analysis

A sensitivity analysis can be done to see how sensitive our model response when a change is made in a data.

Consider there is an error while collecting data. Let's say there is an error ew (where $ew < 1$ of P_{a_j} 's typing speed of the i^{th} word. If the ew is large in some case, the new speed of typing the i^{th} word will be $HS_i \times (1 - ew)$. The person may suddenly type a common short word slowly, while he is not used to typing at this speed. For the calculation finding the average speed, it will only be slightly affected when each group have sufficient many words.

So, the error of TS_p^m will be small enough, a lot smaller than ew . With this new speed, the error of $1 + \frac{|TS_{a_j}^l - TS_{b_l}^l|}{TS_{b_l}^l}$ for all l will be even smaller, and denote the error es . For an l , there will be an original $S_{j,l}$. The new similarity rate will be obtained by

$$S_{j,l} \times \frac{1 + (1 + es\%) \times spd_p^m}{1 + spd_p^m} \quad (5)$$

The actual error in the similarity rate will be

$$S_{j,l} \times spd_p^m \times es\% \quad (6)$$

Hence if the value of $S_{j,l} \times spd_p^m$ is large, the error will be large, else if the value of $S_{j,l} \times spd_p^m$ is small, the error will be small, and probably tends to 0. However, when $S_{j,l} \times spd_p^m$ is sufficient large, it means the two people are not similar to each other, so even though the error is big, we would not map the two people together, and hence this error will not affect our final solution. When $S_{j,l} \times spd_p^m$ is small, and tends to zero, the error will be sufficient small. It will not affect our calculation too.

For the Hungarian Algorithm, we use subtraction instead of multiplication, therefore the error is not exaggerated.

As we mention above, the similarity rate will barely change in the useful data. Hence, even if we make some minor mistakes while typing or leads to a speed calculation error, it will not be too sensitive and can still get the appropriate answer.

3.8 Strengths and Weaknesses

Strengths:

Our solution consider very detailed information, not only by words, but by alphabets and hands. This may increase the accuracy of mapping the unknown person to the known. Base on our sensitivity analysis, we know that our model will not be too sensitive to some error of typing speed. Hence our model are quite stable.

Also we first divide the people into different zone. This can completely increase the accuracy of mapping the the given person to the unidentified person. As the probability of mapping the given person to the unidentified person will be smaller.

Weaknesses:

Our approach is well defined already. However, the input of the data is one of our weakness. As we only have broken line graphs as the given data, it is hard for us to quantify the broken line graph without some graphic software. So we cannot do the case analysis here.

In the section 3.4.2, the way we determine whether a person is a left-hander, right-hander or undefined is by doing finitely many examination of different people, which is time consuming for us. Hence this will be the second weakness for us.

3.9 Conclusion of Solution 1

In this section, we have first quantified both the given data and text data in order to analyze. We first categorize the words into groups by their lengths and commonalities. Then, arrays are generated to represent each person's performance, or probably typing patterns. Also, we consider whether the person is left-hander or right-hander and the performance of cooperation between both hands.

Then, we map them in order to identify them, pairing up the unknown person to known person by the **Hungarian Algorithm**.

4 Problem 2 Solution

4.1 Introduction of Model

About 500,000 data are provided. By having sufficient large amount of data, we can have accurate estimations. Due to the major factors of the decision are the money cost, time cost and taxi fare, accurate estimation is of utmost important.

We divide the New York City into zones by its latitudes and longitudes. We can analyze the provided data to find the information of each zone, including the pick-up number, drop-off number, average taxi fare of trips. It is too time consuming to make the decision by considering each node or road. Zone is the most suitable unit for routing.

We divide the given data into time intervals. As the number of passengers hailing on the road may change over time, taxi driver will have different strategies at different time in order to maximize his profit.

For the taxi driver, we built the mathematical models mainly base on the expected fares and the probabilities of meeting passengers which can be written as geometric distribution.

For the searching case, the taxi driver is eager to get a passenger and starting a trip as soon as possible required that the destination of the trip is convenient enough for the taxi driver to find the next passenger. As there are no means to compare the currency and time with different units, we transform both factors into indexes which are able to be compared. By weighting the factors by the user perception, we can conclude the taxi drivers concerns and find the most suitable route and decisions for them.

Calling case is another major job for taxi driver to take. Actually, this case can be treated as a special case for searching case which provided destination zone and without probabilities and expected searching times. This problem can also be treated as the shortest path problem. It can be easily solve by the Dijkstra's Algorithm.

For the taxi company, we built the mathematical models mainly based on normal distribution and the probability density function. It can be a simple guideline for a company to distribute their taxis. By following the guideline, the company is able to minimize the total vacant time of all taxis. In order to improve the strategy, we consider the market occupancy. What we have to do is just slightly modify the probability density function a little bit. In addition, we want to develop a strong communication system between taxi drivers and taxi company. It can help taxi drivers find the best route much easier.

4.2 Assumptions and Justifications

Assumption: Green taxis do not pick-up passengers in excluded area.

Justification: According to the taxi law for green taxis in New York City, they are only allowed to pick up passengers in outer boroughs (excluding JFK International Airport and LG Airport) and in Manhattan above East 96th and West 110th Street.

Assumption: Taxi drivers do not skip and ignore passengers.

Justification: Taxi drivers are obligated to pick up the passengers hailing on the street.

Assumption: Taxi drivers are able to meet a passenger on the road in a sufficient long time.

Justification: As there are still a probability for a taxi driver to meet a passenger, he will finally meet one for sufficient long time.

Assumption: Those taxis with store and forward system are considered to be our opponent.

Justification: Those taxis cannot cooperate with our taxi drivers and company as they did not install the communication system. The head quarter do not know their positions and conditions. The company is not able allocate them.

4.3 Variables and Parameters

| Variables | Explanation |
|----------------|---|
| F_n^t | The expected taxi fare of zone n in time interval t. |
| S_n^t | The expected searching time to meet a passenger of zone n in time interval t. |
| N_n^t | The expected searching time of the next trip of zone n in time interval t. |
| M_n^t | The expected money cost of zone n in time interval t. |
| \bar{F}_n^t | The average taxi fare of whole New York City in time interval t. |
| \bar{S}_n^t | The average searching time to meet a passenger of whole New York City in time interval t. |
| \bar{M}_n^t | The average money cost of whole New York City in time interval t. |
| \bar{N}_n^t | The average searching time of the next trip of whole New York City in time interval t. |
| w_F | The user perception factor of the taxi fee. |
| w_S | The user perception factor of the searching time. |
| w_M | The user perception factor of the money cost. |
| w_N | The user perception factor of the next trip searching time. |
| p_n^t | The probability of meeting a passenger in zone n in time interval t. |
| u_n^t | The total pick-up number of zone n in time interval t. |
| b_n | The total number of blocks of zone n. |
| Y_n^t | The time required to pass travel through zone z in time interval t. |
| q | The length of each time interval. |
| \bar{d}_n^t | The average distance traveled by taxi drivers starting from zone n in time interval t. |
| $PT_{H_l^k}^t$ | The passing time of a taxi from hub $l - 1$ to hub l . |
| Z_n | The n^{th} zone. |
| R^k | The k^{th} route. |
| H_l^k | The l^{th} hub in rout k. |
| p_l^{kt} | The probability to travel to the l^{th} hub in route k. |
| V_n^t | The expected value of zone n in time interval t. |
| $V_{H_l^k}^t$ | The expected value of hub l in route k in time interval t. |
| V^{kt} | The expected value of route k in time interval t. |
| D_n^t | The pick-up density of zone n in time interval t. |

| Variables | Explanation |
|------------|--|
| a_n | The area of zone n. |
| V_{nC}^t | The expected value the route provided destination C as the calling location. |
| V_{nS}^t | The expected value the best route of searching case. |
| W | The waiting time from the moment he take the job to the scheduled time. |
| V_{nC}^t | The expected value the route provided destination C as the calling location. |
| P_n^t | The pick-up number of our company taxis of zone n in time interval t. |
| O_n^t | The pick-up number of the opponent company taxis of zone n in time interval t. |

4.4 Solution of Problem 2a – Searching Case

4.4.1 Approach 2a.1: Finding the expected income of each zone

The first step of devising an optimal strategy for the taxi driver is to find the expected profit of travelling in each zone. The expected profit of travelling in each zone is different in every time interval. The factors which have to be considered are the expected taxi fare (F), expected searching time (S), expected money cost (M) and expected searching time for next trip (N).

Now, the taxi driver is in Z_n . There are K different routes (R^K) passing through zones and end at a destination zone. In route R^k , the taxi driver starts at Z_n , then go to the next zone, on and on, till it reaches its destination zone. We treat the zones passing through as hubs and using H_l^k to denote the l^{th} hubs of route k , while there are in total L hubs in route k .

There is a probability of meeting a passenger in every zone n in every time interval t which can be define as the pick-up number u per number of blocks b in that zone times travel time R per time interval q .

$$p_n^t = \frac{u_n^t}{b_n} \times \frac{Y_n^t}{q} \quad (7)$$

Firstly, we consider the expected taxi fare F_n^t of every zone n in different time interval t by averaging the trip distance travelled.

| | |
|----------------|-----------------------|
| Charge | Fee |
| Initial charge | \$2.50 |
| Mileage | \$0.4 per 0.2 mile |
| Waiting charge | \$0.4 per 120 seconds |

Table 1: Calculation of taxi fare

The value will be given by:

$$F^{nt} = \$2.50 + \$0.4 \times \frac{\bar{d}_n^t}{0.2} \quad (8)$$

Divide the average taxi fare of zone n by the average taxi fare of the whole New York City in time interval t (\bar{F}^t). Then, we may have the expected taxi fare index.

Secondly, we consider the expected searching time S_n^t of every zone n in different time interval t . We can obtain the passing time for each hub l by multiplying the velocity to the distance. In order to find the expected searching at the destination zone n , we set the probability = 1 as we assume that we will finally meet a passenger for a

certain time.

$$1 = \frac{u_n^t}{b_n} \times \frac{Y_n^t}{q}$$

$$Y_n^t = \frac{b_n \times q}{u_n^t}$$

By having the time required for the taxi driver to meet a passenger in the destination zone n , we can obtain the full equation for the expected searching time.

$$S_n^t = \sum_{l=1}^L \left(\left(\sum_{i=1}^l PT_{H_i^k}^t \right) p_{H_i^k}^t \prod_{i=1}^l (1 - p_{i-1}^{kt}) \right) + \frac{b_n \times q}{u_n^t} \quad (9)$$

Divide the average searching time of zone n by the average searching time of the whole New York City in time interval t (\bar{S}^t). Then, we may have the expected searching time index.

Thirdly, we consider the expected money cost M_n^t of every zone n in each time interval t . The money cost is equal to the gas cost. The gas cost is affected by the distance travelled, gas price and the MPG. It can be obtained by

$$M_n^t = \frac{\text{gas price}}{MPG} \times \bar{d}_n^t \quad (10)$$

By inputting the expected searching time, the money cost will be obtained. Similarly, we divided the average searching time of zone n by the average money cost of the whole New York City in time interval t (\bar{M}^t).

Lastly, we consider the expected searching time for the next trip. As taxi driver is unlikely to travel to some zones which required him a long period to find another passenger, this factor is cannot be ignored. Once the taxi driver dropped off the passenger, he starts searching a new passenger immediately. We treat the drop-off zone as the new starting zone. As we are just calculating the expected searching time, we can obtain the value (N) by equation (3) with a different time interval t and a different starting zone n .

$$N_n^t = \sum_{l=1}^L \left(\left(\sum_{i=1}^l PT_{H_i^k}^t \right) p_{H_i^k}^t \prod_{i=1}^l (1 - p_{i-1}^{kt}) \right) + \frac{b_n \times q}{u_n^t} \quad (11)$$

Combining the four indexes with independent weights, we can obtain the expected value of every zone n in every time interval t by the following equation.

$$V_n^t = w_F \times \frac{\bar{F}_n^t}{\bar{F}^t} + w_M \times \frac{\bar{M}_n^t}{\bar{M}^t} + w_N \times \frac{\bar{N}_n^t}{\bar{N}^t} + w_T \times \frac{\bar{S}_n^t}{\bar{S}^t} \quad (12)$$

4.4.2 Approach 2a.2: Finding the expected value of each route

For route k , we have a starting zone H_0^k and a destination zone H_L^k . The taxi, will pass through $L - 1$ zones and finally stay at the destination zone L until he meets and pick up a passenger.

By geometric distribution, we can find the probability of meeting a passenger in every hub l of route k . The probability function is given by

$$p_l^{kt} = p_{H_l^k} \times \prod_{i=1}^{l-1} (1 - p_{i-1}^k) \quad (13)$$

The expected value of meeting a passenger in every hub l can be obtained by multiplying the probability that might be gained by the taxi driver of the hub. By summing all possible expected values with its probability, we can find the expected value of each route k . The expected value is given by

$$V^{kt} = \sum_{l=1}^L V_{H_l^k}^{kt} \times p_l^{kt} \quad (14)$$

The final equation can be written as:

$$V^{kt} = \sum_{l=1}^L V_{H_l^k}^t \times p_{H_l^k} \times \prod_{i=1}^{l-1} (1 - p_{i-1}^k) \quad (15)$$

Finally, a list of expected values of each route will be generated. The taxi driver may easily find the best routes by sorting the list.

4.4.3 Approach 2a.3: Determining whether stay or move

However, there are still two strategies remaining for a taxi driver to choose: i) stay at a position to wait a passenger or ii) searching around in the zone to meet a passenger.

The decision is mainly determined by the pick-up density in the zone. If the pick-up density of the zone is dense enough, it is easy to meet a passenger by searching around. If the density of the zones is not dense enough, it is hard to meet a passenger by searching around, such that the taxi driver is suggested to stay at a position and wait until he meets a passenger. The position can be found by determining the pick-up number of each point.

The pick-up density of every zone n at time t can be obtained by:

$$D_n^t = \frac{u_n^t}{a_n} \quad (16)$$

The pick-up distribution is also considered in this section. If the distribution is discrete, the taxi driver is suggested to search around in the zone as it can highly increase the probability of meeting a passenger. If the distribution is concrete, the taxi driver is suggested to stop and wait at the concrete point as it can not only reduce the gas cost usage but also increase the probability to meet a passenger.

People in New York City have different activities at different times. For example, people usually have meals at 7 a.m., 1p.m. and 7p.m. Also, people may usually go to work at 8 a.m. and get off work at 6 p.m. Then, the taxi driver is suggested to wait outside the restaurant at 8a.m., 1p.m. and 7p.m. He is also suggested to wait outside the apartments or housing at 8a.m. and around the office at 6p.m. By considering the human habit, the probabilities of meeting a passenger of such waiting strategies are largely increased.

4.5 Solution of Problem 2a – Calling Case

4.5.1 Approach 2a.4: Dijkstras Algorithm

It is known that passengers can use the calling service to call taxis at a certain time instead of hailing taxis on the street. In this section, we are going to discuss how the taxi driver make the decision, whether he should take that passenger or not. If yes, what is the best route to meet the passenger?

First, we have to find the best route for the taxi driver to arrive at the passenger location. Second, we want to consider if the taxi driver is able to pick up a few more passengers before scheduled time of the taxi booking, as long as the best route.

Then, we have to minimize the time cost and the money cost. As the money cost is proportional to the time cost, we only have to minimize the time required to arrive the location of the passenger.

By considering and analyzing the traffic condition, we find the expected time required to pass through each road. With all of these data, we can use Dijkstra's algorithm to find out the minimum time required from the starting point to the destination, and the corresponding route.

Dijkstra's Algorithm is an algorithm for finding the shortest paths. We treat every zone as node. The taxi driver starting zone is called the initial node. Assign to every node a tentative time value: set it to 0 for our initial node and to infinity for the others. For the current node, consider all the neighbors and calculate their tentative time. Compare the newly calculated tentative time to the current assigned value and assign the smaller one. Keep looping until the destination node has been marked visited.

4.5.2 Approach 2a.5: Take it? Or leave it?

Although the shortest time to arrive to the location has been found, this is not the end of the story. The taxi driver can make a decision whether he should take this job or not. If not, he is going to search around by following routes. As the travelling time, destination and the passenger location have already been known, we are going to compare the job value with the expected value obtained in the approach 2a.2 equation (6).

We modify equation (6) a little such that it is suitable for the calling case. The expected searching time of the calling case job is canceled as we have already known a trip. However there is still a time for the taxi driver to wait for the passenger. We replace the expected searching time into waiting time. The new equation can be written as

$$V_{nC}^t = w_F \times \frac{\bar{F}_n^t}{\bar{F}^t} + w_M \times \frac{\bar{M}_n^t}{\bar{M}^t} + w_N \times \frac{\bar{N}_n^t}{\bar{N}^t} + w_T \times \frac{W}{\bar{T}^t} \quad (17)$$

By comparing the expected value of calling case V_{nC}^t and searching case V_{nS}^t , we can know whether he should take the job or not. If $V_{nC}^t > V_{nS}^t$, the taxi driver is told to take that job. If $V_{nC}^t < V_{nS}^t$, the taxi driver is told not to take that job.

4.6 Solution of Problem 2b

4.6.1 Approach 2b.1: Minimize the total vacant time of all taxis

We use normal distribution as a basic model, as well as a guideline, for the company allocating their taxis to different zones. According to the probability density function of the pick-up number of the whole taxi business model, we distribute corresponding amount of taxis to different zones. The probability density function is used to specify the probability of random variable. Pick-up number represents the number of passengers in that zone. As we have already analyzed the pick-up data of the past year, and assuming that the recent statistic and street-hail cases distribution are similar to that of last year, we can conclude that we should allocate more taxis to those zones which have a high pick-up number, vice versa. Taxi drivers are much easier to meet a passenger hailing on the street.

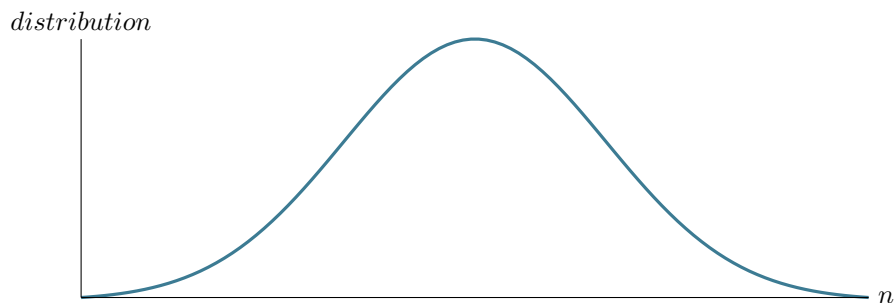


Figure 1: Normal distribution

A normal distribution can tell us how many taxis we have to allocate to different zones. For example, assume that there are 10 zones and we have 100 taxis. If the pick-up rate of one zone is 20% of the whole area, we are going to allocate 20 taxis

So, for a company, what they have to do is simply following the probability density function, and allocating the corresponding amount of taxis to different zones and to pick up an expected corresponding amount of passenger in that zone. This can increase the probability of a taxi driver picking up a passenger as long as minimizing the vacant time of all taxis to gain the greatest profit.

4.6.2 Approach 2b.2: Enlarge the market occupancy

Is it always the best way to compete with our opponent by following the normal distribution of passengers? Enlargement of the market occupancy is another task for a company. However, the number of taxis is limited by the government. How can a company win the competition by remaining the number of taxis?

The original market occupancy of our company is the percentage of our pick-up number to the total pick-up number. The total pick-up number can be described as the sum of the pick-up number of both our taxis and opponent taxis. The equation can be written as:

$$\text{Market occupancy of zone } n = \frac{P_n}{P_n + O_n} \tag{18}$$

The company is eager to win the competition in the denser pick-up zones. The pick-up density represents the number of passengers hailing per area. If the pick-up density is high, it means there are comparatively many passengers hailing on the street. However, those passengers may be picked up by the opponent taxis. In order to gain a better benefit and win the competition, we are more eager to maintain higher pick-up percentages in the high pick-up density zones than that of the opponent company. Such that we put more taxis to the corresponding zones. The pick-up percentage can be defined as Formula.

Assuming the opponent company do not do any actions or strategies to adjust their taxis distribution, increasing the number of taxis can increase the pick-up percentages of our company in those zones. It can be shown by

$$\text{Market occupancy of zone } n = \frac{P_n + \Delta P_n}{P_n + \Delta P_n + O_n} \tag{19}$$

By simply allocating more taxis to the zone, we can easily increase the company pick-up number in that zone. Base on the probability density function obtained in approach 1, we can slightly modify it. We want the function to become thinner with a smaller value of σ .

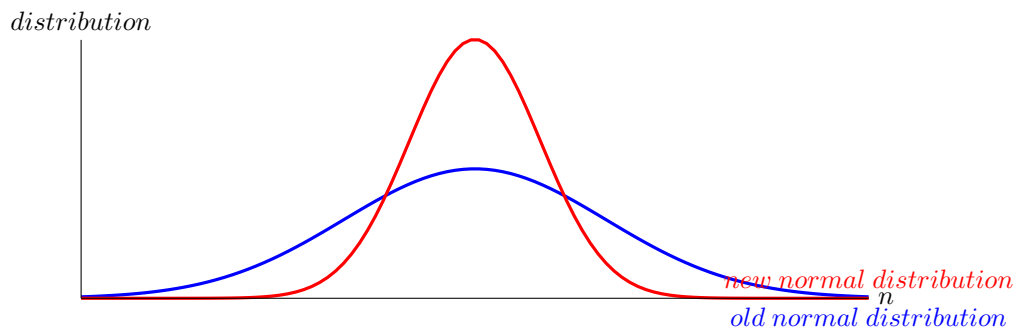


Figure 2: New normal distribution

4.6.3 Approach 2b.3: Communication system

For all taxi drivers, they are obligated to report the road conditions immediately, including the traffic and the position of passengers hailing on road. By letting the company know the passengers positions, they can allocate their taxis to the point by finding the minimum time.

We treat that passenger hailing on the street as a calling case. This is as same as our approach 2a.4. We can consider not only the given data but the By inputting the positions of the hailing passengers and positioning the recent vacant taxis, we generate an array of the required time for taxis to arrive and meet the hailing passengers. We can find the shortest time among all routes by the Dijkstra's Algorithm.

If the shortest time required is smaller than the opponent average pick-up time, it is worth for the company to allocate the corresponding taxi and pick up that passenger, vice versa. That passenger is said to be taken by the opponent taxis. The opponent average pick-up time can be estimated by considering the given data.

4.7 Case Analysis

We divided the whole New York City into 324 zones. Now, we are going to find the route with maximum expected value for a taxi driver in zone Z_{91} at 12n.n. By inputting these data into our algorithm, we may have the following consideration.

First we considered the probability of each zone which can pick up a passenger and the expected value of each zone to calculate the best route by using the 500,000 entries of given data. For the probability of each zone, we estimate the number of blocks as 50, the length of each time interval as 120 mins.

Then, for the taxi driver we are now helping, his user perception is mainly to reduce the searching time to find a passenger. The user perception is said to be the following:
 $w_F = 2, w_S = 8, w_M = 2, w_M = 1$

After generating a list of array and sorting, the best route has been found. The best decision for him is to stay in the zone Z_{91} . It is sensible to stay in zone Z_{91} as the pick-up number is the largest at 12 n.n.

4.8 Sensitivity Analysis

Our models for problem 2a is not sensitive. The solution of the problem is considered by the weighted value of each factor. Our algorithm is a general strategy applicable to all taxi drivers by letting them input their own user perceptions. However, the searching time may be affected by the traffic condition. If there is an accident happened in a zone or block, the taxi driver may not know the traffic condition immediately.

Our models for problem 2b is a little bit sensitive than that of problem 2a. Our models are only for the case provided that there is a sufficient large number of taxis of a company and is about the same as the opponent company in our case. If the company size is dramatically decreased or increased and is fall short by a large amount to the opponent, our strategies are not suitable. There are no mean for a large company to compete with a small company due to the large market occupancy. At the same time, there are also no mean for a small company to compete with a large company due to the lack of taxis.

4.9 Strengths and Weaknesses

Strengths:

Our solution of obtaining the expected values of routes and zones are constraint by four factors. The four factors are weighted by the user perceptions. They can be changed by the taxi driver. Also, we considered the value changed by time. The solution is changed by time. In addition, we considered the expected congestion time in the expected searching time. The road condition can be also calculated. This solution is all-round as we put different factors into our consideration.

Weaknesses:

Our solution is based on the given data. The values we used in our calculation are only estimated with the given data. The expected profit and searching time are not accurate enough as the pick-up number of each zone may change by years. However, we are not able to exactly predict the pick-up number of each zone in the future. We can only analyze the past data and predict the future.

In addition, our expected values are based on probabilities. There might be a big error to the reality. There are still probabilities for a driver travels to get a voided trip in a high expected-value zone or to pick up a passenger easily in a low pick-up number zone.

4.10 Conclusion of Solution 2

In problem 2, we attempted to make the decision for both taxi driver and taxi company. We divided the problem into three parts: searching case, calling case and company decision.

Searching case refers to the taxi driver travels through and around zones to meet a passenger. To find the expected value of each zone and route, we use **the geometric distribution** and weighting by taxi driver.

Calling case refers to the taxi driver travels through zones to a given zone at a scheduled time. To find the shortest path as well as the shortest time, we use **the Dijkstra's Algorithm**.

Company decision refers to the strategies of the company to allocate their taxis to different zones. We use **the normal distribution** as a guideline.

References

- [1] English Letter Frequency Counts
<http://norvig.com/mayzner.html>

```

#include<bits/stdc++.h>
using namespace std;
#define N 100005
#define M 100000
#define INF 100000000.0
#define inf -100000000.0
char pickup_date[N][15], pickup_time[N][15], dropoff_date[N]
[15], dropoff_time[N][15], c;
double pickup_long[N], pickup_lat[N], dropoff_long[N], drop
off_lat[N], dist[N], fare[N];
double extra[N], tax[N], tip[N], toll[N], impro[N], total[N];
double M_long = inf, m_long = INF, M_lat = inf, m_lat = INF,
long_h, lat_w, dist_r, dist_c;
double fare_cnt[20][20], tot_fare, prob[25][25], gas_fee;
double sum_fare[20][20][N], sum_dist[20][20][N], VAL[20][20],
P[20][20], ANS[20][20], dp[20][20];
int tally[105][105], num_ban = 0, _time[N];
int p_h[N], p_m[N], d_h[N], d_m[N], tmp2, sum_cnt[20][20][N];
int cnt = 0, sum = 0, ttime = 9360 /*2016/1/6 12:00*/;
bool ban[N], visited[20][20];
struct region {
    int row, col;
} where_pick[N], where_drop[N];
int t_int(char z) {
    return z - 48;
}
int binary_search_L(int __time) {
    int left = 1, right = M, mid;
    while(left < right) {
        mid = (left + right) / 2;
        if(_time[mid] < __time) left = mid + 1;
        else right = mid;
    }
    return left;
}
int binary_search_l(int __time) {
    int left = 1, right = M, mid;
    while(left < right) {
        mid = (left + right) / 2;
        if(_time[mid] > __time) right = mid - 1;
        else left = mid;
    }
    return left;
}
int main() {

```

```

freopen("infile.txt", "r", stdin);
for(int i = 1; i <= M; i++) {
    scanf("\n%s%s%s%s", pickup_date[i], pick
up_time[i], dropoff_date[i], dropoff_time[i]);
    tmp2 = strlen(pickup_date[i]);
    _time[i] += t_int(pickup_date[i][tmp2 - 1]);
    if(tmp2 == 9) _time[i] += t_int(pickup_date
[i][tmp2 - 2]) * 10;
    _time[i] *= 1440;
    scanf("%lf%lf%lf%lf", &pickup_long[i], &pick
up_lat[i], &dropoff_long[i], &dropoff_lat[i]);
    scanf("%lf%lf%lf%lf%lf%lf%lf%lf", &dist[i],
&fare[i], &extra[i], &tax[i], &tip[i], &toll[i],
&impro[i], &total[i]);
}
m_long = -74.85704041;
m_lat = 40.49094009;
M_long = -73.40166473;
M_lat = 40.99926376;
long_h = (M_long - m_long) / 18;
lat_w = (M_lat - m_lat) / 18;
for(int i = 1; i <= M; i++) {
    p_h[i] = t_int(pickup_time[i][0]) * 10 + t_int
(pickup_time[i][1]);
    p_m[i] = t_int(pickup_time[i][3]) * 10 + t_int
(pickup_time[i][4]);
    d_h[i] = t_int(dropoff_time[i][0]) * 10 + t_int
(dropoff_time[i][1]);
    d_m[i] = t_int(dropoff_time[i][3]) * 10 + t_int
(dropoff_time[i][4]);
    _time[i] += p_h[i] * 60 + p_m[i];
    //printf("%d:%d, %d:%d\n", p_h, p_m, d_h, d_m);
    sum += fare[i];
    if(pickup_long[i] < m_long || pickup_lat[i] < m_l
at || pickup_long[i] > M_long || pickup_lat[i] >
M_lat) {
        ban[i] = 1;
        ++num_ban;
        continue;
    }
    dist_r = M_long - pickup_long[i];
    dist_c = pickup_lat[i] - m_lat;
    where_pick[i].row = floor(dist_r / long_h);
    where_pick[i].col = floor(dist_c / lat_w);
    if(!where_pick[i].row) ++where_pick[i].row;
    if(!where_pick[i].col) ++where_pick[i].col;
}

```

```

        for(int k = 1; k <= 18; k++) for(int l = 1; l <=
18; l++) sum_fare[k][l][i] = sum_fare[k][l][i - 1];
        sum_fare[where_pick[i].row][where_pick[i].col][i]
= sum_fare[where_pick[i].row][where_pick[i].col]
[i - 1] + to
tal[i];
        for(int k = 1; k <= 18; k++) for(int l = 1; l <= 18;
l++) sum_cnt[k][l][i] = sum_cnt[k][l][i - 1];
        sum_cnt[where_pick[i].row][where_pick[i].col][i]
= sum_cnt[where_pick[i].row][where_pick[i].col]
[i - 1] + 1;
        for(int k = 1; k <= 18; k++) for(int l = 1; l <= 18;
l++) sum_dist[k][l][i] = sum_dist[k][l][i - 1];
        sum_dist[where_pick[i].row][where_pick[i].col][i]
= sum_dist[where_pick[i].row][where_pick[i].col]
[i - 1] + dist[i];
        ++tally[where_pick[i].row][where_pick[i].col];
        fare_cnt[where_pick[i].row][where_pick[i].col]
+= total[i];
        tot_fare += total[i];
}
for(int i = 1; i <= M; i++) {
    //printf("%d: %d\n", i, _time[i]);
    if(ban[i] || dropoff_long[i] < m_long || dropoff_l
at[i] < m_lat || dropoff_long[i] > M_long || dropoff_lat[i]
> M_lat) {
        if(!ban[i]) ++num_ban;
        ban[i] = 1;
        continue;
    }
    dist_r = M_long - dropoff_long[i];
    dist_c = dropoff_lat[i] - m_lat;
    where_drop[i].row = floor(dist_r / long_h);
    where_drop[i].col = floor(dist_c / lat_w);
    if(!where_drop[i].row) ++where_drop[i].row;
    if(!where_drop[i].col) ++where_drop[i].col;
}
freopen("Distribution_P2.txt", "w", stdout);
printf("Max_Longitude: %.5lf\nMax_Latitude: %.5lf\n",
M_long, M_lat);
printf("Min_Longitude: %.5lf\nMin_Latitude: %.5lf\n",
m_long, m_lat);
printf("Number_of_banned_rides: %d\nPickup_place
distribution:\n", num_ban);
printf("Fixed_(Const)_time_interval_(Average_trip_time):%
%.5lf_minutes\n", (double) sum / M);

```

```

double avg_fare = (double)sum / M;
for(int i = 1; i <= 18; i++) for(int j = 1; j <= 18; j++) {
    if(!tally[i][j]) printf("_____"); else
    printf("%*d", 6, tally[i][j]);
    if(j == 18) printf("\n"); else printf("_");
}
printf("Average_fare_for_zone_(i,_j)\n");
for(int i = 1; i <= 18; i++) {
    for(int j = 1; j <= 18; j++) if(tally[i][j])
    printf("%.5lf_", 9, fare_cnt[i][j] / (double) tally[i][j]);
    else printf("_____");
    printf("\n");
}
int L = binary_search_L(ttime), R = binary_search_L
(ttime + 120);
for(int i = 1; i <= 18; i++) for(int j = 1; j <= 18; j++) {
    VAL[i][j] = 4.0 * (sum_fare[i][j][R] -
sum_fare[i][j][L - 1]) / avg_fare;
    VAL[i][j] += 3.0 * 17.5291326;
    VAL[i][j] += 2.0 * 21.0 * (sum_dist[i][j][R] -
sum_dist[i][j][L - 1]);
}
for(int i = 1; i <= 18; i++) for(int j = 1; j <= 18; j++) {
    P[i][j] = (sum_cnt[R] - sum_cnt[L]) / 100.0 * 60.0
* 0.292152221 / 24.0;
}
}

```