Estimation of OD matrix for traffic flow with incomplete data

Jiaming Cui

Qifan Yang

Yuyan Zhao

Directed by Qing Yan

Nanjing Foreign Language School

August 2014

Estimation of OD matrix for traffic flow with incomplete data

August 2014

Abstract

This paper lays its focus on OD matrix for traffic flows. As an important basis of traffic network planning and management, OD matrix cannot be directly acquired in real world because of all kinds of restrictions. We propose an algorithm that bases on incomplete traffic data, combines real-world data with theoretical model and estimates OD matrix. This paper obtains the prior matrix by computation geometry, acquires sample matrix, and applies Bayesian inference to acquire the trip production and trip attraction of OD nodes. Based on above, we apply the gravity model and use iteration to optimize the resulting OD matrix. The RFID data in this passage comes from several monitoring stations at Gulou district, Nanjing. With the help of Hash algorithm, traffic analysis and program modeling, this integrated algorithm can provide great insight for improvement in the estimation of OD matrix.

Key words: OD Matrix; Bayesian inference; gravity model

1 Introduction

1.1 Current Problems

Traffic is the basis of a well-functioning modern city. With the accelerating process of urbanization, however, traffic congestion problem is getting severer, and intelligent transportation system (ITS) is in serious need. As basic data of ITS, the *OD (Origin-Destination) matrix* and its estimation method play a primary role in optimizing urban traffic management and control. The accuracy of OD matrix directly influences the error magnitude of following steps and deserves our full attention.

In traffic planning, the most widely used source of OD matrix estimation is still traffic flow detection, and the study of OD matrix still mainly concentrates in backward estimation, using relatively complete data of traffic volume and applying various models and algorithms to acquire final OD matrix. However, with the application of technology such as *radio frequency identification (RFID)*, this expensive but accurate data source has become the new basis of estimation of OD matrix. Nevertheless, current RFID monitoring spots fail to cover most sections of traffic network and the lack of RFID data is an obstacle of wide application of such method.

1.2 Concepts

OD (Origin-Destination) Matrix is a table showing the *trip distribution* between origins and destinations in a traffic network. The trip distribution between node i and node j is the number of vehicles that goes from i to j in unit time period. Within a certain region, *OD nodes* refer to traffic nodes that can be origins or destinations, while the rest nodes are not OD nodes. Number of vehicles entering the region from an OD node in unit time is called the *trip production* of the node; number of vehicles leaving the region from an OD node in unit time is called the *trip attraction* of the node.

1.3 Estimation of OD Matrix

Traditional method of obtaining OD matrix is large-scale sampling investigation, which costs considerable human and financial resources. Meanwhile, the accuracy of OD matrix gained in such way is severely compromised. Therefore, using appropriate model and data collected by traffic monitoring equipment to estimate OD matrix becomes a valuable topic.

Backward estimation is another way to obtain OD matrix. The applied algorithms include generalized least square (GLS) estimation, maximum likelihood estimation, maximum entropy method, and Kalman filtering estimation. GLS estimation is based on least squares principle, and takes the weight matrix into consideration. With GLS, the error of prior matrix directly hurts the accuracy of final OD matrix. Maximum likelihood estimation has the most promising future, but the algorithm itself is too complicated. Maximum entropy method has the advantages of simple structure and convenient application, yet it has high requirements on traffic flow information. Kalman filtering estimation can effectively estimate time-dependent (dynamic) OD matrix, but filtering divergence remains a problem.

This paper proposes an integrated algorithm that combines theoretical model and real-world data, and uses incomplete RFID data to estimate OD matrix. In the future of urban traffic development, this algorithm can offer great assistance in analyzing RFID stations distribution and estimating OD matrix from minimum traffic data.

2 Theoretical Model

2.1 Symbol Description

t_{ij}: trip distribution between node i and node j

O_i: trip production of node i

D_j: trip attraction of node j

Example of OD Matrix:

$$\begin{array}{c} O/D \\ 1 \\ 2 \\ 3 \\ \cdots \\ n \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & \cdots & n \end{array} \\ \begin{bmatrix} t_{11} & t_{12} & t_{13} & \cdots & t_{1n} \\ t_{21} & t_{22} & t_{23} & \cdots & t_{2n} \\ t_{31} & t_{32} & t_{33} & \cdots & t_{3n} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ t_{n1} & t_{n2} & t_{n3} & \cdots & t_{nn} \end{bmatrix} \end{array}$$

g(ij): *traffic resistance function*, the resistance of human, vehicle, road condition, etc. to traffic distribution

h_{ij}: minimum trip distance between node i and node j

2.2 Ideal Assumptions

1. Assume that the chosen traffic region does not include a parking lot, so that every vehicle that enters the region must leave it, or, the sum of trip production equals the sum of trip attraction. Therefore, the model satisfies two conservation premises:

I. $O_i = \sum_j t_{ij}$

II. $D_j = \sum_i t_{ij}$

2. Assume that during peak hour, every road in the region has similar condition, that is, the degrees of comfort and safety and the traffic fee are the same. In such cases, we can assume that the traffic resistance function between two nodes only depends on the trip distance.

3. Work with idealized or simplified traffic region as shown in Figure 1.

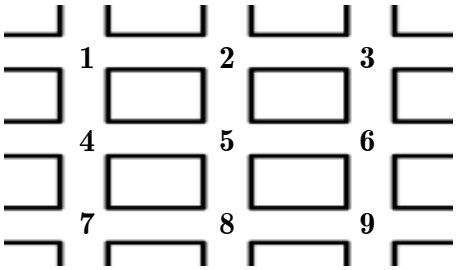

Figure 1

In figure 1, nodes 1~8 are OD nodes; node 9 is not.

2.3 Sample Matrix and Prior Matrix

I. Method for obtaining sample matrix

Hash algorithm is applied to the automatic-classification process of recorded vehicle data at stations.

When vehicles pass, RFID stations identify their information. Radio Frequency Identification, abbreviated to RFID, is a non-contacting automatic identifying technique. It recognizes target based on radio frequency signal and gathers relevant data. No manual operation is needed, and the system can work in hash conditions. The RFID technique can distinguish object in high speed or multiple objects at one time automatically.

When the RFID tag reaches the magnetic field, it receives the radio frequency signal and sends out product information inside the chip relying on inducted current. After reading and decrypting the information, the reader transfers the original data to the main frame to analyze. At last, RFID tags with 24 digits formed by 0-9 and A-Z are generated. Every vehicle has an exclusive RFID tag. When compared to image recognition, RFID demonstrates its accuracy and flexibility.

The RFID tags are the original data of hashing process. Due to the formation of 24 digits of 0-9 and A-Z, the RFID tag can be seen as a number in base-37 and can be translated into a decimal number. The number is modulo by 100003, a prime number about 10 times larger than the total number of cars. At a result, the probability of conflicts diminishes to 10%.

According to the traveling status of each recorded vehicle, mostly the first and the last record, the fragmentary data can be deduced from road condition analysis together with probability. The completed sample matrix is obtained.

II. Method for obtaining prior matrix

In computation geometry, we use cross product of vectors to determine whether two line segments intersect with each other.

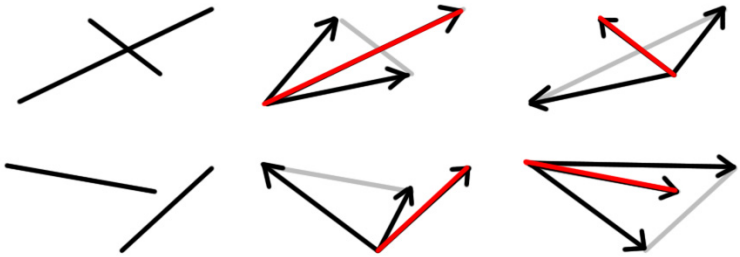

Figure 2 line segments intersection conditions

Suppose there are two vectors $\alpha(x_1, y_1)\,\beta(x_2, y_2)$, then we can get the cross product of $\alpha \times \beta$:

$$\alpha \times \beta = x_1 y_2 - x_2 y_1$$

The sign（positive or negative）of the product indicates the relative

position of the two vectors. As shown by Figure 2, we construct three vectors from one of the endpoints of those segments to others. If the two segments intersect, then the vector consists of one of the segments must lie inside the other two vectors. If the two segments don't overlap, then there must exist a situation when the vector consists of one of the segments doesn't lie inside the other two vectors.

Adopting this method, we are able to decide if the segment between two points intersects with the outline of a region, and thus determine whether commuting form one point to another will pass certain location. After obtaining the intersection, the specific path taken can be acquired through stochastic simulation under certain probability.

Simulation program stochastically generates points around the target region as gatherings of dense population. The program simulates the path of a vehicle by generating its origin and destination and then find out if the segments between the two points intersect with that region. If the intersection exists, then we calculate the probability of every OD nodes taken as origin and destination by analyzing the position of the intersection. We then decide an O node and a D node for that vehicle. As we run the program based on the map of the city to collect an amount of data, a prior matrix that is relatively objective can be obtained.

2.4 Bayesian Inference

Bayesian Inference is a method of inference based on incomplete information that obtains subjective probability of the unknown and applies Bayes' Rule to update the probability estimation. In backward estimation of OD matrix, a commonly researched field, Bayesian Inference, together with Monte Carlo method can be used in time-dependent (dynamic) models. Inspired by it, this paper introduces Bayesian Inference in the estimation of OD matrix with incomplete data to optimize the probability.

The method is as follows:

In a traffic zone with n OD nodes, there are n rows in the matrix, noted as $r_1, r_2, \ldots r_n$. Similarly, note the n columns as $c_1, c_2, \ldots c_n$.

Using the method stated in 2.3-II, we can obtain a prior matrix based on subjective estimation and experience. Here, we first consider the situation

for each row.

The trip production corresponding to $r_1, r_2, ...r_n$ are notes as $O_1, O_2, ...O_n$ respectively. Thus we establish a probability function of parameter r_i to be:

$$\pi(r_i) = \frac{O_i}{\sum_{j=1}^{n} O_j} = \varepsilon_i \quad \text{s.t.} \quad \sum_{i=1}^{n} \varepsilon_i = 1$$

As stated in 2.3-I, we select the record of m cars randomly as samples and obtain a sample matrix. Note the cars as $x_1, x_2, ...x_m$ and get a joint probability function with r_i, noted as:

$$P(x_1, x_2, ...x_m, r_i)$$

According to Bayes' Rule:

$$\pi(r_i \mid x_1, x_2, ...x_m) P(x_1, x_2, ...x_m) = P(x_1, x_2, ...x_m \mid r_i)\pi(r_i) = P(x_1, x_2, ...x_m, r_i)$$

Thus,

$$\pi(r_i \mid x_1, x_2, ...x_m) = \frac{P(x_1, x_2, ...x_m, r_i)}{P(x_1, x_2, ...x_m)} = \frac{P(x_1, x_2, ...x_m \mid r_i)\pi(r_i)}{\sum_{i=1}^{n} P(x_1, x_2, ...x_m \mid r_i)\pi(r_i)}$$

in which,

$P(x_1, x_2, ...x_m \mid r_i)$ represents the probability of the distribution of total trip production to node i in sample matrix;

$\pi(r_i)$ represents the probability of the distribution of total trip production to node i in prior matrix;

$\pi(r_i \mid x_1, x_2, ...x_m)$ represents the posterior probability, or the estimated probability of the distribution of total trip production to node i.

Then adjust the posterior probability according to the conservation premise

$$\sum_{i=1}^{n} \pi\left(r_i \mid x_1, x_2, \ldots x_m\right) = 1$$

Based on the average trip production of the traffic zone O,

$$O_i = O \times \pi\left(r_i \mid x_1, x_2, \ldots x_m\right)$$

We can obtain the estimated trip production of every node in the matrix. Similarly, we can obtain the estimated trip attraction of each OD node. The resulting one-dimension trip production table and trip attraction table will be further utilized in gravity model.

2.5 Gravity Model

I. Gravity model is deduced from Newton's famous *Law of Universal Gravitation*. Its original form is,

$$t_{ij} = \alpha \frac{O_i D_j}{h_{ij}^2}$$

The conservation premises stated before show that,

$$\sum_j t_{ij} = \alpha O_i \sum_j D_j d_{ij}^{-2} = O_i$$

$$\sum_i t_{ij} = \alpha D_j \sum_i O_i d_{ij}^{-2} = D_j$$

However, there does not exist an α that satisfies both equations. Therefore, we modify gravity model to the following

$$t_{ij} = k O_i^{\alpha} D_j^{\beta} g(c_{ij})$$

k, α, β are parameters. Since we assumed that $g(c_{ij})$ is only relevant to distance between two nodes, we have

$$g(c_{ij}) = \frac{1}{h_{ij}^r}$$

Therefore, known O_i, D_j, gravity model can be used to estimate OD matrix, with its formula

$$t_{ij} = k\frac{O_i^{\alpha} D_j^{\beta}}{h_{ij}^{r}}$$

II. Accuracy of gravity model

The gravity model is simple and visual, but several disadvantages of gravity model compromise the accuracy of its result. First, assuming the traffic resistance function only depends on distance may result in error. Second, to find the parameters, reference to historical data is preferred and reasonable speculation is needed in cases without sufficient data.

Considering all these shortcomings of gravity model, we need to modify its results in order to obtain final OD matrix. Notice the conservation premises stated before,

$$\sum_j t_{ij} = O_i, \sum_i t_{ij} = D_j$$

We can use the trip distribution results from gravity model to calculate the trip production $O_i^{(0)}$ and trip attraction $D_j^{(0)}$, compare the error from O_i and D_j, and adjust accordingly.

III. Optimization of results from gravity model

Let the trip distribution result between i and j from gravity model be $t_{ij}^{(0)}$

$$\sum_j t_{ij}^{(0)} = O_i^{(0)}, \sum_i t_{ij}^{(0)} = D_j^{(0)}$$

$$\sum_j t_{ij}^{(k)} = O_i^{(k)}, \sum_i t_{ij}^{(k)} = D_j^{(k)}$$

Let $F_{O_i}^{(k)} = \frac{O_i}{O_i^{(k)}}, F_{D_j}^{(k)} = \frac{D_j}{D_j^{(k)}}$, $0 \le k \le m, t_{ij}^{(m)}$ being the final result for trip distribution between i and j.

On one hand, $t_{ij}^{(k+1)'} = O_i \cdot \frac{t_{ij}^{(k)}}{\sum_j t_{ij}^{(k)}} = O_i^{(k)} \cdot F_{O_i}^{(k)} \cdot \frac{t_{ij}^{(k)}}{\sum_j t_{ij}^{(k)}}$

On the other hand, $t_{ij}^{(k+1)"} = D_j \cdot \frac{t_{ij}^{(k)}}{\sum_i t_{ij}^{(k)}} = D_j^{(k)} \cdot F_{D_i}^{(k)} \cdot \frac{t_{ij}^{(k)}}{\sum_i t_{ij}^{(k)}}$

Therefore, take the arithmetic mean of $t_{ij}^{(k+1)'}$ and $t_{ij}^{(k+1)"}$, we have

$$t_{ij}^{(k+1)} = \frac{1}{2}(t_{ij}^{(k+1)'} + t_{ij}^{(k+1)"})$$

$$t_{ij}^{(k+1)} = \frac{1}{2}(O_i^{(k)} \cdot F_{O_i}^{(k)} \cdot \frac{t_{ij}^{(k)} \cdot F_{D_j}^{(k)}}{\sum_j t_{ij}^{(k)} \cdot F_{D_j}^{(k)}} + D_j^{(k)} \cdot F_{D_j}^{(k)} \cdot \frac{t_{ij}^{(k)} \cdot F_{O_i}^{(k)}}{\sum_i t_{ij}^{(k)} \cdot F_{O_i}^{(k)}})$$

$$t_{ij}^{(k+1)} = t_{ij}^{(k)} \cdot F_{D_j}^{(k)} \cdot F_{O_i}^{(k)} \cdot \frac{1}{2}(\frac{O_i^{(k)}}{\sum_j t_{ij}^{(k)} \cdot F_{D_j}^{(k)}} + \frac{D_j^{(k)}}{\sum_i t_{ij}^{(k)} \cdot F_{O_i}^{(k)}})$$

Let $G_{O_i}^{(k)} = \frac{O_i^{(k)}}{\sum_j t_{ij}^{(k)} \cdot F_{D_j}^{(k)}}, G_{D_j}^{(k)} = \frac{D_j^{(k)}}{\sum_i t_{ij}^{(k)} \cdot F_{O_i}^{(k)}}$, we have

$$f(F_{D_j}^{(k)}, F_{O_i}^{(k)}) = F_{D_j}^{(k)} \cdot F_{O_i}^{(k)} \cdot \frac{G_{O_i}^{(k)} + G_{D_j}^{(k)}}{2}$$

Therefore,

$$\sum_j t_{ij}^{(k)} = O_i^{(k)}, \sum_i t_{ij}^{(k)} = D_j^{(k)}$$

$$F_{O_i}^{(k)} = \frac{O_i}{O_i^{(k)}}, F_{D_j}^{(k)} = \frac{D_j}{D_j^{(k)}}$$

$$t_{ij}^{(k+1)} = t_{ij}^{(k)} \cdot f(F_{D_j}^{(k)}, F_{O_i}^{(k)})$$

If after m times of iteration, $F_{O_i}^{(m)}, F_{D_j}^{(m)}$ are close enough to 1, then $t_{ij}^{(m)}$ is the trip distribution in final OD matrix, else the iteration process continues.

With the help of Pascal, we are able to generate the final OD matrix.

3 Application of the Model

3.1 Sample Choice

We chose a region in downtown Nanjing, as shown in figure 3 below. The region has 7 OD nodes, which are noted in the figure.

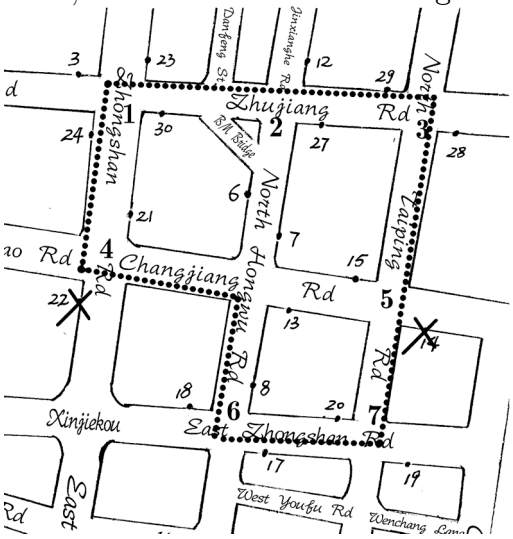

Figure 3

The region chosen in this paper is relatively geometric and suits the idealized example in Figure 1. Moreover, the region is located in downtown Nanjing and usually has a lot of traffic. Thus, it is an optimal choice and provides valuable insight for general cases.

It should be noted that Xinjiekou is not in the region. Because Xinjiekou is the very center of Nanjing, it has significantly large traffic flow but no RFID station. Therefore, it is not included in the region for the sake of accuracy of our result and reference value to general cases.

3.2 Data Collection

This paper collected RFID data from thirty monitoring stations at Gulou district, Nanjing from 8:00 to 9:00 a.m. during 19th-22nd August 2014. The selected stations noted as $A_1, A_2, ...A_{30}$ locate around an "L"-shaped region which consists of Zhujiang Road, Changjiang Road, East Zhongshan Road and Zhongshan Road, North Hongwu Road and North Taiping Road. While most stations lie inside the area, some distribute

outside, serving as auxiliaries. Details are as follow:

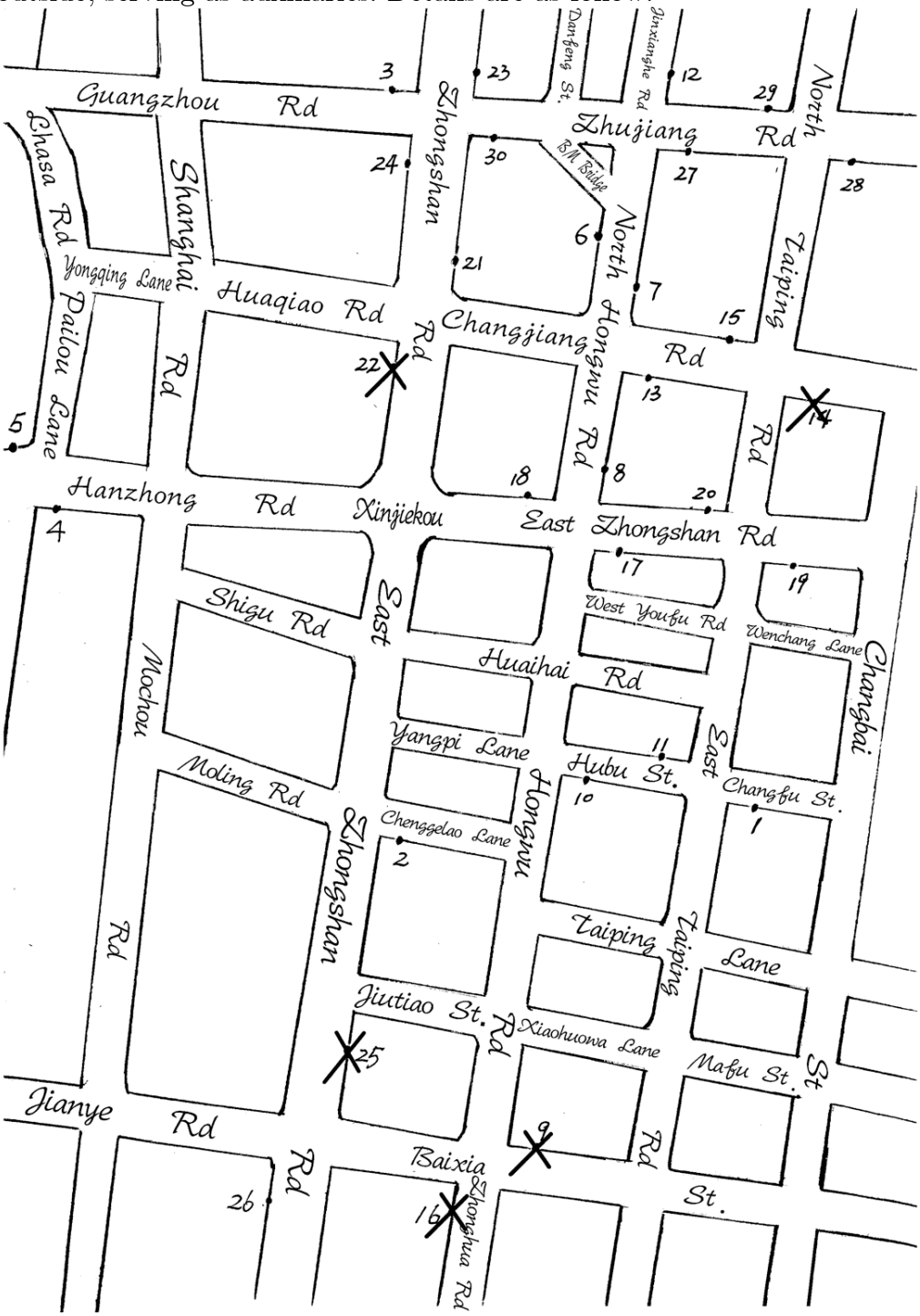

Figure 4

As shown by Figure 4, data from stations $A_9, A_{14}, A_{16}, A_{22}, A_{25}$ are missing due to the machine fault.

At any crossing, there must exist four monitoring stations in order to obtain complete trip production and attraction of that node. However, there is only one crossing (crossing No. 1) among the seven in the selected area that satisfies the requirement for complete data. Thus, the data collected are as incomplete as expected.

Considering that the complete information of all vehicles may not be recorded during the one-hour time interval, we further analyze data collected from the first and last five minutes. If a car only appears once on the record, which takes place during the first or last five minutes, we consider the path of that vehicle not completely recorded and thus delete that piece of information.

3.3 Prior Matrix

The seven OD nodes mark an "L"-shaped region in the map (as shown in Figure 5). Measure the distances between each node, and we get a table of coordinate of each node (as shown in Table 1). The coordinate of Xinjiekou is (400, 400).

We stochastically generate 80-100 points in a 1000*1000 region around the "L"-shaped chosen region, randomly choose a starting point and an ending point each time, and see if the line segment between the two points intersects with the "L"-shaped region. Using method stated in 2.3-II, we can obtain a prior matrix as shown in Table 2.

	x coordinate	y coordinate
1	400	682
2	550	660
3	733	626
4	400	533
5	702	448
6	509.5	392
7	695	377
8	497.5	519

Table 1

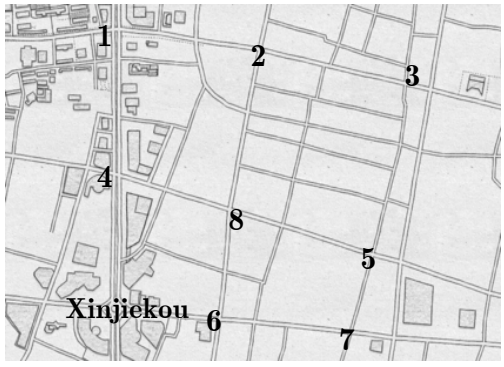

Figure 5

	1	2	3	4	5	6	7
1	0	374	108	322	138	245	103
2	239	0	147	145	172	327	122
3	292	239	0	123	205	183	159
4	217	83	132	0	488	77	151
5	135	231	249	330	0	270	79
6	135	255	120	220	199	0	335
7	177	140	548	161	101	124	0

Table 2

3.4 Sample Matrix

By studying records, especially the first and last records of each vehicle passing this region, we are able to analyze the possible routes for each vehicle, as shown in Table 3. Further calculation results in sample matrix shown in Table 4.

Considering that most vehicles do not normally take detours in the center of the city, we ignore the possibility of vehicles turning backwards in our analysis.

	Single Record	Multiple Records	
		First recorded at the station	Last recorded at the station
A1	No enter	Impossible	If recorded at A10 then 50% exit from 6, 50% exit from 4; If recorded at A17 then exit from 7
A2	5% enter from 7, 95% no enter	50% no enter, 30% enter from 6, 20% enter from 7	If recorded at A4 then no enter; If recorded at A24 then exit from 4
A3	67% enter from 2 exit from 1, 33% no enter	Impossible	Exit from 1
A4	No enter	67% enter from 6, 33% enter from 4	Impossible
A5	No enter	Impossible	If recorded at A18 then exit from 6; If recorded at A3 then exit from 1; If recorded at A24 then exit from 4
A6	Enter from 2; 33% exit from 4; 67% exit from 6	Enter from 2	33% exit from 4, 67% exit from 6
A7	Enter from 4; Exit from 2	Enter from 4	Enter from 2
A8	Enter from 6; Exit from 4	Enter from 6	Enter from 7; Exit from 4
A9	-	-	-
A10	60% no enter; 20% enter from 7 exit from 5; 20% enter from 7 exit from 3	If recorded by A1 or A19 then no enter, else enter from 7	50% exit from 4, 50% exit from 6
A11	No enter	If recorded by A26 then no enter, else 50% enter from 6, 50% enter from 4	Impossible
A12	Impossible	Impossible	Exit from 2

A13	Enter from 4; 50% exit from 5, 25% exit from 3, 25 % exit from 7	Enter from 4	50% exit from 5, 25% exit from 3, 25 % exit from 7
A14	-	-	-
A15	60% enter from 5, 20% enter from 3, 20% enter from 7; 67% exit from 4, 33% exit from 6	20% enter from 3, 20% enter from 7, 20% enter from 5	Impossible
A16	-	-	-
A17	50%no enter, 50% enter from 7	If recorded by A1 or A19 then no enter; If recorded by 28 then enter from 7, exit from 3	50% exit from 4, 50% exit from 6
A18	No enter	75% no enter, 25% enter from 4 exit from 1	50% exit from 6, 50% exit from 7
A19	No enter	Impossible	Exit from 7
A20	20% enter from 3， 30% enter from 5，50% enter from 7	70% enter from 7, 15% enter from 5, 10% enter from 3	Impossible
A21	Impossible	Enter from 4	Impossible
A22	-	-	-
A23	Impossible	Impossible	Exit form 1
A24	Enter from 1, 67% exit from 4, 33% exit from 6	Enter from 1	Exit from 4
A25	-	-	-
A26	No enter	Impossible	If recorded by A4 then no enter; If recorded by A24 then enter from 1, exit from 4
A27	Enter from 2, 33% exit from 5, 67% exit from 7	Enter from 2	42% exit from 3, 25% exit from 5, 33% exit from 7
A28	Enter from 7	Impossible	Exit from 3
A29	42% enter from 3, 25% enter from 5, 33% enter from 7; Exit from 2	50% enter from 3, 20% enter from 5, 30% enter from 7	Impossible
A30	Enter from 1; Exit from 2	Enter from 1	Exit from 2

Table 3

O/D	1	2	3	4	5	6	7
1	0	135	188	303	190	192	206
2	131	0	222	304	143	272	208
3	92	218	0	102	221	318	227
4	306	229	125	0	222	162	119
5	116	114	205	226	0	220	275
6	210	287	355	164	191	0	157
7	304	302	245	116	171	107	0

Table 4

3.5 Final Matrix

By Bayes Inference, we have trip production and trip attraction of each OD node in Table 5.

	1	2	3	4	5	6	7
O	1274	1200	1151	1087	1217	1403	1268
D	1125	1380	1420	1285	1205	1266	919

Table 5

In gravity model, let α =1.152, β =1.152, k=0.15, r=0.5. Through iteration, we have the final matrix in Table 6.

O/D	1	2	3	4	5	6	7
1	0	308	239	305	115	153	88
2	202	0	338	220	126	178	81
3	143	331	0	183	188	154	104
4	190	252	221	0	133	198	71
5	114	195	302	194	0	161	196
6	179	259	232	297	170	0	197
7	115	177	261	187	266	194	0

Table 6

4 Conclusions and Outlook

This paper proposes a new method to estimate OD matrix for traffic flow that is based on incomplete RFID data. We acquire the prior matrix by computation geometry, generate a sample matrix according to traffic analysis and optimize the trip production and attraction between each OD node using Bayesian Inference. By applying gravity model to the one-dimension trip production table and trip attraction table and repeating iteration, we obtain an estimated OD matrix in theory. In application, we collect sample RFID data in several monitoring stations at Gulou district, Nanjing in August 2014. Utilizing Hash algorithm, traffic analysis and program modeling, we are able to achieve considerable improvement in the estimation of OD matrix.

Unlike the traditional method of obtaining OD matrix or backward estimation, this paper offers a method that makes full use of the incomplete data and integrates various algorithms to estimate OD matrix. Surely, this method can provide significant reference in RFID station distribution planning, cost saving and accuracy improvement as

intelligent transportation system develops.

Yet there are still plenty to improve: This paper idealizes the traffic region and simplifies the commuting condition. In future study, we can take traffic fee, safety and other factors into consideration when establishing traffic resistance function of gravity model.

5 References

[1] 汪成亮，张晨，黄文龙.面向车联网的高速路 OD 矩阵估计模型. 西南交通大学学报 2013.12
[2] 黎华林.增长系数法与 Fratar 法的类比分析. 中国水运 2007.07
[3] 孙剑, 冯羽. 基于车辆自动识别技术的动态 OD 矩阵估计新方法 2013.09
[4] 郝光, 岳辉, 王翠红, 马倩玉. 动态 OD 矩阵推算模型及算法应用 2006.08
[5] 徐锦强, 林宇洪, 丁艺. 基于 Fratar 模型的交通分布预测系统设计 2011.06
[6] Mapbox 地图赏析
http://chiangbt.github.io/webcontent/Mapboxmap.html
[7] Anders Peterson. The Origin–Destination Matrix Estimation Problem — Analysis and Computations May, 2007

（1）

```
program DataGen;
  type point = record
                 x, y:real;
               end;
  var od:array[1..7, 1..7] of longint;
      sod:array[1..7, 1..2] of longint;
      g:array[1..100] of point;
      ng, i, j, co, cd, num, numo, numd, cars:longint;
      cp1, cp2, cp3, cp4, cp5, cp6, cp7, cp8, st, ed, tp:point;
  function cp(p0, p1, p2:point):real;
    begin
      cp:=((p2.x - p0.x) * (p1.y - p0.y) -
           (p1.x - p0.x) * (p2.y - p0.y));
    end;
  function dir(d:real):integer;
    begin
      if d < 0 then
        dir:=-1
      else
        if d > 0 then
          dir:=1
        else
          dir:=0;
    end;
  function cc(p0, p1, p2, p3:point):boolean;
    begin
      if (dir(cp(p2, p0, p1)) <> dir(cp(p3, p0, p1))) and
         (dir(cp(p0, p2, p3)) <> dir(cp(p1, p2, p3))) then
        cc:=true
      else
        cc:=false;
    end;
  function rst(var p1, p2:point; a, b:integer):point;
    var p:integer;
    begin
      p:=random(3);
      if p = 2 then
        begin
          rst:=p2;
          num:=b
        end
      else
        begin
          rst:=p1;
          num:=a
        end
    end;
```

```
begin
  writeln('Input the total number of cars:');
  readln(cars);
  assign(output, 'data.out');
  rewrite(output);
  randomize;
  for i:=1 to 8 do
    for j:=1 to 8 do
      od[i, j]:=0;
  for i:=1 to 8 do
    for j:=1 to 2 do
      sod[i, j]:=0;
  cp1.x:=497.5; cp1.y:=519;
  cp2.x:=400; cp2.y:=682;
  cp3.x:=733; cp3.y:=626;
  cp4.x:=695; cp4.y:=377;
  cp5.x:=400; cp5.y:=533;
  cp6.x:=550; cp6.y:=660;
  cp7.x:=702; cp7.y:=448;
  cp8.x:=509.5; cp8.y:=392;
  ng:=trunc(random(20) + 80);
  for i:=1 to ng do
    begin
      repeat
        g[i].x:=random * 1000;
        g[i].y:=random * 1000
      until not ((g[i].x >= 300) and (g[i].x <= 700) and
                 (g[i].y >= 300) and (g[i].y <= 700));
    end;
  {g[1].x:=320;
  g[1].y:=200;
  g[2].x:=800;
  g[2].y:=500;}
  i:=0;
  while i < Cars do
    begin
      co:=random(ng) + 1;
      cd:=random(ng) + 1;
      {co:=1; cd:=2;}
      st.x:=-1;
      if (co <> cd) then
        begin
          if cc(cp5, cp2, g[co], g[cd]) then
            st:=rst(cp5, cp2, 1, 4)
          else
            if cc(cp1, cp5, g[co], g[cd]) then
              st:=rst(cp5, cp5, 4, 4)
            else
              if cc(cp2, cp3, g[co], g[cd]) then
                if cc(cp2, cp6, g[co], g[cd]) then
                  st:=rst(cp2, cp6, 1, 2)
                else
                  st:=rst(cp3, cp6, 2, 3)
              else
                if cc(cp3, cp4, g[co], g[cd]) then
                  if cc(cp3, cp7, g[co], g[cd]) then
                    st:=rst(cp3, cp7, 3, 5)
                  else
                    st:=rst(cp4, cp7, 5, 7)
                else
                  if cc(cp4, cp8, g[co], g[cd]) then
                    st:=rst(cp4, cp8, 7, 6)
                  else
                    if cc(cp8, cp1, g[co], g[cd]) then
                      st:=rst(cp8, cp8, 6, 6);
        if st.x = -1 then continue;
        numo:=num;
        if cc(cp4, cp8, g[co], g[cd]) then
          st:=rst(cp4, cp8, 7, 6)
        else
          if cc(cp8, cp1, g[co], g[cd]) then
            st:=rst(cp8, cp8, 6, 6)
          else
            if cc(cp3, cp4, g[co], g[cd]) then
              if cc(cp3, cp7, g[co], g[cd]) then
                ed:=rst(cp3, cp7, 3, 5)
              else
                ed:=rst(cp4, cp7, 5, 7)
            else
              if cc(cp2, cp3, g[co], g[cd]) then
                if cc(cp2, cp6, g[co], g[cd]) then
                  ed:=rst(cp2, cp6, 1, 2)
                else
                  ed:=rst(cp3, cp6, 2, 3)
              else
                if cc(cp5, cp2, g[co], g[cd]) then
                  st:=rst(cp5, cp2, 1, 4)
                else
                  if cc(cp1, cp5, g[co], g[cd])then
                    st:=rst(cp5, cp5, 4, 4);
        numd:=num;
        if (random(2) = 1) then
          begin
            tp:=st;
            st:=ed;
            ed:=tp;
            num:=numo;
            numo:=numd;
            numd:=num;
          end;
        inc(od[numo, numd]);
        inc(i);
      end;
  end;
for i:=1 to 7 do
  begin
    for j:=1 to 7 do
      begin
        inc(sod[i, 1], od[i, j]);
        inc(sod[j, 2], od[i, j]);
        write(od[i, j]:4, ' ')
      end;
    writeln
  end;
```

```
    writeln;
    num:=0;
    for i:=1 to 7 do
      begin
        write(sod[i, 1]:4, ' ');
        inc(num, sod[i, 1])
      end;
    writeln(num:4);
    num:=0;
    for i:=1 to 7 do
      begin
        write(sod[i, 2]:4, ' ');
        inc(num, sod[i, 2])
      end;
    writeln(num:4);
    close(output);
  end.
(2) program dataanal;
  const largepri = 100003;
        sumrfid = 30;
        matchnum:array[1..sumrfid] of integer = (6330, 6437, 6293,
6264, 6265, 6431, 6432, 6434, 6438, 6332, 6331, 6430,
                                                 6424, 6255,
6423, 6439, 6260, 6261, 6258, 6259, 6237, 6238, 6235, 6236,
                                                 2762, 6242,
6290, 6288, 6289, 6292);
        bnum = 7;
  type link = ^node;
       node = record
                time, place:integer;
                next:link;
              end;
       platestr = string[24];
  var st1, st2, st3, st4, st5:array[1..sumrfid + 1] of integer;
      data1:array[1..500, 1..3] of integer;
      data2, data4:array[1..500, 1..2] of longint;
      data3, data5:array[1..500, 1..2] of longint;
      hash:array[0..largepri - 1] of link;
      hashstr:array[0..largepri - 1] of platestr;
      occur1:array[1..sumrfid] of longint;
      od:array[0..bnum, 0..bnum] of longint;
      sumin:array[0..bnum] of longint;
      sumout:array[0..bnum] of longint;
      tin, tout:longint;
  function find(var num:longint):longint;
    var i, ans:longint;
    begin
      ans:=0;
      for i:=1 to 30 do
        if matchnum[i] = num then
          begin
            ans:=i;
            break
          end;
      if ans <> 0 then find:=ans else find:=9
    end;
  procedure addhash(var num, time:longint; var plate:platestr);
    var i, h, mul, cur:longint;
        p:link;
    begin
      h:=0; mul:=1;
      for i:=1 to 24 do
        begin
          if plate[i] in ['0'..'9'] then
            cur:=ord(plate[i]) - 48
          else
            cur:=ord(plate[i]) - 55;
          h:=(h + cur * mul) mod largepri;
          mul:=(mul * 37) mod largepri
        end;
      while (hash[h] <> nil) and (hashstr[h] <> plate) do
        h:=(h + 1) mod largepri;
      new(p);
      p^.time:=time;
      p^.place:=num;
      p^.next:=hash[h];
      hash[h]:=p;
      hashstr[h]:=plate
    end;
  procedure init;
    var i, j, p1, p2, p3, p4, p5, tot, sum:longint;
        tmp:real;
        ch:char;
    begin
      randomize;
      for i:=0 to largepri - 1    do
        begin
          hash[i]:=nil;
          hashstr[i]:=''
        end;
      for i:=1 to bnum do
        for j:=1 to bnum do
          od[i, j]:=0;
      for i:=1 to bnum do
        begin
          sumin[i]:=0;
          sumout[i]:=0
        end;
      for i:=1 to sumrfid do
        occur1[i]:=0;
      assign(input, 'consts.txt');
      reset(input);
      p1:=1; p2:=1; p3:=1; p4:=1; p5:=1;
      for i:=1 to sumrfid do
        begin
          st1[i]:=p1;
          st2[i]:=p2;
          st3[i]:=p3;
          st4[i]:=p4;
          st5[i]:=p5;
          sum:=0;
          while sum < 100 do
            begin
              read(tmp);
```

```
                tmp:=tmp * 100;
                sum:=sum + trunc(tmp + 0.5);
                data1[p1, 1]:=sum;
                read(data1[p1, 2], data1[p1, 3]);
                inc(p1)
              end;
            read(ch);
            read(tot);
            for j:=1 to tot do
              begin
                read(data2[p2, 1], data2[p2, 2]);
                inc(p2)
              end;
            sum:=0;
            while sum < 100 do
              begin
                read(tmp);
                tmp:=tmp * 100;
                sum:=sum + trunc(tmp + 0.5);
                data3[p3, 1]:=sum;
                read(data3[p3, 2]);
                inc(p3)
              end;
            read(ch);
            read(tot);
            for j:=1 to tot do
              begin
                read(data4[p4, 1], data4[p4, 2]);
                inc(p4)
              end;
            sum:=0;
            while sum < 100 do
              begin
                read(tmp);
                tmp:=tmp * 100;
                sum:=sum + trunc(tmp + 0.5);
                data5[p5, 1]:=sum;
                read(data5[p5, 2]);
                inc(p5)
              end
          end;
        st1[31]:=p1 + 1; st2[31]:=p2 + 1;
        st3[31]:=p3 + 1; st4[31]:=p4 + 1;
        st5[31]:=p5 + 1
      end;
    procedure main;
      var curnum, i, j, k, curt, st, ed, len, rand, last:longint;
          plate:platestr;
          ch, ch2:char;
          p, q:link;
          f:boolean;
          sec:array[1..1000] of integer;
      begin
        assign(input, 'data.in');
        reset(input);
        j:=0;
        while not eof do
          begin
            inc(j);
            curnum:=0;
            read(ch);
            while ch in ['0'..'9'] do
              begin
                curnum:=curnum * 10 + ord(ch) - 48;
                read(ch)
              end;
            curnum:=find(curnum);
            if curnum = 13 then
              begin
                curnum:=curnum;
              end;
            read(ch2);
            read(ch);
            read(ch);
            curt:=ord(ch) - 48;
            read(ch);
            curt:=curt * 10 + ord(ch) - 48;
            curt:=curt + (ord(ch2) - 56) * 60;
            read(ch);
            read(ch);
            curt:=curt * 6 + ord(ch) - 48;
            read(ch);
            curt:=curt * 10 + ord(ch) - 48;
            read(ch);
            plate:='';
            for i:=1 to 24 do
              begin
                read(ch);
                plate:=plate + ch
              end;
            read(ch);
            read(ch);
            addhash(curnum, curt, plate);
          end;
        tin:=0; tout:=0;
        for i:=0 to largepri - 1 do
          begin
            p:=hash[i];
            while p <> nil do
              begin
                if i > 96371 then
                  begin
                    p:=p;
                  end;
                len:=1; q:=p;
                sec[len]:=q^.place;
                last:=q^.time;
                while q^.next <> nil do
                  begin
                    q:=q^.next;
                    if last - q^.time > 600 then break;
                    inc(len);
                    sec[len]:=q^.place;
                    last:=q^.time
```

```
    end;
  if len = 1 then
    begin
      rand:=trunc(random * 100);
      j:=st1[sec[1]];
      while data1[j, 1] < rand do inc(j);
      st:=data1[j, 2];
      ed:=data1[j, 3];
      inc(od[st, ed]);
      inc(sumin[st])
    end
  else
    begin
      f:=false;
      for j:=st2[sec[len]] to st2[sec[len] + 1] - 1 do
        begin
          for k:=1 to len - 1 do
            if sec[k] = data2[j, 1] then
              begin
                f:=true;
                break
              end;
          if f then st:=data2[j, 2]
        end;
      if not f then
        begin
          rand:=trunc(random * 100);
          j:=st3[sec[len]];
          while data3[j, 1] < rand do inc(j);
          st:=data3[j, 2];
        end;
      f:=false;
      for j:=st4[sec[1]] to st4[sec[1] + 1] - 1 do
        begin
          for k:=2 to len do
            if sec[k] = data4[j, 1] then
              begin
                f:=true;
                break
              end;
          if f then ed:=data4[j, 2]
        end;
      if not f then
        begin
          rand:=trunc(random * 100);
          j:=st5[sec[1]];
          while data5[j, 1] < rand do inc(j);
          ed:=data5[j, 2];
        end;
    end;
  if (st = 1) and (ed = 1) then
    begin
      st:=st;
    end;
  inc(od[st, ed]);
  inc(sumin[st]);
        inc(sumout[ed]);
                if (st + ed <> 0) then
                  begin
                    inc(tin);
                    inc(tout);
                  end;
                if p = q then inc(occur1[p^.place]);
                p:=q^.next
              end;
          end;
        close(input)
    end;
  procedure print;
    var i, j:longint;
    begin
      assign(output, 'analyze.out');
      rewrite(output);
      for i:=0 to bnum do
        begin
          for j:=0 to bnum do
            write(od[i, j]:5);
          writeln
        end;
      writeln;
      for i:=0 to bnum do
        write(sumin[i]:5);
      writeln('      ', tin:5);
      for i:=0 to bnum do
        write(sumout[i]:5);
      writeln('      ', tout:5);
      writeln;
      for i:=1 to sumrfid do
        write(occur1[i]:5);
      writeln;
      close(output)
    end;
  begin
    init;
    main;
    print;
  end.
```