This is a repository copy of *A method for private car transportation dispatching based on a passenger demand model*.

A Private Car Transportation Dispatching Method Based on Passenger Demand Model

Wenbo Jiang, Tianyu Wo, Mingming Zhang, Renyu Yang, and Jie Xu

[1] School of Computer Science & Engineering
Beihang University
Beijing, China, 100191
`{jiangwb,woty,zhangmm,yangry}@act.buaa.edu.cn`
[2] School of Computing
University of Leeds
Leeds, UK
`j.xu@leeds.ac.uk`

Abstract. Although the demand for taxis is increasing rapidly with the soaring population in big cities, the taxi number grows relatively slowly during these years. In this context, private transportation such as Uber is emerging as a flexible business model, supplementary to the regular forms of taxis. At present, many works mainly focus on how to effectively reduce the taxi cruising miles. However, these taxi-based approaches still have some limitations in private car transportation scenario because they do not fully utilize the sufficient order information introduced by the new type of business model. In this paper, we present a dispatching method based on a passenger demand model to further reduce the private car cruising mileage. In particular, we firstly split the urban areas into many separate regions by using spatial clustering algorithm and partition one day into several time slots according to statistics of historical orders. Secondly, Locally Weighted Linear Regression is adopted to depict the passenger demand model in one specified region over a time slot. Finally, a dispatching process is formalized as a weighted bipartite graph matching problem and we leverage this dispatching approach to schedule private vehicles. To evaluate the effectiveness of our methods, we conduct several experiments based on real datasets derived from a private car hiring company in China. The experimental results illustrate that at most 74% accuracy could be achieved on passenger demand inference. Additionally, the cruising mileage could be reduced by 22.5% in simulation test.

Keywords: Spatial-temporal data, Data mining, Private car transportation, Vehicle scheduling

1 Introduction

The number of taxis in Beijing grows slowly during these years although demands for taking taxi has been sharply increasing with the soaring population.

At present, it is extremely hard for citizens to hail a taxi especially on the occasion of congested traffic. However, at the same time, taxi occupancy is still very low even in busy time[11]. This contradictory phenomenon leads to great requirements for customers vehicle transportation. Consequently, there are companies such as Uber[1] which provides private car services to satisfy the above demands. In this manner, private cars become a supplementary to the current taxi service in which citizens could reserve a private car through mobile applications in any place at any time. However, taxicab is inefficient due to characteristics such as low capacity utilization, high fuel costs, heavily congested traffic, and a low ratio(*live index*) of *live miles* (miles with a fare) to *cruising miles*(miles without a fare)[2]. The same challenges exist in private car service as illustrated in Figure 1. The rate represents the ratio of live miles to total miles. Apparently, the rate of approximately 80% vehicles is no more than 0.6. The average rate is merely 0.4601, indicating a typical low live index among most private cars. Namely, idle cruising miles take a large portion of the total miles.

Despite some similar challenges within the taxi service and private car service, the differences and the root causes still need to be deeply investigated. As for the taxi service, drivers can pick up a passenger easily in rush hours. However, drivers might cruise several street blocks to pick up a passenger if they lack the experience of when and where the citizens usually expect to catch a taxi. Therefore, experienced taxi drivers have less cruising miles than the inexperienced[4]. To mitigate the empirical impacts, many solutions generate and leverage knowledge of experience by using historical GPS records, and recommend pick-up positions or cruising routes to drivers for the sake of reducing cruising miles [2][3][9]. However, unlike the taxi service, passengers reserve the cars by Apps in private car service. In fact, each passenger order includes sufficient information such as the current position, destination, appointment pickup time, car type and etc. Intuitively, the abundant information could provision sufficient space to dispatch and schedule private vehicles based on the knowledge mining, giving rise to significant optimization potentials. Without fully utilizing of these information, larger cruising miles and longer waiting time will be produced and will result in increasing operating costs and negative user experience. All these strongly motivate our proposal in this paper.

Additionally, determining a specific position or a region is extraordinarily important in scheduling private cars. Presently, a private car driver could only accept orders according to the instructions from the dispatch center passively. The problem will not protrude in rush time due to the large requirements of reservation orders. In other cases, however, drivers have great needs to be instantly notified a specific potential position in order to diminish the time and monetary waste before receiving a new order. Therefore, an effective scheduling should improve the cruising miles whilst reducing passenger waiting time. In order to accurately obtain the specifying designated places, the dispatch center chooses the places based on subjective cognition and the dynamic passengers' demand should be also taken into account when determining the specified places because it is variable along with time. To summarize, we have to cope

(a) Rate CDF (b) Rate histogram
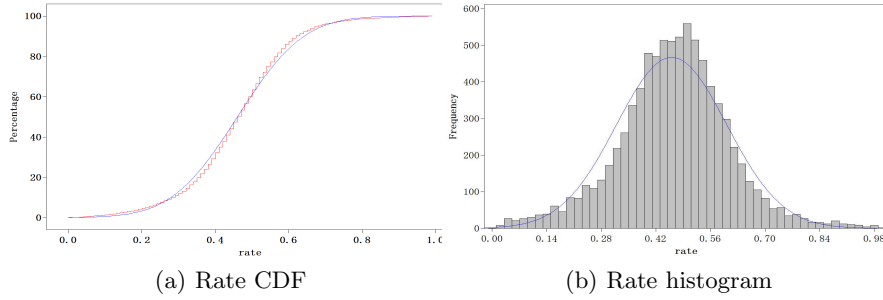
Fig. 1. rate of live miles compared to total miles

with the following challenges appropriately: 1) predicting the time-varied and dynamic passenger demand accurately; 2) dispatching vehicles to satisfy the requirements of different passengers who are free to choose different type of cars; 3) scheduling in real-time manner and provisioning current orders as well as future orders. In particular, the major contributions of the work in this paper can be summarized as follows:

- We propose an innovative study to reduce private vehicle cruising miles and a scheduling method based on a passenger demand model.
- A demand specification model is advocated based on Locally Weighted Linear Regression, which could reach 74% accuracy at most.
- The real-time scheduling is formalized as a weighted bipartite graph matching problem and 22.5% cruising miles could be reduced by using the described scheduling method.

The rest of this paper is organized as follows. Section2 describes the passenger demand model. Section 3 introduces algorithms to dispatch private cars. Furthermore, we evaluate the model and dispatching algorithms on a real dataset in Section 4. Section 5 concludes this paper and discusses the future works.

2 Demand Model

In this section, we elaborate the Passenger Demand Model in detail. Firstly, the training set, formulas and algorithms are respectively described to expose the model's details. We also demonstrate how to predict the passenger demand in real time manner. To begin with, We make some definitions in terms of the pasenger demand in different contexts. In taxi service, passenger demand mainly indicates the citizen requirements to take a taxi; In comparison, we define passenger's demand in the private car service as the orders that preserve private cars in a region over a time slot. As Table 1 shows, an order mainly contains such information.

The region is a much more fine-grained space definition than district. We split Beijing urban area into R regions using a density based spatial clustering algorithm DBSCAN[10] in which $i(1 <= i <= R)$ represents the i-th region.

Name	Description
Order ID	Order identifier
Order Type	Different types: "1" indicates that a passenger expects to be picked up immediately and "2" represents a passenger expects to be on board at a certain time in the future.
Book Time	The time when a passenger submits a new order.
Expected Onboard Time	The time when a passenger expects to be picked up.
Start Time	The time when a passenger is picked up.
Start Longitude	The longitude of GPS coordinates where a private car picks up a passenger.
Start Latitude	The latitude of GPS coordinates where a private car picks up a passenger.
End Time	The time when a passenger is dropped off.
End Longitude	The longitude of GPS coordinates where a private car drops off a passenger.
End Latitude	The latitude of GPS coordinates where a private car drops off a passenger.
Car Type	Three types of cars which passengers can book with different charges.
Status	Order status: "0" indicates a new submited order, "1" indicates an already matched order to a private car, and "2" means a completed order.

Table 1. Order specifications and descriptions

A convex polygon is used to indicate a region space coverage. After statistical analyzing of historical orders, one day could be divided into T time slots and $j(1 <= j <= T)$ represents the j-th time slot. In addition, each time slot's duration is regarded as an hour for the sake of understandability, and most works adopt this dividing method[6]. Therefore, the passenger demand in the i-th region over the j-th time slot could be expressed as $P_{i,j}(1 <= i <= R, 1 <= j <= T)$.

Furthermore, based on the mobility of cars and passengers, we assume that passenger demand in one region in the next time slot is relevant to orders in all regions over current time slots. In section 4, we conduct some experiments on a real dataset to confirm these assumptions. In particular, the orders in each region over the current time slot is an argument while the passenger demand in one region in future time slot is independent variable. A set of coefficients can be used to estimate the relation between the orders in current time slot and the predicted orders in the future time slot. To this end, the linear regression model targets our demand and the core philosophy is the coefficients estimation. With the input data stored in matrix X, and regression coefficients stored in vector w, predicted result can by depicted as Equation 1.

$$Y = X^T \cdot w \tag{1}$$

To achieve an ideal prediction, the error margin between the real demand and predicted one is supposed to be miminized as possible. The model is utilized to obtain the optimal set of regression coefficients w, given a training set X and Y. Typically, we aim to find the vector w with the minimum error. In order to improve the computation accuracy, we use the *squared error* instead of the *accumulated error* to represent the required difference. More specifically, the sum of the squared errors(SSE) can be described by Equation 2. The utilized least squares approach find the optimal w with the least SSE.

$$SSE = \sum_{i=1}^{m}(y_i - x_i^T \cdot w)^2 \tag{2}$$

We also depict the SSE in matrix form, as shown in Equation 3 shows.

$$SSE = (Y - X{\cdot}w)^T(Y - X{\cdot}w) \tag{3}$$

We solve SSE's derivative, and make it equal to zero (Equation 4).

$$X^T(Y - X{\cdot}w) = 0 \tag{4}$$

The best regression coefficients w could be found by using Equation 5.

$$\hat{w} = (X^T{\cdot}X)^{-1}{\cdot}X^T{\cdot}Y \tag{5}$$

It is the best estimation we could achieve from w. It is also worth noting that the above formula contains the requirement for the matrix inversion. Therefore, the equation only works on the occasion of the inversion of matrix $X^T{\cdot}X$ exists. Typically, the linear regression using the proposed least squares method could obtain the ideal result. However, it might be very likely to encounter with the under-fitting phenomenon. This is because it tries to get unbiased estimation of minimum mean square error and obviously we cannot get the best infer result once the model is under-fitting.

To handle with this problem, we introduce the Locally Weighted Linear Regression(LWLR) to produce some bias in the estimation to reduce the mean square error of prediction. In LWLR, we assign certain weights to the sample points which are close to the predicted sample point. In this context, the closest sample point gets the biggest weight. If the weight of a sample point exceeds a certain threshold, it will be selected. Afterwards, the selected points will constitute a sample subset. We use least square method to learn the linear regression model on this subset.

We can use the following Equation to get the optimal regression coefficient:

$$\hat{w} = (X^T{\cdot}W{\cdot}X)^{-1}{\cdot}X^T{\cdot}W{\cdot}Y \tag{6}$$

W is a weight matrix with each element indicating every sample point's weight. In order to generate the weighted matrix W, LWLR uses the kernel function to ensure the fact that the sample point closest to the predicted sample point could get the maximum weight. Specifically, there are a number of types of kernel function and the most commonly-used Gaussian Kernel Function is chosen in our solution, as demonstrated in 7.

$$W(i,i) = exp(\frac{|x_i - x|}{-2k^2}) \tag{7}$$

In this way, we can build a weight matrix W, containing only diagonal element and the sample point x_i closer to x could finally get the greater weight. Additionally, k is a user-specified parameter, which determines the weight given to x_i.

Algorithm 1 is introduced to predict passenger demand in i region over $(j+1)$ time slot. Firstly, we build the training set and test set according to the above

---

Algorithm 1 predictPassengerDemandInDifferentRegions()

1: initialize training set X $Matrix$ $xMat$;
2: initialize training set Y $Matrix$ $yMat$;
3: initialize test set X $Matrix$ X;
4: **for all** $Region \in R$ **do**
5:     add p(i,j) to X ;
6: **end for**
7: create weights matrix W;
8: $xTx \leftarrow xMat.Transpose * (W * xMat)$;
9: **if** determinant of xTx equals to 0 **then**
10:    return $NULL$;
11: **else**
12:    $ws \leftarrow xTx.Inverse * (xMat.Transpose * (W * yMat))$
13:    return $X * ws$;
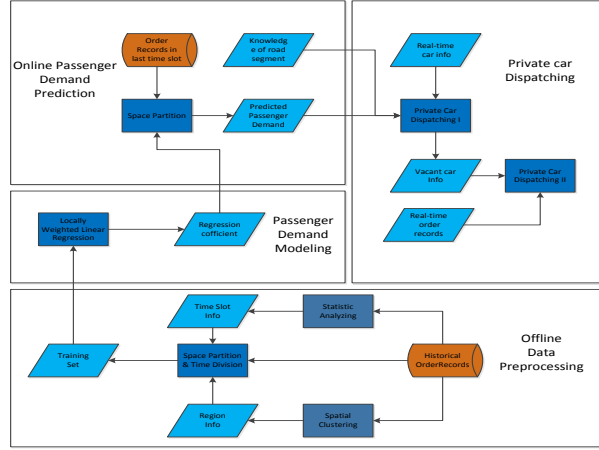14: **end if**

Fig. 2. Dispatching process overview

description. Secondly, we use the kernel method as described in Equation 7 to generate weight matrix. Finally, after examining if xTx has an inverse matrix, we obtain the coefficients using Equation 6 and predict the passenger demand in different regions. If xTx does not have an inverse matrix, we cannot figure out the coefficients and the passenger demand. In such conditions, parameter estimation and optimization algorithms are used to facilitate the computation of the approximately optimal coefficients. Due to the fact that xTx's inverse matrix could be easily obtained in most cases, the parameter estimation is out of the scope of this paper and will not be discussed.

3   Dispatching Algorithm

In this section, we describe in detail the core idea of the private vehicle dispatching algorithm. In Section 2, the prediction model is advocated and the passenger

demand count in different regions could be generated. Before introducing our solution, some fundamental definitions are given as follows:

Definition 1 Bipartite graph matching. *Given a bipartite graph G = (C,O,E), C is a vertex set in which each vertex represents a vacant car and each vertex in set O represents a predicted order. M is a subgraph of G. Any two edges in M's edge set E are not attached to the same vertex and M is a match of vacant cars and predicted orders.*

Definition 2 Road Segment. *If a road meets a junction or the road offset exceeds a threshold, a new road segment is generated. A road segment could be uniquely identified by the combination of ID, start and end GPS coordinates.*

Assuming that we have estimated passenger demand $\hat{P}(i, j + 1)$ in i region over $(j + 1)$ time slot. What we need is to assign vacant cars to passengers with the minimum total cruising mileage. A bipartite graph model $G(C, O, E)$ is therefore used to formalize our problem. Intuitively, distance between a vacant car and a predicted order becomes the weight of an edge connecting those two kinds of vertices. Accordingly, a matching with the minimum weight could be formalized as weighted bipartite graph matching problem.

Before performing the matching algorithm, how to represent the weight of the edges needs to be determined properly. Because the objectives are to find a scheduling method with the minimum cruising mileage, The distance between vacant cars and predicted orders could be used to represent the weights of the edges. In addition, C is the set of current vacant cars, with a vector S recording real-time status of all vehicles. Therefore, we can always get an updated C from S. In particular, "1" indicates that the current passenger vehicle is occupied, whereby "0" represents a vacant status of the current vehicle. Furthermore, in order to calculate the distance, we have to gain the real-time location of the vehicle. To deal with this, GPS coordinates are adopted to display the car's position. L represents real-time location information of all vehicles. In this way, we can get current status of the car k from S_k and current position from L_k.

With respect to the predicted orders, We use vertex set O to cover order information such as the location to get on the car, expected on-board time, destination etc. In order to simplify the problem, we assume that all orders' expected on-board time is the current time. However, as the exact position of the predicted order is unknown, we have to figure out effectively how to estimate a predicted order's position. As mentioned in section 2, the predicted order count is based on pre-divided region, but region itself is a coarse-grained geography definition. Region should be divided in a much more fine-grained manner. To cope with these issues, we firstly match the historical on-board positions to road segment in a region based on historical order analysis. Thereafter, the number of passengers who boarding on private car in a road segment can be computed and further used to decide which road segment a predicted order belongs to. If the probability of an order appears in a road segment which is greater than a certain threshold value, the order could be definitely identified on the road

segment. Eventually, the position of the predicted order can be determined by this road segment.

Moreover, we will discuss how to set the weight of edge. In bipartite graph G (C, O, E), E is the set of edges. Intuitively, the distance between the car and the order is used to represent the weight of edge between them. Despite other factors including the weather, real-time traffic etc, we use Baidu Web APIs to get the shortest distance between two GPS coordinates in our experiment for the clearness and simplicity of the model.

Given above definitions and descriptions, the scheduling problem could be formalize as a weighted bipartite graph matching problem. Kuhn-Munkras algorithm is used to find the optimal matching with minimum cost. Due to the special nature of KM algorithm, set C and O should have equal vertexes. In fact, we can add some vertexes to the minimum set on the occasion of two unequal set. Additionally, the weights of edges associated to those added vertexes are set to be zero. The detailed descriptions of the scheduling process could be found in Algorithm 2 .

Algorithm 2 schedulePrivateCar()

1: initialize each vertex in set C in j+1 time slot;
2: predictPassengerDemandInDifferentRegions ();
3: initialize each vertex in set O in j+1 time slot;
4: find a minimum cruising mileage matching;
5: **while** a real order occurs **do**
6:    select vacant cars near the real order's position ;
7:    **if** a vacant car accepts the order **then**
8:       update car's status;
9:       update order's status;
10:      update L and S;
11:   **end if**
12: **end while**

After the matching decision, vacant cars will be assigned to the road segments where predicted order exists.

In the real scenario, a real order might be placed while vacant cars are forwarding to a matched road segment. The real order contains information including the passenger's position, destination, etc. Meanwhile, vacant cars' information can be extracted from vector S and L. In case of the emerging real orders, the closest car will be scheduled and appointed to satisfy the real order demand. Particularly, the status and location of the vehicle need to be updated in real-time manner. Once a vehicle is assigned to a real order, the corresponding status related to real order will be occupied and updated. The status "1" will turn into "0" when the passenger gets off. Finally, the scheduling process proceeds until all private cars are out of service (finish working).
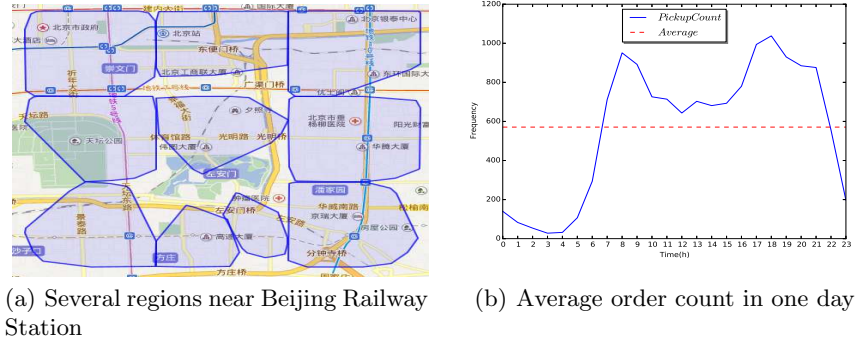
(a) Several regions near Beijing Railway Station

(b) Average order count in one day

Fig. 3. Regions and time slots

4   Experimental Evaluation

In this section, we set up several experiments to evaluate the effectiveness of the proposed passenger demand model and dispatching algorithm.

4.1   Dataset and Experiment Environment

We use a real dataset from a private car hiring company (for commercial reasons we cannot identify the name), which contains 2720 vehicles and 356706 order records for a whole month in Beijing. The test machine is equipped with Debian 6.0 operation system, 8GB Memory and Intel i5-4570 with a frequency of 3.20GHZ is set up.

4.2   Data Processing

Because the real dataset can not be directly used, the data processing starts with data cleaning and pre-operations. Specifically, we filter the order with a time duration less than 10 minutes and larger than 60 minutes, resulting in the remaining 324512 records. Subsequently, spatial clustering is utilized to split Beijing urban area into multiple regions. To simplify our experiment, DBSCAN algorithm is used to divide urban areas in Beijing four-rings into 136 areas. Two parameters (Epsilon and MinPoints) are set to be 0.15 and 100 respectively according to the engineering experiences. Actually, the Epsilon indicates the density of a region , while MinPoints shows how many points a region need to possess at least. Besides, a day is partitioned into several time slots based on the order distribution in one day.

Figure 4.2 illustrates a good example of regions partition near the Beijing Railway Station and the average count of orders in each time period. We select a time period in which the order count is larger than the average count, leading to a more convenient and reliable prediction. The time duration between 7:00 AM to 10:00 PM is selected as the experimental time interval.
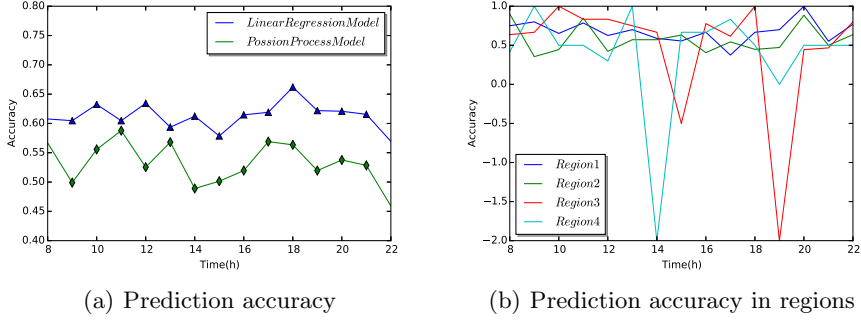
(a) Prediction accuracy

(b) Prediction accuracy in regions

Fig. 4. Passenger Demand Model Evaluation

4.3   Model Evaluation

We firstly introduce Non-homogeneous Poisson Process (NHPP) Model. Poisson process is a stochastic process that is often used to study the occurrence of events. It assumes that the arriving rate of events λ is constantly stable, and has the Poisson distribution of counting and exponential distribution of inter-event time. In fact, NHPP[8] is a specific Poisson process with a time-dependent arriving rate function $\lambda(t)$. The model is more flexible and appropriate to depict the human-related activities because these activities often vary over time but have strong periodicity. We use ΔT to indicate time increment and $N(\Delta T)$ to represent order increment in ΔT. Specifically, $N(\Delta T + j) - N(j)$ follows the Poisson distribution with a parameter $\lambda \Delta T$.

$$P\{N(\lambda T + j) - N(j) = k\} = \frac{e^{-\lambda \Delta T}(\lambda \Delta T)^k}{k!} \tag{8}$$

Here we set ΔT to one hour and λ can be obtained by maximum likelihood estimation:

$$\hat{\lambda} = \frac{\overline{N}}{\Delta T} \tag{9}$$

where $\overline{N}$ can be denoted by average orders in one hour. Therefore, we could use the average order count to predict passenger demand. We use the following equation to calculate the prediction accuracy:

$$Accuracy = \frac{P_{i,j+1} - |P_{i,j+1} - \hat{P}_{i,j+1}|}{P_{i,j+1}} \tag{10}$$

It is observable from Figure 4(a) that the prediction accuracy of our model outperforms NHPP, and the accuracy in different time periods varies in an acceptable range. Meanwhile, a peak accuracy could be achieved at 18:00. This is due to an obvious passenger demand trend around 18:00. Figure 4(b) also

Time	Vacant car count	Predicted order count	Reduced cruising miles
08:00	325	296	21.5%
09:00	332	369	23.1%
10:00	275	285	19.2%
11:00	247	212	24.9%
12:00	203	245	27.2%
13:00	243	210	26.3%

Table 2. Dispatching Results

reveals the prediction accuracy of different regions. Apparently, the accuracy for Region 1 and 2 is steady all the time with little fluctuation. However, the inference accuracy in Region 3 and 4 suffers from some degrees of variations, even with some outlier results. This is because the number of orders in these Regions are so scarce that the proposed model could not capture a reasonable value. To solve this problem, we not only enlarge the size of required dataset, but increase the region's coverage area as well.

4.4   Dispatching Algorithm Evaluation

In order to replay the real behaviors of the moving vehicles and the user demands, we emulate their moving traces obtained from the historical data and apply them into our algorithms. As shown in Table 2, we choose the real orders and car traces from 08:00 to 13:00 on May 6th,2015. At 8:00, there are 324 vacant cars and 296 predicted orders in this evaluation. The results illustrates that the proposed model could finish matching of vacant cars with predicted orders in 52 seconds. In addition, we leverage Baidu Web API to obtain the distance between a vacant car and a predicted order with average speed recorded. Once a real order emerges, the nearest vacant car could be selected to pick up the passenger. On average, 22.5% cruising miles could be reduced in this simulated experiment.

5   Conclusions and Future Work

At present, taxis in urban area could not fully satisfy the booming citizens' demand of convenient transportation tools. Fortunately, the emergence of private car hiring becomes an alternative and one of the significantly important business models, greatly increasing the variety of ways for a short trip. It could be observable that serving vehicles have to spend a large portion of their service time on cruising around streets, resulting in large amount of monetary waste. To solve this problem, this paper presents a private car dispatching method based on passenger demand model. We use Locally Weighted Linear Regression to depict passengers demand in detail, and the dispatching problem could be formalized as a weighted bipartite graph matching problem. The experimental results demonstrate our model outperforms non-homogeneous poisson process model with a more stable prediction accuracy. Finally, we also use a real dataset to simulate the private car dispatching process. By using the proposed approach, 22.5% cruising mileage can be reduced. In the future, we will further enlarge our

dataset to optimize the proposed model and to take more factors into consideration which affect passengers demand, such as weather, holidays etc. Moreover, we are also planning to parallelize the dispatching algorithm to further reduce the total execution time.

References

1. Uber, `https://www.uber.com`
2. Powell, J. W., Huang, Y., Bastani, F., & Ji, M. Towards reducing taxicab cruising time using spatio-temporal profitability maps.. Lecture Notes in Computer Science, 6849, 242-260 (2011)
3. Qu, M., Zhu, H., Liu, J., Liu, G., & Xiong, H. A cost-effective recommender system for taxi drivers. Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM. (2014)
4. Yuan, Jing, et al. "Where to find my next passenger.." Proceedings of the 13th international conference on Ubiquitous computingACM, 2011:109-118. (2011)
5. Huang Y, Powell J W. Detecting regions of disequilibrium in taxi services under uncertainty[J]. Proceedings of International Conference on Advances in Geographic Information Systems, 2012:139-148. (2012)
6. Zhang, D., He, T., Lin, S., Munir, S., & Stankovic, J. A. Dmodel: Online Taxicab Demand Model from Big Sensor Data in a Roving Sensor Network. Big Data (Big-Data Congress), 2014 IEEE International Congress on(pp.152-159). IEEE. (2014)
7. Zhang, Desheng, and T. He. "pCruise: Reducing Cruising Miles for Taxicab Networks." 2013 IEEE 34th Real-Time Systems SymposiumIEEE, 2012:85-94.
8. Zheng, X., Liang, X., & Xu, K. Where to wait for a taxi?. Proceedings of the ACM SIGKDD International Workshop on Urban Computing. ACM. (2012)
9. Yuan, N. J., Zheng, Y., Zhang, L., & Xie, X. T-finder: a recommender system for finding passengers and vacant taxis. IEEE Transactions on Knowledge & Data Engineering, 25(10), 2390-2403. (2013)
10. Ester, Martin, et al. "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise.." Proceedings of International Conference on Knowledge Discovery & Data Mining(1996):226-231.
11. Chang, H. W., Tai, Y. C., & Hsu, Y. J. Context-aware taxi demand hotspots prediction.. International Journal of Business Intelligence & Data Mining, 5(1), 3-18. (2010)
12. Kiam Tian Seow Nam Hai Dang Der-Horng Lee. A Collaborative Multiagent Taxi-Dispatch System[J]. Automation Science & Engineering IEEE Transactions on, 2010, 7(3):607 - 616 (2010)
13. Ge, Y., Xiong, H., Tuzhilin, A., Xiao, K., Gruteser, M., & Pazzani, M. An energy-efficient mobile recommender system. In Proc. KDD 2010(pp.899–908) (2010)
14. Li, Bin, et al. "Hunting or waiting? Discovering passenger-finding strategies from a large-scale real-world taxi dataset." Pervasive Computing and Communications Workshops (PERCOM Workshops), 2011 IEEE International Conference onIEEE, 2011:63-68 (2011)