MCSS: A Model for Car-Sharing System

Enquan Ge, Jian Xu, Ming Xu, Ning Zheng, Youcheng Wang and Jingsheng Jiang
School of Computer Science and Technology
Hangzhou Dianzi University
Hangzhou, China
{131050039, jian.xu, mxu, nzheng, 142050120, 142050123}@hdu.edu.cn

ABSTRACT

Car-sharing systems have become a promising research direction due to the development of the Internet. Previous research mainly focus on how to pick more passengers up, which ignores whether the real income of car-drivers is increasing. In this paper, we exploit a model for car-sharing system named "MCSS", which can make precise prediction on whether a passenger should be picked up while in an order. Our extensive empirical analysis shows the capability of getting maximum profit to drivers.

Keywords

Driver, car-sharing, passenger, maximum profit

1. INTRODUCTION

With the increasing amount of available information in the Internet, taxi systems emerge to be a hot spot for intelligent knowledge management. But taxis are always not enough for passengers' demands. So applications of car-sharing systems,such as Didi(left diagram of Figure1), Uber(right diagram of Figure1), are appeared to relieve the pressure on taxi using the private cars. In order to pick more passengers up and improve the utilization, they explore a car-sharing system. In this system, the driver could accept another order when his current order has one or two passengers. It seems to be a good idea to help drivers improve income. But sometimes driver could get more profit if he do not accept another order. The reason is followed:

First, the driver could only get a proportion of the original price. For example, if an order is a sharing-order, the driver could only get 70% of the original price in Didi and 75% in Uber. In addition, the predictable price is calculated according to the shortest path. It will be the final price no matter how bad traffic conditions are and how many roads the driver goes. They may earn more money because of the more elapsed time. Second, the driver should go through more streets to pick other passengers up and send passengers. So

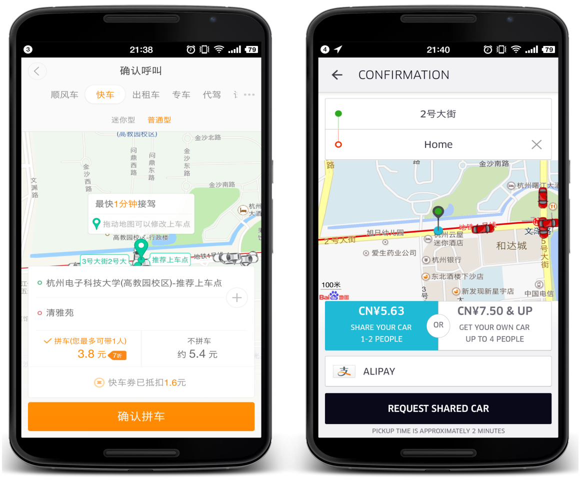

Figure 1: Applications of car-sharing systems

it will cost more by themselves while getting a relative low payment from passengers.

As show in the left diagram of Figure 2, if a driver get an order from P_1 to P_2 and accepts another order from P_3 to P_4. So he adopt the path of solid line on the road network. The driver could get 70% of two orders' original price. Here the original price of an order from P_1 to P_2 is calculated according to the path of dotted line. Thus the driver may get more profit if he does not accept another order.

In order to solve these problems, we propose a new model named model for car-sharing system(MCSS). In this model, we first calculate the original profit of an order. Then we compute the price of two orders. The cost will be the final path of two orders. So the profit of the two orders is minus of the price of two orders and cost of whole path. At last the profit is compared with the original profit and the driver could decide whether he would pick the passenger up.

The remainder of the paper is organized as follows: Section 2 formally defines the problem of whether a passenger should be picked up. Section 3 presents the details of our solution. Section 4 presents the experimental results. Section 5 surveys the related work. Section 6 concludes the paper.

2. PROBLEM STATEMENT

A road network G is a directed graph with a set of vertices V and a set of edges E. An original order O_1 is a path with a start position p_1 and a destination position p_2. Another

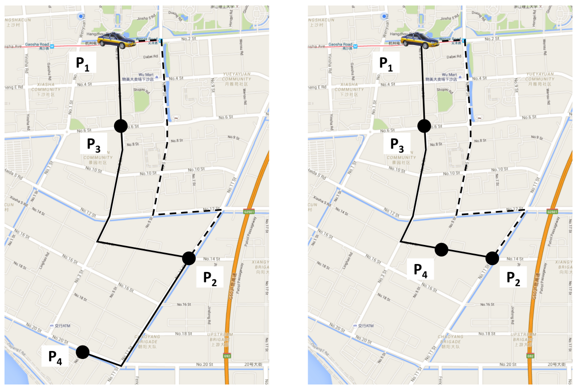

Figure 2: Car-sharing system - example

order O_2 is a path with a start position p_3 and a destination position p_4. Here p_1,p_2,p_3,$p_4 \in V$.

Problem Statement: Given a road network G, an original order O_1, another order O_2, return "accept" or "do not accept".

3. SOLUTION

To solve the problem we defined section 2, we expose algorithm MCSS as show in Algorithm 1. First, the shortest path of O_1 is found. Here is path $P < p_1, p_2 >$. Second, we compute the cost[2] of original order C_{before} and original income I_{before}. Third, the actual path is calculated. Note that there are two cases: 1) O_1 is finished before O_2, which is shown in left diagram of Figure 2; 2)O_2 is finished before O_1, which is shown in right diagram of Figure 2. So two paths $P < p_1, p_3, p_2, p_4 >$, $P < p_1, p_3, p_4, p_2 >$ are computed and the shortest one of them is selected. Fourth, the cost of shortest path C_{after} and income of two orders $I_{after}(I_{passenger1} + I_{passenger2})$are calculated. At last we compare the profit of the original order with the two orders. Here we use 0.7 which is the proportion of Didi as the ratio of the two orders. If the profit of the original order is higher, Algorithm 1 returns "accept". Otherwise, Algorithm 1 returns "do not accept".For multiple orders, the driver could adopt MCSS repeatedly till MCSS return "accept" for a car-sharing order.

Algorithm 1 MCSS
```
Require:
    $G(V,E)$; $O_1(p_1,p_2)$:original order; $O_2(p_3,p_4)$:another order
Ensure:
    "accept" or "do not accept"
1: Find the shortest route $P_{before} < p_1, p_2 >$ between current position of taxi and the destination point of first passenger;
2: Calculate the cost $C_{before}$ and income $I_{before} = I_{passenger1}$ of $P_{shortest} < p_1, p_2 >$;
3: Find shortest route $P < p_1, p_3, p_2, p_4 >$ between $p_1$,$p_3$,$p_2$,$p_4$ and shortest route $P < p_1, p_3, p_4, p_2 >$ between $p_1$,$p_3$,$p_4$,$p_2$;
4: $P_{after} \leftarrow argminP < p_1, p_3, p_2, p_4 >, P < p_1, p_3, p_4, p_2 >$;
5: Get the income of second passenger $I_{passenger2}$;
6: Calculate the cost $C_{after}$ of $P_{after}$;
7: Return "accept" if $I_{before} - C_{before} \leq 0.7 * (I_{passenger1} + I_{passenger2}) - C_{after}$, else return "do not accept";
```

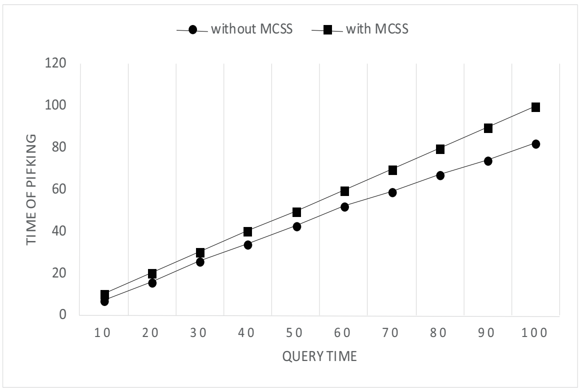

Figure 3: The times of picking passengers and query times

4. EXPERIMENT

In this section we perform an analysis of a system without MCSS and a system with MCSS. All the experiments are conducted on a PC with an Intel CPU of 3.20GHz and 4GB memory. This paper uses a real map Oldenburg. The map Oldenburg has 6105 nodes and 7035 edges. We simulate the orders and drivers using the famous Network-based Generator of Moving Objects by Tomas Brinkhoff[1].

To evaluate the advantages of MCSS, we compare the results of the system with MCSS and the system without MCSS. We apply 10 to 100 queries of whether a passenger should be picked up. For each pair of sharing orders(O_1 and O_2), the similarity of O_1 and O_2 is about 50%, which means the number of same road sections between them are about 50% of the number of the order with small number of road sections. The results is shown in Figure 3, the system with MCSS help driver earn maximum profit each time. But the system without MCSS help drivers earn about 80% of more profit. This shows that the system with MCSS is effective to get maximum profit. In addition, the extra complexity of MCSS is only O(1). So it has no influence on the system with MCSS.

5. RELATED WORK

Urban computing with taxicabs is well-studied in the late years. Taxi based mobile applications are designed for taxi-drivers and passengers to make better decisions while hailing taxicabs or picking up passengers. In recent years, mobile applications like Uber and Didi-Taxi are widely used among citizens in metropolis. Yuan et al.[7],[5],[8] and Zheng et al.[9] provide taxi-based systems to help passengers hailing taxis and increasing drivers' income as well. However, these recommender systems do not contain car-sharing module and they also do not consider drivers' actual earnings.

In order to pick more passengers up and improve car utilization, applications of car-sharing systems are appeared to relieve the pressure on taxi using the private cars. Huang et al.[3] and Ma et al.[5] explore car-sharing systems. In these car-sharing systems, a driver could accept another order if his current remaining-seat is enough for the new request. These application systems provide car-sharing service with searching and schedule module inside. They aim to reduce passengers' payment for a taxi and increase drivers' income per trip in the meanwhile. But if we consider the taxi-charging strategy[6], travel time[4] and fuel consumption[2], the drivers who use the car-sharing system might probably without much respectable earning comparing to non-car-sharing situation. Thus many extra costs are not considered while recommending one more request to a driver in this situation. They are therefore not suitable for MCSS.

6. CONCLUSIONS

In this paper, we propose and study a new model for car-sharing system named MCSS. We also introduce algorithm MCSS to compare the income before and after car sharing. Experiment shows the effectiveness of the car-sharing system with MCSS. So it is convenient for drivers to make a decision whether to pick up a second passenger and earn maximum profit while using MCSS.

7. ACKNOWLEDGMENTS

This work is supported by the National Natural Science Foundation of China (No. 61572165).

8. REFERENCES

[1] T. Brinkhoff. Generating network-based moving objects. In *Scientific and Statistical Database Management, 2000. Proceedings. 12th International Conference on*, pages 253–255, 2000.

[2] C. Guo, Y. Ma, B. Yang, C. S. Jensen, and M. Kaul. Ecomark: Evaluating models of vehicular environmental impact. In *International Conference on Advances in Geographic Information Systems*, pages 269–278, 2013.

[3] Y. Huang, R. Jin, F. Bastani, and X. S. Wang. Large scale real-time ridesharing with service guarantee on road networks. *Proceedings of the Vldb Endowment*, 7(14), 2013.

[4] Y. Wang, Y. Zheng, and Y. Xue. Travel time estimation of a path using sparse trajectories. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 25–34, 2014.

[5] O. Wolfson, Y. Zheng, and S. Ma. T-share: A large-scale dynamic taxi ridesharing service. In *Data Engineering, IEEE International Conference on*, pages 410 – 421, 2013.

[6] Z. Yang, L. Sun, M. Ke, Z. Shi, and J. Chen. Optimal charging strategy for plug-in electric taxi with time-varying profits. *IEEE Transactions on Smart Grid*, 5(6):2787–2797, 2014.

[7] J. Yuan, Y. Zheng, X. Xie, and G. Sun. T-drive: Enhancing driving directions with taxi drivers' intelligence. *IEEE Transactions on Knowledge Data Engineering*, 25(1):220–232, 2013.

[8] N. J. Yuan, Y. Zheng, L. Zhang, and X. Xie. T-finder: A recommender system for finding passengers and vacant taxis. *IEEE Transactions on Knowledge Data Engineering*, 25(10):2390–2403, 2013.

[9] Y. Zheng, Y. Liu, J. Yuan, and X. Xie. Urban computing with taxicabs. In *Proceedings of the 13th international conference on Ubiquitous computing*, pages 89–98, 2011.