Efficient Indexing and Scheduling For Real Time Android App with Taxi Sharing

M. Dhavamani[1], G. Sathish Kumar[2]
[1, 2] Department of Computer Science and Engineering
[1, 2] MNSK College Engineering, Pudhukkottai

Abstract- *Proposed and developed a taxi-sharing system that accepts taxi passengers' real-time ride requests sent from smart phones and schedules proper taxis to pick up them via ridesharing, subject to time, capacity, and monetary constraints. The monetary constraints provide incentives for both passengers and taxi drivers: passengers will not pay more compared with no ridesharing and get compensated if their travel time is lengthened due to ridesharing; taxi drivers will make money for all the detour distance due to ridesharing. Taxi riders and taxi drivers use the taxi-sharing service provided by the system via a smart phone App. The Cloud first finds candidate taxis quickly for a taxi ride request using a taxi searching algorithm supported by a spatio-temporal index. A scheduling process is then performed in the cloud to select a taxi that satisfies the request with minimum increase in travel distance. A ride request generator is developed in terms of the stochastic process modeling real ride requests learned from the data set.*

Keywords- Spatial databases and GIS, Taxi-Sharing, GPS trajectory, Ridesharing, Urban Computing, Intelligent Transportation Systems

I. INTRODUCTION

Proposed and developed a taxi-sharing system that accepts taxi passengers' real-time ride requests sent from smart phones and schedules proper taxis to pick up them via ridesharing, subject to time, capacity, and monetary constraints. The monetary constraints provide incentives for both passengers and taxi drivers: passengers will not pay more compared with no ridesharing and get compensated if their travel time is lengthened due to ridesharing; taxi drivers will make money for all the detour distance due to ridesharing.

Taxi riders and taxi drivers use the taxi-sharing service provided by the system via a smart phone App. The Cloud first finds candidate taxis quickly for a taxi ride request using a taxi searching algorithm supported by a spatio-temporal index. A scheduling process is then performed in the cloud to select a taxi that satisfies the request with minimum increase in travel distance. A ride request generator is developed in terms of the stochastic process modeling real ride requests learned from the data set.

II. PROBLEM DESCRIPTION

Multiple taxi statues may satisfy a ride request, an objective function is usually applied to find the optimal taxi. A variety of objective functions have been used in the existing literature, where a weighted cost function combining multiple factors such as travel distance increment, travel time increment and passenger waiting time, is the most common aim to find the taxi status which satisfies the ride request with minimum increase in travel distance, formally defined as follows: given a fixed number of taxis traveling on a road network and a sequence of ride requests in ascending order of their submitted time, we aim to serve each ride request Q in the stream by dispatching the taxi V which satisfies Q with minimum increase in V 's scheduled travel distance on the road network.

This is obviously a greedy strategy and it does not guarantee that the total travel distance of all taxis for all ride requests is minimized. However, we still opt for this definition due to two major reasons. First, the real-time taxi sharing problem inherently resembles a greedy problem. In practice, taxi riders usually expect that their requests can be served shortly after the submission. Given the rigid real-time context, the taxi-sharing system only has information of currently available ride requests and thus can hardly make optimized schedules based on a global scope, i.e., over a long time span. Second, the problem of minimizing the total travel distance of all taxis for the complete ride request stream is NP-complete.

2.1 The Real-Time Taxi-Sharing

The real-time taxi-sharing problem consists of a data model, constraints, and an objective function. We describe each part separately below before giving the formal definition of the problem.

Propose a framework to address the formulated accuracy estimation problem which has the following layers.

1. Taxi Auto Updating
2. Rider Request
3. Taxi Search
4. Request Scheduling

5. Ride Sharing

The framework firstly activates all the coexisting sensor systems that can monitor the measurements on the queried states for a given time stamp, and the raw observed measurements pulled from the systems are firstly processed through the pre-processing layer. Then the processed measurements are forwarded to the state estimation layer, where the states of monitored measurements are estimated based on sensor observations and available prior over single timestamp [16]. The next accuracy estimation layer takes the estimated states of the monitored value together with the pre-processed sensor measurements as input, and evaluates the accuracy of measurements with the help of accuracy metric.

2.2 Taxi Search

The taxi searching module quickly selects a small set of candidate taxis with the help of the spatio-temporal index. The spatio-temporal index of taxis is built for speeding up the taxi searching process. Specifically, we partition the road network using a grid. (Other spatial indices such as R tree can be applied as well, but we envision that the high dynamics of taxis will cause prohibitive cost for maintaining such an index.). This module mainly used for two type of search techniques first Single Side Taxi Searching techniques and second Dual Side Taxi Search Techniques. These two techniques used for taxi searching for very fast way.

Spatio-Temporal Index

The spatio-temporal index of taxis is built for speeding up the taxi searching process. Specifically, we partition the road network using a grid. (Other spatial indices such as R tree can be applied as well, but we envision that the high dynamics of taxis will cause prohibitive cost for maintaining such an index.)

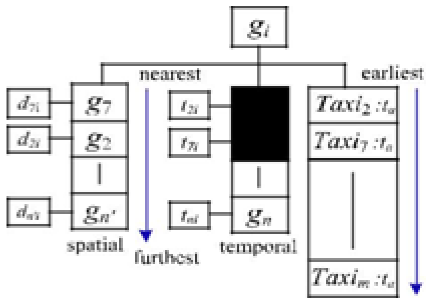


Fig. 1. Spatio-temporal index of taxis

.

Searching Algorithms

4.2.1 Single-Side Taxi Searching

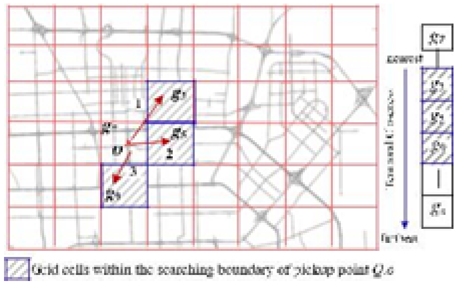

g_7
nearest
furthest
Temporal Closeness
Grid cells within the searching boundary of pickup point Q.o

Fig.2. Single Side Taxi Searching

Now we are ready to describe our first taxi searching algorithm. For the sake of the clarity of description, please con-sider the example shown in Fig. 2. Suppose there is a query Q : and the current time is t_{cur} : g_7 is the grid cell in which Q: o is located. g_7's temporally-ordered grid cell list g_7 : l^t_g is shown on the right of Fig. 7. g_7 is the first grid cell selected by the algorithm. Any other arbitrary grid cell g_i is selected by the searching algorithm if and only if Eq. (1) holds, where t_{i7} represents the travel time from grid cell g_i to grid cell g_7. Eq. (1) indicates that any taxi currently within grid cell g_i can enter g_7 before the late bound of the pickup window using the travel time between the two grid cells (if we assume that each grid cell collapses to its anchor node)

4.2.2 Dual-Side Taxi Searching

The dual-side searching is a bi-directional searching process which selects grid cells and taxis from the origin side and the destination side of a query simultaneously. To dive into the details of the algorithm, consider the ride request illustrated in Fig. 9 where g_7 and g_2 are the grid cells in which Q : o and Q : d are located respectively. Squares filled with stripes stand for all possible cells searched by the algorithm at Q : o side. These cells are determined by scan-ning g_7 : l^t_c, the temporally-order grid cell list of g_7. That is, each grid cell in g_7 : l^t_c which holds Eq. (2) is a candidate cell to be searched at the origin side.

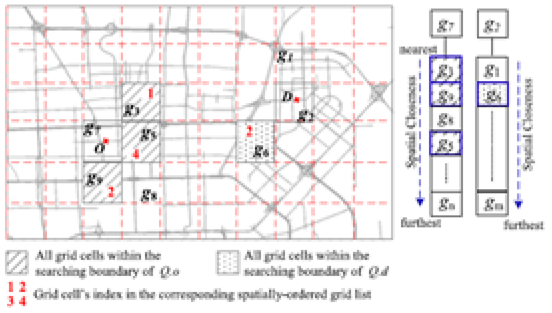
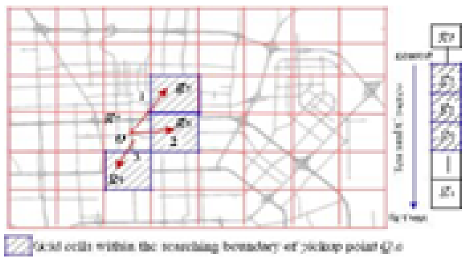

g_7
g_2
nearest
furthest
Spatial Closeness
All grid cells within the searching boundary of Q.o
All grid cells within the searching boundary of Q.d
Grid cell's index in the corresponding spatially-ordered grid list

Fig 3. Dual-Side Taxi Searching algorithm

Eq. (2) indicates that any taxi currently within grid cell g_i can enter g_7 before the late bound of the pickup window using the latest travel time between the two grid cells (assuming each grid cell collap-ses to its anchor node). The red number in each such grid cell indicates its relative position in $g_7 : l^s_c$, the spatially-ordered grid list of g_7

$$t_{cur} þ t_{i7} \quad Q : dw : l: \qquad (2)$$

Squares filled with dots indicate the candidate grid cells to be accessed by the searching algorithm at Q : d side. Like-wise, each such grid cell g_j is found by scanning $g_2 : l^t_c$ to select all grid cells which holds Eq. (3), which indicates that any taxi currently in g_j can enter the g_2 before the late bound of the delivery window (assuming that each grid cell collap-ses to its anchor node). In this example, g_6 is the only satisfy-ing grid cell as shown by Fig. 9


Grid cells within the searching boundary of pickup point g7.o

$$t_{cur} þ t_{j2} \quad Q : dw : l: \qquad (3)$$

K-Nearest Neighbors

The first algorithm we shall investigate is the k-nearest neighbor algorithm, which is most often used for classification, although it can also be used for estimation and prediction. K-Nearest neighbor is an example of instance-based learning, in which the training data set is stored, so that a classification for a new unclassified record may be found simply by comparing it to the most similar records in the training set. Let's consider an example. Recall the example from Chapter 1 where we were interested in classifying the type of drug a patient should be prescribed based on certain patient characteristics, such as the age of the patient and the patient's sodium/potassium ratio. For a sample of 200 patients, Figure 5.6 presents a scatter plot of the patients' sodium/potassium (Na/K) ratio against the patients' age. The particular drug prescribed is symbolized by the shade of the points. Light gray points indicate drug Y; medium gray points indicate drug A or X; dark gray points indicate drug B or C.

III. PROPOSED SYSTEM

A system based on the mobile cloud architecture, which enables real-time taxi-sharing in a practical setting. Taxi drivers independently determine when to join and leave the service using an App installed on their smart phones. Passengers submit real-time ride requests using the same App (if they are willing to share the ride with others).Each ride request consists of the origin and destination of the trip, time windows constraining when the passengers want to be picked up and dropped off (in most case, the pickup time is present).

On receiving a new request, the Cloud will first search for the taxi which minimizes the travel distance increased for the ride request and satisfies both the new request and the trips of existing passengers who are already assigned to the taxi, subject to time, capacity, and monetary constraints. The existing passengers assigned to the taxi will be inquired by the cloud whether they agree to pick up the new passenger given the possible decrease in fare and increase in travel time.

3.1 Objective

Easily to identify the available taxi and also use the single-side taxi searching algorithm, the dual-side taxi searching algorithm reduced the computation cost by over 50 percent, while the travel distance was only about 1 percent higher on average.

3.2 System Architecture

System design is the process of defining the architecture, components, modules, and data for a system to satisfy specified requirements. One could see it as the application of systems theory to product development. There is some overlap with the disciplines of systems analysis, systems architecture and systems engineering. If the broader topic of product development blends the perspective of marketing, design, and manufacturing into a single approach to product development, then design is the act of taking the marketing information and creating the design of the product to be manufactured. System design is therefore the process of defining and developing systems to satisfy specified requirements of the user.

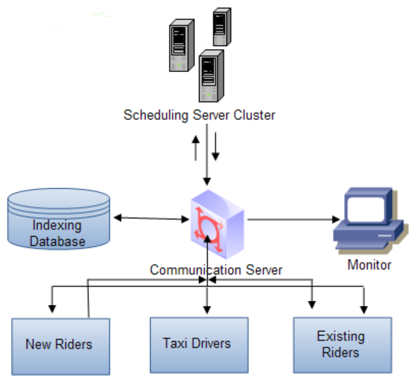


Fig.3.1 System Architecture Diagrams

3.3 Taxi Auto Updating

The taxi driver registers our taxi info in storage server and a taxi automatically reports its location to the server via the mobile App when the taxi establishes the connection with the system, or a rider gets on and off a taxi, or at a frequency (e.g., every 20 seconds) while a taxi is connected to the system. The taxi driver view and modify the taxi information through our mobile app and also view rider info for searching the taxi.

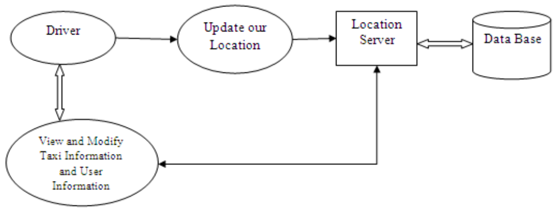


Fig.3.2 Taxi Auto Updating

3.4 Ride Request

A Rider submits a new ride request Q to the Communication Server, the corresponding interface on a rider's smart phone where the stands for the current location of the rider. All incoming ride requests of the system are streamed into a queue and then processed according to the first-come-first serve principle. For each ride request Q, the communication server sends it to the Indexing Server to search for candidate taxis SV that are likely to satisfy Q, Using the maintained spatio-temporal index, the indexing server returns SV to the communication server, communication server sends ride request Q and the received candidate taxi set SV to the Scheduling Server Cluster. After schedule a ride request finally to response for nearest taxi to rider mobile app.

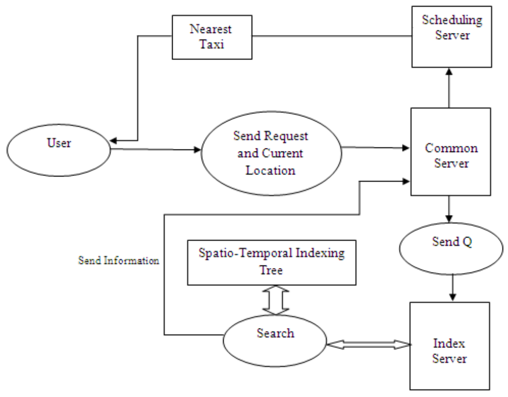


Fig.3.3 Ride Request

3.5 Taxi Search

The taxi searching module quickly selects a small set of candidate taxis with the help of the spatio-temporal index. The spatio-temporal index of taxis is built for speeding up the taxi searching process. Specifically, we partition the road network using a grid. (Other spatial indices such as R tree can be applied as well, but we envision that the high dynamics of taxis will cause prohibitive cost for maintaining such an index.). This module mainly used for two type of search techniques first Single Side Taxi Searching techniques and second Dual Side Taxi Search Techniques. These two techniques used for taxi searching for very fast way.

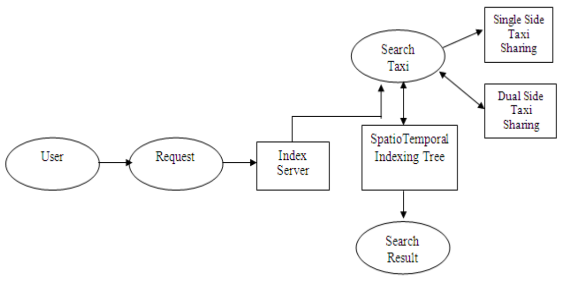


Fig.3.5 Taxi Search

3.6 Request Scheduling

The set of taxi statuses SV retrieved for a ride request Q by the taxi searching algorithm, the purpose of the taxi scheduling process is to find the taxi status in SV which satisfies Q with minimum travel distance increase. To this end, given a taxi status, to try all possible ways of inserting Q into the schedule of the taxi status in order to choose the insertion which results in minimum increase in travel distance. All possible ways of insertion can be created via three steps: (i) reorder the points in the current schedule, subject to the precedence rule, i.e., any origin point precedes the corresponding destination point. (ii) insert Q:o into the schedule (iii) insert the Q:d into the schedule. The capacity and time window constraints are checked in all three steps, during which the insertion fails immediately if any constraint is violated. The monetary constraints are then checked for the insertion after all three steps have been done successfully. Finally, among all insertions that satisfy all constraints, choose the insertion that results in minimum increase in travel distance for the given taxi status.

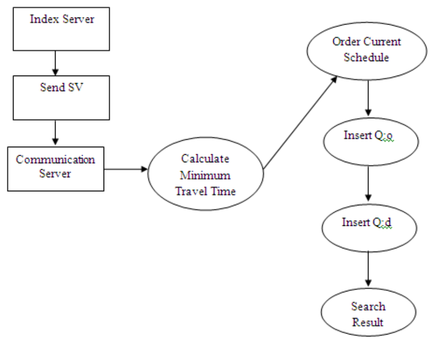


Fig.3.3 Ride Scheduling

3.7 Ride Sharing

This module imposes two constraints which encourage riders to participate in taxi-sharing by rewarding them with certain monetary gains. The first rider monetary constraint says that any rider who participates in taxi-sharing should pay no more than what she would pay if she takes a taxi by herself. The second rider monetary constraint says that if an occupied taxi V is to pick up a new rider Q, then each rider P currently sitting in V whose travel time is lengthened due to the pickup of Q, should get a decrease in taxi fare; and the fare decrease should be proportional to P's increase in travel time. One constraint which gives the driver motivation to participate in taxi-sharing. This constraint says that a driver should charge for all distances she has travelled. Intuitively the driver should make money for the distance of reroutes incurred by the join of any new passenger

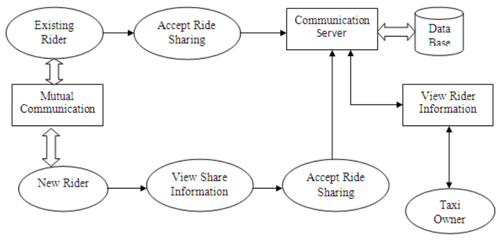
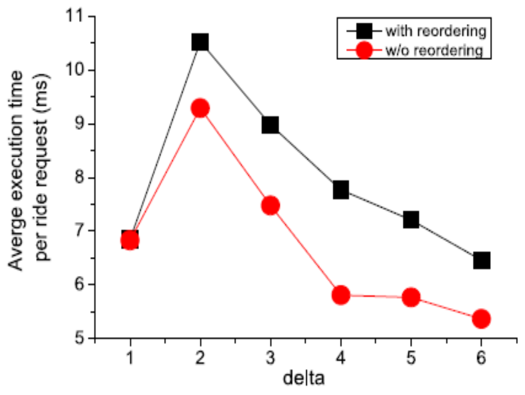


Fig.3.7 Ride Sharing

IV. RESULT AND DISCUSSION

All ridesharing methods first show a slight increase and then keep declining in SR as D increases. This is because as the value of parameter D starts increasing, the ridesharing opportunist increase as well. When D is small, the increase in ridesharing opportunies is larger than the increase in the number of request, as a result, the satisfaction rate surges.

Two first-fit based taxi-sharing methods show clearly higher relative distance rate. From the picture, we can see that taxi-sharing methods save up to 12 percent in travel distance, depending on delta. Given the fact that there are 67,000 taxis in Beijing (not in the data set) and each taxi runs 480 km per day (learned from the data set), the saving achieved by taxi-sharing here means over 1.5 billion kilometers in distance per year, which equals to 120 million liter of gas per year (supposing a taxi consumes 8 liter of gasoline per 100 km) and 2.2 million of carbon dioxide emission per year (supposing each liter of gas consumption generates 2.3 kg of carbon dioxide).



V. CONCLUSION

This paper proposed and developed a mobile-cloud based real-time taxi-sharing system. We presented detail interactions between end users (i.e. taxi riders and drivers) and the Cloud. We validated our system based on a GPS trajectory data set generated by 33,000 taxis over three months, in which over 10 million ride requests were extracted. The experimental results demonstrated the effectiveness and efficiency of our system in serving real-time ride requests. Firstly, our system can enhance the delivery capability of taxis in a city so as to satisfy the commute of more people. For instance, when the ratio between the number of taxi ride requests and the number of taxis is 6, our proposed system served three times as many ride requests as that with no taxi-sharing. Secondly, the system saves the total travel distance of taxis when delivering passengers, e.g., it saved 11 percent travel distance with the same ratio mentioned above.

Supposing a taxi consumes 8 liters of gasoline per 100 km and given the fact learned from the real trajectory data set that the average travel distance of a taxi in a day in Beijing is about 480 km, the system can save over one third million liter of gasoline per day, which is over 120 million liter of gasoline per year (worth about 150 million dollar). Thirdly, the system can also save the taxi fare for each individual rider while the profit of taxi drivers does not decrease compared with the case where no taxi-sharing is conducted. Using the proposed monetary constraints, the system guarantees that any rider that participates in taxi-sharing saves 7 percent fare on average. In addition, the experimental results justified the importance of the dual-side searching algorithm. Compared to the single- side taxi searching algorithm, the dual-side taxi searching algorithm reduced the computation cost by over 50 percent, while the travel distance was only about 1 percent higher on average. The experimental results also suggest that reordering the points of a schedule before the insertion of the new ride request is not necessary in practice for the purpose of travel distance minimization.

REFERENCES

[1] R. Baldacci, V. Maniezzo, and A. Mingozzi, "An exact method for the car pooling problem based on lagrangean column generation," Oper. Res., vol. 52, no. 3, pp. 422–439, 2004.

[2] R. W. Calvo, F. de Luigi, P. Haastrup, and V. Maniezzo, "A distributed geographic information system for the daily carpooling problem," Comput. Oper. Res., vol. 31, pp. 2263–2278, 2004.

[3] S. Ma, Y. Zheng, and O. Wolfson, "T-Share: A large-scale dynamic ridesharing service," in Proc. 29th IEEE Int. Conf. Data Eng., 2013, pp. 410–421.

[4] E. Kamar and E. Horvitz, "Collaboration and shared plans in the open world: Studies of ridesharing," in Proc. 21st Int. Jont Conf. Artif. Intell., 2009, pp. 187–194.

[5] K. Wong, I. Bell, and G. H. Michael, "Solution of the dial-a-ride problem with multi-dimensional capacity constraints," Int. Trans. Oper. Res., vol. 13, no. 3, pp. 195–208, May 2006.

[6] Z. Xiang, C. Chu, and H. Chen, "A fast heuristic for solving a large-scale static dial-a-ride problem under complex constraints," Eur. J. Oper. Res., vol. 174, no. 2, pp. 1117–1139, 2006.

[7] J. Yuan, Y. Zheng, C. Zhang, W. Xie, X. Xie, G. Sun, and Y. Huang, "T-drive: Driving directions based on taxi trajectories," in Proc. 18th SIGSPATIAL Int. Conf. Adv. Geographic Inf. Syst., 2010, pp. 99–108.

[8] J. Yuan, Y. Zheng, X. Xie, and G. Sun, "Driving with knowledge from the physical world," in Proc. 17th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, 2011, pp. 316–324.

[9] O. Wolfson, A. P. Sistla, B. Xu, J. Zhou, S. Chamberlain, Y. Yesha, and N. Rishe, "Tracking moving objects using database technology in DOMINO," in Proc. 4th Int. Workshop Next Generation Inf. Technol. Syst., 1999, pp. 112–119.

[10] J. Yuan, Y. Zheng, C. Zhang, X. Xie, and G.-Z. Sun, "An interactive-voting based map matching algorithm," in Proc. 11th Int. Conf. Mobile Data Manage. 2010, pp. 43–52.