Dynamic Ridesharing

Bilong Shen[1], Yan Huang[2], Ying Zhao[3]
[1]Computer Science, Tsinghua University, China
[2]Computer Science and Engineering, University of North Texas, U.S.A
[3]Computer Science, Tsinghua University, China

Abstract

Ridesharing, which offers empty seats in a car to other passengers, is an efficient way of transportation. In this way, the utilization of seats can be improved and the number of cars used can be reduced. Ridesharing has the potential to solve the problems of congestion, pollution, high travel cost, and energy. The development of internet, smart phone, GPS allows dynamic matchings of travel requests with available cars through real-time travel planning systems. However, matching requests and cars under certain constrains in large scale remains challenging. In this paper, we formally address the problem of dynamic ridesharing and introduce the solution framework of filter and refine, under which we summarize existing state-of-the-art works. Finally, we point out possible research directions and problems needed to be solved.

1 Introduction

With the development of the urban and metropolitan, the number of private cars is growing tremendous. The private cars not only bring convenience, but also bring worsening traffic congestion, increasingly serious pollution, and increasing energy consumption. On the one hand the number of vehicles is growing, but on the other hand the rate of empty seats on the moving vehicles has not been improved. The average occupancy rate of the private car in United States is only 1.6 persons per vehicle mile [10]. Ridesharing is a pattern of transportation in which people with similar itineraries and time schedules utilize spare seats in the vehicle and share the travel cost [9]. Numerous studies show that ridesharing is a good solution with triple Win [4, 9]. First win is the participants(e.g., reducing of the cost of the drivers and the riders by splitting the cost of gas, toll ,and parking fee, more convenient for traveling). Second win is the environment(e.g., less emissions, less noise, less fuel consumption). Third win is the social(e.g., alleviating traffic jams, integration of idle resources). As the so many benefits, more and more people be attracted to the travel pattern of ridesharing.

Currently, many mobile phone applications are providing their ridesharing services, e.g. Avego, Mitfahrgelegenheit, Zimride, Carpooling, Blablacar, and Didi. Mobile transportation platforms such as Uber also provide ridesharing travel options. Participants, society, and environment can get more and more benefit from ridesharing. Ridesharing can be either static or dynamic [9, 10]. Most ridesharing systems operating today belong to **static ridesharing**, which arrange the driver and passengers before trips start and the matching can't be made after a trip starts. **Dynamic ridesharing** is a service which can match real-time trip requests with running vehicles, is more convenient, and provide more flexibility to the passengers and drivers. Because of the potential advantage of real-time dynamic ridesharing, much research effort has be launched recently.

The real-time ridesharing at urban scale brings benefits and challenges at the same time. The core technical challenge is the complexity of the matching process. The static ridesharing route planing algorithm is not

suitable for real-time matching. The first challenge is that the vehicles are moving fast on the road network. The second challenge is that the route planning must not only satisfy the constraints of the new request but also the requests confirmed. Various research work has been done, e.g. minimizing the vehicle's traveling distance [2, 17, 8, 1, 15, 7, 3], maximizing the rate of the requests been matched [1, 15, 11, 3], and minimizing the system response time [17, 15, 11, 20, 7, 3, 12, 16, 22, 2] . And many studies have been conducted based on real data sets. In this paper we organize existing work of real-time ridesharing under a filter and refine framework.

We recognize that computation is one of the technical problems to be solved in order to enable wide adoption of ridesharing. Other factors such inter-personal interaction, safety, social discomfort, and pricing are also important. The reputation systems and the pricing mechanism for dynamic real-time ridesharing have been studied studied by the researchers. Ridesharing is an social activity and trust is very important to enable such an activity. The research in the trust area is still in its early stage. Researchers have tried to use call description records to quantify the potential of ridesharing [5] and others investigated the reputation system in a ridesharing system [21]. Different pricing mechanisms are also been studied with different goals including fairness, deficit control, and promoting the rate of matching [14, 23] and privacy issues are also be considered[18].

2 Dynamic Ridesharing System

2.1 Overview

In a dynamic ridesharing service system, as shown in Figure 1, a set of vehicles running over a road network. Real time requests, consisting of two points, an origin and a destination, are received in real-time. Each request also specifies two constrains, a *waiting time*, defining the maximal time the rider can wait after making the request, and a *service constraint*, defining the acceptable extra detour time from the shortest possible trip duration for ridesharing. After the request is submitted to the service server, the server processes it immediately. The server will match a request with an appropriate vehicle based on road network and travel constraints of the request and plan an efficient scheduling. When matching and scheduling are made, the constraints of the new request and the requests already assigned to the vehicle must be both satisfied.

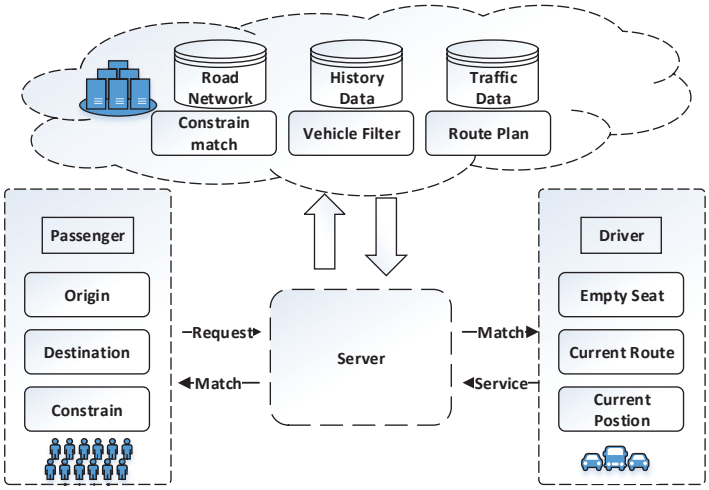

Figure 1: Dynamic ridesharing system framework

2.2 Basic Definition

A road network $G = \langle V, E, W \rangle$ consists of a vertex set V and an edge set E. V represents the intersection of the road. Each edge $(u, v) \in E(u, v \in V)$ is associated with a weight $W(u, v)$ which represents the cost form u to v. The cost can be either time or distance which can be converted from one to the other. The minimal cost from x to y ($x, y \in V$) on the road network is denoted by $dist(x, y)$.

Definition 1: Trip Request. Over a road network, a trip quest $tr = \langle o, d, p_n, t_{ow}, t_{dw}, \epsilon \rangle$ is defined by an origin $o \in V$, a destination $d \in V$, a number of passengers p_n, a pick up time window $t_{ow} = \langle t_{o.s}, t_{o.e} \rangle$ ($t_{o.s}$ and $t_{o.e}$ represent the start and the end time when the passenger can be picked up) , a drop off time window $t_{dw} = \langle t_{d.s}, t_{d.e} \rangle$ ($t_{d.s}$ and $t_{d.e}$ represent the start and the end time when the passenger can be dropped off at the destination), a service constraint ϵ (the extra detour acceptable in a trip, bounding the overall time from o to d by $(1 + \epsilon)dist(o, d)$).

In a particular road network G, the total trip requests set of the current time is denoted by TR.

Definition 2: Car State. A cruising car is represented by $car = \langle id, t, loc, TR_{rec}, shchedule, n_{cap}, n_{emp} \rangle$, where id is a unique ID, t is a timestamp, loc is the current location of car, TR_{rec} is a set of accepted trips, $schedule$ is the route schedule of accepted trips, n_{cap} is the total number of seats of the car, and n_{emp} is the available seats at current time . A schedule is represented by $schedule = \langle v_0, v_1, ..., v_k \rangle, (v_i \in V, (v_i, v_{i+1}) \in E)$, where the cost of the schedule is $\Sigma_0^k W(v_i, v_{i+1}), (v_i \in V)$.

In a particular road network, all the cars of the current time is represented by $C = \langle car \rangle$.

Definition 3: Dynamic Ridesharing. Given a set of cars C on the road network G at a particular time, the real-time trip requests set TR, the goal is to match $tr \in TR$ to $car \in C$ in a particular optimization goal under the dynamic ridesharing constraints.

In the subsection below, the optimization goal and the constraints will be discussed.

2.3 Constraints

Different ridesharing system has different constraints such as waiting time, detour as well as safety, pricing concerns, and social comfort. In this paper we focus on the service constraints. The following are the major constraints in dynamic ridesharing.

- **Available Seats Constraint.** The number of riders of a trip tr_i is not allowed to exceed the number of available seats of the car:

$$tr_i.p_n \leq car.n_{emp}. \tag{1}$$

- **Time Constraint.** As mentioned above, after a trip is assigned to a car, the time window of all the trip tr, including the new trip received and the trips already assigned to the car, should be satisfied. Let $tr.t_{vo}$ represent the time that the car can pick the rider at $tr.o$, and $tr.t_{vd}$ represent the time that the car can drop off the rider at $tr.d$.

$$\forall tr_i \in car.TR_{rec}, tr_i.t_{o.s} \leq tr_i.tr_{vp} \leq tr_i.t_{o.e}, tr_i.t_{d.s} \leq tr_i.t_{vd} \leq tr_i.t_{d.e} \tag{2}$$

- **Detour Constraint.** Equally important is the detour constraint. A trip can be matched to a car, only if the detour rate of all the trip tr including the new trip received and the trips already assigned to the car, are satisfied. The cost of a trip tr_i from $tr_i.o$ to $tr_i.d$ without ridesharing is $dist(tr_i.o, tr_i.d)$. During ridesharing a route may include detour. Let $dist_r(tr_i.o, tr_i.d)$ represent the total distance of the trip on a possibly detoured ridesharing route between the origin $tr_i.o$ at time $tr_i.t_{vo}$ and the destination $tr_i.d$ at time $tr.t_{vd}$. Then $\delta_i = dist_r(tr_i.o, tr_i.d)/dist(tr_i.o, tr_i.d)$.

$$\forall tr_i \in car.TR_{rec}, \delta_i \leq \epsilon \tag{3}$$

Table 1: Constraints and Optimization Goals

Research	Method	Author	Driver Cost	Matching Rate	Time window	Detour Cost
SIGMOD13 [20]; VLDB14 [12]	Noah	Yan Huang et al.	√		√	√
EUR J OPER RES14 [16]	People and parcels sharing taxis.	Baoxiang Li et al.	√		√	√
ICTAI14 [2]	Minimising the Driving Distanc	Vincent Armant and	√		√	
ICDE12 [17]	T-Share	Shuo Ma et al.	√		√	
ITSC12 [8]	Distributed Taxi-Sharing System	Pedro M. dOrey et al.	√		√	
TRANSPORT RES B-METH11 [1]	A simulation study in metro Atlanta	Niels A.H. Agatz et al.	√	√	√	
IJCAI11[15]	Parallel Auctions	Alexander Kleiner et al.	√	√		
MATES09 [22]	SMIZE	Xin Xing et al.				√
EDBT08 [11]	Highly scalable trip grouping	GyozoGidofalvi et al.		√		
EUR J OPER RES06 [7]	A two-phase insertion technique	Luca Coslovich et al.	√		√	
PARALLEL COMPUT04 [3]	Parallel Tabu search heuristics	Andrea Attanasio et al.	√	√	√	

2.4 Optimization Goals

Different optimization methods have been studied in recent years. From the driver side, the optimization goal may be minimizing the driving distance. For the riders, the goal could be maximizing the chance of finding a vehicle with given service constraints. Table **??** summarizes studies with different constraints and optimization goals.

3 Optimization Method

The main challenge of dynamic ridesharing is to deal large number of trip requests and cars in real-time. The most effective way to deal with this problem is to use the framework of **Filter and Refine**. In this section, we introduce the Filter and Refine framework and discuss different studies under this framework.

3.1 The Challenges

In a dynamic ridesharing system, assigning a rider request to a car is not a simple pair matching problem. When the ridesharing system gets a new request, the cars moving on the road network may already been assigned some trips. To find the car to assign the tr to, we need to consider not only $tr.o$ and $tr.d$, but also the trips already in TR_{rec}. The combined route should be replanned on the road network to satisfy all the constraints of the new trip and the trips in TR_{rec}. As can been seen in Figure 2, when a request is submitted to the ridesharing system, a car is cruising to position P and $TR_{rec} = \langle tr_1, tr_2, tr_3 \rangle$. For tr_2 the trip has finished. The rider of tr_1 is already in the car, but has not arrived $tr_1.d$, the rider of tr_3 has not been picked up, and the schedule at this time is $schedule = \langle tr_1.d, tr_3.o, tr_3.d \rangle$. To determine whether the constraints of tr_4 can be satisfied by the car, we need to add tr_4 and reshuffle the new $schedule = \langle tr_1.d, tr_3.o, tr_3.d, tr_4.o, tr_4.d \rangle$ to find the valid schedules that meet all the time window constraints and detour constraints of tr_1, tr_3, and tr_4. This problem is a form of Traveling Salesman Problem and has been proved to be a NP-hard problem[19, 17, 12]. The large

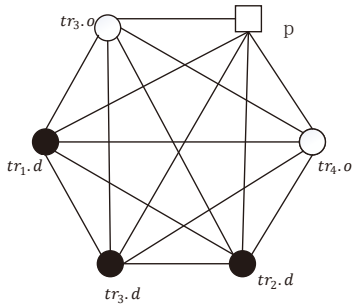

Figure 2: Possible-schedules with a new request

number of trip requests and the enormous number of running cars in mega cities challenges the scalability of matching algorithms. For example, in Beijing, there are more than 5,591,000 vehicles finishing around 9,090,000 passenger trips per day in 2013. In addition, a request not matched to a car may be resubmitted one or more times and generate more requests.

The route searching algorithm such as branch-and-bound [13] and integer programming [6] are designed for offline computation. Their calculation time was measured in minutes or hours while the real-time dynamic ridesharing requires millisecond response time.

3.2 The Filter and Refine Framework

Obviously, it is very expensive to go through all of the cars to match a trip request. The solution of the real-time problem is the framework of **Filter and Refine**. The core idea of this framework is to filter and conquer: filter the vehicles and trip requests and then conquer the problem in a small scale. **Step 1 Filter**: given $G = \langle V, E, W \rangle$, trip request set TR, and the cars on the road network set C, remove the elements from the set of C and the set of TR which do not have matching possibility. An alternative is to cluster the requests into some groups and process them as a batch. After the filtering, the $C_{filtered}, (|C_{filtered}| \leq |C|)$ and $TR_{filtered}, (|TR_{filtered}| \leq |TR|)$ are prepared for the next step. **Step 2 Refine**: specific algorithms are applied to get the matching pairs under the constraints such as time and detour.

3.3 The Filter Method

The target of filter step is to reduce the scale of the problem. When a request is received, the ridesharing system matches the request with the cars cruising on the network. The challenge of this step is to filter the moving objects on the road network using a quick and efficient mechanism. Though spatial index method as $R - tree$, $R^* - Tree$,$TPR - Tree$ is proposed, they are not designed for the ridesharing problem. So the direct usage of these structures may bring lots of update cost on the nodes. Some research has been done to speed up the filtering process. For example in [7], the system calculates the feasible neighbourhoods of the current route between the stops so that when a new request comes, the candidate stops can be retrieved quickly from the stops in the neighborhoods. Depending on method, for a route with n possible stops, the pre-calculation complexity is $O(n^4) or O(n^3)$ and the complexity of the intersection shrinks to $O(n)$. The limitation of this method is that the pre-calculation is not suitable for large scale ridesharing problem in real-time.

A searching algorithm using a spatio-temporal index to quickly filter the candidate cars that may satisfy a trip request is proposed. This method first partitions the road network using a gird, then uses the anchor nodes to denote girds as shown in Figure 3(A). Each cell maintains a temporally-ordered grid cell list sorted in ascending order of the grid's travel time, a spatially-ordered grid cell list stored in travel distance, and a car list recording the cars scheduled to enter the gird as shown in Figure 3(C). When a new request is submitted, the system uses grids from near to far to filter the candidate cars as shown in Figure 3(D).

Combining similar trip requests to some clusters is also an efficient method to reduce the problem size. A Data Stream Management System with different space-partitioning policies is used for trip grouping using a parallel implementation [11, 3].

3.4 The Refine Method

The scale of solution space can be effectively reduced by the filter step. In the Refine step, a trip $tr \in TR_{filter}$ and a $car \in C_{filter}$ will be matched. As be mentioned above, this step needs to reschedule the route to include the trip in set of trips TR_rec already accepted by the car, while satisfying all the constraints. This is a NP-hard problem. As mentioned above, algorithms as branch-and-bound [13] and integer programming [6] are

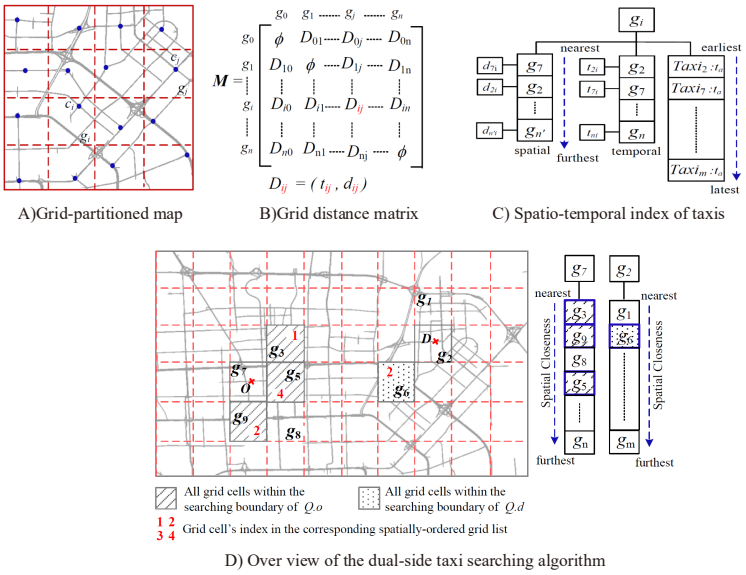

Figure 3: Spatio-temporal index [17]

not suitable for real-time dynamic ridesharing problem. So new methods need to be developed to solve the ridesharing routing problem in this step.

Some simple methods are used to reduce the computational complexity of the system. For example, one can convert the rescheduling problem to a insertion problem. For a trip with k different points, the complexity is reduced from $O(k!)$ to $O(k^2)$. However, insertion method does not try to achieve optimality.

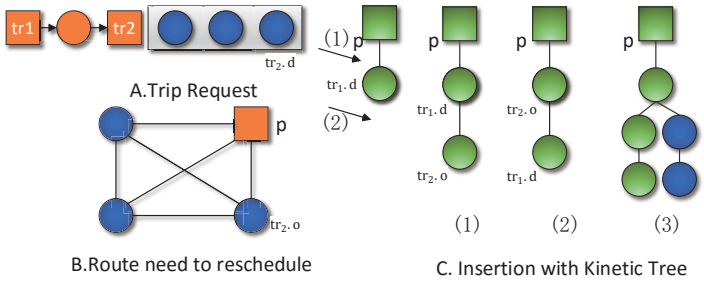

Figure 4: Kinetic Tree for Trip Schedules [12]

Traditional routing algorithms such as branch-and-bound reschedule unfinished origins and destinations with the new request from scratch. Thus the computations of previous scheduling is not used. The repeated computation make dramatically increase the rescheduling time. To solve this problem, a Kinetic Tree structure that maintains the calculations performed so far and use them effectively for a new requests is proposed[12]. As can be seen from Figure 4(C), the kinetic tree structure stores all the valid trip schedules as a tree. When the car is running on the road network, the visited points and unvisited branches are dropped from the tree. The root of the tree tracks the current location loc of the car. The rest of the tree represent all valid schedules in a compact tree structure. When a new request comes in, the system does not need to do the reschedule from scratch, instead it tries to insert the new trip into the Kinetic Tree structure. In order to do so, it extends all valid and active schedules in the prefix tree into a new valid schedule to include tr_i. It deals with the origin point $tr_i.o$ first and then the destination $tr_i.d$. The prefix needs to be scanned to determine which edge can have $tr_i.o$ inserted while staying maintaining the service constraints, and then it tries to insert $tr_i.d$ after the position of

$tr_i.o$ in a similar fashion.

As shown in Figure 4(A), the car accepts a request tr_1 and picks up the rider at position $tr_1.o$. The new request of tr_2 is submitted to the ridesharing system, when the car is at position P. At this time the car should reschedule $\langle tr_1.d, tr_2.o, tr_2.d \rangle$. If we use the branch-and-bound method, the graph needs to be searched to reschedule the whole route, which can not utilize the computation performed before and the complexity of reschedule is $O(n!)$. In contrast, the Kinetic Tree could efficiently use the tree structures to store the computation result. To justify the new trip tr_2, we first determine if an insertion is valid at position (1),(2) as shown in Figure 4(C) to satisfy the constraints of available seats, time window, and detour. And then $tr_2.d$ is inserted at a position after $tr_2.o$. After the insertion, the new route is shown as in Figure 4(C)-(3). The main problem with the basic tree algorithm is the exponential explosion of the size of the tree when there are multiple clustered origin points and destination points. For example, if 8 origin points occur in spatial-temporal proximity, any permutation of pickups may result in a valid schedule, which yields $8! = 40,320$ possibilities. The hotspot clustering algorithm proposed to deal with this problem. Once a point is combined with any hotspot, the insertion to the other edges is stopped. The experiments on a large Shanghai taxi dataset has be performed, showing that the kinetic tree algorithms outperform other algorithms significantly.

4 Conclusion and Future Directions

Dynamic ridesharing brings both opportunities and challenges. In this article, we summarize some real-time ridesharing algorithms under the framework of Filter and Refine. And introduce the different methods could be used in different step of the framework in dynamic ridesharing. We believe pervasiveness of location enabled mobile devices will make large scale ridesharing a reality in the near future. However, there are many research questions left. On the scheduling side, the impact of the traffic and the associated uncertainty problem has not been studied well in dynamic ridesharing and is a challenge research problem. Another important research question still in its early stage is trust and privacy. Tapping into social network analysis may help with these problems. With the advent of driverless vehicles, ridesharing will involve more research issues such as pre-route empty cars to maximize rideshare through learning from historical trips.

Acknowledgments

References

[1] N. A. Agatz, A. L. Erera, M. W. Savelsbergh, and X. Wang. Dynamic ride-sharing: A simulation study in metro atlanta. *Transportation Research Part B: Methodological*, 45(9):1450–1464, 2011.

[2] V. Armant and K. N. Brown. Minimizing the driving distance in ride sharing systems. In *Tools with Artificial Intelligence (ICTAI), 2014 IEEE 26th International Conference on*, pages 568–575. IEEE, 2014.

[3] A. Attanasio, J.-F. Cordeau, G. Ghiani, and G. Laporte. Parallel tabu search heuristics for the dynamic multi-vehicle dial-a-ride problem. *Parallel Computing*, 30(3):377–387, 2004.

[4] N. D. Chan and S. A. Shaheen. Ridesharing in north america: Past, present, and future. *Transport Reviews*, 32(1):93–112, 2012.

[5] B. Cici, A. Markopoulou, E. Frías-Martínez, and N. Laoutaris. Quantifying the potential of ride-sharing using call description records. In *Proceedings of the 14th Workshop on Mobile Computing Systems and Applications*, page 17. ACM, 2013.

[6] J.-F. Cordeau. A branch-and-cut algorithm for the dial-a-ride problem. *Operations Research*, 54(3):573–586, 2006.

[7] L. Coslovich, R. Pesenti, and W. Ukovich. A two-phase insertion technique of unexpected customers for a dynamic dial-a-ride problem. *European Journal of Operational Research*, 175(3):1605–1615, 2006.

[8] P. M. d'Orey, R. Fernandes, and M. Ferreira. Empirical evaluation of a dynamic and distributed taxi-sharing system. In *Intelligent Transportation Systems (ITSC), 2012 15th International IEEE Conference on*, pages 140–146. IEEE, 2012.

[9] M. Furuhata, M. Dessouky, F. Ordez, M.-E. Brunet, X. Wang, and S. Koenig. Ridesharing: The state-of-the-art and future directions. *Transportation Research Part B: Methodological*, 57:28 – 46, 2013.

[10] K. Ghoseiri, A. E. Haghani, M. Hamedi, and M.-A. U. T. Center. *Real-time rideshare matching problem*. Mid-Atlantic Universities Transportation Center, 2011.

[11] G. Gidofalvi, T. B. Pedersen, T. Risch, and E. Zeitler. Highly scalable trip grouping for large-scale collective transportation systems. In *Proceedings of the 11th international conference on Extending database technology: Advances in database technology*, pages 678–689. ACM, 2008.

[12] Y. Huang, F. Bastani, R. Jin, and X. S. Wang. Large scale real-time ridesharing with service guarantee on road networks. *Proceedings of the VLDB Endowment*, 7(14):2017–2028, 2014.

[13] B. Kalantari, A. V. Hill, and S. R. Arora. An algorithm for the traveling salesman problem with pickup and delivery customers. *European Journal of Operational Research*, 22(3):377–386, 1985.

[14] E. Kamar and E. Horvitz. Collaboration and shared plans in the open world: Studies of ridesharing. In *IJCAI*, volume 9, page 187, 2009.

[15] A. Kleiner, B. Nebel, and V. Ziparo. A mechanism for dynamic ride sharing based on parallel auctions. 2011.

[16] B. Li, D. Krushinsky, H. A. Reijers, and T. Van Woensel. The share-a-ride problem: People and parcels sharing taxis. *European Journal of Operational Research*, 238(1):31–40, 2014.

[17] S. Ma, Y. Zheng, and O. Wolfson. T-share: A large-scale dynamic taxi ridesharing service. In *Data Engineering (ICDE), 2013 IEEE 29th International Conference on*, pages 410–421. IEEE, 2013.

[18] K. Radke, M. Brereton, S. Mirisaee, S. Ghelawat, C. Boyd, and J. G. Nieto. Tensions in developing a secure collective information practice-the case of agile ridesharing. In *Human-Computer Interaction–INTERACT 2011*, pages 524–532. Springer, 2011.

[19] M. W. Savelsbergh. Local search in routing problems with time windows. *Annals of Operations research*, 4(1):285–305, 1985.

[20] C. Tian, Y. Huang, Z. Liu, F. Bastani, and R. Jin. Noah: a dynamic ridesharing system. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, pages 985–988. ACM, 2013.

[21] J. Witkowski, S. Seuken, and D. C. Parkes. Incentive-compatible escrow mechanisms. In *AAAI*, 2011.

[22] X. Xing, T. Warden, T. Nicolai, and O. Herzog. Smize: a spontaneous ride-sharing system for individual urban transit. In *Multiagent System Technologies*, pages 165–176. Springer, 2009.

[23] D. Zhao, D. Zhang, E. H. Gerding, Y. Sakurai, and M. Yokoo. Incentives in ridesharing with deficit control. In *Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems*, pages 1021–1028. International Foundation for Autonomous Agents and Multiagent Systems, 2014.