A Reinforcement Learning Based Algorithm for Multi-hop Ride-sharing: Model-free Approach

Ashutosh Singh, Abubakr Alabbasi, and Vaneet Aggarwal*

Abstract

The growth of autonomous vehicles and self driving technology will bring a shift in the way ride hailing platforms plan out their services. In this paper, we propose a novel multi-hop ride-sharing (MHRS) algorithm that uses deep reinforcement learning to learn optimal vehicle dispatch and matching decisions by interacting with the external environment. By allowing customers to transfer between vehicles, i.e., ride with one vehicle for sometime and then transfer to another one, MHRS helps in attaining 30% lower cost and 20% more efficient utilization of fleets, as compared to the ride-sharing algorithms. This flexibility of multi-hop feature gives a seamless experience to customers and ride-sharing companies, and thus improves ride-sharing services.

1 Introduction

As the adoption of autonomous vehicles becomes more widespread, the need for ride-sharing and carpooling transportation will become even more prominent. Autonomous driving could lead to fewer accidents, fewer traffic deaths, greater energy efficiency, and lower insurance premiums [12, 2, 5]. It is estimated that savings with autonomous vehicles for the United States alone will be around $1.3 trillion [10]. Carmakers race to develop self-driving cars to use in fleets of robo-taxis to tap into the potentially lucrative new market. While ride-sharing (joint travel of two or more passengers in a single vehicle) has long been a common way to share the costs and reduce the congestion, it remains a nook transportation option with limited route choice and sparse schedules (only few rides per route). This work explores the benefits of allowing customers to transfer between vehicles to further improve the ride-sharing services.

Multi-hop ride-sharing (MHRS) plays a crucial role in revolutionizing the transport experience in ride-sharing and logistics transportation [7]. With the advancement of self-driving vehicle technology and proliferation of ride-sharing apps (e.g., Uber and Lyft), optimal distributed dispatching of vehicles is essential to minimize the supply-demand mismatch, reduce the waiting time of customers, reduce the travel cost and utilize the fleet effectively. Also, multi-hop ride sharing is a more competitive and reliable mode of transportation in terms of time and cost when compared with other modes of transport such as trains and buses [19]. However, hopping and dispatch decisions over a large city for carpooling requires instantaneous decisions for thousands of drivers to serve the ride requests over an uncertain future-demand. Moreover, the dispatch decision for each vehicle could be time consuming, exacerbating the uncertainty of the optimization over a dynamically changing demand. We note that vehicle can only be dispatched if it is not fully occupied, and the travel time of a passenger, if any, within the vehicle should be minimized. Thus, the dispatch decision depends on the future potential trip requests which are uncertain in nature. Yet, realistic models are necessary to assess the trade-offs between possibly conflicting interests, minimizing the un-served ride requests, waiting time per request, total trip times of passengers, and the total trip distance of every vehicle.

*The authors are with Purdue University, West Lafayette IN 47907, email: {singh596,aalabbas,vaneet}@purdue.edu.

Machine Learning for Autonomous Driving Workshop at the 33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

1.1 Related Work

Multi-hop ride-sharing poses myriad challenges that need to be addressed for efficient ride platforms services. While fleet management and ride-sharing problems have been widely studied, most of the approaches in the literature are model-based, see [24, 23, 13, 8, 3] and the references therein. In such models, pickup request locations, travel time, and destinations are assumed to follow certain distributions and then propose dispatching policies that would improve the performance. Using realistic data from ride-sharing services, the authors in [9] have modeled the customer requests and their variations over service area using a graph. The temporal/spatial variability of ride requests and the potentials for ride-pooling are captured. In [6], a matching algorithm based on the utility of the desired user is proposed. However, the proposed approach is a model-based that lacks adaptability and thus becomes impractical when the number of vehicles is very large, which is the scenario in our setting.

Recently, different approaches for dispatching vehicles and allocating transportation resources in modern cities are investigated. In [23], an optimal policy for autonomous vehicles is proposed, which considers both global service fairness and future costs. Yet, idle driving distance and real-time GPS information are not considered. In [21, 4], scheduling temporal-based methods are developed to reduce costs of idle cruising, however, the proposed algorithm does not utilize the real-time location information. Several studies in the literature, e.g., [14, 22, 16, 17], have shown that learning from past taxi data is possible. Thus, taxi fleet management and minimizing both waiting-time for passengers and idle-time for drivers can be obtained. In [11], an open source based on reinforcement learning for traffic control is developed, where traffic flow in a wide variety of traffic scenarios is improved. However, the settings, objective and formulations in our MHRS is different. The authors in [15, 17] have used deep learning to solve traffic problems, such as travel mode choice predication. In [16], authors used DQN for dynamic fleet management. Using realistic data, distributed DQN based approaches have shown to outperform the centralized solutions. This work is extended to accommodate the ride-sharing scenarios in [1]. In this paper, we generalize these models and consider a multi-hop ride-sharing system where a rider can reach a destination via a number of transfers (hops) and at the sametime one or more passengers can share a single vehicle.

1.2 Contributions

In this paper, we propose a distributed model-free multi-hop ride-sharing algorithm for (i) dispatching vehicles to locations where future demand is anticipated, and (ii) matching vehicles to customers. We provide an optimized framework for vehicle dispatching that adopts deep reinforcement learning to learn the optimal policies for each vehicle individually by interacting with the external environment. Our approach allows customers to have one (or more) hops along the way to the destination.

Different from the majority of existing model-based approaches, we do not need to accurately model the system (so we call it model-free), rather, we use reinforcement learning technique in which each vehicle solves its own deep Q-network (DQN) problem in a distributed manner without coordination with cars in its vicinity. By doing so, a significant reduction in the complexity is obtained. To the best of our knowledge, this work is the first to cast the MHRS problem to a reinforcement learning problem. We aim to optimally dispatch the vehicles to different locations in a service area as to minimize a set of performance metrics including customers waiting time, extra travel time due to participating in MHRS and idle driving costs. Using real-world dataset of taxi trip records in New York City, our extensive simulation results show that, with the same overhead time as DeepPool [1], i.e., model-free algorithm for ride-sharing, passengers can complete their trips with ~30% lower costs. Further, MHRS can achieve upto ~99% accept rates, and thus minimizes the supply-demand mismatch. In addition, MHRS also helps reduce both waiting time and (idle) cruising time by~40%, with ~20% better utilization rate of vehicles.

2 Problem Statement

We consider a MHRS scenario where customers can complete their trips in multiple hops. We will develop an algorithm that dispatches vehicles to pick-up passengers from their pickup and/or hop locations. The vehicle location, capacity, next destination, number of passengers, etc. can be tracked in real time, thanks to mobile internet technologies. Without loss of generality, we consider the New York City area as our area of operation. The area is divided into multiple non-overlapping zones (or

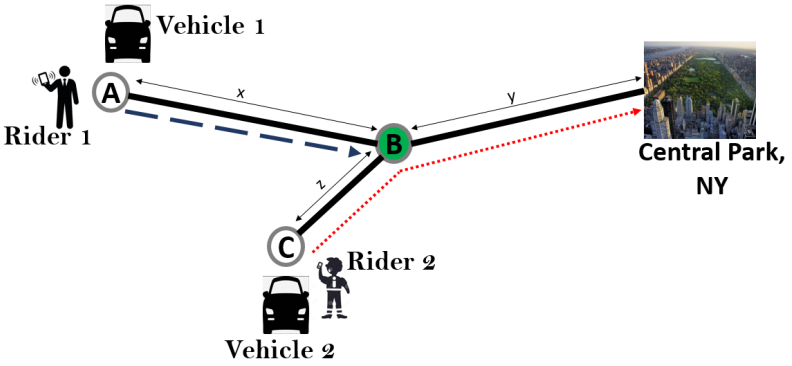

Figure 1: A schematic illustrating the multi-hop ride-sharing in a simple ride matching scenario.

regions), each of which is 1 square mile. This allows to discretize the area of operation and thus makes the action space (i.e, where to dispatch the vehicles) tractable.

Multi-hop Example: Consider a simple multi-hop ride-sharing scenario as depicted in Figure 1. In this figure, both Rider 1 and Rider 2 want to go to the Central Park, NY. However, Rider 1 and Rider 2 are in different zones. Rider 1 is in zone A, while Rider 2 is in zone C. Since zone B lies along the route of both Riders, Zone B is a hop zone, where one vehicle (say Vehicle 1) can drop its passenger and join the another vehicle (Vehicle 2). Here, Rider 1 initially gets into the nearest vehicle (Vehicle 1) and Rider 2 gets into Vehicle 2. Vehicle 1 drops rider 1 at zone B. As Vehicle 2 reaches the hop zone (zone B), it picks up Rider 1 and drops both at their destination, i.e., Central Park. As shown in the figure, x represents the distance between zone A and zone B, y represents the distance between zone B and Central Park, and z is the distance between zone B and zone C. We define the effective distance of Vehicle 2, D_2, which gives the effective distance traveled by Vehicle 2, as follows:

$$D_2 = \frac{z+y+y}{z+y} = 1 + \frac{y}{z+y} \tag{1}$$

Thus, effective distance is formally defined as the ratio of total distance covered if no hoping and sharing was allowed to the total distance covered when hoping and sharing is allowed.

We use $\boldsymbol{X}_t = \{\boldsymbol{x}_{t,1}, \boldsymbol{x}_{t,2}, \ldots, \boldsymbol{x}_{t,N}\}$ to denote the vehicles status at time t. $\boldsymbol{x}_{t,k}$ is a vector which consists of the current zone of the vehicle k, available vacant seats, the time at which a passenger is picked up, and destination zone of each passenger. Using this information, we can also predict the time slot at which the unavailable vehicle v will be available, $d_{t,\tilde{t},i}$. Thus, for a set of dispatch actions at time t, we can predict the number of vehicles in each zone for T time slots ahead, denoted by $\boldsymbol{V}_{t:t+T}$. We use $hop_{l,i,p}$ to denote the passenger l hopped down at zone i, and p denotes the p^{th} hop of the passenger. We can improve upon our dispatching policies by estimating demand in every zone through historical weekly/daily distribution of trips across zones[20]. We use $\boldsymbol{D}_{t:T} = \left(\overline{\boldsymbol{d}}_t, \ldots, \overline{\boldsymbol{d}}_{t+T}\right)$ to denote the predicted future demand from time t_0 to time $t+T$ at each zone. Combining this data, the environment state at time t can be written as $\boldsymbol{s}_t = (\boldsymbol{X}_t, \boldsymbol{V}_{t:t+T}, \boldsymbol{D}_{t:t+T})$. Note that when a passenger's request is accepted, we will append the user's expected pick-up time, the source, and destination to $\boldsymbol{s}_t$.

Objectives: The objective of this paper is to optimally dispatch the vehicles to various locations and achieve these motives: a) ensure minimum waiting-time and dispatch time for passengers and vehicles, respectively, b) minimize the gap between the supply and demand, c) minimize the number of vehicles used, d) minimize the number of hops for a customer, e) minimize the extra travel time for a customer for a given trip due to hops, and f) minimize the fuel consumption for each vehicle.

3 Multi Hop Framework Design

In this section, we present our framework to solve the MHRS problem. We propose a distributed model-free approach using DQN, where the agent learns the best actions based on the reward it receives from the environment.

3.1 Reward

The first component of the reward is to minimize the gap between the supply and demand for the agents/vehicles. Let $\boldsymbol{v}_{t,i}$ denote the number of available vehicles at time t at zone i. Supply demand difference within time t at zone i can be written as $(\boldsymbol{v}_{t,i} - \overline{\boldsymbol{d}}_{t,i})$, i.e.,

$$\text{diff}_t^{(D)} = \sum_{i=1}^{M} \left(\overline{\boldsymbol{d}}_{t,i} - \boldsymbol{v}_{t,i}\right)^+ \tag{2}$$

The second component of the reward is to minimize the dispatch time for the vehicles. Vehicles can be dispatched in two cases, either to serve a new request or to move to such locations where a high demand is anticipated in the future. We note that only available vehicles can be dispatched. Dispatch time refers to the estimated travel time to go to zone j, i.e., $h_{t,j}^n$ if the vehicle n is dispatched to zone j. Thus, for all available vehicles within time t, we wish to minimize the total dispatch time, $T_t^{(D)}$,

$$T_t^{(D)} = \sum_{n=1}^{N} \sum_{j=1}^{M} h_{t,j}^n u_{t,j}^n \tag{3}$$

where $u_{t,j}^n = 1$ only if vehicle n is dispatched to zone j at time t. Next, we want to minimize the extra travel time for every passenger (the third component). For vehicle n, rider ℓ, at time t, we need to minimize $\delta_{t,n,\ell} = t' + t_{t,n,\ell}^{(a)} - t_{n,\ell}^{(m)}$, where $t_{t,n,\ell}^{(a)}$ is the updated time the vehicle will take to drop off passenger ℓ because of change in route and/or addition of a rider from the time t. $t_{t,n,\ell}^{(m)}$ is the travel time that would have been taken if the user ℓ would not have shared the vehicle with any other passenger. Also, t' is the time elapsed after the user ℓ has requested its ride. Therefore, we can write

$$\boldsymbol{\Delta}_t = \sum_{n=1}^{N} \sum_{\ell=1}^{U_n} \delta_{t,n,\ell} \tag{4}$$

To ensure that the passengers do not have to go through any inconvenience (e.g., many hops along the way to destination) in order to complete their trips, we minimize the number of hops a passenger has to make to complete the trip. Recall that $hop_{\ell,i,p}$ denote the passenger ℓ hopped down at zone i, and p denotes the p^{th} hop of the passenger ℓ. Then,

$$\boldsymbol{H}_\ell = \sum_{i=1}^{M} \sum_{p=1}^{P} hop_{\ell,i,p} \tag{5}$$

Lastly, the number of vehicles at any time $\boldsymbol{e}_t$ needs to be optimized to better utilize the vehicles usage (i.e., available resources), this can be written as

$$\boldsymbol{e}_t = \sum_{n=1}^{N} [\max\{\boldsymbol{e}_{t,n} - \boldsymbol{e}_{t-1,n}, 0\}] \tag{6}$$

where $e_{t,n}$ shows whether a vehicle has been occupied. We want to minimize the number of vehicles in the fleet that change their status from being inactive to active. This would ensure that the vehicles in the fleet are utilized efficiently.

Now that we have obtained our main objective equations, we can define the reward equation at time t as follows:

$$\overline{r}_t = -[\beta_1 \text{diff}_t^{(D)} + \beta_2 T_t^{(D)} + \beta_3 \boldsymbol{\Delta}_t + \beta_4 \boldsymbol{e}_t + \beta_5 \boldsymbol{H}_\ell] \tag{7}$$

The minus sign indicates we want to minimize those terms, and $\beta_1, \beta_2, \beta_3, \beta_4, \beta_5$ are weights and can be changed based on any specific objective we want to meet. Example, we can increase β_4 if we decide our main goal to minimize the size of the fleet. The reward stated above is defined within the action. However, this reward is not an explicit function of the action. Here, model-free approach means that the exact relationship between the action and reward will the learned by our algorithm.

Algorithm 1 MHRS Simulator

Initialize vehicles states $\boldsymbol{X}_0$.
for $t \in T$ **do**
    Get all ride requests at time t
    for Each ride request in time slot t **do**
        Choose a vehicle n to serve the request
        Calculate the dispatch time using ETA model
        Update the state vector $\boldsymbol{\Omega}_{t,n}$.
    end for
    Send the state vector $\boldsymbol{\Omega}_t$ to the agent.
    Get the best actions (dispatch orders) $\boldsymbol{a}_t$ from
    the agent.
    for all vehicles $n \in \boldsymbol{a}_t$ **do**
        Find the shortest path to every dispatch location
        for every vehicle n.
        Estimate the travel time using the ETA model
        Update $\delta_{t,n}$, if needed, and generate the trajectory
        of vehicle n.
    end for
    Update the state vector $\boldsymbol{\Omega}_{t+1}$.
end for

3.2 Selection of hop zones

Hop zones are the zones where customers hop down from a vehicle and wait for the next vehicle to complete the rest of their trips. We start by dividing the city into 212 x 219 square grids. To run our DQN, we further combine these grids into 41 x 43 to prevent our action space from exploding. Every third intersection of zones in the horizontal and vertical direction is considered a hop zone. Since we want to make sure that the hop zones are in the busy area of the city and are not too closely placed, we consider only the zones with at least 10 ride requests in the day. We obtain 148 hop zones in the whole New York city through this method.

3.3 Deep Q-Network Architecture

For the DQN, we divide the New York City area map to obtain 41 x 43 grids. We do so to limit the action space of vehicles to 7 grids vertically and horizontally. Thus the action space for each vehicle is 15 x 15 grid around its present location. The input to the deep Q network is the state of vehicles, supply and demand. Specifically, it consists of predicted requests in the next 15 minutes obtained from demand model, current location of each vehicle, future location of each vehicle in the next 15, and 30 minutes. The network architecture consists of 16 convolution layers of 5 x 5, 32 convolution layer of 3 x 3, 64 convolution layer of 3 x 3, 16 convolution layer of 1 x 1. All convolution layers have a rectifier non linearity activation. The output of the deep Q network is 15x 15 Q values, where each value corresponds to the reward a vehicle could get if dispatched to that particular zone.

Reinforcement learning becomes unstable for non linear approximations, hence we use experience replay to overcome this issue. We define a new parameter, α to address this issue. We memorize the experiences and randomly sample from these experiences to avoid correlation between immediate actions and experiences. The value of α is an indication of how frequently the target Q values are updated, indicating how much the agents are learning from their past experiences. In our case, we have taken α as 0.9. The details of the multi hop framework design can be found in [18].

4 Evaluation and results

We compare the results of our MHRS algorithm with two scenarios: no ride-sharing (No-RS) and ride-sharing (RS). No-RS means the vehicles are dispatched using DQN policies but only one rider can occupy a vehicle at any given time. This policy is akin to that in [16], which we refer in our simulation comparisons as "MOVI". For the RS case, in addition to DQN dispatch policies, more than one passenger can be assigned to one vehicle but no multi-hop is performed. This policy is

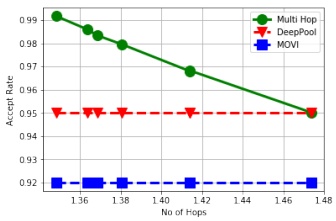

Figure 2: Accept rate versus the average number of hops.

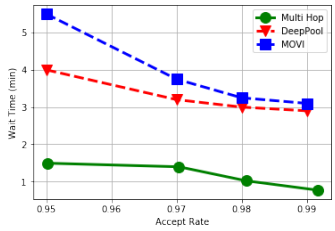

Figure 3: Average wait time per request for different accept rate

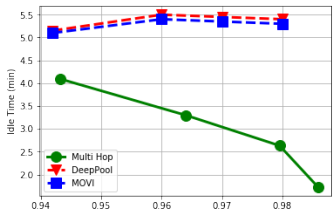

Figure 4: Average idle time per request for different accept rate.

proposed in [1], so-called DeepPool. In our MHRS policy, passengers can complete their trips in more than one hop.

We set up the simulator with 5000 vehicles. The time step was taken as one minute and the agents were trained on data of one month duration and tested on data of two weeks (first two weeks of June 2016). The initial location of the vehicles are corresponding to the locations of the first 5000 trips.

Figure 2 plots the average accept rate versus the number of hops. Note that if a passenger is dropped at a hop zone and not served further till the final drop location, this passenger request is deemed rejected. Since DeepPool and MOVI are not function of the number of hops, both have fixed acceptance rates at 0.95 and 0.92, respectively. We note that by 1.40 hops, only 40% of the total trips have 2 hops. While the accept rate decreases as number of hops increases, our MHRS approach still performs the best as compared to both DeepPool and MOVI. This decrease of accept rate is due to the possibility for our algorithm to fail serving some of the passengers at the hop zones. In addition, some passengers complete their trips in more than one hop, thus making some vehicles available at the hop zones. Hence, vehicles get free more often as compared to when they had to travel long distances in one trip.

Figure 3 shows that the wait time decreases as the accept rate increases. MHRS outperforms DeepPool and MOVI, and thus suggesting more than one hop allows customers to have sufficient vehicles in their vicinity and hence less waiting time on average.

The idle time per vehicle is captured in Figure 4. Clearly, MHRS leads to a significant decrease in the idle cruising time of vehicles. Small idle time would encourage incessant revenue generation for the taxi provider firms as well as quick fulfillment of customer demands

5 Conclusion

We have looked at the problem of multi-hop ride-sharing, where passengers can be dropped one or more times before reaching their destination. We have proposed an efficient model-free algorithm for dispatching and matching policies where reinforcement learning techniques are used to to learn the optimal decisions for each vehicle individually. We have observed better accept rates, more efficient utilization of vehicles and reduced idle time for vehicles This indicates that searching for shared rides with three (or more) hops does not give significantly better results, on average. Surprisingly, a negligible improvement is noticed for more than two hops.

As a future research, utilizing the interaction between vehicles in a multi-agent reinforcement learning settings is a nice future direction. Further, optimal incentive techniques to influence passengers to opt for the MHRS is another promising research area.

References

[1] AL-ABBASI, A. O., GHOSH, A., AND AGGARWAL, V. Deeppool: Distributed model-free algorithm for ride-sharing using deep reinforcement learning. *IEEE Transactions on Intelligent Transportation Systems* (2019), 1–14.

[2] ANDERSON, J. M., NIDHI, K., STANLEY, K. D., SORENSEN, P., SAMARAS, C., AND OLUWATOLA, O. A. *Autonomous vehicle technology: A guide for policymakers*. Rand Corporation, 2014.

[3] BEI, X., AND ZHANG, S. Algorithms for trip-vehicle assignment in ride-sharing. In *Thirty-Second AAAI Conference on Artificial Intelligence* (2018).

[4] BERTSIMAS, D., JAILLET, P., AND MARTIN, S. Online vehicle routing: The edge of optimization in large-scale applications. *Operations Research* (2019).

[5] BLYTH, P.-L., MLADENOVIC, M. N., NARDI, B. A., EKBIA, H. R., AND SU, N. M. Expanding the design horizon for self-driving vehicles: Distributing benefits and burdens. *IEEE Technology and Society Magazine 35*, 3 (2016), 44–49.

[6] DE LIRA, V. M., TIMES, V. C., RENSO, C., AND RINZIVILLO, S. Comewithme: An activity-oriented carpooling approach. In *2015 IEEE 18th International Conference on Intelligent Transportation Systems* (2015), IEEE, pp. 2574–2579.

[7] DREWS, F., AND LUXEN, D. Multi-hop ride sharing. In *Sixth annual symposium on combinatorial search* (2013).

[8] GOPALAKRISHNAN, R., MUKHERJEE, K., AND TULABANDHULA, T. The costs and benefits of sharing: Sequential individual rationality and sequential fairness. *arXiv preprint arXiv:1607.07306* (2016).

[9] JAUHRI, A., FOO, B., BERCLAZ, J., HU, C. C., GRZESZCZUK, R., PARAMESWARAN, V., AND SHEN, J. P. Space-time graph modeling of ride requests based on real-world data. In *Workshops at the Thirty-First AAAI Conference on Artificial Intelligence* (2017).

[10] KEARNEY, A. How automakers can survive the self-driving era. *Retrieved November 14* (2016), 2017.

[11] KHETERPAL, N., VINITSKY, E., WU, C., KREIDIEH, A., JANG, K., PARVATE, K., AND BAYEN, A. Flow: Open source reinforcement learning for traffic control.

[12] LITMAN, T. *Autonomous vehicle implementation predictions*. Victoria Transport Policy Institute Victoria, Canada, 2017.

[13] MA, S., ZHENG, Y., AND WOLFSON, O. Real-time city-scale taxi ridesharing. *IEEE Transactions on Knowledge and Data Engineering 27*, 7 (2015), 1782–1795.

[14] MIAO, F., HAN, S., LIN, S., STANKOVIC, J. A., ZHANG, D., MUNIR, S., HUANG, H., HE, T., AND PAPPAS, G. J. Taxi dispatch with real-time sensing data in metropolitan areas: A receding horizon control approach. *IEEE Transactions on Automation Science and Engineering 13*, 2 (2016), 463–478.

[15] NAM, D., KIM, H., CHO, J., AND JAYAKRISHNAN, R. A model based on deep learning for predicting travel mode choice. In *Proceedings of the Transportation Research Board 96th Annual Meeting Transportation Research Board, Washington, DC, USA* (2017), pp. 8–12.

[16] ODA, T., AND JOE-WONG, C. Movi: A model-free approach to dynamic fleet management. *arXiv preprint arXiv:1804.04758* (2018).

[17] ODA, T., AND TACHIBANA, Y. Distributed fleet control with maximum entropy deep reinforcement learning.

[18] SINGH, A., ALABBASI, A., AND AGGARWAL, V. A distributed model-free algorithm for multi-hop ride-sharing using deep reinforcement learning. *arXiv preprint arXiv:1910.14002* (2019).

[19] TEUBNER, T., AND FLATH, C. M. The economics of multi-hop ride sharing. *Business & Information Systems Engineering 57*, 5 (Oct 2015), 311–324.

[20] WYLD, D. C., JONES, M. A., AND TOTTEN, J. W. Where is my suitcase? rfid and airline customer service. *Marketing Intelligence & Planning 23*, 4 (2005), 382–394.

[21] YANG, J., JAILLET, P., AND MAHMASSANI, H. Real-time multivehicle truckload pickup and delivery problems. *Transportation Science 38*, 2 (2004), 135–148.

[22] ZHANG, D., HE, T., LIN, S., MUNIR, S., AND STANKOVIC, J. A. Taxi-passenger-demand modeling based on big data from a roving sensor network. *IEEE Transactions on Big Data 3*, 3 (2017), 362–374.

[23] ZHANG, R., AND PAVONE, M. Control of robotic mobility-on-demand systems: a queueing-theoretical perspective. *The International Journal of Robotics Research 35*, 1-3 (2016), 186–203.

[24] ZHENG, H., AND WU, J. Online to offline business: urban taxi dispatching with passenger-driver matching stability. In *Distributed Computing Systems (ICDCS), 2017 IEEE 37th International Conference on* (2017), IEEE, pp. 816–825.