A DISTRIBUTED MODEL-FREE ALGORITHM FOR MULTI-HOP RIDE-SHARING USING DEEP REINFORCEMENT LEARNING

by

Ashutosh Singh

A Thesis

Submitted to the Faculty of Purdue University

In Partial Fulfillment of the Requirements for the degree of

Master of Science in Industrial Engineering

School of Industrial Engineering

West Lafayette, Indiana

December 2020

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF COMMITTEE APPROVAL

Dr. Vaneet Aggarwal, Chair

School of Industrial Engineering

Dr. Hua Cai

School of Industrial Engineering

Dr. Satish Ukkusuri

Lyles School of Civil Engineering

Approved by:

Dr. Abhijit Deshmukh

ACKNOWLEDGMENTS

First and foremost, I would like to thank my adviser, Dr. Vaneet Aggarwal. I started working with him on an independent study and was lucky to continue working with him during my masters thesis. He gave me ample freedom to work on the problem of my choice and was always available and accessible to provide his guidance. His support helped me dive deeper in the field of machine learning and software development and make my understanding more concrete.

I would like to thank my lab mate Dr. Abubakr Alabbasi, who shared his expertise on ride sharing and automated vehicles, and helped me understand the problem better. His guidance on my research and academic writing was very instrumental in helping me complete my research. I would also like to thank my committee members Dr. Hua Cai and Dr. Satish Ukkusuri for providing their valuable comments and thoughtful suggestions to my thesis.

Last but not the least, I would like to thank my parents Kapil Deo Singh and Chandrawati Singh for their constant guidance and support throughout my graduate studies and research, my sisters Dr. Priyanka Singh, Dr. Priyabala Singh, Priyadarshani Singh and Kshama Singh for always boosting my morale and for their encouragement.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS

N	number of vehicles
$v_{t,\mathrm{i}}$	number of available vehicles at region i at time slot t
$d_{t,\mathrm{i}}$	number of requests at zone i at time t
$d_{t,\widetilde{t},\mathrm{i}}$	the number of vehicles that are not available at time t but will drop-off a passenger(s) at zone i and then become available at $\widetilde{t}$
$x_{t,k}$	current zone of vehicle k, available vacant seats, the time at which a passenger is picked up, and destination zone of each passenger.
$\Omega_{t,n}$	state of vehicle n at time t
$a_{t,n}$	action of vehicle n at time t
$r_{t,n}$	reward of vehicle n at time t
β_{i}	weight of component i in reward expression
$b_{t,n}$	number of customers served by vehicle n at time t
$c_{t,n}$	time taken by vehicle n to hop or take a detour to pick up extra customers if the vehicle has available seats
$\delta_{t,n,\ell}$	additional time vehicle n takes because of sharing/hoping

U_n	total number of chosen customers for pooling at vehicle n
$\mathcal{L}_{t,n}$	set of all customers assigned to vehicle n at instant t
$hop_{\ell,\mathrm{i},p}$	p^{th} hop of passenger ℓ at zone i

ABSTRACT

The growth of autonomous vehicles, ridesharing systems, and self driving technology will bring a shift in the way ride hailing platforms plan out their services. However, these advances in technology coupled with road congestion, environmental concerns, fuel usage, vehicles emissions, and the high cost of the vehicle usage have brought more attention to better utilize the use of vehicles and their capacities. In this paper, we propose a novel distributed 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

1.1 Motivation

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 [1]–[3]. It is estimated that savings with autonomous vehicles for the United States alone will be around $1.3 trillion [4]. 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 [5]–[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. The idea of multi hop ride sharing is interesting in the sense that it allows more efficient use of the fleet and lower costs for the customer without any substantial change in the infrastructure. Moreover, this model of travel is successfully followed in other domains such as air travel where passengers have a layover of some hours at connecting destinations. Although transfers between vehicles would lead to increase in the travel time of the customers, this can be compensated by offering them lower costs. Ride hailing platforms such as Uber have tried these strategies in crowded cities such as Seattle [8], where there is a high congestion of vehicles during peak hours. MHRS would be able to solve this problem through

more efficient use of fleet, lower congestion of vehicles on roads and offering reduced cost to customers for their travel. 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 [6], [9]. 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.

One of the necessary components for a successful dynamic MHRS system is an efficient optimization techniques that matches drivers and riders in real-time. However, the majority of work in this domain (e.g., [10]–[12]) requires pre-specified model for demand prediction, user's utilities for waiting times and travel times, and cost functions governing the cars' total trip distance. These impact the decisions of route planning for the vehicles in real-time. Thus, a model based approach is unable to consider all the intricacies involved in the optimization problem. Further, a distributed algorithm is preferable where each vehicle can take its own decision without coordinating with the others. This paper aims to consider a model-free distributed approach for ride-sharing with carpooling with possible passenger transfer from one vehicle to another one. Our approach can adapt to the changing distributions of customer arrivals, the changing distributions of the active vehicles, the vehicles' locations, and user's valuations for the total travel time. The tools from deep reinforcement learning [13] are used for this dynamic modeling, where the vehicle dispatching is dynami-

cally learned using the deep neural network, based on which the action is chosen with the Q-learning strategy.

We use deep reinforcement learning technique in which each vehicle solves its own double deep Q-network (DDQN) problem in a 'distributed' manner. A distributed approach means that each vehicle solved its own DDQN without coordination with the car in the vicinity. This leads to a significant reduction in the time complexity because each vehicle is learning its own optimal policy independently. This helps us achieve our goal to learn the optimal dispatch actions for each vehicle individually.

1.2 Related Work

Multi-hop ride-sharing poses myriad challenges that need to be addressed for efficient ride platforms services [5], [6], [14]. While fleet management and ride-sharing problems have been widely studied, most of the approaches in the literature are model-based, see [7], [15]–[20] and the references therein. In such models, pickup request locations, travel time, demand prediction, route decisions, and destinations are assumed to follow certain distributions and then propose dispatching policies that would improve the performance [21], [22]. Using realistic data from ride-sharing services, the authors in [23] 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 [24], 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 [16], 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 [25], [26], scheduling temporal-based methods are developed to reduce costs of idle cruising, however, the pro-

posed algorithm does not utilize the real-time location information. Minimizing the total customer waiting time by concurrently dispatching multiple taxis and allowing taxis to exchange their booking assignments is investigated in [27]. Several studies in the literature, e.g., [28]–[31], 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 [30], 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 [32]. 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 same time one or more passengers can share a single vehicle. With MHRS, the availability and range of the rideshare service increase, and also the total travelled vehicular miles can decrease.

1.3 Our Contribution

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. We note that customers who accept to have hop(s) along their paths to destination may experience a slightly longer time than direct path. However, this incurred time could be compensated by giving incentives to the customers (e.g., lower payment rate) to opt for the carpooling with multi-hop. Our results show a small increase in the total trip times as compared to the rid-sharing scenario.

Figure 2.1 illustrates with a simple example how a multi hop ride sharing scenario can look like.

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 deep reinforcement learning technique in which each vehicle solves its own double deep Q-network (DDQN) 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. Our MHRS algorithm is flexible and can accommodate other ride options as special cases by properly choosing the coefficients of the reward function. The extensive simulation results show new interesting results and reveal useful insights for fleet management. 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 [32], i.e., model-free algorithm for ride-sharing, passengers can complete their trips with $\sim 30\%$ lower costs. Further, MHRS can achieve upto $\sim 99\%$ accept rates, and thus minimizes the supply-demand mismatch. In addition, MHRS algorithm also helps reduce both waiting time and (idle) cruising time by $\sim 40\%$, with $\sim 20\%$ better utilization rate of vehicles.

1.4 Organization

This paper is organized as follows. Section II explains the problem and present an example for MHRS and distinguishes it from ridesharing problem. In addition, the used notations and performance metrics used in this paper are briefly explained. Section III explains the proposed MHRS framework and describes the main components in the system design. In addition, a distributed version of our algorithm is further presented in this section. Section IV presents the simulator design and the evaluation results. It also highlights the main findings out of this work. Section V concludes the paper and gives possible venues for promising directions and future research.

2. PROBLEM STATEMENT

We consider a MHRS scenario where customers can complete their trips in multiple hops. We develop an algorithm that dispatches vehicles to either pick-up passengers from their pickup and/or hop locations or to serve potential customers in the dispatched zones. The vehicle location, capacity, next destination, and 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 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.

2.1 Multi-hop Example

Consider a simple multi-hop ride-sharing scenario as depicted in Figure 2.1. 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 can be 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 relative distance gain of Vehicle 2, D_2, which gives the relative distance gain traveled by Vehicle 2, as follows:

$$D_2 = \frac{z + y + y}{z + y} = 1 + \frac{y}{z + y} \tag{2.1}$$

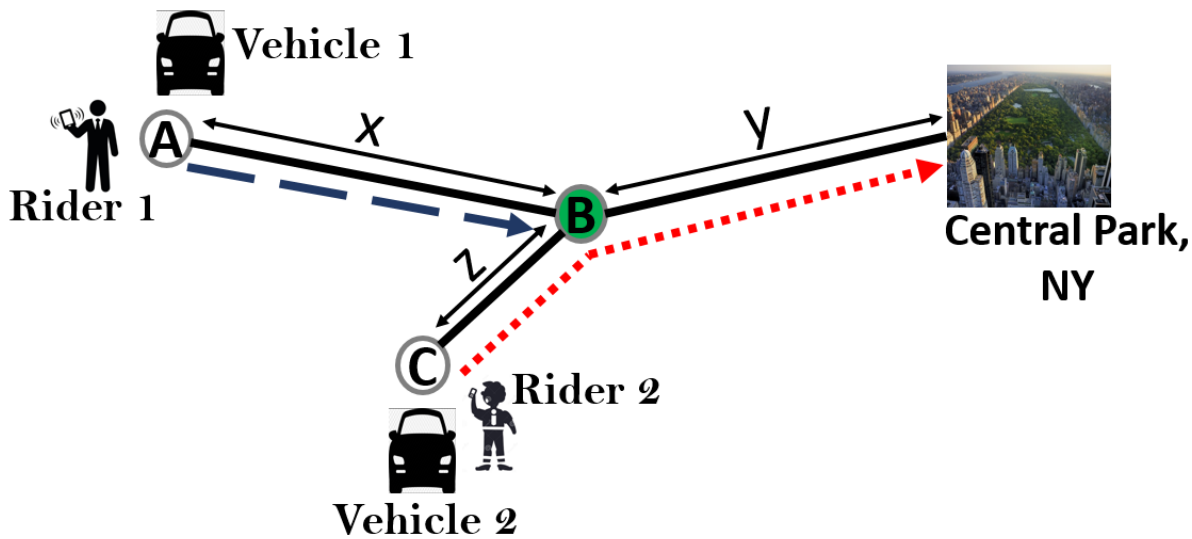

Figure 2.1. : A figure showing the MHRS in a simple ride matching scenario. There are two vehicles (Vehicle 1 and Vehicle 2) and two ride requests (Ride 1 and Ride 2) at zones A and B, respectively. Zone B repesents at "hop" zone where one passenger changes/transfers to another vehicle in zone B.

Thus, relative distance gain 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. Relative distance gain gives the number of times the efficiency of a single vehicle increases when it follows MHRS or DeepPool. The added efficiency comes from the fact that the same vehicle is used to fulfill more than one request. More gain in distance makes overall gains in the amount each vehicle travels and thus leads to better packing of customers. We note that efficient packing of vehicles allows for less distance traveled by the vehicles in completing service of same number of requests. The multi hop scenario shown in Figure 2.1 reduces the traffic that goes from Zone B to the Central Park. Further, MHRS allows vehicles to be free more frequently and thus can improve the accept rate of ride requests as in our example Vehicle 1 is free after dropping Rider 1 and can serve new customers accordingly.

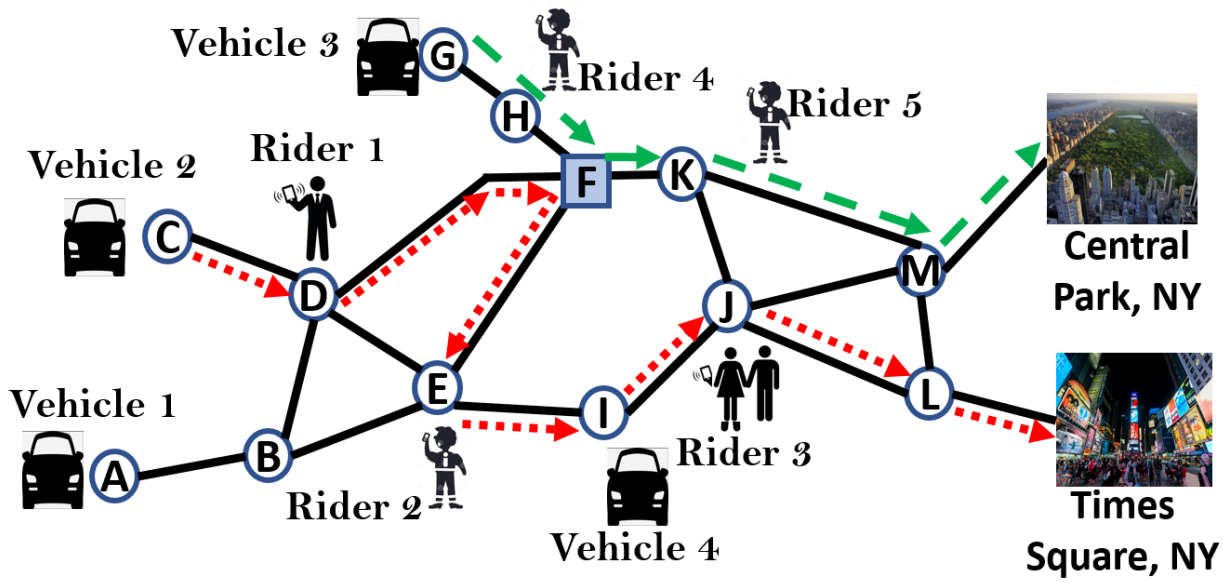

Figure 2.2. : A schematic to illustrate the multi hop ride-sharing routing in a region graph, consisting of 13 regions, A to M. There are five ride requests and four vehicles. The locations of both customers and vehicles are shown in the figure above. The zone 'F' (colored in blue with a square shape) is the hop zone. The red dotted line shows the path of Vehicle 2 and green dotted line shows the path of Vehicle 3 in the multi-hop ride-sharing scenario.

2.2 MHRS, Ride-sharing, and No-ride-sharing Scenarios

Figure 2.2 highlights the differences between different scenarios include: ride-sharing with multi-hop, ride-sharing with no-hops, and the case where only one rider per vehicle is allowed (no ride-sharing case). The figure shows the locations of vehicles and the passengers at a given time. Riders 1, 4 and 5 want to go to the Central Park, NY, while Riders 2 and 3 want to go to the Times Square, NY. There can be a scenario where Vehicle 2 takes Rider 1, and Vehicle 3 takes Rider 4 to the Central park. Further, vehicle 1 takes Rider 2 and Vehicle 4 takes Rider 3 to Times Square. In this case, no ride-sharing is involved and thus four vehicles are used to serve part of the demand. Since no more vehicles are available, Rider 5 request deemed rejected. We now consider a scenario where Vehicle 2 can serve Rider 1 (due to its proximity to Rider 1) and Vehicle 3 serves Riders 4 and 5. Vehicle 1 serves Rider 2 and Vehicle 4 serves Rider 3. In this case, ride-sharing allows efficient use of vehicles, thus no request is rejected. Still, there is a room for improvement through multi-hop ride-sharing. In such settings, Vehicle 2 picks-up Rider 1 from zone D and drops Rider 1 at zone F (i.e., a hop zone). Then, Vehicle 3 takes Rider 1, Rider 4 and Rider 5 to the Central Park (see the green dashed-line in Figure 2.2). In addition, Vehicle 2 moves from zone F to zone E to pick up Rider 2 and then picks up Rider 3 a long its way to the Times square, NY (the red dotted-line in Figure 2.2). In this case, only 2 vehicles are used to fulfill the demand. We note that under MHRS scenario, the non-used vehicles (Vehicle 1 and Vehicle 4) are free and able to serve other customers and thus improve the acceptance rate of ride-sharing systems.

From the proceeding discussion, we observe that MHRS leads to better utilization of vehicles, reduced travel cost due to sharing the cost by passengers, and reduced traffic congestion since less number of vehicles are used to serve the user's demand. Since traffic is reduced, pollution could be also mitigated. Moreover, since fewer number of vehicles are used, the incurred fuel cost is reduced, which results in less cost for customers. Table 2.1 shows a comparison of the no hop (MOVI), ride sharing (DeepPool) and multi hop ride

Table 2.1. : Comparison of no hop, ride sharing and multi hop ride sharing

Comparison of no hop, ride sharing and multi hop ride sharing			
Algorithm	Number of vehicles used	Rejected Requests	Free Vehicles
No hop & No ride sharing	4	1	0
Ride Sharing	3	0	1
Multi Hop Ride Sharing	2	0	2

sharing (MHRS) scenarios in terms of vehicle utilisation, rejected requests and free vehicles. It can be seen that MHRS performs the best out of all the three scenarios.

2.3 Model Parameters and Notations

We use the following notations in our paper to define state, supply and demand. We use $\mathrm{i} \in \{1, 2, 3, \ldots, M\}$ for the zones. The number of vehicles is taken as N. We optimize the multi hop system on T time slots, where one time slot has length Δt. The vehicles take decisions at each time slot $\tau = t_0, t_0 + 1, t_0 + 2, \cdots, t_0 + T$, where t_0 is the current time slot. Let the number of available vehicles at region i at time slot t be $v_{t,\mathrm{i}}$. A vehicle is marked as *available* if at least one of its seats is vacant. A vehicle is marked unavailable if it is completely full or does not consider taking an extra passenger. We can only dispatch those vehicles which are available. Vehicle v has a maximum capacity of C_v passengers. We take the number of requests at zone i at time t as $d_{t,\mathrm{i}}$. $d_{t,\widetilde{t},\mathrm{i}}$ is the number of vehicles that are not available at time t but will drop-off a passenger(s) at zone i and then become available at $\widetilde{t}$. We can estimate the $d_{t,\widetilde{t},\mathrm{i}}$ using the estimated time of arrival (ETA) model [32]. 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 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, $d_{t,\widetilde{t},\mathrm{i}}$. 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 $hop_{l,\mathrm{i},p}$ to denote the passenger l hopped down at zone i, and p denotes the pth hop of the passenger. We can improve upon our dispatching policies by estimating the demand in every zone through historical weekly/daily distribution of trips across zones [33]. 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 $\boldsymbol{s}_t$.

2.4 Objective

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/idle time for passengers and vehicles, respectively, b) minimize the gap between the supply and demand, c) minimize the number of used vehicles to serve the demand, d) minimize the number of hops (or transfers) for customers 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.

The metrics mentioned above will be explained in detail mathematically in the subsequent sections. We can associate different weights with these metrics. These weights help in deciding which metrics should be given more importance over others. For example, we can assign more weight to accept rate if we want to ensure high overall accept rate for the trips.

For every time step, the agent takes an action, in our case the vehicles choose to be dispatched to the zones where future demand is anticipated. The agent (i.e., vehicle) receives a reward based on the action taken. Ideally, the agent takes the actions that would maximize its expected discounted future reward, i.e.,

$$\sum_{k=t}^{\infty} \eta^{k-t} \boldsymbol{r}_k(\boldsymbol{a}_t, \boldsymbol{s}_t), \tag{2.2}$$

where $\eta < 1$ is a time discounting factor. In the context of this paper, the reward $\boldsymbol{r}_k(.)$ is defined as a weighted sum of different performance components that best capture the motive of our system as mentioned above. Since the number of vehicles is limited, it is likely that some rides may not be served, so we minimize the supply and demand gap. Since we want to motivate sharing and hopping but, at the same time, do not want the customer to have a bad experience, we minimize the overhead time and the number of hops per customer. In addition, we aim to minimize the wait time and idle cruising time to promote effective utilization of vehicles and smooth ride experience for the customer. The agent takes an action $\boldsymbol{a}_t$ based on the environment state $\boldsymbol{s}_t$ and receives the corresponding reward $\boldsymbol{r}_t$.

2.5 Assumptions

We make certain assumptions in building the algorithm for Multi-hop ride sharing (MHRS) and evaluating our algorithm. The assumptions are as follows:

1. All rides are requested with an app (such as Uber, Lyft).
2. Vehicle states are available in real time without any lag.
3. Users are willing to share rides with other passengers.
4. Passengers have uniform preferences (they are willing to hop from one vehicle to another) and incentives.
5. All vehicles are of the same type and will take shortest time path to fulfill requests.
6. Requests are rejected if no vehicle available in 5-km radius.
7. Simulator cannot take instantaneous decisions; simulators takes decisions (if any) every minute.
8. Matching of vehicles to passengers and vehicles is not dynamic.

3. MULTI-HOP RIDE-SHARING FRAMEWORK DESIGN

In this section, we present our framework to solve the MHRS problem. We propose a distributed model-free approach using DDQN, where the agent learns the best actions based on the reward it receives from the environment. It should be noted that the agent here refers to the vehicles, and our algorithm learns the optimal policy for each vehicle independently.

3.1 Reward

The first component of the reward is to minimize the gap between the supply and demand for the agents/vehicles. Let $\boldsymbol{v}_{t,\mathrm{i}}$ 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 $(\boldsymbol{v}_{t,\mathrm{i}} - \overline{\boldsymbol{d}}_{t,\mathrm{i}})$, i.e.,

$$\mathrm{diff}_t^{(D)} = \sum_{\mathrm{i}=1}^{M} \left(\overline{\boldsymbol{d}}_{t,\mathrm{i}} - \boldsymbol{v}_{t,\mathrm{i}}\right)^+ \tag{3.1}$$

where $(.)^+ = \max(0,.)$. 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 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 from the current zone to a particular zone, say zone j, i.e., $h_{t,\mathrm{j}}^n$ 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, $T_t^{(D)}$,

$$T_t^{(D)} = \sum_{n=1}^{N} \sum_{\mathrm{j}=1}^{M} h_{t,\mathrm{j}}^n u_{t,\mathrm{j}}^n \tag{3.2}$$

where $u_{t,\mathrm{j}}^n = 1$ only if vehicle n is dispatched to zone j at time t. We seek to minimize the dispatch time sine drivers are consuming fuel without gaining revenue. 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. $t^{(m)}_{t,n,\ell}$ is the travel time that would have been taken if the passenger 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

$$\boldsymbol{\Delta}_t = \sum_{n=1}^{N} \sum_{\ell=1}^{U_n} \delta_{t,n,\ell} \tag{3.3}$$

To ensure that the passengers do not have to go through any inconvenience (e.g., many hops a long 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 $hop_{\ell,\mathrm{i},p}$ denote the passenger ℓ hopped down at zone i, and p denotes the pth hop of the passenger ℓ. Then,

$$H_\ell = \sum_{\mathrm{i}=1}^{M} \sum_{p=1}^{P} hop_{\ell,\mathrm{i},p} \tag{3.4}$$

Hence, the sum over all passengers ℓ at time t, gives the total number of hops of all customers at instant t, i.e., $\boldsymbol{H}_t = \sum_\ell H_\ell$. Lastly, the number of vehicles at any time e_t needs to be optimized to better utilize the vehicles usage (i.e., available resources), this can be written as

$$\mathrm{e}_t = \sum_{n=1}^{N} (\mathrm{e}_{t,n} - \mathrm{e}_{t-1,n}, 0)^+ \tag{3.5}$$

where $\mathrm{e}_{t,n}$ is an indicator variable, it takes a value of 1 if the vehicle n is occupied at time t and 0 otherwise,

and where $(.)^+ = \max(0,.)$. 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. This would help in reducing the idle cruising time of the vehicles, hence would lead to better vehicle utilization.

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

$$\overline{r}_t = -[\beta_1 \text{diff}_t^{(D)} + \beta_2 T_t^{(D)} + \beta_3 \boldsymbol{\Delta}_t + \beta_4 \text{e}_t + \beta_5 \boldsymbol{H}_t] \tag{3.6}$$

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. For example, β_{i} could be the cost per measure unit, e.g., cost per second. Moreover, we can increase β_4 if we decide our main goal to minimize the size of the fleet. Also, $\beta_{\text{i}}, \forall \text{i}$, can be vehicle dependent (i.e., $\beta_{\text{i},n}, \forall \text{i}, n$) where each vehicle can choose its weights based on the drivers preference. We note that the reward is not an explicit function of the action. Through the model free approach, we learn the optimal relationship between action and reward by our algorithm.

We note that equation (3.6) gives the accumulated reward of all vehicles at time t in a centralized manner. To solve the problem in a distributed fashion where each vehicle solves its own DDQN individually, we need to write the reward (and its related components) of every vehicle independent of other vehicles. In the next section, we provide the reward expression for each vehicle individually and show that by summing over all vehicles the total reward function is analogous to the objective function described in Section 3.

3.2 Distributed MHRS Framework

We build a simulator to train and evaluate our framework. We assume that a central unit is responsible for maintaining the states of all vehicles, e.g., current locations, destinations, occupancy state, etc. These states are updated in every time step based on the dispatching and assignments decisions. When needed, a vehicle n communicates with the central unit to either request new information of other vehicles or update its own status. The goal of our distributed MHRS policy is to learn the optimal dispatch actions for each vehicle individually. Towards that goal, at every time slot t, the idle vehicles decide in *parallel* the next step (which zone to go) taking into consideration the locations of all other nearby vehicles in its vicinity. When the vehicles are idle, they can go to the same location, but their

states would be different if current location of vehicles is different. The state of two vehicles can't be the same as they match with different passengers. Overtime, the algorithm will learn the optimal policies as the vehicles explore more their environment and converge towards optimal decisions rather than making similar decisions. However, their current actions do not anticipate the future actions of other vehicles. We note that it is unlikely for two (or more) drivers to take actions at the same exact time since drivers know the location updates of other vehicles in real time, through the GPS for example. The advantage of MHRS policy stems from the fact that each vehicle can take its own decision without coordination with other vehicles. We note that the algorithm is distributed because the dispatching and matching decisions are made independently by all the vehicles, without any prior information of other vehicles' action. The dispatch decisions depend on the current state and matching depends on the nearby passengers. Next, we explain the state, reward and action.

State

The state variables capture the environment status and thus affect the reward feedback of different actions. The state at time t is captured by a three tuple t: $(\boldsymbol{X}_t, \boldsymbol{V}_{t:t+T}, \boldsymbol{D}_{t:t+T})$. These elements are combined in one vector denoted as $\boldsymbol{\Omega}_{t,n}$. When a set of new ride requests are generated, the MHRS engine updates its own data to keep track the environment status. The three-tuple state variables in $\boldsymbol{\Omega}_{t,n}$ are pushed as an input to the neural network input layer to make a certain decision.

Action

We denote the action of vehicle n by $\boldsymbol{a}_{t,n}$. This action consists of two components –i) if the vehicle is partially filled it decides whether to serve the existing passengers or to accept new customer, and ii) if it decides to serve a new customer or the vehicle is totally empty, it decides the zone it should head (or hop) to at time slot t, which we denote by $w_{t,n,\mathrm{i}}$. Here, $w_{t,n,\mathrm{i}} = 1$ if the vehicle decides to serve a new customer and/or head to zone i, otherwise it

is 0. Note that if vehicle n is full it can not serve any additional customer. In contrast, if a vehicle decides to serve current customers, it opts the shortest optimal route for reaching the destinations of the users.

Reward

We now represent the reward fully, and the weights which we put towards each component of the reward function for each vehicle if it is not full. The reward $r_{t,n}$ for vehicle n at time slot t in this case is given by

$$\begin{aligned} r_{t,n} &= r(\boldsymbol{s}_{t,n}, \boldsymbol{a}_{t,n}) \\ &= \beta_1 b_{t,n} - \beta_2 c_{t,n} - \beta_3 \sum_{\ell=1}^{U_n} \delta_{t,n,\ell} \\ &- \beta_4 \max\{\mathrm{e}_{t,n} - \mathrm{e}_{t-1,n}, 0\} - \beta_5 \boldsymbol{H}_{t,n} \end{aligned} \tag{3.7}$$

where β_{i} is the weight for component i in the reward expression. Note that $\boldsymbol{H}_{t,n} = \sum_{\ell \in \mathcal{L}_{t,n}} \{\boldsymbol{H}_\ell\}$. This denotes the average number of hops made by every vehicle. Further, $b_{t,n}$ denotes the number of customers served by vehicle n at time t while $c_{t,n}$ denotes the time taken by vehicle n to hop or take a detour to pick up extra customers if the vehicle has available seats. $\delta_{t,n,\ell}$ denotes the additional time vehicle n takes because of carpooling compared to the scenario if the vehicle would have served customer ℓ solely without any carpooling and/or hoping. Note that while taking the decision, the vehicle n exactly knows the destination of the passenger. When a user is added, the route is updated for dropping the passengers. U_n is the total number of chosen customers for pooling at vehicle n till time t. $\mathcal{L}_{t,n}$ contains the set of all customers assigned to vehicle n at instant t. Both U_n and $\mathcal{L}_{t,n}$ are not known a priori and will be adapted dynamically in the MHRS policy. These two quantities will also vary as the passengers are picked or dropped by the vehicle n. The last term captures the vehicle status, where $\mathrm{e}_{t,n}$ is set to one if empty vehicle n becomes occupied (even if by one passenger), however, if an already occupied vehicles takes a new customer it is 0. The

intuition behind this reward term is that if an already occupied vehicle serves a new user, the congestion cost and fuel cost will be less rather than an empty vehicle serves the user. If we make both β_3 and β_5 very large, we resort to the scenario where there is no carpooling. Since high β_3 indicates that passengers will not prefer detours to serve another passenger. Thus, the setting becomes similar to the one in [30]. Moreover, by setting β_5 to a very large value, our MHRS algorithm will resort to that of ride-sharing with no passenger transfer capability, which is similar to the algorithm presented in [32].

3.3 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 DDQN algorithm, we further combine grids to form a new map of 41 x 43 square grids, preventing 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. Through this method, we obtain 148 hop zones in the whole New York city.

3.4 MHRS Algorithm Description

In this section, we explain our proposed algorithm for MHRS in detail. The full algorithm is shown in Algorithm 2. We first construct the state vector $\boldsymbol{\Omega}_{t,n}$, using the ride requests and available vehicles, which are extracted from real records (line 1). $\boldsymbol{\Omega}_{t,n}$ is defined as the state of the vehicle n at time t. Then, we initialize the number of vehicles and generate some ride requests in each time step based on the real record in the dataset (see lines 2 to 5). Next, the agent determines the actions $\boldsymbol{a}_t$ using our MHRS algorithm and matches the vehicle to the appropriate ride request (or requests), lines 6 to 14. Every vehicle n selects the action that maximizes its own reward, i.e., taking the *argmax* of the DDQN-network output.We use

the Double Deep Q Networks (DDQN) to learn the policy for the agents. DDQN combines Q learning with a deep neural network. It has been shown in [34] that the DDQN keeps over estimations in check. In DDQN, two value functions are learnt resulting in two sets of weights, θ and θ. For each update, one set of weights is used to determine the greedy policy and the other to determine its value. DDQN decouples the process of taking action and getting the Q value.At any time step t the target Q value is defined as:

$$\boldsymbol{Y}_t^{DDQN} = R_{t+1} + \gamma Q(S_{t+1}, argmax_a Q(S_{t+1}, a; \theta_t); \theta_t) \tag{3.8}$$

where θ_t is used to determine the greedy policy and θ_t is used to evaluate the value of this policy. The DDQN algorithm is presented in Algorithm 1.

The output of the DDQN are the Q-values corresponding to every possible movement, while the input is the environment status governed by the vector $\boldsymbol{\Omega}_t$. $\boldsymbol{\Omega}_t$ is defined as the state at any time t. Note that each vehicle updates its status at the agent based on the chosen action (line 15). We note that state of a vehicle is dependent on the location and capacity of other vehicles in the fleet, but is not dependent on the future actions of the other vehicles. This makes the dispatching process *non-sequential*. It may so happen that during matching, two vehicles are matched with the same passenger. In such cases, we allow the passenger to randomly pick one vehicle (equally likely among all vehicles that are matched to the same passenger). Each vehicle does not anticipate the future actions of other vehicles, thus limiting the coordination among them.

Then, the trajectory is decided based on the shortest path on the road network graph. In the algorithm, the trajectory of a vehicle is defined as the path the vehicle will follow after a dispatch decision. The dispatched vehicles travel to the destination location in the time estimated from the ETA model (Section IV-B), see lines 18 to 23. The extra travel time $\delta_{t,n}$ and hop counter $H_{t,n}$ are updated as needed (lines 19-25).

Algorithm 1 Double Q-learning with experience replay

Initialize replay memory D, Q-network parameter θ, and target Q-network θ^{-}.
for e : 1 : *Episodes* **do**
    Initialize the simulation with arbitrary number of vehicles and ride requests based on real data.
    for $t : \Delta t : T$ **do**
        Perform the dispatch and match order
        Update the state vector $\boldsymbol{\Omega}_t = (\boldsymbol{X}_t, \boldsymbol{V}_{t:T}, \boldsymbol{D}_{t:T})$.
        Update the reward $\boldsymbol{r}_t$ based on actions $\boldsymbol{a}_t$.
        for all available vehicles n **do**
            Create $\boldsymbol{\Omega}_{t,n} = (\boldsymbol{X}_{t,n}, \boldsymbol{V}_{t:T}, \boldsymbol{D}_{t:T})$.
            Store transition
            $(\Omega_{t-1,n}, a_{t-1,n}, r_{t,n}, \Omega_{t,n}, c_{t,n})$
        end for
        Sample random transitions
        $(\Omega_{\mathrm{i}}, a_{\mathrm{i}}, r_{\mathrm{i}}, \Omega_{\mathrm{i+1}}, c_{\mathrm{i+1}})$ from D.
        Set $a_{\mathrm{i}}^{*} = \mathrm{argmax}_a Q(\Omega_{\mathrm{i+1}}, a_{\mathrm{i}}^{*}; \theta^{-})$.
        Set $z_{\mathrm{i}} = r_{\mathrm{i}} + \gamma^{1+c_{\mathrm{i+1}}} \hat{Q}(\Omega_{\mathrm{i+1}}, a_{\mathrm{i}}^{*}; \theta^{-})$.
        minimize $(z_{\mathrm{i}} - Q(\Omega_{\mathrm{i}}, a_{\mathrm{i}}; \theta))$ w.r.t. θ.
        Set $\theta = \theta^{-}$ every N steps.
        Update the set of available vehicles A_t
        for n in A_t **do**
            Create $\boldsymbol{\Omega}_{t,n} = (\boldsymbol{X}_{t,n}, \boldsymbol{V}_{t:T}, \boldsymbol{D}_{t:T})$.
            Choose, with prob. ϵ, a random action from
            $a_t^{(n)}$.
            Else set $a_t^{(n)} = \mathrm{argmax}_a Q(\Omega_t^{(n)}, a; \theta)$.
            Send vehicle n to its destination, based on
            $a_t^{(n)}$.
            Update $\boldsymbol{\Omega}_{t,n}$.
        end for
    end for
end for

MHRS can be scaled to a large number of vehicles. This is because Double Deep Q Networks (DDQN) have $O(1)$ time complexity during inference. Moreover, since our algorithm is distributed, we will be able to avoid action space explosion because we do not consider the joint action space to make the decisions. Thus, MHRS can be extended and scaled to accommodate a large number of vehicles and passengers.

Algorithm 2 MHRS Algorithm

```
Form a state vector Ω_{t,n} = (X_t, V_{t:t+T}, D_{t:t+T}).
Initialize vehicles states X_0 as location of first N requests.
for t ∈ Total time do
    Get all ride requests at time t
    Group all the requests assigned to the same hop zone
    for Each request group in time slot t do
        Match a vehicle n to serve the requests
        Update the vehicle capacity
        Update the pickup and destination for hop trips
        Calculate wait time and the number of passengers
        without any hops
        Calculate the dispatch time using ETA model
        Update the state vector Ω_{t,n}.
    end for
    Send the state vector Ω_t to the Q network.
    Get the best dispatch action a_t for each agent from
    the network.
    for  all the available vehicles n ∈ a_t  do
        Update H_{t,n}, if needed, and update the current
        location of vehicle
        Find the shortest path to every dispatch location
        for every vehicle n.
        Estimate the travel time using the ETA model
        Update δ_{t,n}, if needed, and generate the trajectory
        of vehicle n.
    end for
    Update the state vector Ω_{t+1}.
end for
```

4. SIMULATOR SETUP AND EVALUATION

4.1 Setup and Initialization

Through extensive simulations, we show how MHRS algorithm works in real time scenario as compared to ride-sharing [32] and no ride-sharing [30] scenarios. We trained our algorithm using real world taxi trips in New York City for May 2016. Then, we tested our algorithm using the trip records of the first two weeks of June 2016 data. We divide the New York city into grids of size 41×43, each grid has a an area of 800 x 800 m^2.

We built a discrete-event Fleet Simulator in Python in which taxi dispatch, matching with passengers and pickup and drop-off of the passengers are simulated. The Fleet Simulator keeps a track of all attributes related to the ride requests and vehicles such as: incoming ride requests, vehicle status, current location of vehicles, availability of vehicles, reward collected by the vehicles, relative distance gain of the vehicles, etc. When the simulator takes a time step forward (e.g., every one minute for instance), existing ride requests are matched with some of the available vehicles (based on our proposed matching policy) and the rest of the vehicles are dispatched to the new locations where future demand is anticipated as to maximize their rewards. Once a vehicle changes its state, it sends its new attributes to the central unit that keeps entire fleet's information. The technical details of the fleet simulator can be found in the source code in [35].

To initialize the environment, we run the simulation for 20 minutes without dispatching the vehicles. Further, we set the number of vehicles to $N = 5000$. The initial locations of these vehicles correspond to the first 5000 ride requests. We set the maximum horizon to $T = 30$ step and $\Delta t = 1$ minute. We breakdown the reward and investigate the performance of the different policies. Further, unless otherwise stated, we set $\beta_1 = 5$, $\beta_2 = 1$, $\beta_3 = 3.5$, $\beta_4 = 0.05$, and $\beta_5 = 2$.

Recall that the reward function is composed of five different metrics, which we seek to minimize: (i) mismatch between supply and demand, (ii) dispatch time for vehicles, (iii)

extra travel time because of ride-sharing and/or detouring for hops, (iv) number of used vehicles (resources), and finally (v) number of hops per customer. We note that the supply-demand mismatch is reflected in our simulation through the accept rate metric. This metric is computed as the number of ride accepts divided by the total number of requests per day.

Recall that a request is deemed rejected if there is no vehicle in the $5km$ radius, or no vehicle is available to serve the request. Also, the metric of idle time represents the time at which a vehicle is not occupied while still incurring gasoline cost and not gaining revenue. In addition, the waiting time is composed of two sub-components: the time from the ride request till the time at which the vehicle arrives to pick up the customer and the incurred wait time in hop zone(s), if any. Below, we explain our models to estimate the time of arrival, demand and our proposed DDQN architecture.

4.2 Estimated Time of Arrival (ETA) and Demand Prediction

We use the New York City taxi data set [36] to build the ETA and demand prediction models.

The traffic is unpredictable and is susceptible to spontaneous change. In [37], the shortest route between two locations is found to calculate the travel time of a vehicle in each route and the travel route. We take into account the dynamic and spontaneous nature of the traffic through the features of our Estimated time of Arrival (ETA) model. The ETA model is trained on real traffic data of NYC and uses features such as pickup latitude and longitude, drop off latitude and longitude, sine and cosine for day of week and sine and cosine for hour of day and distance calculated from the road network graph. We divide our training data into 70% training and 30% test sets. We obtained RMSE of 3.2 and 3.4 on the training set and the test set for the model respectively. The non parametric and non linear prediction model for ETA we have built learns the traffic congestion and density fluctuations based on origin and destination of vehicles (e.g., vehicles originating from airport may face a lot of congestion), time of day when the vehicle is travelling (e.g., there will be more congestion

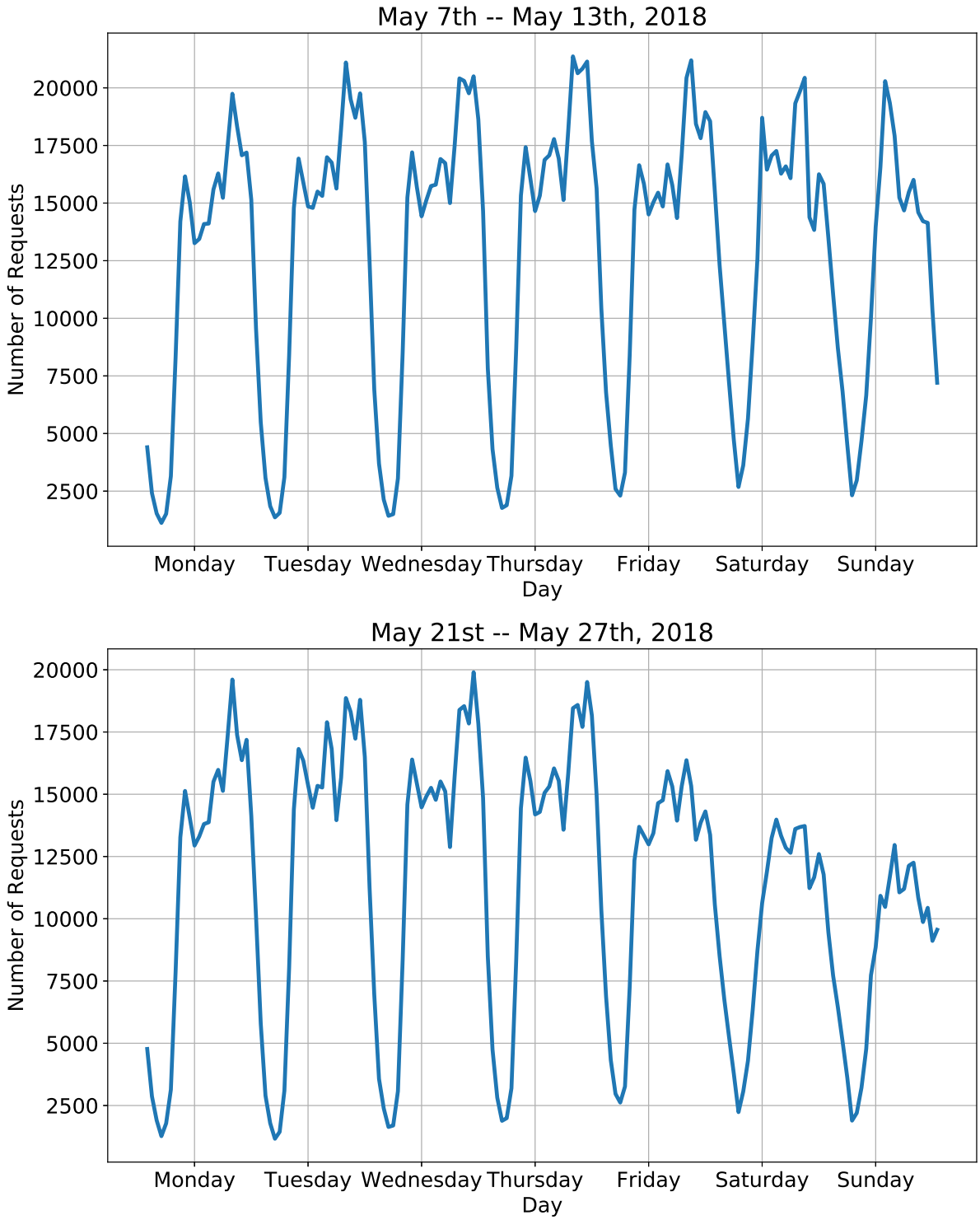

Figure 4.1. : Demand pattern for two weeks in May 2018.

during office hours) and day of week (e.g., certain routes will be busier during the weekends). Thus, the ETA model closely resembles the actual traffic conditions in the New York City. This helps us to correctly predict the correct ETA for vehicles and direct them to the correct path.

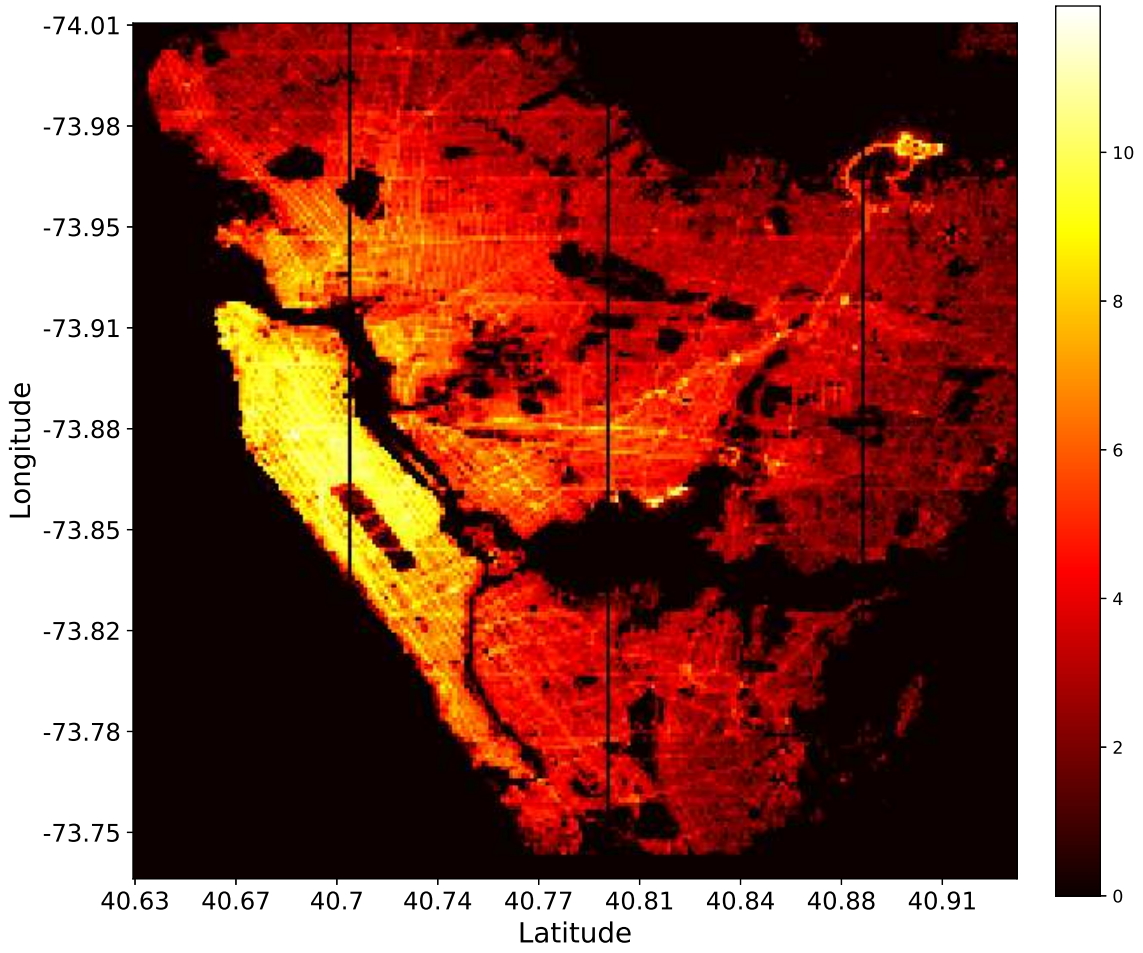

Figure 4.2. : Demand Heat Map in New York City

For the demand prediction model, we predict the expected number of requests in a given zone for 15 and 30 minutes ahead. We plot in Figure 4.1 the weekly number of ride requests for two different weeks: May 7th-May 14th and May 21st-May 27th. We observe that both two weeks data exhibits the same daily behavior. In particular, both datasets experience daily periodicity with lower demand on the weekend, and thus the demand has

a daily/weekly pattern that can be predicted. We simulate our algorithm using 12 million of real taxi trips in NYC and thus MHRS algorithm accounts for trips variability and rely on realistic distribution in matching/dispatching the vehicles-to-customers. In Figure 4.2 we show the demand pattern in the form of a heat map. The New York city is plotted in the form of a rectangular grid of size 212 x 219. The brightest areas on the heat map, yellow zones, belong to the Manhattan area, where the trip demands are very high throughout the day.

We use convolutional neural networks for our demand predictions. The input to the network is 2 planes of 212 x 219. Each grid in the planes represents the ride request in that zone for the previous 2 time steps. In the network, we have 16 of 5x5 filters, 32 of 3x3 filters and finally one 1x1 filter. The output is a plane of 212x219, with each grid in the plane representing the demand in that zone in the next 30 minutes. The output of the demand prediction model will be used in the state in our model free algorithm. It should be further noted that we are not explicitly learning the transition probabilities from one state to another, which makes our approach model free.

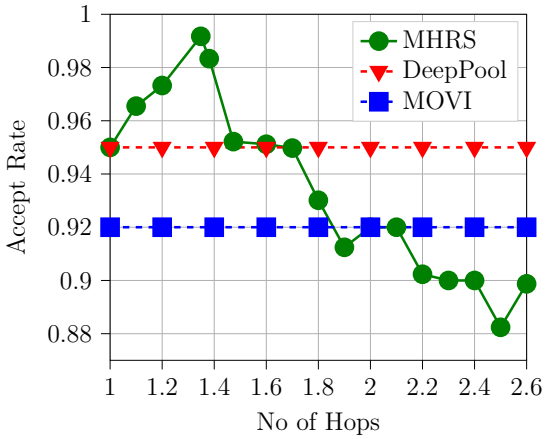

Figure 4.3. : Accept rate versus the average number of hops.

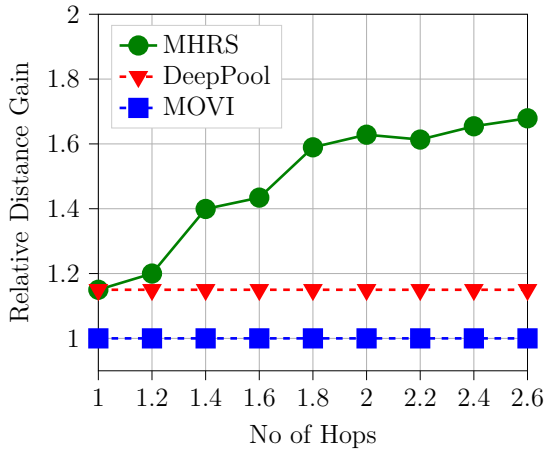

Figure 4.4. : Relative distance gain versus average number of hops.

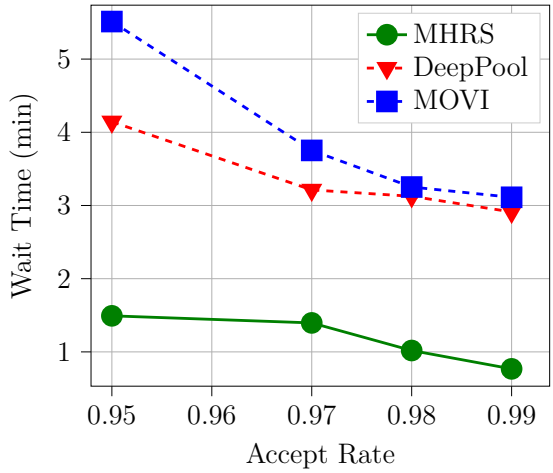

Figure 4.5. : Average wait time per request for different accept rate.

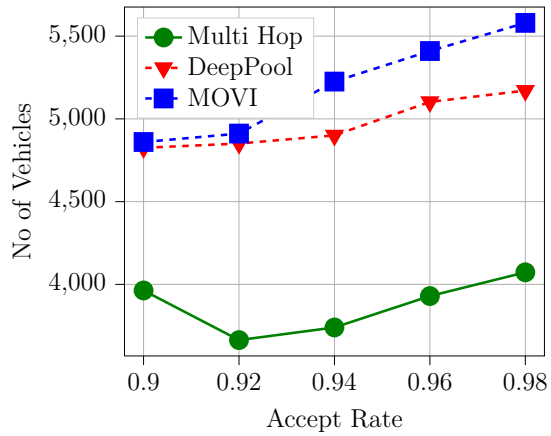

Figure 4.6. : Average used vehicles for different accept rate.

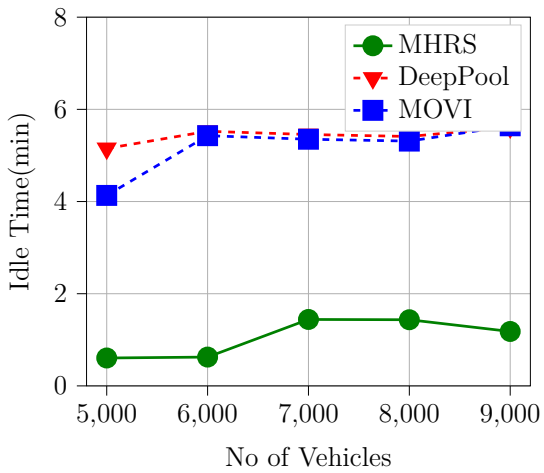

Figure 4.7. : Average idle time for different number of vehicles.

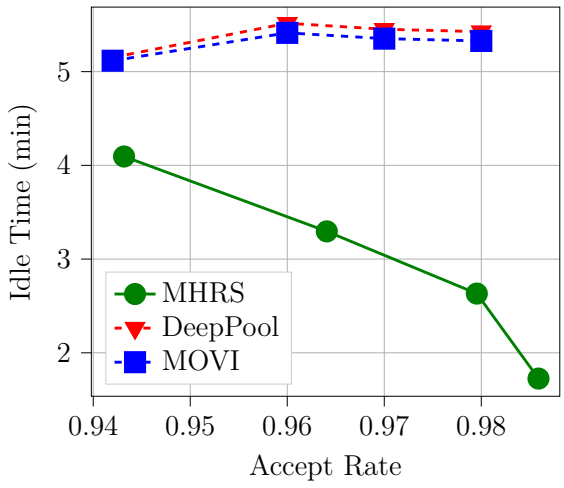

Figure 4.8. : Average idle time per request for different accept rate.

4.3 Double Deep Q-Network Architecture

For the DDQN, 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

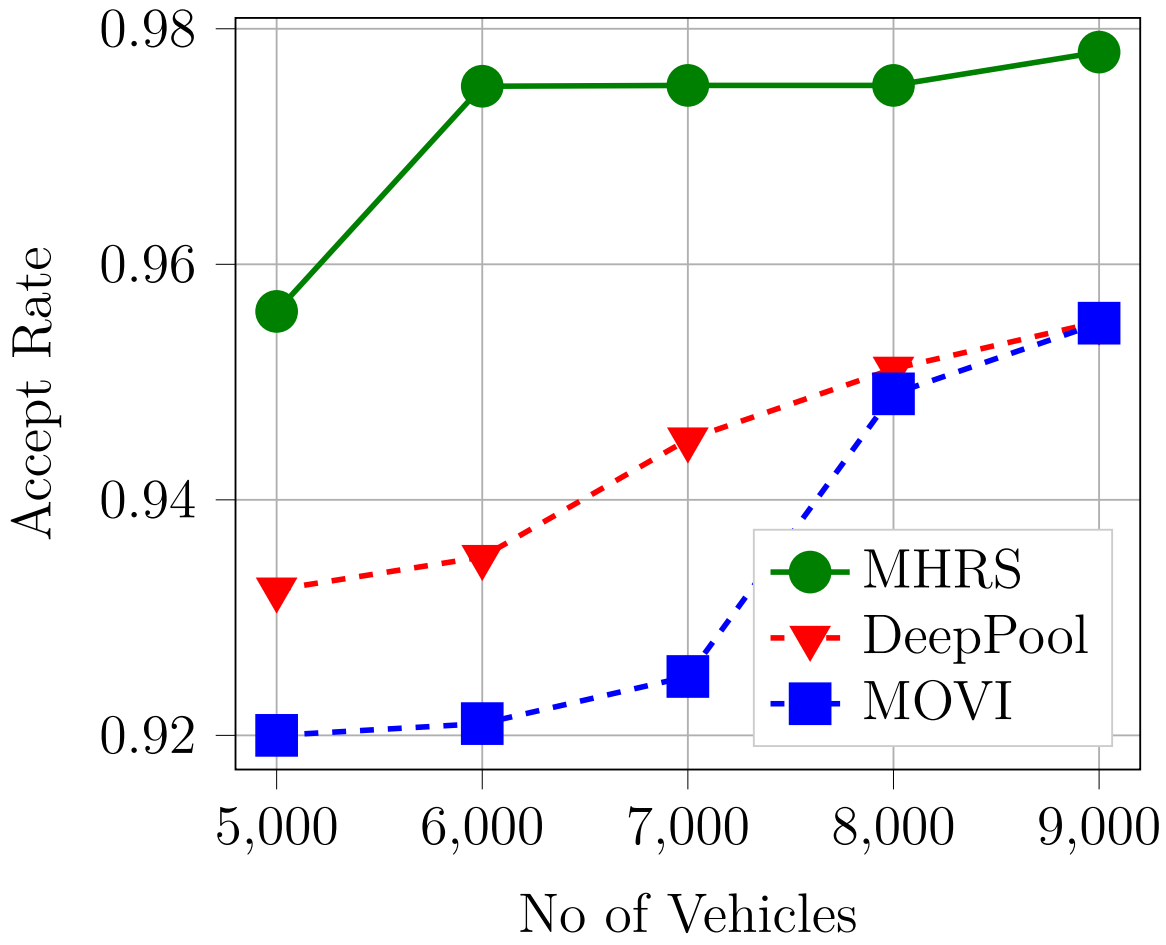

Figure 4.9. : Average accept rate for different number of vehicles.

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

4.4 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 DDQN policies but only one rider can occupy a vehicle at any given time. This policy is akin to that in [30], which we refer in our simulation comparisons as "MOVI". For the RS case, in addition to DDQN dispatch policies, more than one passenger can be assigned to one vehicle but no multi-hop is performed. This policy is proposed in [32], so-called DeepPool. In our MHRS policy, ridesharing is allowed and passengers could complete their trips in more than one hop.

Unless otherwise stated, 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.

Figures 4.3 and 4.4 plot accept rates and relative distance gain as a function of hops respectively. Although MOVI and DeepPool are not a function of hops, the figures help us understand how MHRS compares to DeepPool and MOVI as a service. Figure 4.3 plots the average accept rate versus the number of hops. We change the parameter β_5 to show the effect of hops on the accept rate. The values of β_5 used are - 13, 11, 9, 8, 7.5, 6.5, 5.5, 3.5, 2.8, 2.3, 1.9, 1.8, 1.7, 1.6, 1.5 and 1.4. Note that if a passenger is dropped at a hop zone and is not served in $\delta_{t,n,\ell}$, 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 at 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 till an average of 1.6 hops. This decrease of

accept rate after 1.6 is due to the possibility for our algorithm to fail serving some of the passengers at the hop zones. As number of hops increases, vehicles become more densely packed and thus take a long time to complete the trips of existing customers. Due to this reason, a lot of passengers waiting for a car have their trips rejected. 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 4.3 shows that we can achieve upto 99% accept rates with MHRS with 1.38 average hops and savings in relative distance gain. Figure 4.3 shows that we can achieve upto 99% accept rates with MHRS. Figure 4.4 shows the relative distance gain of vehicles versus the number of hops. Relative distance gain tells us the number of times the efficiency of a single vehicle increases when it follows MHRS or DeepPool. The added efficiency comes from the fact that the same vehicle is used to fulfill more than one request. We change the parameter β_5 to show the effect of hops on the relative distance gain. The values of β_5 used are - 13, 11, 9, 8, 7.5, 6.5, 5.5, 3.5, 2.8, 2.3, 1.9, 1.8, 1.7, 1.6, 1.5 and 1.4. Increasing relative distance gain with number of hops suggests that the vehicle is more efficiently packed as number of hops increases. Since only one passenger is assigned to a vehicle in MOVI, its relative distance gain is 1. It is evident that MHRS is efficient as compared to DeepPool since the former achieves better overall utilization of vehicles even when the number of hops is close to 1. The vehicles following MHRS show at least 20% improvement in relative distance gain (thus 20% cost effective) as compared to MOVI and DeepPool at 1.6 average hops. The gap increases as average hops increase at a cost of rejects.

Figures 4.8, 4.5 and 4.6 plot the idle time, wait time and number of vehicles used respectively for different accept rates. We change the values of β_1 to obtain the figures. The values of β_1 used are - 5, 7, 8, 9 and 10.

The idle time per vehicle is captured in Figure 4.8. Idle time is defined as total time when the cars are not serving any request divided by the total number of requests. This metric gives an indication on how much time a vehicle is burning extra fuel without serving

any customer or gaining revenue. 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 (This can also be inferred from Figure 4.3).

Figure 4.5 shows that the wait time decreases as the accept rate increases. Wait time is defined as the average difference between the time of request for a passenger and the time when the passenger starts his/her ride. 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.

Figure 4.5 shows that with MHRS, up to 40% reduction is observed in waiting time.

The number of used vehicles for different accept rate is shown in Figure 4.6. The number of used vehicles is defined as the average number of vehicles used to fulfill the requests. We note that MHRS needs fewer number of vehicles as compared to DeepPool and MOVI. This indicates that MHRS contributes towards reduced fuel consumption and less traffic. Further, it leads not only to a constant circulation of supply but also helps in minimizing the fleet size and maintenance cost.

The number of vehicles utilised from the fleet for similar accept rates is at least 30% lower for vehicles running under the multi hop ride sharing (MHRS) algorithm. Figure 4.6 shows at least 30% better utilization of the vehicle fleet using MHRS algorithm as compared to MOVI and DeepPool.

The effect of varying the number of vehicles on waiting time and idle time is captured in Figures 4.7 and 4.9. We only change the number of vehicles and set our parameters as - $\beta_1 = 5$, $\beta_2 = 1$, $\beta_3 = 3.5$, $\beta_4 = 0.05$, and $\beta_5 = 2$. Intuitively, as the number of vehicles increases, the idle time increases since more vehicles will compete the existing passengers and further cruise searching for potential customers. However, our approach of MHRS still achieves the lowest idle time even in the regime of low number of vehicle. While DeepPool allows pooling customers together, it remains a nook option with limited route choice and

few rides per route can be pooled. This highlights the importance of allowing customers to transfer vehicles to further improve the quality of experience of passengers.

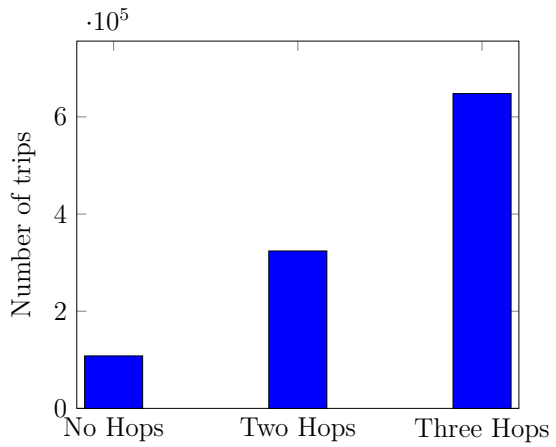

Figure 4.10. : Distribution of customer trips (average hops = 2.5)

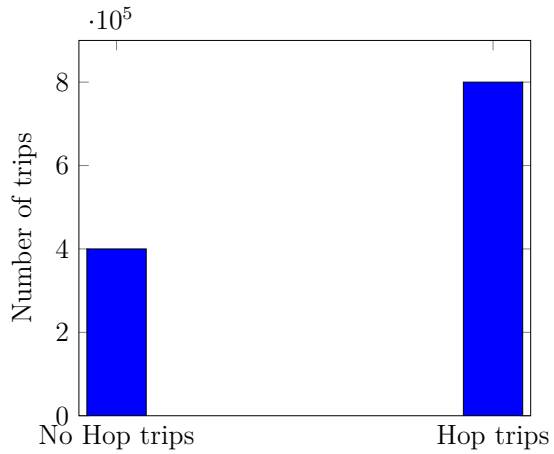

Figure 4.11. : Distribution of customer trips (average hops = 1.4)

Figure 4.9 plots the accept rate for different number of vehicles. The gain of the MHRS over both DeepPool and MOVI policies is remarkable even for low number of vehicles, which highlights the benefits of ride-sharing with passenger transfer capabilities in meeting customer demand at small travel time overhead. We observe that both DeepPool and MOVI achieve higher accept rate as the number of vehicle increase. This is because more vehicles can be dispatched. Further, DeepPool algorithm can serve more customers using a single car by better utilizing the passengers proximity. However, this comes at an additional increased in the idle time of vehicles as depicted in Figure 4.7. By allowing passengers transfer, vehicles can pool even more customers which results in increasing the accept rate of passengers. Figure 4.7 shows that with MHRS, up to 40% reduction is observed in idle cruising time. The distribution of customer trips that encounter hops/transfers during their paths to destination is shown in Figure 4.10 and Figure 4.11. The figures also show the total number of customer trips without hops (i.e., total number of passenger trips that reach their destinations using a single vehicle). We can see that approximately 40% of the total trips involve customer transfers in Figure 4.11 and approximately 90% of the total trips involve customer transfers

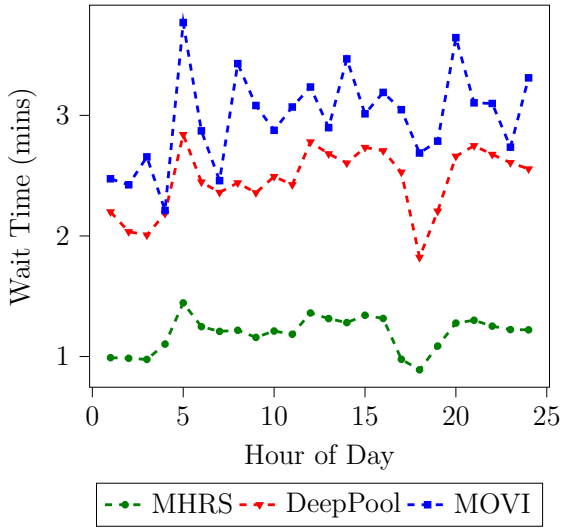

Figure 4.12. : Distribution of wait times in a day

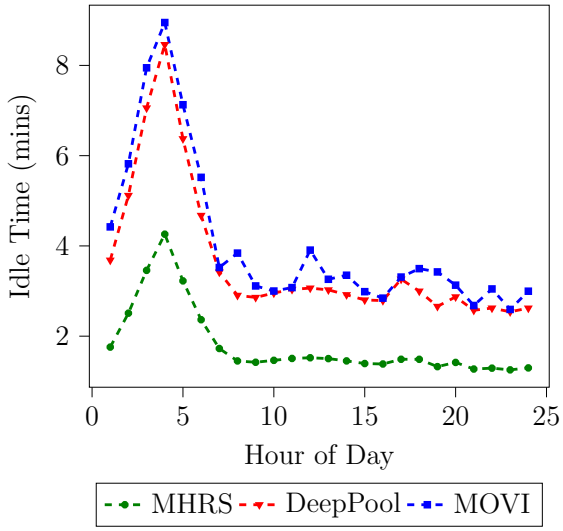

Figure 4.13. : Distribution of idle times in a day

in Figure 4.10. By seeing Figures 4.10 and 4.11 along with Figures 4.3–4.8, we conclude that as number of hops increases beyond 1.6, the accept rate decreases. The scenario where passengers have to transfer more than once reduces (going from one direct trip to three hops), the accept rate decreases by a large amount. Hence, with only one transfer (going from one direct trip to two hops) on 40% of the total trips, we notice significant reductions in customers waiting times, vehicle idle times, and the total number of used vehicle. This shows that it would be sufficient to search only for one transfer since shared rides with two (or more) hops do not give significantly better performance, on an average.

Figure 4.12 shows the improvement in wait times for MHRS as compared to DeepPool and MOVI. This saving comes from efficient customer grouping and the flexibility of MHRS in performing detours to further serve more customers, while maintaining lower idle/crusing time (as shown in Figure 4.13), when compared to DeepPool and MOVI. Hence, the cost of fuel and pollution could be drastically reduced in MHRS. Further, due to efficient customers packing and passenger transfers capabilities, MHRS achieves the most efficient utilization of the fleet as depicted in Figure 4.14 and Figure 4.15.

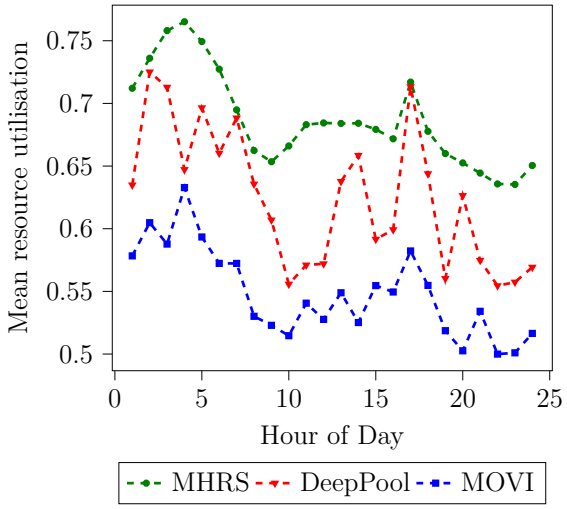

Figure 4.14. : Mean resource utilization in a day

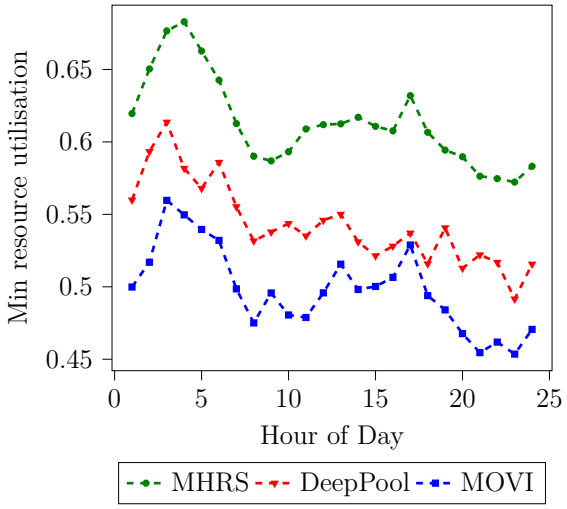

Figure 4.15. : Minimum resource utilization in a day

We also increase the action space and evaluate the time taken during prediction for different sizes of action space. We plot the prediction time latency for different action space sizes in Figure 4.16. We observe that the prediction latency increases with increase in the action space. It should be notes that action space of 4 in the figure suggests that there are 4 possible actions for the agent in all the four directions, thus a total of 81 possible actions are possible at any given time step for the agent. From figure 4.16 we can conclude that in addition to the training time, the prediction latency also increases as we increase the action space.

We plot the accept rates for different action space sizes in Figure 4.17. We observe that MHRS consistently performs better than DeepPool and MOVI. Further, we conclude that the performance drops when the action space is increased. One possible reason for this could be that once vehicles are dispatched to farther off locations, less number of vehicles are available at locations with high demand.

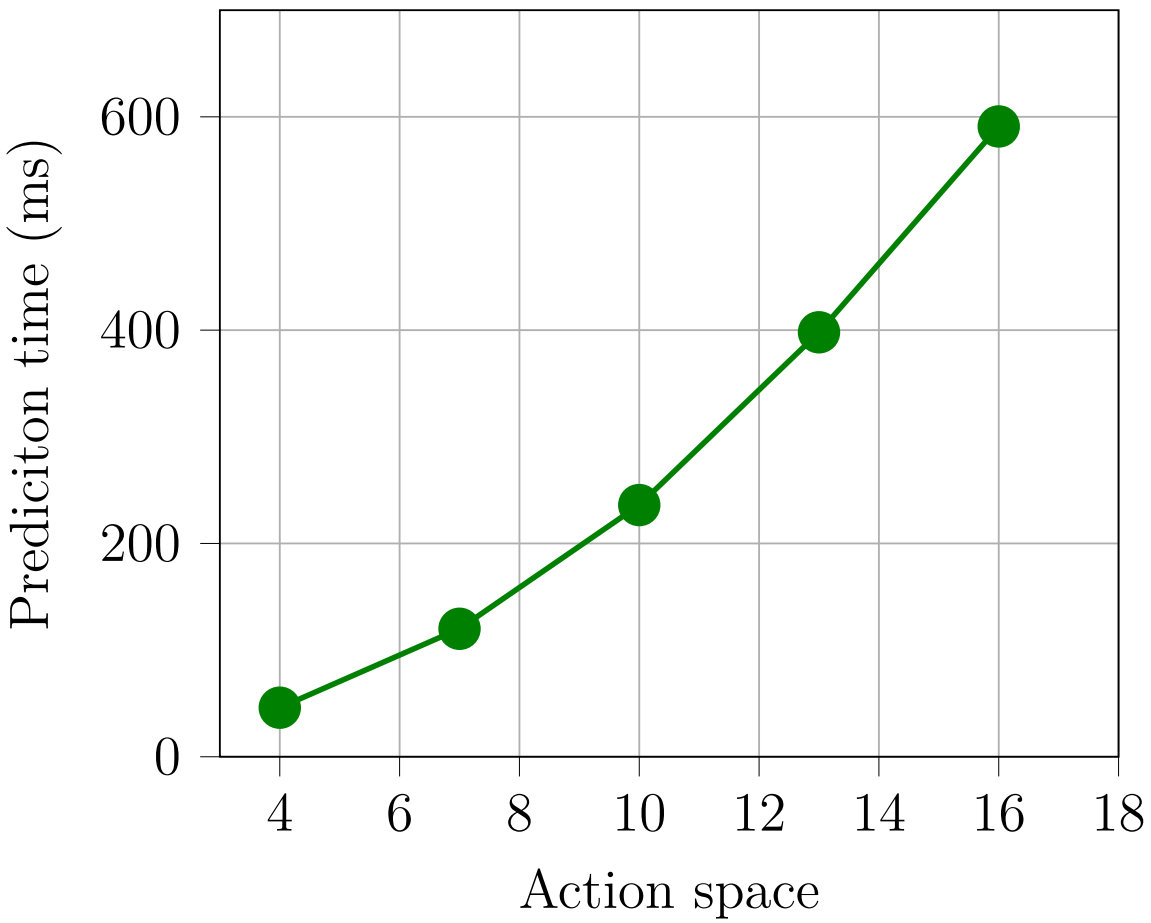

Figure 4.16. : Prediction latency for different action space sizes

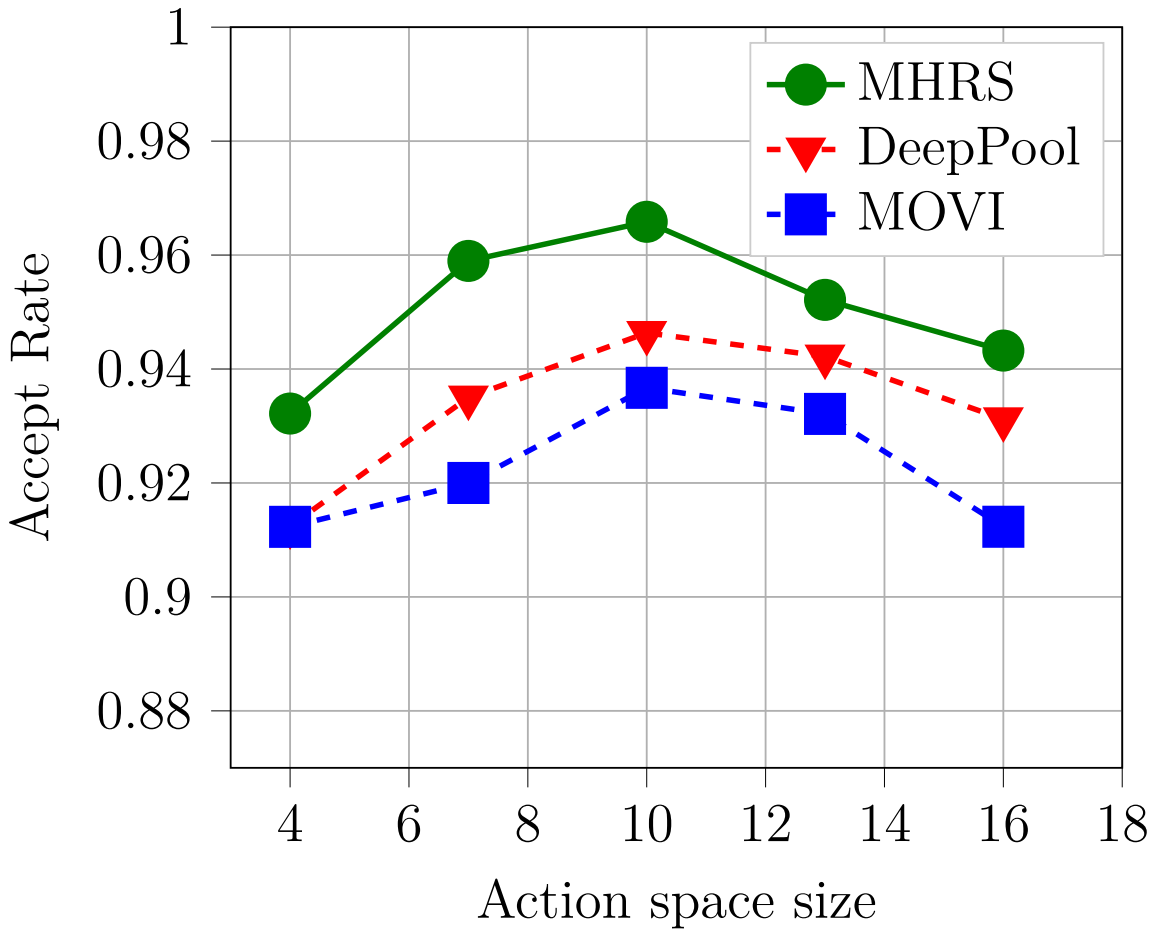

Figure 4.17. : Accept rates for different action space sizes

5. CONCLUSION AND FUTURE WORK

We have looked at the problem of multi-hop ride-sharing, where passengers can be dropped one or more times before reaching their destinations. We have proposed an efficient distributed 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, a constant revenue cycle and reduced prices for customers. Further, we have seen a significant improvement in waiting time, idle time, and the number of used vehicles when going from one hop to two hops, but we have not seen any considerable changes when further increasing the hop count. This indicates that searching for shared rides with three (or more) hops does not give significantly better results, on an 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. MHRS can also be tested in cities other than NYC and such verification is an interesting future direction. It would also be interesting to see how customer satisfaction will play a role in deciding customer's preference for multi hop ride sharing. This is an interesting direction for future work. The choice of β_{i}'s impact social welfare, vehicle profits and customer satisfaction, and efficient choice of such metrics based on pricing mechanism is another fruitful direction for the future.

Further, optimal incentive techniques to influence passengers to opt for the MHRS is another promising research area. Lastly, as a promising alternative to the inefficient traditional package delivery techniques, MHRS systems represent a potential delivery capability through ridesharing within an urban environment. By allowing the passengers and the packages with similar itineraries and time schedules to share one vehicle, the delivery cost and time can be reduced significantly.

Limitations Of The Work and Future Directions

We make certain assumptions in building the algorithm for Multi-hop ride sharing (MHRS) and evaluating our algorithm. The assumptions are as follows:

1. All rides are requested with an app (such as Uber, Lyft).
2. Vehicle states are available in real time without any lag.
3. Users are willing to share rides with other passengers.
4. Passengers have uniform preferences (they are willing to hop from one vehicle to another) and incentives.
5. All vehicles are of the same type and will take shortest time path to fulfill requests.
6. Requests are rejected if no vehicle available in 5-km radius.
7. Simulator cannot take instantaneous decisions; simulators takes decisions (if any) every minute.
8. Matching of vehicles to passengers and vehicles is not dynamic.

As a future research, relaxing some of the assumptions made above could be good direction to explore. It will be interesting to study the effect of behavioural implications of passengers on MHRS and willingness of passengers to transfer between vehicles. Allowing dynamic matching and fleet route optimization will be a good future direction. In addition to this, modeling pricing and weather in a multi-hop ride sharing scenario is another possible area to explore.

Further, it will be fruitful to consider factors that could cause unknown uncertainties in demand, such as hailstorms, accidents etc. in the demand prediction model. Another area of research could be to group the rides with similar origins and destinations as discussed in [38], thus making more optimized decisions in the multi-hop assignment policy.

REFERENCES

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

[2] J. M. Anderson, K. Nidhi, K. D. Stanley, P. Sorensen, C. Samaras, and O. A. Oluwatola, *Autonomous vehicle technology: A guide for policymakers*. Rand Corporation, 2014.

[3] P.-L. Blyth, M. N. Mladenovic, B. A. Nardi, H. R. Ekbia, and N. M. Su, "Expanding the design horizon for self-driving vehicles: Distributing benefits and burdens," *IEEE Technology and Society Magazine*, vol. 35, no. 3, pp. 44–49, 2016.

[4] Kearney, AT, "How automakers can survive the self-driving era," *Retrieved November*, vol. 14, p. 2017, 2016.

[5] F. Drews and D. Luxen, "Multi-hop ride sharing," in *Sixth annual symposium on combinatorial search*, 2013.

[6] T. Teubner and C. M. Flath, "The economics of multi-hop ride sharing," *Business & Information Systems Engineering*, vol. 57, no. 5, pp. 311–324, 2015.

[7] Y. Chen, D. Guo, M. Xu, G. Tang, T. Zhou, and B. Ren, "Pptaxi: Non-stop package delivery via multi-hop ridesharing," *IEEE Transactions on Mobile Computing*, 2019.

[8] A. Hawkins, *UberHop is Uber's latest idea for killing mass transit*, https://www.theverge.com/2015/12/8/9873544/uber-hop-commute-mass-transit-seattle-chicago, 2015.

[9] B. Coltin and M. Veloso, "Ridesharing with passenger transfers," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2014, pp. 3278–3283.

[10] M. Zhu, X.-Y. Liu, F. Tang, M. Qiu, R. Shen, W. W. Shu, and M.-Y. Wu, "Public vehicles for future urban transportation.," *IEEE Trans. Intelligent Transportation Systems*, vol. 17, no. 12, pp. 3344–3353, 2016.

[11] N. Agatz, A. Erera, M. Savelsbergh, and X. Wang, "Optimization for dynamic ride-sharing: A review," *European Journal of Operational Research*, vol. 223, no. 2, pp. 295–303, 2012.

[12] B. N. Greenwood and S. Wattal, "Show me the way to go home: An empirical investigation of ride-sharing and alcohol related motor vehicle fatalities.," *MIS quarterly*, vol. 41, no. 1, pp. 163–187, 2017.

[13] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.

[14] K. M. H. Sonet, M. M. Rahman, S. R. Mehedy, and R. M. Rahman, "Shary: A dynamic ridesharing and carpooling solution using advanced optimised algorithm," *International Journal of Knowledge Engineering and Data Mining*, vol. 6, no. 1, pp. 1–31, 2019.

[15] H. Zheng and J. Wu, "Online to offline business: Urban taxi dispatching with passenger-driver matching stability," in *Distributed Computing Systems (ICDCS), 2017 IEEE 37th International Conference on*, IEEE, 2017, pp. 816–825.

[16] R. Zhang and M. Pavone, "Control of robotic mobility-on-demand systems: A queueing-theoretical perspective," *The International Journal of Robotics Research*, vol. 35, no. 1-3, pp. 186–203, 2016.

[17] S. Ma, Y. Zheng, and O. Wolfson, "Real-time city-scale taxi ridesharing," *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 7, pp. 1782–1795, 2015.

[18] R. Gopalakrishnan, K. Mukherjee, and T. Tulabandhula, "The costs and benefits of sharing: Sequential individual rationality and sequential fairness," *arXiv preprint arXiv:1607.07306*, 2016.

[19] X. Bei and S. Zhang, "Algorithms for trip-vehicle assignment in ride-sharing," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[20] N. Masoud and R. Jayakrishnan, "A decomposition algorithm to solve the multi-hop peer-to-peer ride-matching problem," *Transportation Research Part B: Methodological*, vol. 99, pp. 1–29, 2017.

[21] B. Coltin, "Multi-agent pickup and delivery planning with transfers," 2014.

[22] B. J. Coltin and M. Veloso, "Towards ridesharing with passenger transfers," in *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, International Foundation for Autonomous Agents and Multiagent Systems, 2013, pp. 1299–1300.

[23] A. Jauhri, B. Foo, J. Berclaz, C. C. Hu, R. Grzeszczuk, V. Parameswaran, and J. P. Shen, "Space-time graph modeling of ride requests based on real-world data," in *Workshops at the Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[24] V. M. de Lira, V. C. Times, C. Renso, and S. Rinzivillo, "Comewithme: An activity-oriented carpooling approach," in *2015 IEEE 18th International Conference on Intelligent Transportation Systems*, IEEE, 2015, pp. 2574–2579.

[25] J. Yang, P. Jaillet, and H. Mahmassani, "Real-time multivehicle truckload pickup and delivery problems," *Transportation Science*, vol. 38, no. 2, pp. 135–148, 2004.

[26] D. Bertsimas, P. Jaillet, and S. Martin, "Online vehicle routing: The edge of optimization in large-scale applications," *Operations Research*, 2019.

[27] K. T. Seow, N. H. Dang, and D.-H. Lee, "A collaborative multiagent taxi-dispatch system," *IEEE Transactions on Automation Science and Engineering*, vol. 7, no. 3, pp. 607–616, 2010.

[28] F. Miao, S. Han, S. Lin, J. A. Stankovic, D. Zhang, S. Munir, H. Huang, T. He, and G. J. Pappas, "Taxi dispatch with real-time sensing data in metropolitan areas: A receding horizon control approach," *IEEE Transactions on Automation Science and Engineering*, vol. 13, no. 2, pp. 463–478, 2016.

[29] W. Zhang, R. He, Q. Xiao, and C. Ma, "Taxi carpooling model and carpooling effects simulation," *International Journal of Simulation and Process Modelling*, vol. 12, no. 3-4, pp. 338–346, 2017.

[30] T. Oda and C. Joe-Wong, "Movi: A model-free approach to dynamic fleet management," in *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*, IEEE, 2018, pp. 2708–2716.

[31] T. Oda and Y. Tachibana, "Distributed fleet control with maximum entropy deep reinforcement learning," in *NIPS 2018 Workshop MLITS*, 2018.

[32] A. Alabbasi, A. Ghosh, and V. Aggarwal, "DeepPool: Distributed model-free algorithm for ride-sharing using deep reinforcement learning," *IEEE Trans. Intelligent Transportation Systems (to appear)*, 2019.

[33] D. C. Wyld, M. A. Jones, and J. W. Totten, "Where is my suitcase? rfid and airline customer service," *Marketing Intelligence & Planning*, vol. 23, no. 4, pp. 382–394, 2005.

[34] H. van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," *CoRR*, vol. abs/1509.06461, 2015. arXiv: 1509.06461. [Online]. Available: http://arxiv.org/abs/1509.06461.

[35] A. Singh, *MHRS Code*, https://github.rcac.purdue.edu/Clan-labs/MultiHopRideSharing, 2019.

[36] NYC, *NYC Taxi & Limousine Commission-Trip Record Data- NYC.gov*, www.nyc.gov, 2018.

[37] Z.-j. Ding, Z. Dai, X. Chen, and R. Jiang, "Simulating on-demand ride services in a manhattan-like urban network considering traffic dynamics," *Physica A: Statistical Mechanics and its Applications*, Nov. 2019. DOI: 10.1016/j.physa.2019.123621.

[38] X. Qian, W. Zhang, S. V. Ukkusuri, and C. Yang, "Optimal assignment and incentive design in the taxi group ride problem," *Transportation Research Part B: Methodological*, vol. 103, pp. 208–226, 2017, Green Urban Transportation, ISSN: 0191-2615. DOI: https://doi.org/10.1016/j.trb.2017.03.001. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0191261516306166.