Multi-agent Reinforcement Learning for Fleet Management: A Survey

Haoyang Chen[1, *, †]
[1]Computer science and Engineering. Southeast University
Nanjing, China
[*]hy_chen@seu.edu.cn

Zhuoming Li[2, *, †]
[2]Computer science and Engineering. Southeast University
Nanjing, China
[*]leezhuoming@seu.edu.cn

Yuxin Yao[3, †]
[3]Department of Computer Science. University College London
London, UK
yuxin.yao.19@ucl.ac.uk

[†]These authors contributed equally.

***Abstract*—Fleet management has achieved great success benefiting from the application of deep reinforcement learning (DRL) in recent years and has yielded many successful commercial applications like ride-hailing services, whose basic goal is to efficiently manage the fleet of vehicles to meet the demand separated temporally and spatially. However, research that provides insight about how existing methods succeeded in dealing with massive agent interactions from a multi-agent perspective is still missing. In this paper, we review the RL methods of order dispatching and vehicle re-positioning in recent years, and classify them from the perspective of multi-agent reinforcement learning (MARL). We provide a comparison of vehicle-based methods, grid-based methods, and order-based methods, along with the popular datasets and open simulators. Afterward, we discuss several challenges and opportunities for the application of DRL in this domain.**

Keywords- multi-agent reinforcement learning; fleet management; ride sharing; order dispatching; vehicle re-positioning;

I. INTRODUCTION

As a well-investigated programming problem, fleet management aims to manage the fleet of vehicles to meet the customers' requests distributed temporally and spatially, which has shown a wide range of application prospects because it involves different types of vehicles such as commercial delivery vehicles, taxis, locomotives, truck tractors. Many of them operate in a demand-responsive mode, i.e., the demands for services are not known beforehand and the fleet has to be deployed and managed in real-time [1], which brings great challenges to fleet management.

According to the previous surveys [1, 2], the modeling and algorithm around order dispatching, vehicle routing, and re-positioning, are the main topics in fleet management. Order dispatching refers to matching the customers' requests to available vehicles. Vehicle routing, also known as vehicle routing problem (VRP), focuses on the route programming between targets under given constraints. Vehicle re-positioning is to guide idle vehicles to specific locations in anticipation of fulfilling more requests in the future [3].

Benefiting from the rapid development of artificial intelligence, Deep Reinforcement Learning (DRL) is gradually playing a more and more important role in Intelligent Transportation Systems (ITS) and has been proved to be essential in solving the problems of fleet management [2, 3]. Since the decision-making during the fleet management process is sequential and relies on the state and time, it could be modeled as a Markov Decision Process, which could be solved by reinforcement learning algorithms [3]. To achieve efficient vehicle allocation, order dispatching and vehicle re-positioning could be improved and optimized with different deep reinforcement learning algorithms. Most papers implied and developed model-free algorithms including DQN, PPO, REINFORCE, while some model-based algorithms are employed as well.

A multi-agent system is a group of autonomous, interacting entities sharing a common environment, which they perceive with sensors and upon which they act with actuators [4]. Since the fleet management system can be typically modeled as a multi-agent system, multi-agent reinforcement learning (MARL), which provides a more natural perspective to deal with the distributed agents, is playing an increasingly significant role in this field. This method is based on DRL but focuses on its identical challenges, e.g., the formal statement of the multi-agent system's learning goal [4].

This survey provides a comprehensive analysis on how reinforcement learning helps address the problem of fleet management. We want to focus especially on the MARL aspect, as well as mentioning several innovative and instructive ideas based on MARL. Chapter 2 mainly focus on the basic concepts and advanced algorithms in reinforcement learning. The core of this survey lies in Chapter 3 and 4. First we break the problem of fleet management into three sub-

problems --- order dispatching, vehicle routing and vehicle re-positioning. In each section of the sub-problems, we categorize the investigated methods into three groups, the grid based algorithms and vehicle based algorithms as well as other special methods. Non-deep reinforcement learning methods already provided some sufficiently efficient solution for vehicle routing problem, resulting few DRL methods are applied on vehicle routing. Thus in this paper we only focus on the other two problems.

Due to the underdevelopment of multi-agent reinforcement learning in the past years, previous surveys mostly focus on the single-agent view, thus not paying enough attention to multi-agent methods. In the section of MARL of fleet management, we further classify the methods into three categories according to their selection of agents, which are grid-based methods, vehicle-based methods and other special methods at last. For each of the methods included, we analyze their performance and their unique metrics. In the final chapter of our survey, we provide an insightful analysis of currently confronted challenges as well as possible development directions of this field.

II. Reinforcement Learning

The basic concepts and advanced algorithms applied in fleet management system using reinforcement learning are briefly explained below. This paper focus on multi-agent solutions, which will also be introduced.

A. Single-agent RL

Due to the feature that the decisions made by a fleet management system are sequential and they depend on real-time environment situations and time, the Markov decision process could be applied to solve the problem [3].

A Markov decision process could be represented as a tuple $< S, A, P, R >$ where S represents for the set containing possible states of the environment, A represent for the set of possible actions of agents, P represent for the transition probability function, the R represents for the reward function [4]. The agent takes the current environment state $s_t \in S$ at step t and then decide about the next action $a_t \in A$, which lead to a transition from s_t to $s_t + 1$ in step $t + 1$ because the agent performing action interact with the environment. The reward r_t is produced by reward function after agent takes action at [5]. The action chosen relies on the policy function $\pi(s)$: $S \rightarrow A$. The sequential process ends when it achieves a terminal state. The agent is trained with different algorithms to achieve the maximum cumulative reward of the process [3].

The state-value function V_π indicates the expected cumulative reward starting at a specific state under policy π:

$$V_\pi(s) = \mathbb{E}_\pi\left(\sum_{t=0}^{T} R_t \mid s_0 = s\right) \quad (1)$$

where $R_t = \gamma^t r_t$ and γ is the discount factor, which indicates importance of the future reward.

The action-value function Q_π, similarly, indicates the expected cumulative reward starting at state s, taking action a under policy π [3, 6]:

$$Q_\pi = \mathbb{E}_\pi\left(\sum_{t=0}^{t=T} R_t \mid s_0 = s, a_0 = a\right) \quad (2)$$

The Bellman equations of value function are shown below, which indicates the relationship between the value of the current state and the value of its future states [3].

$$V(s_t) = \sum_{a_t} P(s_{t+1}, r_t)(r_t(s_t, a_t) + \gamma V(s_{t+1})) \quad (3)$$

With the model-based method, the decisions of action rely on predicting future states. With the estimated transition function and reward function, the action that could result in maximum value is computed by iterating the Bellman value equation [5]. Conversely, model-free methods cannot predict how the action they take changes the environment state but interacts with the environment [6]. However, they are more useful under complex environments where the reliable transition function and reward function are hard to estimate [2]. The policy and value function are learned from data. Thus, the model-free methods are more commonly used in fleet management.

Value-based methods and policy-based methods are the two categories of model-free methods. In value-based algorithms, the value of a given state at step t is estimated. The Q-learning is applied to learn the optimal actionvalue function, where the action-value function is iteratively updates by [3]:

$$Q(s, a) \leftarrow Q(s, a) + \alpha\big(r + \gamma max_{a'} Q(s', a') - Q(s, a)\big) \quad (4)$$

Whereas for the policy-based algorithms, the state-value pair is not required to be estimated, but they search for an optimal policy. A parameterized policy is chosen at first, and then the parameters in the policy are updated to maximize the value function [5].

B. Multi-agent RL

However, the single-agent may face difficulties during working with the fleet management problems due to the nature of multiple drivers or grid cooperating to achieve the same goal. Thus, employing multiple agents for decision-making is a reasonable choice. The Markov decision process is generalized to the Markov game involving multi-agents. Markov game could be described as a tuple $< S, A_1, \ldots, A_N, P, R_1, \ldots R_N >$, where S is the finite set of the environment state, N is the number of agents, A_i is the finite set of the action space of agent i, yielding $\mathcal{A} = \mathcal{A}_1 \times \ldots \times \mathcal{A}_\mathcal{N}$ is the joint action space, $f: \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0,1]$ denotes the state function of transition probability, $R_i: \mathcal{S} \times \mathcal{A}_i \rightarrow R$ denotes the reward function of agent i. The policy of agent i is defined as $\pi_i : \mathcal{S} \times \mathcal{A}_i \rightarrow [0,1]$, yielding $\pi : \mathcal{S} \times \mathcal{A} \rightarrow [0, 1]$ as the joint policy. According to the definition, the transition of state is decided by the joint action of all the agent, therefore the expected return $R_i^\pi(s)$ for agent i depends on the joint policy π,

$$R_i^{\pi}(s) = \mathbb{E}\left\{\sum_{t=0}^{T} \gamma^t r_{i,t+1}\right\} \quad (5)$$

where the $r_{i,t+1}$ is the reward of agent i in timestep t. The value function and the state-action function of each agent under joint policy π, initial state s and joint action $a = [a^1, \dots a^N]$ could be evaluated with algorithms like DQL, in which the best performing policies of each agent could be discovered.

III. Reinforcement Learning for Fleet Management

A. Order Dispatching

In this section, proposed methods for order dispatching will be discussed from a MARL perspective. We will firstly give a problem statement of order dispatching in fleet management. The necessary mathematical formulation of the problem will be included. Then, the main challenges to the application of MARL in order dispatching will be discussed. After all, the proposed methods for order dispatching will be discussed from the perspective of MARL. These methods will be mainly organized by the different choices of agents.

The Order dispatching module in fleet management, which is also learned as the online matching [3], is to best assign customers' requests to the available drivers in real-time. The geographical position of the available vehicles, the starting position, the destination of the unmatched orders, as well as their price given by the pricing module are provided as the input. Dispatching plans of how and when to match the orders and vehicles are expected to be given as the output.

What is a "best" order dispatching ideally? Many metrics have been proposed to help form an optimization target.

The metrics to be optimized can be roughly classified into short-term metrics and long-term metrics. The short-term metrics, e.g., minimizing the waiting time of the customer, maximizing the immediate income from the current order, can be optimized directly according to the distribution of orders immediately. The long-term metrics, typically the expected accumulated income of the rest of the day, is also related to the current decision, but cannot be learned immediately.

Another classification of these metrics can be driver-centric and customer-centric. From the driver's perspective, the ultimate goal is to maximize its income, or the in-service time, the order response rate as well. From the customer's perspective, the grade of service should be guaranteed by the platform, i.e., the waiting time should be minimized.

These critics are not exclusive [7], e.g., reducing pickup waiting time also decreases the cancellation probability of an order. [8, 9] prefer maximizing Gross Merchandise Volume as the learning goal, while [10] prefers maximizing the accumulated driver income and order response rate. Since the orders will be canceled after a period of waiting, [11] considers the metric of maximizing the acceptance rate of orders as more prominent than the metric of minimizing the waiting time.

The mechanism of how the orders are delivered to the drivers is also various. The simplest mechanism assumes that one vehicle is matched to one order, and the order will not be re-assigned to the others unless the driver refuses it, which is named as a one-to-one dispatching pattern in [11]. Another mechanism is that one order will be dispatched to many drivers, and the first driver who accepts it gets the order, viz. The driver can select one order to accept from a recommended order list provided by the platform, which is named as a one-to-many pattern [12]. The most complicated mechanism is that, based on a one-to-many pattern, the driver can accept multiple orders, and complete them one by one, which is named as a many-to-many pattern [11]. Most researchers take the one-to-one mechanism as the model assumption. [11] implies that the additional complexity of mechanism assumption requires a more complicated design, which will be discussed in the later section.

1) Multi-agent reinforcement learning challenges

In order dispatching, the state $s \in \mathcal{S}$ can be defined as $< D_0, D_v, t >$, where D_0 is the distribution of orders, D_v is the distribution of vehicles, *t* is the current timestep. Commonly, the map is discretized into hexagon girds, therefore the location of order or vehicle is expressed by the index of gird [13]. [14] applies a link-node-based micro-network representation, leveraging the heterogeneous traffic network topology. Besides, it's necessary to append the timestep *t* into the definition since the distribution of orders or vehicles may strongly have a temporal characteristic, e.g., morning rush hour and evening rush hour. Therefore, the characteristic of the Markov Decision Process (MDP) can be guaranteed.

$$P(\mathcal{S}_{t+1} = s_{t+1} | \mathcal{S}_t = s_t, \dots, \mathcal{S}_0 = s_0) = P(\mathcal{S}_{t+1} = s_{t+1} | \mathcal{S}_t = s_t) \quad (6)$$

The game in order dispatching is likely to be a fully cooperative game, where the platform aims to maximize the return of every agent without competition. Therefore, the reward R^{π} of the system can be expressed as the sum of the reward of every agent R_i:

$$R^{\pi}(s) = \sum_{i=1}^{n} R_i^{\pi}(s) \quad (7)$$

In ride-hailing service or ride-sharing service, orders and vehicles can be generally viewed as homogeneous. Prior knowledge of order dispatching is quite abundant, e.g., the driver should pick a nearby order, therefore can be utilized to help the learning. Coordination between agents in order dispatching is quite necessary, which enables the improvement of global optimal dispatching based on cooperation. Thus, the agent should be aware of other agents and estimate their policy. The input of each agent can be global, i.e., each agent can observe the whole part of state $s \in \mathcal{S}$.

However, there are some challenges in MARL [4]. First, the curse of dimensionality, i.e., the exponential growth of the discrete state-action, brings not only heavy computational expense but obstacles to fit the prediction of state-action space from historical data of limited number. Second, nonstationarity arises because the best policy of each agent is changing resulting from the activation of other agents. Third, it's hard to specify a unified learning goal for the MARL

system, because the agents' returns are correlated and cannot be maximized independently. Forth, the coordination between agents is necessary, e.g., the action space of the agent depends on the action of other agents. We are interested in how these difficulties are solved or are avoided in the proposed method. While applying MARL into order dispatching, these challenges above become more specific and background-related. We will mainly discuss how the MARL environment is modeled, i.e., what the agent is on behalf of and what the action means since it is decisive to the characteristic of the state-action space. Therefore, proposed methods will be classified and be discussed in the classification of agent modeling type.

2) Vehicle-agent

Modeling the vehicle as the agent can be intuitive. A typical method for order dispatching is the learning and planning approach [8, 9, 15]. The offline learning step performs Temporal-Difference (TD) update to learn the evaluation of the spatiotemporal state, while the online planning step uses these values to compute a bi-partite graph matching problem. It makes a simplification and takes the single-agent perspective to solve this problem, viz. The state $s =< g_v, t >$ contains the region index g_v where the agent is located in and the timestep t. Correspondingly, the action $a = < g_{src}, g_{dest} >$ represents an order which starts in g_{src} region, and ends in g_{dest} region. Therefore, the state s which performs the action a will transform to the next state $s' =< g_{dest}, t + \Delta t >$, while Δt is the expected time cost for transportation from g_v to g_{src}, then to g_{dest}.

To be more detailed, in the planning step, a dispatching time window, where unmatched orders and available drivers are pooled and matched simultaneously, is created. In the dispatching window, suppose there are n orders and m available drivers. Orders and drivers can be formulated as a bipartite graph, and the edge w_{ij}, representing a potential matching, is evaluated as a reward for assigning order i to driver j. The evaluation of reward is a state-action value given by the Q-network trained in the learning step. Therefore, the dispatching policy is modeled as a Combinatorial Optimization problem and can be solved by some classic matching algorithm, e.g., the Hungarian Method (a.k.a. KM algorithm). In the learning step, the matching is modeled as an MDP, and single-agent reinforcement learning is applied to acquire the evaluation of state-action value. All the agents share the same policy. The application of Deep Reinforcement Learning (DRL) enables the model to generalize beyond the historical training data, as well as to leverage knowledge transfer across multiple cities (Transfer Learning) [9, 15]. Batching more orders and drivers enables a more global optimization, but results in an expense of longer order response time.

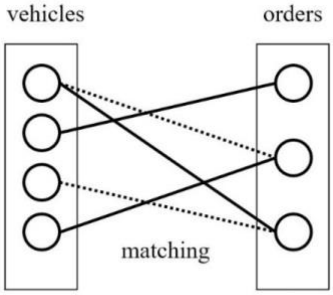

Figure 1. bipartite graph modeling for order dispatching

One of the challenges to the learning and planning approach is that it cannot reflect the fluctuation of supply and demand in real-time. The estimation of spatiotemporal state-action value is based on historical data, without the involvement of the current transition of environment. This challenge can be reduced by redesigning the structure of the Q-network [15, 16].

What makes the single-agent reinforcement learning practical in the multi-agent environment is that the learning and planning approach extracts the necessary interaction between vehicles into the planning procedure, leaving the learning procedure undifferentiated to every agent. It makes full use of the homogeneity of agents and simplification is made. The curse of dimensionality is avoided since the finite set of state $\mathcal{S}$ and action $\mathcal{A}$ have limited size. Nonstationarity in training is also avoided, as the policy of the agent is independent. The dispatching policy in combinatorial optimization is called collectively greedy [9]:

$$\underset{a \in \mathcal{A}}{\operatorname{argmax}} \sum_{s \in \mathcal{S}} Q(s, a(s)) \tag{8}$$

where $Q\big(s, a(s)\big)$ denotes a feasible edge in the bipartite graph. It provides an optimization objective to the system, and the constraints of "one order can only be assigned to one driver", etc., consist of the coordination between agents.

However, the learning and planning approach does not produce many advantages of MARL, like the inherent robustness and parallel computational efficiency which is based on the distributed nature, because planning procedure highly relies on centralized control. [17] also models each vehicle as the agent, but takes the complex interactions between drivers and orders into consideration by applying the Mean Filed MARL method. Instead of giving centralized control to all the vehicles and letting them share the same policy, it assumes that each agent makes decisions independently. By comparison, the centralization of the learning and planning approach has the potential "single point of failure" [18], i.e., the failure of the centralized authority control will fail the whole system. Also, a heterogeneous agent setting with individual-specific features is supported.

The design of state and action space is similar to the learning and planning approach, but jointly in the multi-agent environment, viz. The state is denoted as $s =< g_{v,1}, \dots, g_{v_n}, t >$, while the action is denoted as $a =<$

$a_1, \ldots, a_n >$. The reward of each agent is, therefore, dependant on the joint action of all the agents. The nonstationarity in training arises. For each agent, the training of its state-action value is hard to converge because the policy of other agents is also changing in this procedure. [17] adopts the mean field approximation to simplify the local interactions by taking an average action among neighbourhoods [19]. The learning target of each agent is to maximize its cumulative reward, instead of the global reward, while the demand-supply gap is defined as a constraint. The mean-field method lessens the curse of dimensionality, making the additional cost for the growth of the number of agents acceptable.

Since the fully distributed order-driver matching decisions are made independently without dispatching time window, the matching of order is performed asynchronously to avoid matching collision, viz. The matching is competitive and the first driver who sends the matching request to the platform will get the order dispatched. It also acts as the coordination between agents at the minimum level. The matching collision is intrinsically the problem that dynamic action space is dependent on the behaviour of other agents.

3) Grid-agent

[20, 21] prefer to model the discrete region grid as the agent. They regard the grid as the manager of vehicles and unify the order dispatching problem and re-positioning problem into the same form, i.e., matching the orders and vehicles inside the grid, and distributing the redundant vehicles to the neighbour girds. Specifically, re-positioning the vehicle to neighbour grids or staying at the current grid is treated as fake orders, which is a trick to unify the two tasks in form. Self-organization techniques are implemented to decrease the waiting time as well as to enhance the utilization rate of vehicles. It models the order dispatching as a large-scale parallel ranking problem, instead of a sequential decision-making problem.

A fundamental denotes of the state can be expressed as $s =< N_v, N_o >$, where inner elements represent the number of available vehicles, number of unmatched orders.

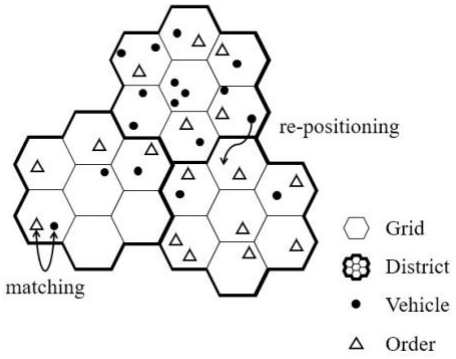

Figure 2. dispatching and re-positioning in gird-agent modeling environment

[21] applies a learning-and-planning-like method. [20], however, integrates a geographical hierarchy reinforcement learning (HRL) to decompose the dispatching in sub-tasks. District of grids is modelled as the manager agent, raising and delivering sub-tasks to the worker agents, i.e., small gird. The spatiotemporal balance of supply and demand corresponds to the tasks of higher level, while the matching of orders and vehicles corresponds to the lower. The manager's action is to generate goals for its workers, while the worker's action is to provide a ranking list of relevant real orders [20].

4) Order-agent

Suppose the platform runs in the mechanism that recommends the most suitable orders to the driver and allows the driver to choose from some selections, the dispatching model based on one-to-one matching assumption cannot work well. Taking the rejection behaviour of drivers into consideration helps to adapt the demand of one-to-many or many-to-many dispatching pattern, i.e., an order is dispatched to many vehicles [11]. The idea of order-agent, i.e., modelling the order as the agent, comes from considering the driver's rejection behaviour. It provides another perspective to see the dispatching problem.

In [11], the order is defined as the agent, while selecting the matching vehicles is defined as the action. [11] proposed a learning and planning solution, which is quite similar to the mentioned approach before. It also has an offline learning step to learn the state-action value, and an online planning step to give centralized dispatching decisions to all the orders. In the learning step, historical data is used to train for predicting the expected response time in a one-to-one dispatching situation, i.e., how the response time will change if the order is dispatched to different vehicles. Reinforcement learning is applied to solve such a response process. In the planning step, the probability distribution of response time is involved in the reward function. Orders and vehicles are treated as nodes in bipartite graph, and combinatorial optimization is also utilized to maximize the weighted matching, which is discussed in the last section. [22] also defines the order as the agent and takes a learning-and-planning-like solution under the one-to-one matching assumption. However, it defines the action as whether to enter the matching pool or to be delayed to the next turn, expecting a nearer driver will be available in the next timestep.

B. Vehicle Re-positioning

In this section, we are focusing on reinforcement learning's application in the problem of vehicle re-positioning which is also a sub-problem of fleet management [23]. In the first part of this section, we present the problem statement of vehicle re-positioning. Next, we put our attention on one of the most important aspects of vehicle re-positioning which is demand/supply predicting, we introduce some of the advanced methodologies developed to address this problem. In the third part, we are going to stress the use of multi-agent reinforcement learning to help solve the problem of vehicle re-positioning, by categorizing the different research papers into two distinct classes according to their two main choices of bases of their reinforcement learning agents, we offer a thorough insight of their pros and cons, state and action space formulation, relationships between agents, etc. respectively. Based on the two categories mentioned above, we are also

introducing some of the basic algorithms designed for this problem as well as their improvements. At last, we mention several innovative ideas which are different from the popular methods and are considered to have offered promising directions for future research.

In a fleet management system, vehicle re-positioning is often considered in parallel with order dispatching. Most literatures develop two different systems at the same level for order dispatching and vehicle re-positioning respectively. Specifically, after the execution of order dispatching, the vehicle re-positioning system operates on every idle vehicle that was not assigned an order and assign them a location to go before they receive an order. To be effective, vehicle re-positioning systems generate the location according to predicted future demand/supply gap, such that the assigned vehicle plays the role of taking some pressure off the exceeding supply of zone it was leaving as well as making up the insufficient supply of its destination zone.

1) Prediction

In order for the vehicle re-positioning system to act effectively as mentioned above, literatures use real-world collected data to train their predictor. These datasets usually include the specific time and location of every order, the trajectory of different drivers as well as their status of occupied or not. Since the actions of dispatching and re-positioning will change the environment from what it is supposed to be from the data sets, at most time researchers needs to generate new data based on the given real-world records.

However, since the reinforcement learning methods in vehicle re-positioning are all making use of MDP and its property that given the present, the future does not depend on the past.

$$P(S_{t+1} = s_{t+1} | S_t = s_t, \dots, S_0 = s_0) = P(S_{t+1} = s_{t+1} | S_t = s_t) \quad (9)$$

So that instead of building a separate system of predicting the future demand and supply information, literatures usually let the reinforcement learning system of re-positioning to handle the predicting job itself. As when the reinforcement learning algorithm is learning how to make appropriate decisions given current states, it is also learning the Markov Property of the environment, which is to say that it makes decisions based on the given current state S and a learned state transition probability P. This approach can be classified as agent learns a model of the environment dynamic. On the other side, researchers also proposed a different way of building a data generator outside the agent [24, 25, 26, 27, 28].

Among these proposed approaches, [24, 26, 27] used convolutional neural networks (CNN) to generate the demand information in a spatio-temporal manner. [25] made use of the recurrent neural networks (RNN) to help predict future demand in a sequential manner. [28] utilized the recently developed graph neural networks (GNN) which uses neighbour information to better formulate demand in a contextual way. Using a model predictor outside the reinforcement learning algorithm can help our agent reduce its massive state space thus improving its scalability, as usual, learning algorithms learn to re-position vehicles by acquiring not only demand/supply information but also weather, time and other complex external factors [28]. However, by using an extra predictor, we can contract external factors into a small state space and predict the demand information as well.

2) MARL algorithms

Recent researches on vehicle re-positioning are showing a trend of division on the choices of agents. In consistent with the traditional methods of re-positioning vehicles without the help of reinforcement learning, many research papers show a way of abstracting vehicles as reinforcement learning agents which are often considered homogeneous [28, 29, 30, 31]. While there are also many researchers who choose to represent the demand and supply in a grid-based manner, which is to segment the region used to conduct the experiment in a square or hexagonal way. In the grid system, each grid is an agent and is treated as heterogeneous. [20, 21, 32] The prime distinction between the two approaches of modelling agents is the difference in the executor of the actions. In a grid-based algorithm, the action is often indicating how many vehicles the grid needs to re-position. While in a vehicle-based algorithm, the action indicates the location the vehicle needs to be re-positioned.

3) Grid-based algorithms

As mentioned above, literatures use a demand predictor to assist the agent in making decisions, since many researches abstract the agents into grids, they simplify the prediction module into a gird-based prediction. The most obvious advantage of grid-based algorithms against vehicle-based algorithms is that its low computational complexity. The number of grid agents that a city usually is segmented is about 100 ~ 400 which is much smaller than that of vehicle-based algorithms. Moreover, according to the grids' location as well as their functions, the grids can be classified into several classes and thus have their unique neural network to maintain. In this manner, the grid-based algorithms treat the grid agents as heterogeneous as different grids have their own geographical features and behavioural patterns.

The state space, at most time, is all the demand and supply information on the map of each grid.

$$S_{\mathrm{i}} = \langle i, D_0, \dots, D_n, S_0, \dots, S_n \rangle \quad (10)$$

This is the simplest but most often used representation of state space of grid-based algorithms, indicating that in a grid system containing n grids, the state of grid i is made up of the id of current grid, the number of orders that has not been dispatched denoted as demand, as well as the number of idle vehicles denoted as supply from $grid\_0$ to $grid\_n$ [32]. This design of state space is sufficient in information needed by reinforcement learning algorithms to generate dispatch actions. However, a vector consisting $(2n + 1)$ elements can sometimes be too large for normal reinforcement learning algorithms, such a large state space can affect the scalability of the algorithm which is a crucial factor in multi-agent systems. To solve the problem of scalability while not compromising the efficiency of the algorithm, [28] proposed

a contextual-based state representation. In a square-based grid system, the state space for $grid\_i$ is designed as follows.

$$S_i = \langle i, D_i, S_i, D_{n\_1}, S_{n\_1}, \dots, D_{n\_8}, S_{n\_8} \rangle \quad (11)$$

The State representation in this setting is composed by the grid id i, the number of demand and supply for its neighbour grids ranging from 1 to 8 denoted as $grid\_n$. They proposed that the information provided by its neighbour grids for the current grid to make re-position actions is sufficient enough in a contextual manner. Since in literatures a re-position action usually moves a taxi to the neighbour grids of its location, after using a GNN which will pass the neighbour grids features into the current grid's when doing prediction on the demand, the contextual way of giving only neighbour information performed well in experiments.

One possible design of the action space is a single ratio, indicating how much vehicle of the current grid agent needed to be re-positioned or to be added [21]. However, this design of the action space only provides a rough description of the amount of re-positioning while ignoring the direction. Thus, this design of one single ratio need to operate in cooperation with an extra matching function, which combine the *Source Grids* that re-position out of itself with the *Sink Grids* that needs extra vehicles together using a bipartite graph combinatorial optimization problem. This seems to be a bit complex and unnecessary compared with the more commonly used design. The action space, could also be designed similar to the state space. As mentioned above, a grid agent can only move its vehicles to neighbour grids. So, if it is in a square grid system, the action should be a vector of nine elements which represent the grid itself and its eight neighbours. This design outputs a vector of nine elements as follows.

$$a_i = \langle r_0, r_1, r_2, r_3, r_4, r_5, r_6, r_7, r_8 \rangle \quad (12)$$

Where each element indicates a ratio of the number of vehicles in current grid and by numbering the nine grids, we set r_4 to be the ratio of vehicles that needs to stay in their current grid and the other elements suggest the amount of vehicles to be moved to neighbour grids.

The reward in a vehicle re-positioning system should reflect the efficiency of the action in balancing demand/supply the gap. For a grid system, an agent grid's goal is set to balance its own demand and supply to the utmost extent. Thus, its own information of demand and supply can be used to formulate the reward function without using any other information of the others [21].

$$r_t^i = 1 - \frac{\left| Num_d^i - Num_s^i \right|}{\max\{Num_d^i, Num_s^i\}} \quad (13)$$

The design of reward varies between papers and above is one possible approach which reflects the equilibrium degree of the agent $grid\_i$. To maximize the reward, the agent has to minimize the gap between demand and supply while keeping as much vehicles and orders in its region.

The design and usage of a grid system to model agent in general is quite a simplification of the real-world dynamics. It reduces computational complexity tremendously and has high scalability due to its small number of agents and simple representation of vehicles and orders. However, it compensates many detail information in order to achieve its simplicity. For example, the grid system treats the supply information as a number of vehicles, which dropped the exact location of each vehicle aside. The location for each order is also blurred [32]. What's more, the agent's action only determines how much vehicles to re-position while not considering the difference between drivers which might affect their willingness to go or to stay. Beside the problem of not being detailed enough, the grid system for vehicle re-position provides a handy environment to test new algorithms.

4) vehicle-based algorithms

Another formulation of agents in solving vehicle re-position problems using multi-agent reinforcement learning is vehicle-based agents.[28, 29, 30, 31] The vehicle-based algorithms, differs a lot from the grid-based algorithms, instead of maintaining multiple neural networks for each agent as the grid-based algorithms, vehicle-based algorithms usually share the parameters between agents and thus maintaining only one or a small number of neural networks after the classification on the types of vehicles. Recent researches treat the vehicles as homogeneous and are not paying special attention to the classification of agents, which makes these algorithms different from the heterogeneous grid-based algorithms.

However, the vehicle-based algorithms still have some similarities with the grid-based algorithms in its basic elements. In most research papers using a vehicle-based system, a grid-like segmentation of the map is widely applied. This is for the convenience of predicting the demand information as well as the management of vehicle agents. In a vehicle-based algorithm, grids are used to store the specific information of orders and vehicles, they are also crucial factors when designing the state space for vehicle-based algorithms.

The design of the state space is quite similar to that of grid-based algorithms. Like a grid system, the optimization target is to balance the demand and supply between the grids, thus in most papers, the design of the state space is based on the grid information. The design used in [28] can still be used in a vehicle-based manner, which only take neighbour information into consideration. However, like what was considered a global view in grid-based algorithms, [30] proposed a global state space design.

$$S = \left(r_j, s_j', \lambda_j \text{ for all } j \in \bar{G} \right) \quad (14)$$

The above equation indicates that the state is represented by r_j which is the number of idle vehicles in $grid\_j$, the availability of in-service vehicles in $grid\_j$ which is denoted as s_j, and the predicted demand around the grid denoted as λ_j. This is slightly different from the global view in grid-based algorithms in that it has an element of $grid\_j$ which is taking into the consideration of the empirical probability of a vehicle be assigned to a new request based on simulation results. However, in real world data, the demand and supply gap

depend on spacial and temporal information, so researchers believe the agents should also learn to make decisions based on spatial and temporal information, not only on the amount of global demand or supply. [29] proposed a different representation of state space which combine the global state together with spatial temporal information together.

$$s_t^i = [s_t, g_t] \tag{15}$$

Where the state of $vehicle_i$ at time t is a concatenation of the global state s_t containing the spacial distribution of available vehicles and orders as well as current time t (using one-hot encoding) and the one-hot encoding of the grid ID g_t. After all, the state space should always provide the demand and supply information to our agent, while the spacial and temporal information are optional.

The action space of vehicle-based algorithms is totally different from that of grid-based algorithms because the entity of executing the action has changed. Since every vehicle is an independent agent, they can make their own decisions of which neighbouring grids to go or to stay. The most popular design of action space is merely a number indicating the destination grid [28, 29]. Where the action is extracted from the output of the neural network which is a probability vector noting the possibility of going to a certain neighbouring grid, the length of the vector is 9 for a square grid system and 6 for a hexagonal grid system. After extracting the action by applying a max operator on the vector in an exploitation manner, we can get the index of the grid for the vehicle to go to. Also, in some paper the action is formulated as certain directions like $\{None, NE, E, SE, S, SW, W, NW, N\}$, which can be treated exactly as the index manner of representation.

The reward design has to reflect the quality of the re-position action of current agent. Like in a grid-based algorithm, no matter what kind of agent the algorithm uses, the optimization goal is always set to balance the demand and supply of both the *Source Grid* and the *Sink Grid* [21]. One design of reward provided by [28] is to calculate the reward of a re-position action conducted at time $t-1$ a time step later, which only use demand and supply information of its leaving grid and destination grid at time $t-1$.

$$\omega_{z_i} = \frac{P_{z_i}^{t_{j-1}}}{D_{z_i}^{t_{j-1}}} \tag{16}$$

$$r_t = \begin{cases} 5 & 0 \le \omega_{z_i} \le 1 \text{ and } i = g \\ -5 & 0 \le \omega_{z_i} \le 1 \text{ and } i \ne g \\ \dfrac{1}{\omega_{z_i}} & \omega_{z_i} > 1 \text{ and } 0 \le \omega_{z_g} \le 1 \\ 0 & \omega_{z_i} > 1 \text{ and } \omega_{z_g} > 1 \text{ and } i = g \\ -\omega_{z_g} & \omega_{z_i} > 1 \text{ and } \omega_{z_g} > 1 \text{ and } i \ne g \end{cases} \tag{17}$$

The ω_{z_i} is a fraction of supply over demand at time step $j-1$ of $grid_{z_i}$, the following reward at time step t is determined by the ratio of supply/demand of the *Source Grid* and the *Sink Grid* and also taking consideration of the case that the vehicle decided to stay rather than re-position. Further, another design of reward function that does not depends on the *Source Grid* was proposed by [29]. In their paper, the individual reward r_t^i.

For the i-th agent associated with the action a_t^i is defined as the averaged revenue of all agents arriving at the same grid as the i-th agent at time $t+1$. This is an example of using vehicle revenue as the re-position rewards. Actually, the optimization goal of balancing demand and supply is proved to be in parallel with that of getting the maximum revenue for each agent.

Generally, vehicle-based algorithms are more often used than grid-based algorithms. Even if the number of vehicles is massive, with the application of parameter sharing, the thousands of agents can learn in a centralized manner with a decentralized execution. Different from a grid-based algorithm, a vehicle-based algorithm can pay special attention to the vehicle and order's location and route. It uses the same order predictor and the state space as the grid-based algorithms while having a much simpler action space. But, after a re-position action is generated, to calculate the reward for this action in a more specific way, a vehicle routing module may be needed to simulate the travel cost. In all, the vehicle-based algorithms are more complicated than a grid-based algorithm but it has more expanding space such as a heterogeneous way of designing agents and a more precise way to treat the interactions between agents.

5) Innovative Algorithm

Apart from the traditional multi-agent reinforcement learning way of solving vehicle re-positioning problem, there are two distinct and innovative methods. One is paying special attention to the interaction between agents and their influences on each other by applying mean-field reinforcement learning (MFRL) in a vehicle-based manner [17]. This technique of MFRL greatly reduces the state space from what used to consider all the other agents' joint action around the current one to a representation of their mean influence on the current one. Since it is not considering precisely the number of other agents or their features, the algorithm is providing a possible solution of treating agents heterogeneously. Another algorithm is based on the grid system. It applies the hierarchical multi-agent reinforcement learning to the hexagonal grid system where the grids are further specified as manager and worker [20]. The hierarchical reinforcement learning algorithms is proved to perform better at solving complex problems than other algorithms.

IV. Challenges and Opportunities

So far, Reinforcement Learning algorithms have been proved to perform well in help solving fleet management problems. Its capability of addressing complicated tasks in a real-world scenario assisted many startups gaining profit. Also, researchers are still paying efforts to make their algorithms able to handle emerging real-world problems. In this section, we discuss some of the problems to be solved in reinforcement learning's application in fleet management as well as providing some of the possible directions for solutions.

A. *Unifying dispatching and re-positioning*

While the problem of order dispatch and vehicle re-position have been both well studied, they are mostly conducted in a separated manner, which means, that most paper either propose an algorithm designed to solve specifically one of the two problems. However, order dispatching and vehicle re-positioning have a strong connection with each other.

When doing order-dispatching, we are not only matching the drivers with a suited order but also evaluating what impact such matching could have on the demand and supply gap of both the zone where the order generated as well as the order's destination. The order dispatching problem can be considered as a limited vehicle re-positioning problem in that the vehicle have limited choices of action where the action determines both the income for the driver and the destination to re-position. Unlike the normal re-position problem, order dispatching has a re-position impact that is highly depending on the available orders. Compared with the broader neighbor zones to choose in vehicle re-positioning, order dispatching's re-position is deeper which often consider zones away from the current zone.

From another aspect, when doing vehicle re-positioning, for a reinforcement learning algorithms to perform as expected, researchers have to design the rewards for every re-position action. Since every re-positioning action is assigned with a reward, this has the same property as the actions in order dispatching. If we represent the reward of an action in the same way as the income in order dispatching and transfer the possible zones to re-position as the destination for an order, then the re-positioning problem can be seen in an order dispatching way where the agent have to choose and order to pick up, thus achieving the same effect as vehicle re-positioning.

There are rarely papers focusing both the order dispatching and vehicle re-positioning problems. [28] proposed an algorithm for vehicle re-positioning but also integrated the order dispatching module into their simulator where vehicles are matched with the nearest order. They treated the two problems respectively rather than a unified manner. The idea of using fake order was first proposed by [20], where they unified the vehicle re-positioning with order dispatching and designed an order dispatching algorithm to solve the two problems.

Future research should focus on integrating the order dispatching and vehicle re-positioning in one general system. Finding a way to unify the two problems is promising considering the relationship is not to be neglected while solving two problems respectively is computationally expensive.

B. *Map representation*

Map representation acts as a necessary module in fleet management to embed the geographical position into a discrete state. The majority of existing methods follow a similar convention and use gird system to split the spatial world, with the shape of square [15, 33], or hexagon [10, 16, 17, 20, 34]. By selecting the neighbors of the grid, we can simply get a rather rational range of consideration in dealing with the interactions of agents. [15] supports multiple resolutions of hexagonal grids, which helps the information aggregation to happen at a different level. [33] uses cluster to divide the megacity into independent regions and focus on each of them separately, reducing the complexity.

However, a more reasonable representation of the map should consider the topological of the road. The coarse-grained grid representation cannot capture the real distance from place to place. To be worse, the complicated road in urban may deteriorate the planning and lead to infeasible operation strategies. [14] builds a link-node-based micro-network representation and have successfully applied MARL on it, which is prospective.

The previous learning on order dispatching and re-positioning put little concentration on the selection of the map representation method. In fact, map representation based on real-world transportation networks has been learned in vehicle routing problem (VRP) rather sufficiently [35, 36]. Further exploration on the integration of the map representation module with fleet management will be productive.

C. *Adaptability to emergencies*

According our survey, the learning and planning approach [8, 9, 15], and its variants [14, 21], have been widely used in fleet management. Based on historical data, the off-line learning step enables the model to predict the distribution of demand and supply, therefore capable of arranging the fleets to fill the gap in advance. The centralized online planning step, helps the platform to coordinate the actions of agents, optimize the reward of the whole system.

However, while researchers are trying to improve the performance of the algorithm, the risks of emergencies are ignored. The off-line training doesn't extract the causality of the transition of the supply and demand but directly learns the state-value on each timestep, viz. The prediction of state-value is more likely to be a replay of historical tendency and hardly consider the current situation. To deal with the challenge, [15, 16] redesigned the DRL network and take the current state as part of the input in the planning step. This shortage may be fatal when a large difference takes place between the real-world situation and historical situation, let alone the abrupt of the emergent incident. How to improve the robustness of the system from the real-time update part should be learnt in the future.

Another concern about the adaptability to the emergency of the system is the “single point of failure” [18]. [17] goes further on the design of the distributed system, which depends little on the central control.

Although [14] points that such non-cooperative learning may harm the system episode reward, the capability for the vehicle to continually provide a sub-optimal arrangement to itself when temporarily disconnected to the control center is of no doubt significant. This backup system may be of necessity where the connection is not reliable. Considering the recent development of Internet of Things (IoT) and the Internet of Vehicles (IoV), the distributed execution could be practical and prospective.

D. Heterogeneous fleet

To reduce computational complexity, literatures consider the vehicles and orders as heterogeneous and ignore the though obvious distinctions between each driver and costumer.

On one hand, for each driver, they may have their own destinations like their living places at a certain time, or they may have certain preferences of routes since some drivers prefer to take the route that are faster and the others may favor those routes which are easy to drive on. When doing order dispatching, those features or preferences of drivers may influence their degree of satisfaction on the dispatch result. On the other hand, for costumers, they may also have preferences or special needs which will greatly influence their riding experience and thus their degrees of satisfaction also.

The driver's preference has to be stressed when considering vehicle re-positioning. In real-world practice, drivers may have their own pattern of seeking customer, for instance, some drivers may prefer to route around the city railway station while others may show a special favor for the shopping centers. Having this specific preference when doing re-positioning, the system have to make decisions which not only balance the demand and supply but also satisfy the driver's preference. There are a lot of papers using the order reject rate to evaluate their algorithm's advantage, however not any paper uses the driver reject rate to evaluate whether their re-position action is disobeyed by the driver and considered as noneffective which is normally observed in real-world scenarios.

Finding an approach to bring in the heterogeneous features of both the driver and consumer into the reinforcement learning system in fleet management is what future researches should focus on. Researches have already been conducted on modeling drivers routing patterns as well as their driving preferences but are all based on a small scale of drivers and none was used in practice to help addressing fleet management problems [37, 38].

V. Conclusion

The fleet management problems are increasingly important in our daily life now. In this paper the fleet management problems are divided into three parts: order dispatching, vehicle routing and re-positioning. This paper mainly focused on reviewing the methods of order dispatching and re-positioning questions using multi-agent reinforcement learning. Meanwhile, the basic knowledge of single and multi-agent reinforcement learning is presented. Various methods solving the two problems were generally separated to three categories based on the choice of applying vehicle-based agent or grid-based agent, along with some extended work of the former categories. The representative methods are introduced and explained in this paper. Furthermore, the real-life applications of multi-agent reinforcement learning are investigated. The challenges and possible opportunities in improving fleet management system with multi-agent reinforcement learning are also discussed.

References

[1] Bielli M, Bielli A, Rossi R. Trends in modelsand algorithms for fleet management. Procedia-Social and Behavioral Sciences. 2011;20:4–18.

[2] Zong Z, Feng T, Xia T, Li Y, et al. Deep Reinforcement Learning for Demand Driven Services in Logistics and Transportation Systems: A Survey. arXiv preprint arXiv:210804462. 2021.

[3] Qin Z, Zhu H, Ye J. Reinforcement Learning for Ridesharing: A Survey; 2021.

[4] Bu¸soniu L, Babu˘ska R, De Schutter B. Multi-agent reinforcement learning: An overview. Innovations in multi-agent systems and applications-1. 2010:183-221.

[5] Farazi NP, Zou B, Ahamed T, Barua L. Deep reinforcement learning in transportation research: A review. Transportation Research Interdisciplinary Perspectives. 2021;11:100425.

[6] Sutton RS, Barto AG. Reinforcement learning: An introduction. MIT press; 2018.

[7] Qin Z, Tang X, Jiao Y, Zhang F, Xu Z, Zhu H, et al. Ride-hailing order dispatching at DiDi via reinforcement learning. INFORMS Journal on Applied Analytics. 2020;50(5):272{286.

[8] Xu Z, Li Z, Guan Q, Zhang D, Li Q, Nan J, et al. Large-scale order dispatch in on-demand ride-hailing platforms: A learning and planning approach. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining; 2018. p. 905-913.

[9] Wang Z, Qin Z, Tang X, Ye J, Zhu H. Deep reinforcement learning with knowledge transfer for online rides order dispatching. In: 2018 IEEE International Conference on Data Mining (ICDM). IEEE; 2018. p. 617-626.

[10] Zhou M, Jin J, Zhang W, Qin Z, Jiao Y, Wang C, et al. Multi-agent reinforcement learning for order-dispatching via order-vehicle distribution matching. In: Proceedings of the 28th ACM International Conference on Information and Knowledge Management; 2019. p. 2645-2653.

[11] Yang L, Yu X, Cao J, Liu X, Zhou P. Exploring Deep Reinforcement Learning for Task Dispatching in Autonomous On-Demand Services. ACM Transactions on Knowledge Discovery from Data (TKDD). 2021;15(3):1-23.

[12] Zhang L, Hu T, Min Y, Wu G, Zhang J, Feng P, et al. A taxi order dispatch model based on combinatorial optimization. In: Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining; 2017. p. 2151-2159.

[13] Ke J, Yang H, Zheng H, Chen X, Jia Y, Gong P, et al. Hexagon-based convolutional neural network for supply-demand forecasting of ridesourcing services. IEEE Transactions on Intelligent Transportation Systems. 2018;20(11):4160-4173.

[14] Liang E, Wen K, Lam WH, Sumalee A, Zhong R. An Integrated Reinforcement Learning and Centralized Programming Approach for Online Taxi Dispatching. IEEE Transactions on Neural Networks and Learning Systems. 2021.

[15] Tang X, Qin Z, Zhang F, Wang Z, Xu Z, Ma Y, et al. A deep value-network based approach for multi-driver order dispatching. In: Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining; 2019. p. 1780-1790.

[16] Tang X, Zhang F, Qin Z, Wang Y, Shi D, Song B, et al. Value Function is All You Need: A Unified Learning Framework for Ride Hailing Platforms. In: Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining; 2021. p. 3605-3615.

[17] Li M, Qin Z, Jiao Y, Yang Y, Wang J, Wang C, et al. Efficient ridesharing order dispatching with mean field multi-agent reinforcement learning. In: The World Wide Web Conference; 2019. p. 983-994.

[18] Lynch GS. Single point of failure: The 10 essential laws of supply chain risk management. John Wiley and Sons; 2009.

[19] Yang Y, Luo R, Li M, Zhou M, Zhang W, Wang J. Mean Field Multi-Agent Reinforcement Learning; 2020.

[20] Jin J, Zhou M, Zhang W, Li M, Guo Z, Qin Z, et al. Coride: joint order dispatching and fleet management for multi-scale ride-hailing platforms. In: Proceedings of the 28th ACM International Conference on Information and Knowledge Management; 2019. p. 1983-1992.

[21] Liu C, Chen CX, Chen C. META: A City-Wide Taxi Repositioning Framework Based on MultiAgent Reinforcement Learning. IEEE Transactions on Intelligent Transportation Systems. 2021.

[22] Jintao K, Yang H, Ye J, et al. Learning to delayin ride-sourcing systems: a multi-agent deep reinforcement learning framework. IEEE Transactions on Knowledge and Data Engineering. 2020.

[23] O'Keeffe K, Anklesaria S, Santi P, Ratti C. Using reinforcement learning to minimize taxi idle times. Journal of Intelligent Transportation Systems. 2021:1-16.

[24] Yao H, Wu F, Ke J, Tang X, Jia Y, Lu S, et al. Deep multi-view spatial-temporal network for taxi demand prediction. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 32; 2018.

[25] Xu J, Rahmatizadeh R, B¨ol¨oni L, Turgut D. Real-time prediction of taxi demand using recurrent neural networks. IEEE Transactions on Intelligent Transportation Systems. 2017;19(8):2572-2581.

[26] He S, Shin KG. Spatio-temporal capsule-based reinforcement learning for mobility-on-demand network coordination. In: The World Wide Web Conference; 2019. p. 2806-2813.

[27] Oda T, Joe-Wong C. MOVI: A model-free approach to dynamic fleet management. In: IEEE INFOCOM 2018-IEEE Conference on Computer Communications. IEEE; 2018. p. 2708-2716.

[28] Liu Z, Li J, Wu K. Context-Aware Taxi Dispatching at City-Scale Using Deep Reinforcement Learning. IEEE Transactions on Intelligent Transportation Systems. 2020.

[29] Lin K, Zhao R, Xu Z, Zhou J. Efficient largescale fleet management via multi-agent deep reinforcement learning. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining; 2018. p. 1774-1783.

[30] Wen J, Zhao J, Jaillet P. Rebalancing shared mobility-on-demand systems: A reinforcement learning approach. In: 2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC). Ieee; 2017. p. 220-225.

[31] Shou Z, Di X, Ye J, Zhu H, Zhang H, Hampshire R. Optimal passenger-seeking policies on Ehailing platforms using Markov decision process and imitation learning. Transportation Research Part C: Emerging Technologies. 2020;111:91-113.

[32] Yang Y, Wang X, Xu Y, Huang Q. Multiagent reinforcement learning-based taxi predispatching model to balance taxi supply and demand. Journal of Advanced Transportation. 2020;2020.

[33] Li Y, Zheng Y, Yang Q. Cooperative MultiAgent Reinforcement Learning in Express System. In: Proceedings of the 29th ACM International Conference on Information & Knowledge Management; 2020. p. 805-814.

[34] Zhang W, Wang Q, Li J, Xu C. Dynamic Fleet Management With Rewriting Deep Reinforcement Learning. IEEE Access. 2020;8:143333-143341.

[35] James J, Yu W, Gu J. Online vehicle routing with neural combinatorial optimization and deep reinforcement learning. IEEE Transactions on Intelligent Transportation Systems. 2019;20(10):3806-3817.

[36] Garg N, Ranu S. Route recommendations for idle taxi drivers: Find me the shortest route to a customer! In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining; 2018. p. 1425-1434.

[37] Pan M, Li Y, Zhou X, Liu Z, Song R, Lu H, et al. Dissecting the learning curve of taxi drivers: A data-driven approach. In: Proceedings of the 2019 SIAM International Conference on Data Mining. SIAM; 2019. p. 783-791.

[38] Ziebart BD, Maas AL, Dey AK, Bagnell JA. Navigate like a cabbie: Probabilistic reasoning from observed context-aware behavior. In: Proceedings of the 10th international conference on Ubiquitous computing; 2008. p. 322-331.