Reinforcement Learning for Ridesharing: An Extended Survey*

Zhiwei (Tony) Qin†
DiDi Labs
Mountain View, CA 94043

Hongtu Zhu
University of North Carolina
Chapel Hill, NC 27514

Jieping Ye
University of Michigan
Ann Arbor, MI 48109

Abstract

In this paper, we present a comprehensive, in-depth survey of the literature on reinforcement learning approaches to decision optimization problems in a typical ridesharing system. Papers on the topics of rideshare matching, vehicle repositioning, ride-pooling, routing, and dynamic pricing are covered. Popular data sets and open simulation environments are also introduced. Subsequently, we discuss a number of challenges and opportunities for reinforcement learning research on this important domain.

1 Introduction

The emergence of ridesharing, led by companies such as DiDi, Uber, and Lyft, has revolutionized the form of personal mobility. It is projected that the global rideshare industry will grow to a total market value of $218 billion by 2025 [66]. Auto industry experts expect that ride-hailing apps would eventually make individual car ownership optional [20]. However, how to improve operational efficiency is a major challenge for rideshare platforms, e.g., long passenger waiting time [99] and as high as 41% vacant time for ridesharing vehicles in a large city [14]. The success of ridesharing, from the perspectives of the platforms, drivers, and passengers, requires sophisticated optimization of all the integrated components that collectively deliver the services.

Reinforcement learning (RL) is a machine learning paradigm that trains an agent to take optimal actions (measured by total cumulative reward) through interaction with the environment and getting feedback signals. It is a class of optimization methods for solving sequential decision-making problems with a long-term objective in a stochastic environment. Thanks to the rapid advancement in deep learning research and computing power, the integration of deep neural networks and RL has generated explosive progress in solving complex large-scale decision problems [96, 11], attracting huge amount of renewed interests in the recent years. We are witnessing a similar trend in the ridesharing domain, where the demand and supply are highly stochastic and non-stationary, and the operational decisions are often sequential in nature and have strong spatiotemporal dependency. RL, often being data-driven, presents itself as an excellent promising approach to these ridesharing decision optimization problems.

There are excellent surveys on RL for intelligent transportation [35, 130], with in-depth coverage of traffic signals control and autonomous driving. [115] offers a broad review of ridesharing systems,

*This survey is a significantly expanded and refined version of [82].

†Corresponding author, email: zq2107@caa.columbia.edu

Preprint. Under review.

whereas [107] surveys spatial crowdsourcing, which is a more general field than ridesharing. There has been no comprehensive review of the literature on RL for ridesharing, even though the field has attracted much attention and interest from the research communities for both RL and transportation just within the last few years (e.g., [89, 103, 1, 94]). This paper aims to fill that gap by surveying the literature of this domain published in top conferences and journals in transportation (e.g., Transportation Research series, IEEE Transactions on Intelligent Transportation Systems, Transportation Science), data mining (e.g., KDD, ICDM, WWW, CIKM), and machine learning/AI (e.g., NeurIPS, AAAI, IJCAI). We describe the research problems associated with the various aspects of a ridesharing system, review the existing RL approaches proposed to tackle each of them, and discuss the challenges and opportunities.

Reinforcement learning has close relationship with other families of methods, such as stochastic optimization, approximate dynamic programming, and model predictive control. Although it is not our goal in this paper to have a comprehensive review of those methods for problems in ridesharing, we aim to point the readers to the representative works so that they can refer to the further literature therein.

This survey is organized as follows: We lay out the ridesharing system architecture in Section 2 and define the scope of the problems to be reviewed. Within this section, as well as the subsequent sections of the survey, we clarify and draw connections among the different names that the problems are referred to, which are often due to the different communities that the authors are from and are easy to confuse by researchers new to ridesharing. In Section 3, we provide a concise review of the RL basics and the major algorithms adopted by the works in this survey. We review in details in Section 4 the literature for each problem described in Section 2 and the relevant data sets and environments. Finally in Section 5, we discuss some challenges and opportunities that we feel crucial in advancing RL for ridesharing.

2 Ridesharing

We first describe the architecture of a ridesharing system in this section, followed by explanation and clarification on the scopes of the problem associated with each module.

2.1 Architecture

A ridesharing service, in contrast to taxi hailing, matches passengers with drivers of vehicles for hire using mobile apps. In a typical mobile ridesharing system, there are five major modules: pricing, matching, repositioning, pooling, and routing. Figure 1 illustrates the process and decision modules. When a potential passenger submits a trip request, the *pricing* module offers a quote, which the passenger either accepts or rejects. Upon acceptance, the *matching* module attempts to assign the request to an available driver. Depending on driver pool availability, the request may have to wait in the system until a successful match. Pre-match cancellation may happen during this time. The assigned driver then travels to pick up the passenger, during which time post-match cancellation may also occur. The pick-up location is usually where the passenger is making the request or he/she specifies. In some cases, it could be a public designated area, e.g., outside an airport or train station. After the driver successfully transports the passenger to the destination, she receives the trip fare and becomes available again. The *repositioning* module guides idle vehicles to specific locations in anticipation of fulfilling more requests in the future. Following the reposition recommendations is usually on a voluntary basis unless it is an autonomous ridesharing setting. Hence, it is common that the platform offers incentives to drivers for completing the repositions. When each driver takes only one passenger request at a time, i.e. only one passenger shares the ride with the driver, this mode is more commonly called 'ride-hailing'. Ridesharing can refer to both ride-hailing and ride-pooling. In the *ride-pooling* mode, multiple passengers with different trip requests can share one single vehicle, so the pricing, matching, repositioning, and routing problems are different from those for ride-hailing and require specific treatment, in particular, considering the passengers already on board. The *routing* module provides turn-by-turn guidance on the road network to drivers/vehicles either in service of a passenger request or performing a reposition. The goal is to guide the vehicle to its destination efficiently and safely.

2.2 Problem Scopes

First, we start from the pricing module. Since the trip fare is both the price that the passenger has to pay for the trip and the major factor for the income of the driver, pricing decisions influence both demand and supply distributions through price sensitivities of users, e.g., the use of surge pricing during peak hours. This is illustrated by the solid and dotted arrows pointing from the pricing module to orders and idle vehicles respectively in Figure 1. The pricing problem in the ridesharing literature is in most cases dynamic pricing, which adjusts trip prices in real-time in view of the changing demand and supply. The pricing modules sits at the upstream position with respect to the other modules and is a macro-level lever to achieve supply-demand (SD) balance.

The ridesharing matching problem [127, 77, 81] may appear under different names in the literature, e.g., order dispatching [81], trip-vehicle assignment [9], and on-demand taxi dispatching [107]. It is an online bipartite matching problem where both supply and demand are dynamic, with the uncertainty coming from demand arrivals, travel times, and the entrance-exit behavior of the drivers. Matching can be done continuously in a streaming manner or at fixed review windows (i.e., batching). Sophisticated matching algorithms often leverage demand prediction in some form beyond the actual requests, e.g., the value function in RL. Online request matching is not entirely unique to ridesharing. Indeed, ridesharing matching falls into the family of more general dynamic matching problems for on-demand markets [39]. A distinctive feature of the ridesharing problem is its spatiotemporal nature. A driver's eligibility to match and serve a trip request depends in part on her spatial proximity to the request. Trip requests generally take different amount of time to finish, and they change the spatial states of the drivers, affecting the supply distribution for future matching. The drivers and passengers generally exhibit asymmetric exit behaviors in that drivers usually stay in the system for an extended period of time, whereas passenger requests are lost after a much shorter waiting period in general.

Single-vehicle repositioning may refer to as taxi routing or passenger seeking in the literature. Taxi routing slightly differs in the setting from repositioning a rideshare vehicle in that a taxi typically has to be at a visual distance from the potential passenger to take the request whereas the matching radius of a mobile rideshare request is considerably longer, sometimes more than a mile. System-level vehicle repositioning, also known as driver dispatching, vehicle rebalancing/reallocation, or fleet management, aims to rebalance the global SD distributions by proactively dispatching idle vehicles to different locations. Repositioning and matching are similar to each other in that both relocate a vehicle to a different place as a consequence. In theory, one can treat repositioning as matching a vehicle to a virtual trip request, the destination of which is that of the reposition action, so that both matching and repositioning can be solved in a single problem instance. Typically in practice, these two problems are solved separately because they are separate system modules on most ridesharing platforms with different review intervals and objective metrics among other details.

The definition of the routing problem is often confused, especially in the context of ridesharing. The routing module described in Section 2.1 performs route guidance, which could be *dynamic routing* or *route planning* depending on the decision points. Dynamic routing is also called *dynamic route choice*, and route planning is alternatively referred to as the traffic assignment problem [93]. In some cases, the reposition policy directly provides link-level turn-by-turn guidance with the goal of maximizing the driver's income, thus covering the role of dynamic routing albeit with a different objective. Dynamic routing is generally different from the *vehicle routing problem* (VRP) [21]. In VRP, the set of destinations that the vehicle has to visit is known in advance, and hence, it is a static problem. It mainly concerns with the sequence in which the destinations should be visited, considering the estimated travel time from point to point. In contrast, dynamic routing is associated with a road network, and the decision to make is which outgoing road (link) to follow at each intersection (node). The decision is adaptive to the changing traffic condition on the road network in real time. In the context of ride-pooling, there is another emerging problem in which the dynamic routing decisions with passenger(s) on board have to align with the overall objective of ride-pooling.

Some literature refers to the mode of multiple passengers sharing a ride as 'ridesharing'. In this paper, we use term '*ride-pooling*' (or 'carpool') to disambiguate the concept, as 'ridesharing' can refer to both single- and multiple-passenger rides. The seminal paper of [3] shows that through ride-pooling optimization, the number of taxis required to meet the same trip demand can be significantly reduced with limited impact on passenger waiting times. In a pooling-enabled rideshare system, the matching, repositioning, and pricing modules all have to adapt to additional complexity. Compared to the regular ride-hailing problem, the one with ride-pooling has considerably more complexity due to

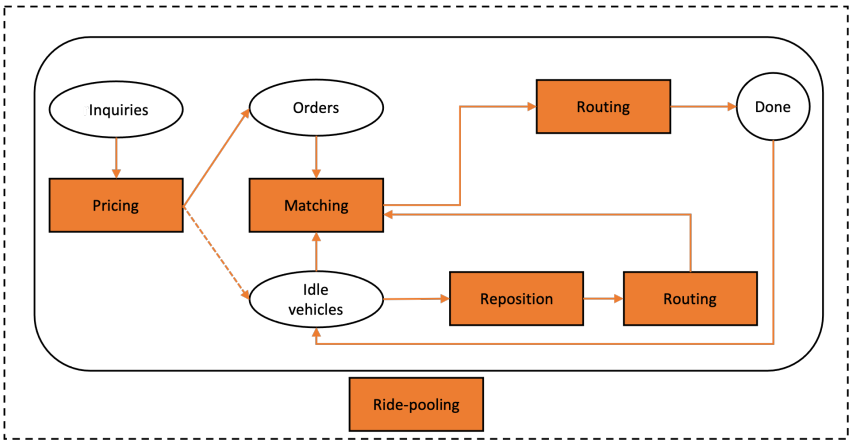

Figure 1: The process flow of ridesharing operations. The solid orange rectangular boxes represent the modules described in Section 2, and the literature on the optimization problems associated with the modules are reviewed in the paper.

the more dynamic nature of the problem and the additional constraints and multiple objectives that have to be considered. In this case, the set of available vehicles are augmented, including both idle vehicles and occupied ones not at capacity. It is non-trivial to determine the set of feasible actions (one or more passengers to be assigned to a vehicle) for matching. Every time a new passenger is to be added to a non-idle vehicle, the route has to be recalculated using a VRP solver to account for the additional pick-up and drop-off, the travel times for all the passengers already on board are updated, and the vehicle capacity, the waiting time and detour distance constraints are checked. In-service routing in the context of ride-pooling is discussed in Section 4.5.

3 Reinforcement Learning

We briefly review the RL basics and the major algorithms, especially those used in the works reviewed by this survey. For a complete reference, see, e.g., [102].

3.1 Basics

RL is based on the Markov decision process (MDP) framework, where the agent (the decision-making entity) has a *state* s in the state space $\mathcal{S}$ and can perform an *action* a defined by an action space $\mathcal{A}$. After executing the action, the agent receives an immediate *reward* $R(s,a)$ from the environment, and its state changes according to the transition probabilities $P(\cdot|s,a)$. The process repeats until a terminal state or the end of the horizon is reached, giving a sequence of the triplets $(s_t, a_t, r_t)_{t=0}^{t=T}$, where t is the epoch index, T is the final epoch at the terminal state or the end of the horizon, and r_t is a sample of R. The objective of the MDP is to maximize the cumulative reward over the horizon. A key quantity to compute is the value function

$$V(s) := E\left[\sum_{t=0}^{t=T} \gamma^t r_t \middle| s_0 = s\right],$$

which satisfies the Bellman equation,

$$V(s_t) = \sum_{a_t} P(s_{t+1}, r_t | s_t, a_t)\Big(r_t(s_t, a_t) + \gamma V(s_{t+1})\Big). \tag{1}$$

Similarly, we have the action-value function

$$Q(s,a) := E\left[\sum_{t=0}^{t=T} \gamma^t r_t \middle| s_0 = s, a_0 = a\right],$$

which conditions on both s and a. A policy is a function $\pi(s) : \mathcal{S} \to \mathcal{A}$, and we denote the state-value and action-value associated with a given π by V^π and Q^π, respectively, and their optimal values by V^* and Q^* evaluated at the optimal policy π^*. The objective of an MDP is to find an optimal policy π^* that maximizes the long-term cumulative discounted reward, i.e., $\pi^* := \arg\max_\pi E_s[V^\pi(s)]$.

3.2 Algorithms

Given P and R, which specifies the MDP, and π, we can compute V^π by iteratively applying the Bellman equation (1),

$$V^\pi(s) \leftarrow \sum_a \pi(a|s) \sum_{s',r} P(s', r|s, a)\Big(r + \gamma V^\pi(s')\Big). \tag{2}$$

This is called *policy evaluation*. Using policy evaluation as a sub-routine, we can again iteratively improve the policy through *policy iterations*, which generate a new policy at each outer iteration by acting greedily with respect to V^π,

$$a^* \leftarrow \arg\max_a \sum_{s',r} P(s', r|s, a)\Big(r + \gamma V^\pi(s')\Big). \tag{3}$$

As a special instance, we can collapse the inner policy evaluation loop to a single iteration and compute $V^*(s), \forall s \in \mathcal{S}$, by the *value iterations*,

$$V(s) \leftarrow \max_a \sum_{s',r} P(s', r|s, a)\Big(r + \gamma V(s')\Big). \tag{4}$$

If we estimate (P, R) from data, we have a basic model-based RL method. A model-free method learns the value function and optimizes the policy directly from data without learning and constructing a model of the environment. A common example is *temporal-difference (TD) learning* [101], which iteratively updates V^π by TD-errors using π-generated trajectory samples and bootstrapping,

$$V^\pi(s) \leftarrow V^\pi(s) + \alpha\Big(r + \gamma V^\pi(s') - V^\pi(s)\Big), \tag{5}$$

where s' is the next state in the trajectory after s, α is the step size (or learning rate), and the term that it scales is the TD error. If learning the optimal action-values for control is the goal, we similarly have *Q-learning* [119], which updates the action-value function $Q(s, a)$ to approximate $Q^*(s, a)$ by

$$Q(s, a) \leftarrow Q(s, a) + \alpha\Big(r + \gamma \max_{a'} Q(s', a') - Q(s, a)\Big). \tag{6}$$

Q-learning is an *off-policy* algorithm, where the behavior policy, which collects the experience data and typically involves exploration, is different from the target policy that we are trying to learn and in this case, is the optimal policy. The *on-policy* counterpart of Q-learning is SARSA, which basically generalizes TD-learning (5) to the action-value function associated with the behavior policy π (same as the target policy):

$$Q^\pi(s, a) \leftarrow Q^\pi(s, a) + \alpha\Big(r + \gamma Q^\pi(s', a') - Q^\pi(s, a)\Big), \tag{7}$$

where a' is the action executed by the agent at state s' in the experience data. The *deep Q-network* (DQN) [72] approximates $Q(s, a)$ by a neural network Q_w parametrized by w along with a few heuristic techniques like experience replay [57] and a target network to improve training stability. These techniques are critical to the successful of DQN in playing Atari games and many other applications, due to the deadly triad issue [102] of reinforcement learning with nonlinear function approximation. The algorithms introduced so far are all value-based methods, which focus on learning the value function, and the policy is derived from the learned value function by, e.g., $\arg\max_a Q(s, a)$.

A policy-based method directly learns π (which is also called the *actor* and parametrized by θ) by performing stochastic gradient descent. The central step is computing the policy gradient (PG), the gradient of the cumulative reward $J(\theta)$ with respect to the policy parameters θ,

$$\nabla_\theta J(\theta) = \sum_s \mu(s) \sum_a Q^\pi(s, a) \nabla_\theta \pi(a|s, \theta), \tag{8}$$

where μ is the on-policy distribution under π. A more common (equivalent) form of the PG (8) is

$$\sum_s \mu(s) \sum_a \pi(a|s,\theta) Q^\pi(s,a) \nabla_\theta \log \pi(a|s,\theta), \tag{9}$$

which most of the policy-based methods are based on. REINFORCE [122] is a classical PG method, which uses Monte Carlo (MC) rollout to obtains the sample-based update

$$\theta_{t+1} \leftarrow \theta_t + \alpha G_t \nabla_\theta \log \pi(a|s,\theta), \tag{10}$$

where G_t is an MC approximation of Q^π. As in a value-based method, we can also use function approximation for the action values. The function approximator (e.g., a neural network) Q_w is called the *critic*, and the resulting algorithm is an *actor-critic* (AC) method. It is well-recognized that the 'baseline' version of (8),

$$\sum_s \mu(s) \sum_a \pi(a|s,\theta) \Big(Q^\pi(s,a) - b(s) \Big) \nabla_\theta \log \pi(a|s,\theta), \tag{11}$$

where $b(s)$ is an action-independent baseline, reduces the variance in the sample gradient and helps speed up learning. Since a natural choice of such a baseline is the state value, the critic often learns the advantage function $Q(s,a) - V(s)$, and the method is called Advantage Actor Critic (A2C). The evaluation of an action is based on how good it can be with respect to the average over all actions, the benefit of which is to reduce the high variance in the actor and to stabilize the model. [71] extends A2C to an asynchronous version (A3C) where independent agents interact with their own copy of the environment and update their model parameters with the master copy asynchronously. This architecture enables much more efficient utilization of the CPU cores through parallel threads and hence accelerates the training. The proximal policy optimization (PPO) [88] optimizes a clipping surrogate objective with respect to the advantage to promote conservative updates to π and is a popular choice of training algorithm for RL problems where policy-based methods are suitable (e.g., with continuous actions).

An MDP can be extended to a Markov game involving multiple agents to form the basis for multi-agent RL (MARL). Many MARL algorithms, e.g., [128, 61] focus on agent communication and coordination, in view of the intractable action space in MARL.

3.3 Approximate Dynamic Programming

A family of methods closely related to RL is approximate dynamic programming (ADP) [79] for solving stochastic dynamic programs (DP), of which the Bellman equation for MDP is an instance. In ADP methods, a post-decision state s_t^x is often defined to represent the intermediate state to which the current state s_t will transition deterministically given the action a_t before the random factors ω_t (e.g., demand appearance and cancellation) in the environment realize. With ω_t fully realized, the state transitions into the next pre-decision state s_{t+1}. The value function in an ADP method is defined on the post-decision state and is approximated by a particular functional form. Given the approximated values, the original optimization problem is solved to obtain the decision solution for the current time step. Linear function approximation is popular (e.g., [97, 133, 1] because the dual variables associated with the solution to the current-stage optimization can be used to update the linear function parameters. Then, the state is advanced to the next pre-decision state, and the iteration continues until convergence. Nevertheless, neural network-based value function approximation [89] has also been adopted and developed due to their higher level of flexibility. In this case, the value function updates largely follow the DQN scheme.

4 Reinforcement Learning for Ridesharing

We review the RL literature for ridesharing in this section grouped by the core operational problems described in Section 2. We first cover pricing, matching, repositioning, and routing in the context of ride-hailing. Then, we will review works on those problems specific to ride-pooling.

4.1 Pricing

RL-based approaches have been developed for dynamic pricing in one-sided retail markets [83, 12], where pricing changes only the demand pattern per customers' price elasticity. The ridesharing

marketplace, however, is more complex due to its two-sided nature and spatiotemmporal dimensions. In this case, pricing is also a lever to change the supply (driver) distribution if price changes are broadcast to the drivers in advance.[3] [16] describes examples of such elasticity functions for both demand and supply for their simulation environment. In addition, dynamic pricing on trip inquiries changes the subsequent distribution of the submitted requests through passenger price elasticity. The requests distribution, in turn, influences future supply distribution as drivers fulfill those requests. Because of its close ties to SD distributions, dynamic pricing is often jointly optimized with order matching or vehicle repositioning. Prior to using RL, dynamic pricing for ridesharing has already been studied and analyzed in conjunction with matching [127, 77] and from the spatiotemporal perspective [62, 13, 38], covering optimality and equilibrium analyses.

Figure 1 summarizes the reviewed works on RL for dynamic pricing in ridesharing. As one of the early works, [124] consider a simplified ridesharing environment which captures only the two-sidedness of the market but not the spatiotemporal dimensions. The state of the MDP is the current price plus SD information. The action is to set a price, and the reward is the generated profit. A Q-learning agent is trained in a simple simulator, and empirical advantage in the total profit is demonstrated against other heuristic approaches. More recent works leverage the spatiotemporal nature of the pricing actions and take into account the spatiotemporal long-term values in the pricing decisions. [17] integrate contextual bandits and the spatiotemporal value network developed in [103] for matching to jointly optimize pricing and matching decisions. In particular, the pricing actions are the discretized price percentage changes and are selected by a contextual bandits algorithm, where the long-term values learned by the value network are incorporated into the bandit rewards. In [110], the RL agent determines both the price for each origin-destination (OD) pairs and the reposition/charging decisions for each electric vehicle in the fleet. The state contains global information such as the electricity price in each zone, the passenger queue length for OD pair, and the number of vehicles in each zone and their energy levels. The reward accounts for trip revenue, penalty for the queues, and operational cost for charging and repositioning. Due to the multi-dimensional continuous action space, PPO is used to train the agent in a simulator. [100] perform a case study of ridesharing in Seoul. They use a tabular Q-learning agent to determine spatiotemporal pricing, and extensive simulations are performed to analyze the impact of surge pricing on alleviating the problem of marginalized zones (areas where it is consistently hard to get a taxi) and on improving spatial equity. [16] adopts PPO to optimize the spatiotemporal pricing decisions for each hexagonal cell in terms of the per-km rate for the excess mileage beyond a base trip distance and the per-km rate for driver wage, for the objective of maximizing profits (revenue minus wage). The agent is modeled as a global decision maker with state information of the numbers of open requests, vacant vehicles, occupied vehicles in each grid cell at time t and historical demand at time $t-1$. Unlike the works above that focus on the pricing decisions, [67] study from a different perspective of the pricing problem. The proposed risk-sensitive inverse RL method [74] recovers the policies of different types of passengers (risk-averse, risk-neutral, and risk-seeking) in view of surge pricing. The policy determines whether the passenger should wait or take the current ride.

4.2 Online Matching

The rideshare matching problem and its generalized forms have been investigated extensively in the field of operations research (see e.g., [77, 39, 60] and the references therein). Typically, both the open trip requests and available drivers are batched within time windows of fixed length as they arrive at the system, and they are matched at predefined discrete review times. See Figure 2 for an illustration. Hence, ridesharing matching is an online stochastic problem [81]. [60] approach the problem through stochastic optimization and use Bender's decomposition to solve it efficiently. To account for the temporal dependency of the decisions, [39] formulate the problem as a stochastic DP and propose heuristic policies to compute the optimal matching decisions. For a related problem, the truckload carriers assignment problem, [97] also formulate a dynamic DP but with post-decision states so that they are able to solve the problem using ADP. In each iteration, a demand path is sampled, and the value function is approximated in a linear form and updated using the dual variables from the LP solution to the resulting optimization problem.

[3] In some cases, the trip price could influence the probability that a driver accepts a given assignment, depending on the rules of the particular ridesharing platform.

Paper	Agent	State	Action	Reward	Algorithm	Environment
[124]	global decision-maker	current trip price (same for all trips), SD info	price	profit	Q-learning	no spatiotemporal dimensions
[17]	global decision-maker	features of the trip request	discretized price change percentage	profit	contextual bandits with action values partly computed by CVNet	ride-hailing simulator with pricing module and passenger elasticity model
[110]	global decision-maker for pricing and EV charging	electricity price in each zone, passenger queue length for each OD pair, number of vehicles in each zone and their energy levels	price for each OD pair, reposition/ charging for each vehicle	trip revenue - penalty for queues - operational cost for charging and reposition	PPO	simulator
[100]	global decision-maker	location, time	price for spatial-temporal grid cells	trip price minus penalty for driver waiting	Q-learning	case study: ride-hailing simulation of Seoul
[67]	passenger	price multiplier, time, if a ride has completed	wait, take current ride	trip price to pay	risk-sensitive inverse RL	historical data
[16]	global decision-maker	number of open requests, vacant vehicles, and occupied vehicles in each grid cell at time t, and demand in time $t-1$	joint actions of price (per-km for excess mileage) and wage (per-km rate) for each grid cell	profit: revenue minus wage	PPO	simulation based on Hangzhou data from DiDi; modeling on both supply and demand elasticity

Table 1: Summary of literature for Pricing.

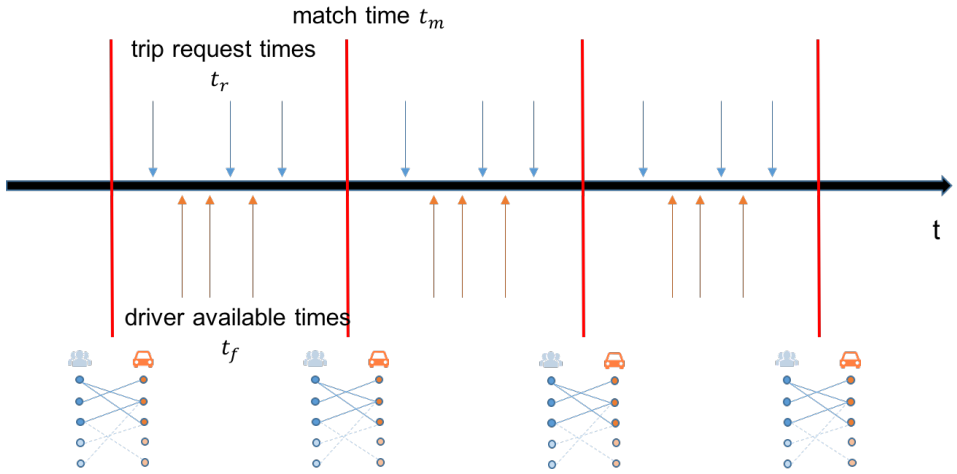

Figure 2: The order matching process with batching from the system perspective [81].

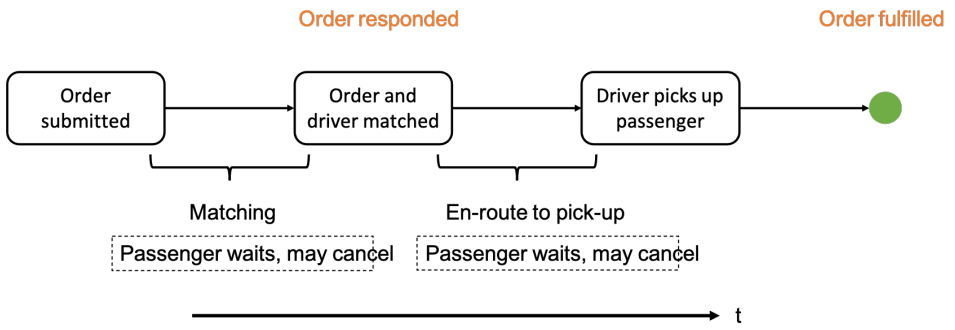

Figure 3: The order matching process from a single request's perspective.

The RL literature for rideshare matching typically aims to optimize the total driver income and the service quality over an extended period of time. Service quality can be quantified by *response rate* and *fulfillment rate*. Response rate is the ratio of the matched requests to all trip requests. Since the probability of pre-match cancellation is primarily a function of response time (pre-match waiting time), the total response time is an alternative metric to response rate. Fulfillment rate is the ratio of completed requests to all requests and is no higher than the response rate. The gap is due to post-match cancellation, usually because of the waiting for pick-up. Hence, the average pick-up distance is also a relevant quantity to observe. Figure 3 shows the detailed flow of matching a single trip request together with the quantities discussed above.

In terms of the MDP formulation, modeling the agent as a driver is convenient choice for its straightforward definition of state, action, and reward, in contrast to system-level modeling where the action space is exponential. In this case, the rideshare platform is naturally a multi-agent system with a global objective. A common approach is to crowdsource all drivers' experience trajectories to train a single agent and apply it to all the drivers to generate their matching policies [126, 118, 103]. Since the system reward is the sum of the drivers' rewards, the system value function does decompose into the individual drivers' value functions computed by each driver's own trajectories. The approximation here is using a single value function learned from all drivers' data. See [81] for detailed discussions. Specifically, [126] learn a tabular driver value function using TD(0), and [118, 103, 36] apply DQN-type of training to learn a value network. In particular, [103] design a spatiotemporal state-value network using hierarchical coarse coding and cerebellar embedding memories for better state representation and training stability. [36] develop an action-value network that leverages global SD information, which is embedded into a global context by attention.

This type of single-agent approach avoids dealing explicitly with the multi-agent aspect of the problem and the interaction among the agents during training. Besides simplicity, this strategy has the additional advantage of being able to easily handle a dynamic set of agents (and hence, a changing action space) [49]. On the other hand, order matching requires strong system-level coordination in that a feasible solution has to satisfy the one-to-one constraints. To address this issue, [126, 103] use the learned state values to populate the edge weights of a bipartite assignment problem to generate a collective-greedy policy [81] with respect to the state values. [36] assume a setting where drivers are matched or repositioned sequentially so that the policy output always satisfies the matching constraints.

Leveraging MARL, [54, 44, 140] directly optimize the multi-agent system. One significant challenge is scalability since any realistic ridesharing setting can easily involve thousands of agents, precluding the possibility of dealing with an exact joint action space. [54] apply mean-field MARL to make the interaction among agents tractable, by taking the 'average' action of the neighboring agents to approximate the joint actions. [140] argue that no explicit communication among agents is required for order matching due to the asynchronous nature of the transitions and propose independent Q-learning with centralized KL divergence (of the supply and demand distributions) regularization. Both [54, 140] follow the centralized training decentralized execution paradigm. [44] take a different approach treating each spatial grid cell as a worker agent and a region of a set of grid cells as a manager agent, and they adopt hierarchical RL to jointly optimize order matching and vehicle repositioning.

Besides the driver-passenger pairing decisions, there have also been research on using RL to learn when to match a request (or a batch of requests). This can be done from the perspective of a request itself [49] or the system [116, 80]. In [49], an agent is modeled as a trip request. An agent network is trained centrally using pooled experience from all agents to decide whether or not to delay the matching of a request to the next review window, and all the agents share the same policy. To encourage cooperation among the agents, a specially shaped reward function is used to account for both local and global reward feedback. They also modify the RL training framework to address the delayed reward issue by sampling complete trajectories at the end of training epochs to update the network parameters. [116] take a system's view and propose a Restricted Q-learning algorithm to determine the length of the current review window (or batch size). They show theoretical analysis results on the performance guarantee in terms of competitive ratio for dynamic bipartite graph matching with adaptive windows. [80] take a similar modeling perspective but use the AC method with experience replay (ACER) [117] that combines on-policy updates (through a queuing-based simulator) with off-policy updates.

Because of its generalizability, matching for ridesharing is closed related to a number of online matching problems in other domains, the RL methods to which are also relevant and can inform the research in rideshare matching. Some examples are training a truck agent using DQN with pooled experience to dispatch trucks for mining tasks [137], learning a decentralized value function using PPO with a shaped reward function for cooperation (in similar spirit as [49]) to dispatch couriers for pick-up services [18], and designing a self-attention, pointer network-based policy network for a system agent to assign participants to tasks in mobile crowdsourcing [91].

4.3 Vehicle Repositioning

Vehicle repositioning from a single-driver perspective (i.e., taxi routing) has a relatively long history of research since taxi service has been in existence long before the emergence of rideshare platforms. Likewise, research on RL-based approaches for this problem also appeared earlier than that on system-level vehicle repositioning.

For the taxi routing problem, the agent is naturally modeled as a driver, and the objective thus focuses on optimizing individual reward. Common reward definitions include trip fare [86], net profit (income - operational cost) [113], idle cruising distance [28], and ratio of trip mileage to idle cruising mileage [27]. Earlier works [34, 121, 113, 28] optimize the objective within a horizon up to the next successful match (i.e., A-to-B and A-to-C in Figure 4), but it is now more common to consider a long-term horizon, where an episode usually consists of a trajectory over one day [56, 95, 43]. We illustrate these concepts in Figure 4.

The type of actions of an agent depends on the physical abstraction adopted. A simpler and more common way of representing the spatial world is a grid system, square or hexagonal[4] [34, 121, 113, 27, 56, 86, 42, 95]. In this setting, the action space is the set of neighboring cells (often including the current cell). [94] explain the justification for this configuration. Determination of the specific destination point is left to a separate process, e.g., pick-up points service [42]. The more realistic abstraction is a road network, in which the nodes can be intersections or road segments [28, 132, 142, 87]. The action space is the adjacent nodes or edges of the current node. This approach supports a turn-by-turn guiding policy but requires more map information at run time.

Most of the papers adopt a tabular value function, so the state is necessarily low-dimensional, including spatiotemporal information and sometimes additional categorical statuses. [95] has a boolean in the state to indicate if the driver is assigned to consecutive requests since its setting allows a driver to be matched before completing a trip request. [86, 142] have the direction from which the driver arrives at the current location. For deep RL-based approaches [121, 42], richer contextual information, such as SD distributions in the neighborhood, can go into the state.

The learning algorithms are fairly diverse but are all value-based. By estimating the various parameters (e.g., matching probability, passenger destination probability) to compute the transition probabilities, [86, 132, 95, 142] adopt a model-based approach and use value iterations to solve the MDP. [95] further uses inverse RL to learn the unit-distance operational cost. Model-free methods are also common, e.g., Monte Carlo learning [113], Q-learning [34, 27], and DQN [121]. [42] is a hybrid approach in that it performs an action tree search at the online planning stage using estimated

[4]The hexagonal grid system is the industry standard.

Paper	Agent	State	Action	Reward	Algorithm	Environment	Notes
[126]	driver	assignment to a specific order or idle	location, time	trip price	tabular TD(0) for learning state values offline + Hungarian method for generating the assignment online	deployed in production; multi-agent, agent-level simulation	
[118]	driver	assignment to a specific order or idle	location, time, SD features	trip price	offline DQN for matching, CFPT network for transfer learning	single-agent simulation	single-vehicle problem
[103], [81]	driver	assignment to a specific order or idle	location, time, SD features	trip price	CVNet (deep TD-like algorithm) for learning state values offline + Hungarian method for generating the assignment online	deployed in production; multi-agent, agent-level simulation	hierarchical sparse coarse coding, cerebellar embedding of spatial info, Lipschitz regularization on network
[36]	driver, system	matching a driver to an order, reposition a driver; matching and repositioning done sequentially	global info of all drivers and open orders	trip price, reposition cost	DQN, PPO	multi-agent, agent-level simulation	attention mechanism to extract global state info into a context vector
[54]	driver	assignment to a specific order or idle	location, time, is_available	trip price	mean-field MARL, AC method	homogeneous vehicles within the same grid cell	The mean action is represented by the peers' destination distribution.
[44]	worker: hex cell manager: group of hex cells (one layer)	worker: ranking for match and reposition manager: abstract goal for workers	number of vehicles, orders, entropy, reposition-guided vehicles, distributions of trip prices and durations in the given hex cell	manager: total driver income + specifically designed quantity to promote high order response rate worker: intrinsic reward for following the goal generated by manager	hierarchical MARL	homogeneous vehicles within the same grid cell	multiple managers, each manager communicates with multiple workers multi-head attention mechanism for coordination
[140]	driver	a trip tuple: (origin cell, destination cell, trip duration, price)	cell index, number of idle vehicles, orders, distribution of trip destinations in the given hex cell	trip price	independent learning with KL divergence regularization	homogeneous vehicles within the same grid cell	
[49]	trip request	match or delay	global: number of idle vehicles, open requests, expected arrival rates of requests and drivers in each cell local: location, cumulative waiting time, expected pick-up distance	local reward based on the ultimate outcome of matching (whether or not matched or cancelled): trip value, pick-up distance, and match window time Global reward is based on average local reward. Final reward is convex combination of local and global rewards. The rewards have to be updated at the end of the epoch.	DQN, PPO, A2C, ACER with delayed reward. Whole episode trajectories are sampled from replay buffer.	agent-based simulation with delayed matching feature	
[116]	system	expected length of current batch	current nodes in the bipartite graph, current batch size	the sum of the edge weights in the batch	restricted Q-learning	DiDi GAIA data set	The adaptive batch-based matching has a guarantee on competitive ratio
[80]	system	match the current batch or continue to batch (decision made at every time interval)	number of batched requests and vehicles and estimated arrival rates of demand and supply in each cell	for each time interval, the negative of total matching wait time for all batched requests and the total pick-up wait time saved (by delaying the current batch)	ACER that combines on-policy updates (through a queuing-based simulator) with off-policy updates	Shanghai taxi data	
[92]	vehicle	remaining battery level when available, next available time and location, global time	matching, EV (charging)	trip price - pick-up cost - charging cost	similar to CVNet [103]	synthetic data	assumes decomposibility of system value into vehicle values
[52]	system	global time, new request info, state of each vehicle	joint matching, reposition, and charging (EV) for each vehicle	revenue - travel cost	DQN + attention mechanism over vehicles embeddings (similar to [36])	NYC taxi data with taxi zones	Decision epoch is either a new request arrives or a vehicle becomes idle.
[1]	system	supply-demand counts in spatiotemporal discretized space	joint matching and reposition, EV (charging)	revenue - charging expense	ADP with value function approx. on post-decision states. Value function approx. by hierarchical aggregation.	simulation with New Jersey ride-hailing trip data	

Table 2: Summary of literature for Online Matching.

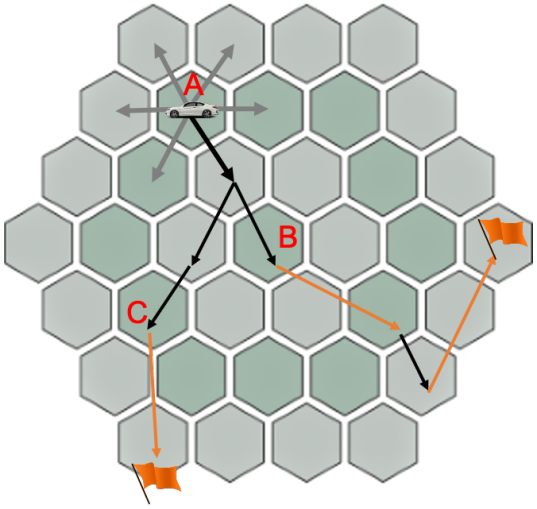

Figure 4: Illustration of the (single-agent) taxi routing problem on a hexagon grid system. The vehicle at its origin position A has the option to reposition to one of the neighboring and current cells. The black arrows represent reposition (idle cruising), and in the two scenarios, the vehicle is matched to a trip request at B and C respectively. The orange arrows represent trip moves, and the orange flags are where the episodes terminate (for long-term horizons).

matching probabilities and a separately learned state value network. [28] is in a similar spirit by augmenting the multi-arm bandits with Monte Carlo tree search.

The problem formulation most relevant to the ridesharing service provider is system-level vehicle repositioning. Similar to order matching, the ridesharing platform reviews the vehicles' states at fixed time intervals which are significantly longer than those for order matching. See Figure 2 for illustration. Idle vehicles that meet certain conditions, e.g., being idle for sufficiently long time and not in the process of an existing reposition task, are sent reposition recommendations, which specify the desired destinations and the associated time windows. The motivation here is to explicitly modify the current distribution of the available vehicles so that collectively they are better positioned to fulfill more requests more efficiently in the future. Figure 5 explains the idea with a concrete example. If the vehicles reposition independently (following the orange arrows), they both move to the orange-circled area and there will be a surplus of supply while the demand in the green-circled area will not be served. In contrast, if the vehicles coordinate and the one in the south repositions by the blue arrow, both vehicles will be matched, and all the requests are served.

The agent can be either the platform or a vehicle, latter of which calls for a MARL approach. All the works in this formulation have global SD information (each vehicle and request's status or SD distributions) in the state of the MDP, and a vehicle agent will additionally have its spatiotemporal status in the state. The rewards are mostly the same as in the taxi routing case, except that [64] consider the monetized passenger waiting time. The actions are all based on grid or taxi zone systems.

The system-agent RL formulation has only been studied very recently, in view of the intractability of the joint action space of all the vehicles. To tackle this challenge of scalability, [26] decompose the system action into a sequence of atomic actions corresponding to passenger-vehicle matches and vehicle repositions. The MDP encloses a 'sequential decision process' in which all feasible atomic actions are executed to represent one system action, and the MDP advances its state upon complete of the system action. They develop a PPO algorithm for the augmented MDP to determine the sequence of the atomic actions. The system policy in [64] produces a reposition plan that specifies the number of vehicles to relocate from zone i to j so that the action space is independent from the number of agents (at the expense of additional work at execution). The agent network, trained by a batch AC method, outputs a value for each OD pair, which after normalization gives the percentage of vehicles from each zone to a feasible destination.

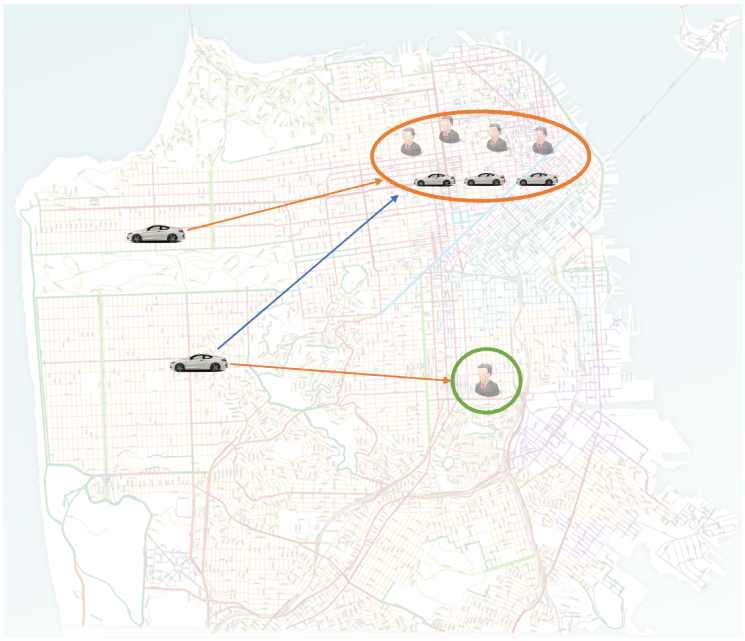

Figure 5: Illustration of system-level vehicle repositioning. The requests in the orange-circled and green-circled areas appear in the future w.r.t. the time of repositioning. The empty vehicles are existing ones in the orange-circled area. The orange and blue arrows represent potential reposition moves.

The vehicle-agent approaches have to address the coordination issue among the agents. [56] develop contextual DQN and AC methods, in which coordination is achieved by masking the action space based on the state context and splitting the reward accrued in a grid cell among the multiple agents within the same cell. [75] treat the global state in grid as image input and develop an independent DQN method. They argue that independent learning, equipped with global state information, works quite well compared to an MPC-based approach. The zone structure in [58] is constructed by clustering a road-connectivity graph. A single vehicle agent is trained with contextual deep RL and generates sequential actions for the vehicles. [137] also train a single DQN agent for all agents, but with global KL distance between the SD distributions similar to [140]. The DQN agent is put in tandem with QRewriter, another agent with a Q-table value function that converts the output of DQN to an improved action. [94] approach the MARL problem with bilevel optimization: The bottom level is a mean-field AC method [54] with the reward function coming from a platform reward design mechanism, which is tuned by the top level Bayesian optimization. Agent coordination is done by a central module in [15], where a vehicle agent executes a mix of independent and coordinated actions. The central module determines the need for coordination based on SD gaps, and explicit coordination is achieved by solving an assignment problem to move vehicles from excess zones to deficit zones.

Existing literature typically assumes the drivers' full compliance to reposition, i.e., the autonomous vehicle setting. How non-compliance affects the overall performance of a reposition algorithm still requires further investigation. It is also interesting and practically necessary to investigate incentives design and strategies that facilitate the repositioning process.

4.4 Route Guidance (Navigation)

Routing in this paper refers to low-level navigation decisions on a road network, typically with output of matching and repositioning algorithms as input. The road network, combined with traffic conditions on the links (exhibited as link costs), forms the traffic network which is a non-stationary stochastic network [cite]. It is known that standard shortest-path algorithms do not find the path with minimum expected cost in this case, and the optimal route is not a simple route but a policy [Hall 1986; Kim 2005]. There are two types of set-up for the routing problem, depending on the decision review time. In the first type of set-up, each vehicle on the road network selects a route

Paper	Type	Agent	State (in addition to ST info)	Action	Reward	Episode	Algorithm	Coordination	Data
[86]	taxi	driver	direction from which driver arrives at the current location	neighboring cells in a grid system	trip fare	long-term	model-based, VI to solve MDP	-	
[34]	taxi	driver	-	neighboring cells in a grid system	trip fare - reposition cost	from idle to completion of next trip(s) and being idle again	Q-learning	-	
[113]	taxi	driver	-	neighboring cells in a grid system	trip fare - reposition cost	up to the next match	MC learning	-	taxi log data from Singapore
[121]	taxi	driver	SD contextual info	neighboring cells in a grid system	saved idle time compared to a counterfactual simulation without reposition. Penalized if no match after reposition.	up to the next match	DQN with greedy action Compared with MINLP and SAR (simple anticipatory rebalancing) on avg passenger wait time.	-	
[28]	taxi	driver	current node on road network	adjacent node or edge	idle cruising distance till the next passenger	up to the next match	MAB + MCTS	-	
[27]	taxi	driver	location, occupied, parking, vacant	idle cruise to a neighboring cell in the grid system, carrying passenger to destination, waiting	trip mileage/idle cruising mileage Problem objective: effective driving ratio = total trip mileage / total idle cruising mileage	long-term (day)	Q-learning	-	Beijing taxi data 2013
[142]	taxi	driver	current road segment, time, previous road segment	adjacent node or edge	trip fare	long-term	model-based, VI to solve MDP	-	a year of taxi data from a major city in China
[133]	taxi	driver	current node on road network	adjacent node or edge	trip fare - operational cost	long-term (day)	model-based, VI to solve MDP; use parallel matrix operations to accelerate computation	-	Shanghai taxi trajectory data in the morning of a weekday
[95]	taxi, system	driver	boolean: whether or not assigned to consecutive requests	neighboring cells in a grid system	trip fare - operational cost operational cost per unit distance learned through IRL	long-term (day)	model-based, VI to solve MDP; inverse RL to learn unit distance op cost	Sequentially make decision for each driver. Adjust the matching prob of each driver's MDP, and solve again.	Beijing ride-hailing trajectory data from DiDi over three weekdays
[42]	taxi	vehicle	SD contextual info in the current cell	neighboring cells in a grid system	trip fare - reposition cost	long-term	offline CVNet + decision-time action search	-	DiDi ride-hailing data

Table 3: Summary of literature for Vehicle Repositioning (taxi routing).

Paper	Type	Agent	State (in addition to ST info)	Action	Reward	Episode	Algorithm	Coordination	Data
[56]	system	driver	global SD contextual features in all cells	neighboring cells in a grid system	trip fare, shared when multiple agents are in the same grid cell	long-term	contextual DQN, AC	contextual state features, action space pruned by context	4 weeks of DiDi data in Chengdu, China
[75]	system	driver	global SD state discretized into cells, treated as an image	reachable cells in a grid system within the reposition cycle	weighted number of pick-ups - reposition time	long-term	independent DQN	-	NYC taxi data
[94]	system	vehicle	-	neighboring cells in a grid system	trip fare	long-term	bilevel optimization: top Bayesian optimization to update reward param, bottom mean-field MARL (AC)	mean-field MARL	NYC taxi data
[43]	system	vehicle	SD contextual info in the current and neighboring cells	neighboring cells in a grid system	trip fare - reposition cost	long-term	deep SARSA	stochastic policy through softmax of action values, SD regularization to action values	DiDi ride-hailing data
[64]	system	system	SD info for each zone	reposition plan: number of repositioned vehicles for each OD pair in a zone map	monetized passenger waiting time	long-term	batch AC	central decision-making	NYC taxi data
[26]	system	system	status of every vehicle and request	atomic action: driver-passenger match or driver-destination match (reposition); system action: sequence of atomic actions	trip fare - operational cost	long-term (day)	PPO applied to MDP with Sequential Decision Process embedded: global actions decomposed into sequential atomic ones	central decision-making	DiDi data: 5 regions, 1000 cars, 360 minutes
[58]	system	vehicle	discrete zone structure constructed by clustering a road connectivity graph	neighboring zones	trip fare	long-term	contextual DQN with shared value function	vehicle actions generated sequentially	real-world taxi data
[137]	system	vehicle	global SD distributions	neighboring cells in a grid system	global KL distance between SD distributions	long-term	DQN + Q-table in tandem with shared value function	global state features	DiDi data
[15]	system	vehicle	cell in an ST table	neighboring cells in a grid system, wait	trip fare - operational cost	long-term	SARSA-like policy evaluation	solve an assignment problem of surplus and deficits in terms of SD gap	NYC taxi data

Table 4: Summary of literature for Vehicle Repositioning (system reposition).

for a given OD pair from a set of feasible routes. The decision is only reviewed and revised after a trip is completed. Hence, it is called route planning or route choice. When the routes for all the vehicles are planned together, it is equivalent to assigning the vehicles to each link in the network, and hence, the problem is called traffic assignment problem (TAP). In the second type of set-up, the routing decision is made at each intersection to select the next outbound road (link) to enter. These are real-time adaptive navigation decisions for vehicles to react to the changing traffic state of the road network. The problem corresponding to this set-up is called dynamic routing, dynamic route choice, or route guidance.

Routing on a road network is a typical multi-agent problem, where the decisions made by one agent has influence on the other agents' performance, simply in that the congestion level of a link depends directly on the number of vehicles passing through that link at a given time and has direct impact on the travel time for all the vehicles on that link within the same time interval. The literature for route planning and TAP often consider the equilibrium property of the algorithms when a population of vehicles adopt them. TAP is typically from a traffic manager's (i.e., system's) perspective. Its goal is to reach system equilibrium (SE, or system optimal solution). Some works focus on route planning or TAP from an individual driver's perspective and maximize individual reward. These algorithms try to reach user equilibrium (UE) or Nash equilibrium, under which no agent has the incentive to change its policy because doing so will not achieve higher utility. This is the best that selfish agents can achieve but may not be optimal for the system.

Value-based RL is by far the most common approach for route planning and TAP aiming to reach UE. In the MDP formulation, the agent is a vehicle (or equivalently, a task) with a given OD pair. The objective is to minimize the total travel time for an individual vehicle (task) [63, 84, 139], i.e., the agent is selfish. The immediate reward is the total travel time of a trip for an individual and a particular run. This MDP is stateless, so strictly speaking, it is a multi-arm bandits or contextual bandits problem [53] if considering time as a contextual feature. The action to take at decision time is to select a route from the set of feasible routes for the associated OD pair [84, 139, 7]. The value function that governs the route choice decisions represents the long-term expected travel time for the trip identified by the given OD pair. Due to the multi-agent nature, the environment w.r.t. each agent is non-stationary in that the reward function is changing with the policy updates from the other agents. Empirical convergence to UE is demonstrated by [84]. [139] further develops a Bush-Mosteller RL scheme for MARL and formally establishes its UE convergence property. We also highlight some unique features of the papers. [84] considers a different objective from the common and minimizes the driver's regret. To do that, the Q-learning updates are modified using the estimated action regret, which can be computed by local observations and global travel time information communicated by an app. [7] proposes a hybrid method, with Q-learning for individual agents and Genetic Algorithm for reaching system equilibrium, minimizing the average travel time over different trips in the network. This method is thus able to achieve SE. [63] adopts Q-iterations with a model set-up similar to that of dynamic routing to be discussed next.

Most applications of RL to routing concern with the *dynamic routing* (DR) problem. The MDP is modeled around a vehicle agent. The basic state information is the traffic state of the current node (i.e., intersection). Some works consider state features of the neighboring nodes [50, 65] so that the agent has a broader view of the environment. The action space comprises the set of outbound links (i.e., roads) or adjacent nodes from the current node, so the policy provides a turn-by-turn navigation guidance until the destination is reached. While it is most common to use travel time on a link as the reward function, [109, 29] stand out by defining a new form called difference reward, which is the difference in average travel time on a link with and without the agent in the system. This applied to only a reward function dependent on the number of agents using the traversed link. In particular, travel distance cannot be used to define a difference reward. Whether solving a specific formulation achieves UE or SE depends on the reward function used. The average travel time on a link is a global reward because it is an aggregate of local rewards (i.e., individual travel times) of all the agents on that link. The difference reward, by definition, is a global reward that also reflects individual effect. If all the agents in the system learn by global reward [109, 29, 93], then the system is expected to achieve SE. Otherwise, the agents learn by their local rewards, and we will have UE or Nash equilibrium [50, 131, 65, 8, 120].

Most works in the literature adopt Q-learning or its variant as the training algorithm. We report several notable developments. To tackle the sample efficiency issue of online model-free methods, [65] propose an offline batch RL approach (fitted Q-iterations) with a tree-based function approximator

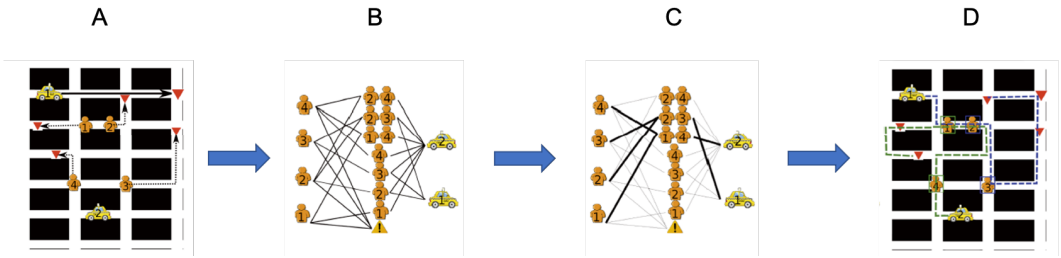

Figure 6: Illustration of the ride-pooling matching process adapted from [3]. Stage A shows the state of the current vehicles and requests. Vehicle 1 has one passenger on board with her destination at the top right-hand corner, while vehicle 2 is empty. The feasible combinations of passengers and the vehicles feasible to serve them are determined at stage B. The corresponding assignment graph is set up and solved at stage C. Stage D shows the resulting routes to fulfill all four new requests as well as the existing trip.

(Extreme Randomized Trees) that empirically shows good convergence property. Hierarchical methods have also been adopted to address the complexity of a large-scale problem. In [120], the global road network is divided into sub-networks by differential evolution-based clustering. The top-level network contains only the boundary nodes of the original network. The top-level policy produces the destination node for a sub-network. The sub-level policy provides link-level guidance to reach its sub-destination. [94] adopts a bilevel optimization scheme. At the lower level, a mean-field MARL algorithm solves for the dynamic routing problem for the travelers, while at the upper level, a Bayesian optimization module optimizes the control (i.e., reward parameter of the travelers) by the city planner.

4.5 Ride-pooling (Carpool)

Ride-pooling optimization typically concerns with matching, repositioning, routing (see e.g., [138, 4, 3, 106]). The RL literature has primarily focused on the first two problems. The ride-pooling matching problem differs from that in Section 4.2 in that a combination of multiple passengers, and hence their combined trip, can be matched to a vehicle that may or may not be empty. See stages B and C in Figure 6 from [3] for an illustration. The repositioning problem is similar to the ride-hailing case, except that the objective is to optimize some pooling-specific metrics that we define next. The routing problem solves for the sequence of pick-ups and drop-offs given the assigned passengers for a vehicle. The routing problem could also concern with route guidance on the road network. See stage D in Figure 6.

Many works have multiple objectives and define the reward as a weighted combination of several quantities, with hand-tuned weight parameters. *Passenger wait time* is the duration between the request time and the pick-up time. *Detour delay* is the extra time a passenger spends on the vehicle due to the participation in the ride-pooling. In some cases, these two quantities define the feasibility of a potential pooled trip instead of appearing in the reward [89]. *Effective trip distance* is the travel distance between the origin and destination of a trip request, should it be fulfilled without ride-pooling. [133] consider minimizing passenger wait time, detour delay, and lost demand. [31] maximize the number of passengers served. [45] maximize the total effective trip distance within an episode, which is just the number of served requests weighted by individual trip distance. Considering a fixed number of requests within an episode (hence fixed maximum effective distance), this metric reflects the efficiency of ride-pooling. [2, 32, 98, 33] all attempt to minimize the SD mismatch, passenger wait time, reposition time, detour delay, and the number of vehicles used. In addition, [32] considers the fleet profit, and [98] study a more general form of ride-pooling, where a passenger can hop among different vehicles to complete a trip, with each vehicle completing one leg. They further consider the number of hops and the delay due to hopping.

The state of an agent usually consists of global SD information, similar to that for matching and reposition, but the vehicle status contains key additional information of occupancy and OD's of the passengers on board.

Paper	Type	Equilibrium	Agent	State	Action	Reward	Algorithm	Road Network
[63]	TAP	UE	vehicle	current node (intersection)	an adjacent link	travel time on the link	Q-iteration. The route is constructed by following the decision at each intersection.	grid network
[7]	TAP	SE	vehicle	-	route from feasible routes for the OD pair	total travel time on the chosen route	Hybrid method: Q learning for individual agent + Genetic Algorithm for system equilibrium, minimizing avg travel time over different trips.	-
[84]	TAP	UE (empirical convergence)	vehicle	-	route from feasible routes for the OD pair	total travel time on the chosen route	Q-learning with action-regret updates to minimize driver's total regret	Braess graphs, OW network
[139]	TAP	UE (theoretical convergence established)	vehicle	-	route from feasible routes for the OD pair	total travel time on the chosen route	B-M RL scheme, similar to a MARL algo with individual reward.	Nguyen–Dupuis network
[50]	DR	UE	vehicle	node, time, binary congestion status vector for all links	an adjacent node	cost accrued by traversing the link	parameters of MDP estimated from data, MDP solved by value iterations	Southeast Michigan network and traffic data
[109]	DR	SE	vehicle	just a single link	start time on the link	difference reward (uplift): with and without the agent in the system	Q-learning	-
[131]	DR	UE	vehicle	current node	an adjacent link; action instruction received from the intersection in real time	travel time on the link	Similar approach to [63], but incremental update is done by a step of SARSA. The updates are done in real time and value functions are sync-ed at each intersection, which is an independent traffic management module.	SOUND/4U simulator based on the road network of Kurosaki, Kitakyushu in Japan
[29]	DR	SE	vehicle	current node	an adjacent link	individual reward: negative travel time experienced by the agent on a link difference reward: as in [109]	Applied to a more sophisticated network than aamas08 paper. Results show that DQ-learning outforms IQ-learning.	abstract network topology
[65]	DR	UE	vehicle	current node, time, discrete congestion states vector of the arcs at most 2 steps away.	an adjacent node	negative travel time experienced by the agent on a link	offline batch RL: fitted Q-iterations with tree-based function approximator (Extreme Randomized Trees). Compared with model-based Q-iterations.	Sioux Falls network
[8]	DR	UE	trip (OD pair)	current node	an adjacent link	negative travel time experienced by the agent on a link	Independent Q learning compared with successive average method	OW network, Sioux Falls network
[120]	DR	UE	vehicle	next approaching node, destination node	an adjacent link	negative travel time experienced by the agent on a link	tabular Q-learning, global road network clustered into subnetworks by differential evolution-based clustering Top-level network contains only boundary nodes of the original network. Top network policy produces the 'destination' node for a subnetwork. Subnetwork policy provides link-level guidance to reach its sub-destination.	SUMO simulator with various networks in Japan and US
[94]	DR	SE (avg travel time), UE (travel distance)	vehicle	current node, time	an adjacent link	negative average travel time of all agents or travel distance on a link	Bilevel optimization: Lower level - mean field MARL to solve for dynamic routing for travelers Upper level - Bayesian optimization to optimize controls by city planners	SUMO with Manhattan network

Table 5: Summary of literature for Route Guidance.

The action space depends on whether the agent is modeled at vehicle level or system level. Existing works all require that a vehicle drops off all the passengers on board according to a planned route before a new round of pooling. An individual vehicle agent can then match to a feasible group of passengers (in terms of capacity and detour delay) [45], reposition to another location [2, 32, 33], or both [31]. A system-level agent has to make action decisions for the entire fleet together [133, 89]. The feasible combinations of passengers are typically determined by a separate process based on a pairwise shareability graph [3]. (See illustration in Figure 6.)

Papers with vehicle-level policy commonly train a single agent and apply to all the vehicles independently (see e.g., [33]). DQN is a convenient choice of training algorithm for this setting. For system-level decision-making, both [133] and [89] employ an ADP approach and consider matching decisions only. [133] follow a similar strategy as [97] and use a linear approximation for the value function. In contrast, [89] decompose the system value function into vehicle-level ones and adopts a neural network for the individual value function, which is updated by mini-batch stochastic gradient descent similar to that in DQN.

It has become increasingly clear that dynamic routing and route planning in the context of ride-pooling require specific attention. In particular, there are two aspects unique to ride-pooling. First, the trips are known only at their request times. Hence, the routes taken by the pooled vehicles (i.e., the sequences of pick-ups and drop-offs) have to be updated dynamically to account for the newly joined passengers. [106, 125] formulate the route planning problem for ride-pooling and develop efficient DP-based route insertion algorithms for carpool. In Section 4.6, we will see that this is also part of the stochastic dynamic vehicle routing problem. Second, within a given route plan, the route taken by a pooled vehicle from an origin to a destination can affect the chance and quality of its future pooling. Hence, dynamic routing (or route choice) between an OD pair can be optimized in that direction, e.g., [134] goes beyond the shortest-path to make route recommendations for better chance of pooling. [30] evaluates the SAMoD system proposed in [31] in a microscopic environment based on SUMO with a traffic congestion-aware (non-RL) routing component. We expect to see more RL-based algorithms for the ride-pooling dynamic routing problems.

4.6 Vehicle Routing Problem (VRP)

VRP has close connection to the ridesharing problem in that variants of VRP could serve as a subroutine of the ride-pooling problem and could even used to model the entire ridesharing problem itself. So reviewing the RL literature for VRP is not only for the completeness of this survey but also essential for one to appreciate the complexity and challenges in tackling ridesharing via RL.

VRP has many variants in its rich literature, so it is important to be clear on their differences and on the variant that each paper claims to solve. The basic setup of a VRP consists of a transportation network $\mathcal{G} := (\mathcal{V}, \mathcal{E}, w)$ and a fleet of K vehicles, where $\mathcal{V}$ is the set of nodes (customer and depot locations), and $\mathcal{E}$ is the set of edges such that $e_{ij} \in \mathcal{E}$ indicates that it is possible to travel from node v_i to node v_j. $w(x_{ij}) := w_{ij}$ is the edge cost, typically distance or travel time. The depot $v_0 \in \mathcal{V}$ is a special node where all the vehicles depart and return at. The vanilla VRP is to find an optimal set of disjoint routes (one for each vehicle) that start and end at the depot, that collectively cover all the nodes, and whose total cost (summing over all the edges in those routes) is minimized. A *traveling salesman problem* (TSP) is a special (simplified) instance of VRP, in which there is no depot, and the fleet consists of a single vehicle. In a *capacitated* VRP (CVRP), each node has a demand quantity to be fulfilled by one of the vehicles. Each vehicle has a limited capacity, and it starts from the depot with a full load of goods to fulfill the demand of the nodes on its route. The total capacities of the fleet is sufficiently large, and all the demand has to be fulfilled. A CVRP with *split delivery* allows the demand of each node to be fulfilled by multiple vehicles. In a VRP with *time windows*, each node has a delivery window within which the node has to be visited or its demand has to be satisfied. If the time window constraints are *soft*, they can be violated with the price of a penalty that contributes to the total cost function. In some variants of the VRP, the goods for the demand of a node (destination) has to be picked up from another designated non-depot location (origin) before being delivered to it. This is known as a VRP with *pick-up and delivery*, also known as the *dial-a-ride* problem (DARP). If the fleet consists of electric vehicles (EV), the set of nodes also include charging stations, and each EV has a limited battery capacity, before the depletion of which the EV has to reach a charging station to recharge. In practical situations, a VRP or variant can be *stochastic* and *dynamic* (SDVRP),

i.e., its parameters (e.g., demand and travel time) are uncertain, and the requests are not known at the beginning but are revealed sequentially throughout the problem period.

The goal of this section is not to provide a complete survey of the VRP literature but rather to point out the representative or unique works that adopt RL to solve VRPs. A recent review of RL-based methods for solving stochastic dynamic VRP can be found in [Hildebrandt, et al., 2021] and a more general one in [112].

Single-vehicle v.s. multi-vehicle problems CVRP may appear in different forms, and sometimes the subtle differences may not be stated clearly. Most papers solve the single-vehicle problem where there is only one active vehicle at any time. In the capacitated single vehicle problem, the vehicle can make multiple tours (i.e., passing through the depot multiple times) to fulfill all the demand, but the number of tours is not set in advance. In the multi-vehicle case, a fleet of K vehicles are active simultaneously. For static VRPs, if the number of tours in the single-vehicle case is fixed, then it is equivalent to the multi-vehicle counterpart by treating each tour as a separate vehicle. (For problems with time windows, this equivalence can be achieved by resetting the clock every time a new tour starts.) Otherwise, they are not equivalent in general because the number of tours in the optimal solution for the single-vehicle problem may not be N. For dynamic problems where the requests are not all known a-priori, it is not possible to generate the fixed number of tours in sequence, since one cannot insert a new request to a previous tour. In this case, equivalence can only be achieved by keeping each tour on the same clock and updating the routes with the newly appeared requests at each time step. As we will see below, this would render essentially a multi-vehicle algorithm.

The majority of the RL-based methodsfor VRP models the agent as a vehicle with the system-state visibility. The state thus consists of two types of information: the vehicle state, which includes the vehicle's current location and remaining capacity (for pick-up and delivery, e.g., [112, 41, 46]) or inventory (for homogeneous goods delivery, e.g., [73, 51, 22]); the system state, which contains the locations of the customer nodes, the demand at each node, and the unserved customers. For pick-up and delivery problems, the system state instead contains the pick-up and delivery locations of the orders. In the case of EVs, the vehicle state additionally contains the vehicle's battery level, and the system state also includes the locations of the charging stations and the number of vehicles available (not in charging). The action of the agent is to specify the next stop (pick-up/delivery location or charging station in the case EV) to visit for the current vehicle. The sequence of actions form a route for the vehicle. When multiple routes/tours are required, the different routes are separated by the insertion of the depot [73, 25, 51, 55]. For (dynamic) multi-vehicle problem where decisions for all the vehicles are made at each time step, the agents would generate their actions sequentially to avoid conflicting actions [41, 135]. Since the objective of VRPs is typically to minimize total travel distance, the reward is naturally defined as the negative travel distance. For problems with (soft) time window constraints, the negative penalty for constraint violation is added to the reward.

Typically, these methods adopt an encoder-decoder agent network architecture. The encoder is responsible for encoding part or all of the state information into an embedding vector (or context), which, potentially with additional input state features, is fed into the decoder to generate the action one at a time. [10] develops an policy network based on the pointer network [114], which consists of an RNN encoder and an RNN decoder. The major novelty over a sequence-to-sequence architecture is that the decoder uses attention mechanism to attend over the embeddings of the input nodes to generate the probability distribution over the input space, thus eliminating the distance disparity in the output with respect to the input, a feature that is particularly suitable for solving TSP and VRP. To reduce the complexity of the encoder and avoid imposing a sequence on the input state features (e.g., customer locations), which is unnecessary in routing problems, [73] modifies the pointer network with a non-sequential encoder which simply embeds each individual input node. They incorporate the policy network into an AC method and validate the design on a CVRP with split delivery. A few more recent works have adopted this network structure. [41] uses structural graph embedding (Struct2Vec) for the encoder, since their agent's state additionally contains a vehicle tour graph. In [55], the encoder has 1D convolution and graph embedding for the input nodes, followed by an attention layer. [25] includes edge features in the state besides the node features of the transportation network. Their encoder is based on graph convolution network with both node and edge inputs. Another work with significant novelty is [51], which develops a policy network with a transformer-based encoder and a self-attention-based decoder to use in a PG method (REINFORCE) with the baseline computed from deterministic greedy rollout. This training framework has also been adopted by [135]

for multi-vehicle VRP with soft time windows, [55] for EV VRP with time windows, and [25], which jointly trains an MLP-based binary classifier on edge encoding with the policy network output as labels. They have tested their method on a CVRP with 400 nodes, the largest among the reviewed works.

For SDVRP, [112] argues that it is a more convenient model, which also aligns better with popular approaches to this problem, that the action contains also the route plan information. They define a new variant of MDP, called route-based MDP, in which the state includes the route plan from the last epoch, and the action contains the updated route plan in addition to the next stop to visit. The 'immediate' reward becomes the difference in route value between the old and new plans. Following this line, [46] models a system agent whose state includes the cost for the remaining route for each vehicle, and the agent assigns a new request to a vehicle at each decision epoch. The rerouting after matching is solved by simulated annealing for VRP. Under this framework, one only needs to learn an action-value function to generate the matching decisions. The algorithm is tested on a multi-vehicle SDVRP with pick-up/delivery and time windows.[5] In a somewhat similar spirit but for static CVRP, the MDP action in [22] is to generate one route (tour). The value network consists of dense layers and ReLU activation and is representable by mixed-integer linear constraints so that the action can be computed through solving a Prize Collecting TSP by MIP.

The connection between VRP and ridesharing exists at both local and fundamental levels. As a subproblem in ride-pooling, the rerouting problem after a new passenger is matched to the vehicle is a TSP with pick-up and delivery (TSPPD), which is one-vehicle single-tour instance of CVRP with pick-up and delivery.

(Multi-vehicle) ride-pooling is basically stochastic dynamic multi-vehicle CVRP with pick-up and delivery. Although there are no explicit time windows, cancellation may occur if waiting time is too long. Ride-hailing is also a special case where the vehicles all have unit capacity, and in this case, matching and routing merge into one single problem. So the ridesharing problem is an SDCVRP, except that repositioning is an intervention strategy not considered in SDCVRP.

4.7 Model Predictive Control

A related family of forward-looking methods that share a similar spirit with model-based RL is model predictive control (MPC). MPC exploits the environment dynamics model more explicitly in that it solves a multi-step planning problem over the prediction horizon at decision time and executes the first-step decision. The process is repeated in a rolling-horizon fashion. For ridesharing applications, both MPC and model-based RL involve online prediction of supply and demand using models trained on historical data. Model-based RL uses the prediction to learn the environment model, whereas MPC utilizes the information to generate the planning problem, which is typically solved by a mixed-integer-linear-programming (MILP) solver.

MPC-based repositioning methods have been developed for both regular ridesharing [40, 136, 69, 19] and ride-pooling [85, 70]. These works solve an MILP for online planning and separately-trained SD forecast models. [136, 40, 19] solve for zone-level decisions. [136] additionally considers charging constraints for electric vehicles. [69, 70] directly solves for vehicle-level decisions, and [85] solves two optimization problems, the second of which assigns individual vehicles according to the plan generated by the first.

4.8 Data Sets & Environments

The problems in ridesharing are highly practice-oriented, and results from toy data sets or environments may present a very different picture from those in reality. Hence, real-world data sets and realistic simulators backed up by them are instrumental to research in RL algorithms for these problems.

The most commonly used data sets are those made available by NYC TLC (Taxi & Limousine Commission) [105]. This large public data repository contains trip records from several different services, Yellow Taxi, Green Taxi, and FHV (For-Hire Vehicle), from 2009 to 2020. The Yellow Taxi data is the most frequently used for various studies. The FHV trip records are submissions from the TLC-licensed bases (e.g., Uber, Lyft) and have a flag indicating pooled trips offered by Uber

[5] This method can be regarded as one for the matching problem in ride-pooling described in Section 4.5.

Paper	Type	State	Action	Reward	Network	Algorithm	Problem Size
[73]	single-vehicle	location and demand of each request	the next request to visit	negative travel distance	Non-sequential encoder for the input with RNN decoder that attends over over the input space (Pointer network without an RNN encoder)	AC	Single-vehicle Capacitated VRP with split delivery: one active vehicle at a time
[51]	single-vehicle	coordinates and original demand of each node, remaining demand of each node, remaining capacity of the vehicle	next stop for a given vehicle	negative travel distance	transformer encoder with input of coordinates and demand of each node + self-attention-based decoder with additional input of remaining demands and capacity at time t	REINFORCE with greedy rollout baseline	TSP and Capacitated VRP with split delivery, 100 nodes
[6]	single-vehicle	current pickup location, vehicle's location and remaining capacity, orders' locations, statuses, waiting times, and values	accept an order, pick up an accepted order, wait	value of delivered order (accept, pickup, deliver) - cost (waiting, traveling, penalty)	two dense layers NN	APE-X DQN [37]	stochastic and dynamic CVRP with pick-up/delivery and time windows, 8 x 8 map, 5 orders 3 pick-up locations
[41]	multi-vehicle	system state (available requests, charging station output, vehicles' status(location, battery levels, next stops)), vehicle tour graph	next stop for a given vehicle; The vehicles generate actions sequentially at each time step.	expected objective value for a tour	pointer network for actor, another critic network; Network architecture is similar to NeurIPS-18 paper, but with structural graph embedding (Struct2Vec) for the encoder.	A3C	multi-vehicle dynamic VRP with pick-up and delivery for EVs: 200 random requests, 100 vehicles
[135]	multi-vehicle	same as [51]	same as [51]; The vehicles generate actions sequentially at each time step.	negative travel distance + negative constraint violation penalty	same as [51]	same as [51]	Multi-vehicle VRP with soft time windows (no split delivery): 150 nodes, 5 vehicles
[112]	single-vehicle	vehicle location, time, num of passengers onboard, info of in-process and outstanding requests, route plan from last epoch	the next stop to visit and the new route plan; The paper defines a new variant of MDP called route-based MDP.	difference in route value between the old route plan and the new one	N.A.	insert new request s into the current route and use variable neighborhood search to improve the route	SDVRP with pick-up and delivery (DDARP)
[46]	multi-vehicle	includes the cost for the remaining route for each vehicle	matching a new order to a vehicle. Rerouting after matching is done by simulated annealing for VRP	cost diff between two consecutive decisions	not specified	NN-based TD learning with experience replay (like in [103]) to learn the action-value function	Multi-vehicle dynamic VRP with pick-up/delivery and delivery windows: 48 nodes, 2 vehicles, avg 22 orders/day
[22]	single-vehicle	the remaining unvisited nodes	to generate one route (tour) through solving a Prize Collecting TSP by MIP	negative tour distance	Value network consists of dense layers + ReLU activation (representable by mixed-integer linear constraints)	MC policy iteration: rollout N trajectories, fit a new NN	CVRP: 51 nodes
[25]	single-vehicle	nodes (location, demand), edges (distance, adjacency)	generate one node at a time sequentially; The resulting sequence may have multiple depot occurrences for different tours.	negative travel distance	GCN-based encoder with both node and edge features; GRU-based decoder similar to the pointer network as policy network and MLP-based decoder on the edge encoding as classifier	REINFORCE with greedy rollout baseline [51] to train the policy network; Cross-entropy loss to train the binary classifier of route edges with policy output as labels	CVRP: 400 nodes
[55]	multi-vehicle	For time t, the state of each vertex (location, time window, remaining demand), and global variables (time, battery level of the active vehicle, number of EVs not in charging)	Next stop for the current route; Unlike [41] the routes of the vehicles are generated sequentially. Every time the depot appears in the sequence, the system time is reset to 0.	negative travel distance + negative penalties for constraint violations	Encoder with 1D conv layer and graph embedding for the nodes and attention layer; LSTM-based decoder	REINFORCE with greedy rollout baseline [51]	EV with time window and charging. Within the planning horizon, a vehicle can visit the depot only once: C100, S12, EV12

Table 6: Summary of literature for VRP.

Pool and Lyft Line. The pick-up and drop-off locations are represented by taxi zones. Manhattan, for example, is divided into 64 zones. There is no driver ID associated with the trip records, so reconstructing historical driver-based trajectories is not possible. A similar subset of the NYC FHV data is available at [47], with GPS coordinates for pick-up and drop-off locations. In addition, travel time data between OD pairs can be obtained through Uber Movement [111].

A more recent rideshare (Transportation Network Providers, TNPs) data set is published by Chicago Data Portal [78]. This data set contains trips, drivers, and vehicles data reported by Transportation Network Providers (TNP, or rideshare companies) in Chicago from 2018. Trip origin and destinations are represented by census tracts. Times are rounded to the nearest 15 minutes. Fares are rounded to the nearest $2.50 and tips are rounded to the nearest $1.00. The driver and vehicle data are not joinable with the trip data.

Developing ridesharing simulators has been a line of research itself. [129] offers a comprehensive review of recent works on ridesharing simulation models, most of them covering a subset of considerations on the number of passengers, the pre-/post-match passenger cancellation behaviors, and driver acceptance/rejection behaviors. In [129], a sophisticated event-based simulation framework is proposed to capture all aspects of the behavior modeling. Although the 'ridesharing' in their paper is known as the hitch service, where the driver is on her own trip as well, the modeling framework is general and accommodates the ridesharing setting in this survey. [15] offer a Gym-compatible, open-source ride-hailing environment for training dispatching agents. The evaluation simulation environment for the KDD Cup 2020 competition is available for public access through the DiDi decision intelligence simulation platform [23]. Although not yet open-sourced, this simulation environment supports both matching and vehicle repositioning tasks and accepts input algorithms through a Python API.

5 Challenges and Opportunities

Given the state of the current literature, we discuss a few challenges and opportunities that we feel crucial in advancing RL for ridesharing.

5.1 Ride-pooling

As seen in Section 4.5, the reward function in ride-pooling is often a hand-tuned combination of multiple objectives. It is desirable to have a principled way to determine the best weighting scheme automatically, potentially leveraging inverse RL and multi-objective learning techniques [143, 5]. Methods for learning to make matching decisions are still computationally intensive [89, 133], in part due to the need to use VRP solver to determine feasible actions (combination of passengers). Moreover, all existing works assume that the action set is pre-determined, and some make only high-level decisions of reposition and serving new passengers or not. A more sophisticated agent may be called for to figure out, for example, how to dynamically determine the desirable passenger combination to match to a vehicle and the routes to take thereafter. Ride-pooling pricing [48], a hard pricing problem itself, is tightly coupled with matching. A joint pricing-matching algorithm for ride-pooling is therefore highly pertinent. As mentioned in Section 4.5, it is also highly anticipated to go beyond using generic routing algorithms and to tailor them to ride-pooling with RL.

5.2 Joint Optimization

The rideshare platform is an integrated system, so joint optimization of multiple decision modules leads to better solutions that otherwise unable to realize under separate optimizations, ensuring that different decisions work towards the same goal. We have already seen development on RL for joint matching-reposition [36, 44] and with ride-pooling [31], pricing-matching [17], and pricing-reposition [110]. An RL-based method for fully joint optimization of all major modules is highly expected. Meanwhile, this also requires readiness from the rideshare platforms in terms of system architecture and organizational structure.

5.3 Heterogeneous Fleet

With the wide adoption of electric vehicles and the emergence of autonomous vehicles, we are facing an increasingly heterogeneous fleet on rideshare platforms. Electric vehicles have limited operational range per their battery capacities. They have to be routed to a charging station when the battery level is low (but sufficiently high to be able to travel to the station). Autonomous vehicles may run within a predefined service geo-fence due to their limited ability (compared to human drivers) to handle complex road situations. For an RL-based approach, a heterogeneous fleet means multiple types of agents with different state and action spaces. Specific studies are required to investigate how to make such a heterogeneous fleet cooperate well to complement each other and maximize the advantage of each type of vehicles.

5.4 Simulation & Sim2Real

Simulation environments are fundamental infrastructure for successful development of RL methods. Despite those introduced in Section 4.8, simulation continues to be a significant engineering and research challenge. We have rarely seen comparable simulation granularity as that of the environments for traffic management, (e.g., SUMO [59], Flow [123]) or autonomous driving (e.g., SMARTS [141], CARLA [24]).[6] The opportunity is an agent-based microscopic simulation environment for ridesharing that accounts for both ride-hailing and carpool, as well as driver and passenger behavior details, e.g., price sensitivity, cancellation behavior, driver entrance/exit behavior. None of the existing public/open-source simulators supports pricing decisions. Those simulators described in the pricing papers all have strong assumptions on passenger and driver price elasticities. A better way might be to learn those behaviors from data through, e.g., generative adversarial imitation learning [90] or inverse RL [67].

No publicly known ridesharing simulation environment has sufficiently high fidelity to the real world to allow an agent trained entirely in it to deploy directly to production. Several deployed works [81, 42] in Section 4 have all adopted offline RL for learning the state value functions and online planning. The robotics community has been extensively investigating ways to close the reality gap [108, 68]. Sim2real transfer algorithms for ridesharing agents are urgently sought after.

5.5 Non-stationarity

We have seen in Sections 4.2 and 4.3 that RL algorithms deployed to real-world systems generally adopt offline training - once the value function or the policy is deployed, it is not updated until the next deployment. Value functions trained offline using a large amount of historical data are only able to capture recurring patterns resulted from day-on-day SD changes. However, the SD dynamics can be highly non-stationary in that one-time abrupt changes can easily occur due to various events and incidents, e.g., concerts, matches, and even road blocks by traffic accidents. To fully unleash the power of RL, practical mechanisms for real-time on-policy updates of the value function (e.g., [104]) is required. In view of the low risk tolerance of production systems in general, sample complexity, computational complexity, and robustness are the key challenges that such methods have to address.

5.6 Business Strategies

The research problems in the ridesharing domain are closely associated with how the ridesharing platforms run the operations. There can be multiple alternative product forms to achieve the same goals or address the same challenges, and they inherently define different research problems. Surge pricing, for example, is a pricing strategy during the peak hours to address the severe shortage of supply with respect to the surging demand. We have explained its motivation in Section 4.1. While surge pricing is a common practice nowadays, it is not the only strategy that the ridesharing platforms adopt. In China, passenger requests submitted to some ridesharing services are queued if there are no vacant vehicles around to immediately serve the requests. As long as the shortage in supply persists, the requests in queue are served on a first-come-first-serve basis, and the passengers are able to see their real-time positions in the queue. The queuing mechanism is perceived in the local market as a more socially acceptable mechanism during the peak hours than surge pricing. Several operational decision questions immediately come up, e.g., how large an area each queue should cover, if the

[6] [30] evaluate the ridesharing algorithms in SUMO, but the environment is not public.

coverage should be dynamically updated, and when the incoming requests should start queuing. In the US, Lyft has released and adopted the Personal Power Zone (PPZ) feature for the drivers [76], which redistributes the additional income from surge pricing to better incentivize drivers for repositioning. The drivers are presented with recommended areas defined by geo-fence, where if they reposition to and stay sufficiently long to serve at least one request, they will receive a bonus calculated based on the estimated income through surge pricing for that area. How the PPZs should be defined and how the incentives should be set are some of the new operational decisions to be made. Innovation in product and business operations in ridesharing will continue to raise new challenging research problems.

6 Closing Remarks

We have surveyed the RL literature for the core problems in ridesharing and discussed some open challenges and future opportunities. As one may have noticed, most of the literature has just appeared in the last four years, and we expect it to continue growing and updating rapidly.

References

[1] L. Al Kanj, J. Nascimento, and W. B. Powell. Approximate dynamic programming for planning a ride-hailing system using autonomous fleets of electric vehicles. *European Journal of Operational Research*, 284(3):1088–1106, 2020.

[2] A. Alabbasi, A. Ghosh, and V. Aggarwal. Deeppool: Distributed model-free algorithm for ride-sharing using deep reinforcement learning. *IEEE Transactions on Intelligent Transportation Systems*, 20(12):4714–4727, 2019.

[3] J. Alonso-Mora, S. Samaranayake, A. Wallar, E. Frazzoli, and D. Rus. On-demand high-capacity ride-sharing via dynamic trip-vehicle assignment. *Proceedings of the National Academy of Sciences*, 114(3):462–467, 2017.

[4] J. Alonso-Mora, A. Wallar, and D. Rus. Predictive routing for autonomous mobility-on-demand systems with ride-sharing. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3583–3590. IEEE, 2017.

[5] S. Arora and P. Doshi. A survey of inverse reinforcement learning: Challenges, methods and progress. *Artificial Intelligence*, page 103500, 2021.

[6] B. Balaji, J. Bell-Masterson, E. Bilgin, A. Damianou, P. M. Garcia, A. Jain, R. Luo, A. Maggiar, B. Narayanaswamy, and C. Ye. Orl: Reinforcement learning benchmarks for online stochastic optimization problems. *arXiv preprint arXiv:1911.10641*, 2019.

[7] A. L. Bazzan and C. Chira. A hybrid evolutionary and multiagent reinforcement learning approach to accelerate the computation of traffic assignment. In *AAMAS*, pages 1723–1724, 2015.

[8] A. L. Bazzan and R. Grunitzki. A multiagent reinforcement learning approach to en-route trip building. In *2016 International Joint Conference on Neural Networks (IJCNN)*, pages 5288–5295. IEEE, 2016.

[9] X. Bei and S. Zhang. Algorithms for trip-vehicle assignment in ride-sharing. In *Thirty-second AAAI conference on artificial intelligence*, 2018.

[10] I. Bello, H. Pham, Q. V. Le, M. Norouzi, and S. Bengio. Neural combinatorial optimization with reinforcement learning. *arXiv preprint arXiv:1611.09940*, 2016.

[11] C. Berner, G. Brockman, B. Chan, V. Cheung, P. Dębiak, C. Dennison, D. Farhi, Q. Fischer, S. Hashme, C. Hesse, et al. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*, 2019.

[12] D. Bertsimas and G. Perakis. Dynamic pricing: A learning approach. In *Mathematical and computational models for congestion charging*, pages 45–79. Springer, 2006.

[13] K. Bimpikis, O. Candogan, and D. Saban. Spatial pricing in ride-sharing networks. *Operations Research*, 67(3):744–769, 2019.

[14] E. Brown. *The Ride-Hail Utopia That Got Stuck in Traffic - WSJ*, 2020. https://www.wsj.com/articles/the-ride-hail-utopia-that-got-stuck-in-traffic-11581742802.

[15] H. A. Chaudhari, J. W. Byers, and E. Terzi. Learn to earn: Enabling coordination within a ride hailing fleet. *Proceedings of IEEE International Conference on Big Data*, 2020.

[16] C. Chen, F. Yao, D. Mo, J. Zhu, and X. M. Chen. Spatial-temporal pricing for ride-sourcing platform with reinforcement learning. *Transportation Research Part C: Emerging Technologies*, 130:103272, 2021.

[17] H. Chen, Y. Jiao, Z. Qin, X. Tang, H. Li, B. An, H. Zhu, and J. Ye. Inbede: Integrating contextual bandit with td learning for joint pricing and dispatch of ride-hailing platforms. In *2019 IEEE International Conference on Data Mining (ICDM)*, pages 61–70. IEEE, 2019.

[18] Y. Chen, Y. Qian, Y. Yao, Z. Wu, R. Li, Y. Zhou, H. Hu, and Y. Xu. Can sophisticated dispatching strategy acquired by reinforcement learning?-a case study in dynamic courier dispatching system. *arXiv preprint arXiv:1903.02716*, 2019.

[19] S.-F. Cheng, S. S. Jha, and R. Rajendram. Taxis strike back: A field trial of the driver guidance system. In *AAMAS'18: Proceedings of the 17th International Conference on Autonomous Agents and Multiagent Systems, Stockholm, July 10*, volume 15, pages 577–584, 2018.

[20] T. Choi. *Uber and Lyft to turn the wheels on car ownership: industry experts*, 2020. Reuters.

[21] G. B. Dantzig and J. H. Ramser. The truck dispatching problem. *Management science*, 6(1):80–91, 1959.

[22] A. Delarue, R. Anderson, and C. Tjandraatmadja. Reinforcement learning with combinatorial actions: An application to vehicle routing. *arXiv preprint arXiv:2010.12001*, 2020.

[23] DiDi. Didi decision intelligence simulation platform. 2021. "https://outreach.didichuxing.com/Simulation".

[24] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun. Carla: An open urban driving simulator. In *Conference on robot learning*, pages 1–16. PMLR, 2017.

[25] L. Duan, Y. Zhan, H. Hu, Y. Gong, J. Wei, X. Zhang, and Y. Xu. Efficiently solving the practical vehicle routing problem: A novel joint learning approach. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 3054–3063, 2020.

[26] J. Feng, M. Gluzman, and J. Dai. Scalable deep reinforcement learning for ride-hailing. *IEEE Control Systems Letters*, 2020.

[27] Y. Gao, D. Jiang, and Y. Xu. Optimize taxi driving strategies based on reinforcement learning. *International Journal of Geographical Information Science*, 32(8):1677–1696, 2018.

[28] N. Garg and S. Ranu. 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*, pages 1425–1434, 2018.

[29] R. Grunitzki, G. de Oliveira Ramos, and A. L. C. Bazzan. Individual versus difference rewards on reinforcement learning for route choice. In *2014 Brazilian Conference on Intelligent Systems*, pages 253–258. IEEE, 2014.

[30] M. Guériau, F. Cugurullo, R. A. Acheampong, and I. Dusparic. Shared autonomous mobility on demand: A learning-based approach and its performance in the presence of traffic congestion. *IEEE Intelligent Transportation Systems Magazine*, 12(4):208–218, 2020.

[31] M. Guériau and I. Dusparic. Samod: Shared autonomous mobility-on-demand using decentralized reinforcement learning. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 1558–1563. IEEE, 2018.

[32] M. Haliem, G. Mani, V. Aggarwal, and B. Bhargava. A distributed model-free ride-sharing algorithm with pricing using deep reinforcement learning. In *Computer Science in Cars Symposium*, pages 1–10, 2020.

[33] M. Haliem, G. Mani, V. Aggarwal, and B. Bhargava. A distributed model-free ride-sharing approach for joint matching, pricing, and dispatching using deep reinforcement learning. *arXiv preprint arXiv:2010.01755*, 2020.

[34] M. Han, P. Senellart, S. Bressan, and H. Wu. Routing an autonomous taxi with reinforcement learning. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 2421–2424, 2016.

[35] A. Haydari and Y. Yilmaz. Deep reinforcement learning for intelligent transportation systems: A survey. *IEEE Transactions on Intelligent Transportation Systems*, 2020.

[36] J. Holler, R. Vuorio, Z. Qin, X. Tang, Y. Jiao, T. Jin, S. Singh, C. Wang, and J. Ye. Deep reinforcement learning for multi-driver vehicle dispatching and repositioning problem. In J. Wang, K. Shim, and X. Wu, editors, *2019 IEEE International Conference on Data Mining (ICDM)*, pages 1090–1095. Institute of Electrical and Electronics Engineers, Washington, DC, 2019.

[37] D. Horgan, J. Quan, D. Budden, G. Barth-Maron, M. Hessel, H. Van Hasselt, and D. Silver. Distributed prioritized experience replay. *arXiv preprint arXiv:1803.00933*, 2018.

[38] B. Hu, M. Hu, and H. Zhu. Surge pricing and two-sided temporal responses in ride hailing. *Manufacturing & Service Operations Management*, 2021.

[39] M. Hu and Y. Zhou. Dynamic type matching. *Rotman School of Management Working Paper*, (2592622), 2020.

[40] R. Iglesias, F. Rossi, K. Wang, D. Hallac, J. Leskovec, and M. Pavone. Data-driven model predictive control of autonomous mobility-on-demand systems. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–7. IEEE, 2018.

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

[42] Y. Jiao, X. Tang, Z. T. Qin, S. Li, F. Zhang, H. Zhu, and J. Ye. A deep value-based policy search approach for real-world vehicle repositioning on mobility-on-demand platforms. In *NeurIPS 2020 Deep Reinforcement Learning Workshop*, 2020.

[43] Y. Jiao, X. Tang, Z. T. Qin, S. Li, F. Zhang, H. Zhu, and J. Ye. Real-world ride-hailing vehicle repositioning using deep reinforcement learning. *Transportation Research Part C: Emerging Technologies*, 130:103289, 2021.

[44] J. Jin, M. Zhou, W. Zhang, M. Li, Z. Guo, Z. Qin, Y. Jiao, X. Tang, C. Wang, J. Wang, 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*, pages 1983–1992, 2019.

[45] I. Jindal, Z. T. Qin, X. Chen, M. Nokleby, and J. Ye. Optimizing taxi carpool policies via reinforcement learning and spatio-temporal mining. In *2018 IEEE International Conference on Big Data (Big Data)*, pages 1417–1426. IEEE, 2018.

[46] W. Joe and H. C. Lau. Deep reinforcement learning approach to solve dynamic vehicle routing problem with stochastic customers. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 30, pages 394–402, 2020.

[47] Kaggle. Uber pickups in new york city - trip data for over 20 million uber (and other for-hire vehicle) trips in nyc. 2017. "`https://www.kaggle.com/fivethirtyeight/uber-pickups-in-new-york-city`".

[48] J. Ke, H. Yang, X. Li, H. Wang, and J. Ye. Pricing and equilibrium in on-demand ride-pooling markets. *Transportation Research Part B: Methodological*, 139:411–431, 2020.

[49] J. Ke, H. Yang, J. Ye, et al. Learning to delay in ride-sourcing systems: a multi-agent deep reinforcement learning framework. *IEEE Transactions on Knowledge and Data Engineering*, 2020.

[50] S. Kim, M. E. Lewis, and C. C. White. Optimal vehicle routing with real-time traffic information. *IEEE Transactions on Intelligent Transportation Systems*, 6(2):178–188, 2005.

[51] W. Kool, H. Van Hoof, and M. Welling. Attention, learn to solve routing problems! *arXiv preprint arXiv:1803.08475*, 2018.

[52] N. D. Kullman, M. Cousineau, J. C. Goodson, and J. E. Mendoza. Dynamic ride-hailing with electric vehicles. *Transportation Science*, 2021.

[53] L. Li, W. Chu, J. Langford, and R. E. Schapire. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th international conference on World wide web*, pages 661–670, 2010.

[54] M. Li, Z. Qin, Y. Jiao, Y. Yang, Z. Gong, J. Wang, C. Wang, G. Wu, and J. Ye. Efficient ridesharing order dispatching with mean field multi-agent reinforcement learning. In *To appear in Proceedings of the 2019 World Wide Web Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 2019.

[55] B. Lin, B. Ghaddar, and J. Nathwani. Deep reinforcement learning for the electric vehicle routing problem with time windows. *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[56] K. Lin, R. Zhao, Z. Xu, and J. Zhou. Efficient large-scale fleet management via multi-agent deep reinforcement learning. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1774–1783, 2018.

[57] L.-J. Lin. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine learning*, 8(3-4):293–321, 1992.

[58] Z. Liu, J. Li, and K. Wu. Context-aware taxi dispatching at city-scale using deep reinforcement learning. *IEEE Transactions on Intelligent Transportation Systems*, 2020.

[59] P. A. Lopez, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y.-P. Flötteröd, R. Hilbrich, L. Lücken, J. Rummel, P. Wagner, and E. Wießner. Microscopic traffic simulation using sumo. In *The 21st IEEE International Conference on Intelligent Transportation Systems*. IEEE, 2018.

[60] M. Lowalekar, P. Varakantham, and P. Jaillet. Online spatio-temporal matching in stochastic and dynamic domains. *Artificial Intelligence*, 261:71–112, 2018.

[61] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. *arXiv preprint arXiv:1706.02275*, 2017.

[62] H. Ma, F. Fang, and D. C. Parkes. Spatio-temporal pricing for ridesharing platforms. *ACM SIGecom Exchanges*, 18(2):53–57, 2020.

[63] M. K. Mainali, K. Shimada, S. Mabu, and K. Hirasawa. Optimal route based on dynamic programming for road networks. *Journal of Advanced Computational Intelligence and Intelligent Informatics*, 12(6):546–553, 2008.

[64] C. Mao, Y. Liu, and Z.-J. M. Shen. Dispatch of autonomous vehicles for taxi services: A deep reinforcement learning approach. *Transportation Research Part C: Emerging Technologies*, 115:102626, 2020.

[65] C. Mao and Z. Shen. A reinforcement learning framework for the adaptive routing problem in stochastic time-dependent network. *Transportation Research Part C: Emerging Technologies*, 93:179–197, 2018.

[66] MarketsAndMarkets. *Ride Sharing Market by Type (E-hailing, Station-Based, Car Sharing & Rental), Car Sharing (P2P, Corporate), Service (Navigation, Payment, Information), Micro-Mobility (Bicycle, Scooter), Vehicle Type, and Region - Global Forecast to 2025*, 2018.

[67] E. Mazumdar, L. J. Ratliff, T. Fiez, and S. S. Sastry. Gradient-based inverse risk-sensitive reinforcement learning. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 5796–5801. IEEE, 2017.

[68] B. Mehta, T. Deleu, S. C. Raparthy, C. J. Pal, and L. Paull. Curriculum in gradient-based meta-reinforcement learning. *arXiv preprint arXiv:2002.07956*, 2020.

[69] 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*, 13(2):463–478, 2016.

[70] J. Miller and J. P. How. Predictive positioning and quality of service ridesharing for campus mobility on demand systems. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1402–1408. IEEE, 2017.

[71] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937, 2016.

[72] 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*, 518(7540):529–533, 2015.

[73] M. Nazari, A. Oroojlooy, L. Snyder, and M. Takác. Reinforcement learning for solving the vehicle routing problem. In *Advances in Neural Information Processing Systems*, pages 9839–9849, 2018.

[74] A. Y. Ng, S. J. Russell, et al. Algorithms for inverse reinforcement learning. In *Icml*, volume 1, page 2, 2000.

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

[76] H. Y. Ong, D. Freund, and D. Crapis. Driver positioning and incentive budgeting with an escrow mechanism for ridesharing platforms. *arXiv preprint arXiv:2104.14740*, 2021.

[77] E. Özkan and A. R. Ward. Dynamic matching for real-time ride sharing. *Stochastic Systems*, 10(1):29–70, 2020.

[78] C. D. Portal. Chicago transportation network providers (rideshare) data. 2020. "`https://data.cityofchicago.org/Transportation/Transportation-Network-Providers-Trips/m6dm-c72p`".

[79] W. B. Powell. *Approximate Dynamic Programming: Solving the curses of dimensionality*, volume 703. John Wiley & Sons, 2007.

[80] G. Qin, Q. Luo, Y. Yin, J. Sun, and J. Ye. Optimizing matching time intervals for ride-hailing services using reinforcement learning. *To appear in Transportation Research: Part C*, 2021.

[81] Z. Qin, X. Tang, Y. Jiao, F. Zhang, Z. Xu, H. Zhu, and J. Ye. Ride-hailing order dispatching at didi via reinforcement learning. *INFORMS Journal on Applied Analytics*, 50(5):272–286, 2020.

[82] Z. Qin, H. Zhu, and J. Ye. Reinforcement learning for ridesharing: A survey. In *appearing in Proceedings of the IEEE Intelligent Transportation Systems Conference*, 2021.

[83] C. Raju, Y. Narahari, and K. Ravikumar. Reinforcement learning applications in dynamic pricing of retail markets. In *IEEE International Conference on E-Commerce, 2003. CEC 2003.*, pages 339–346. IEEE, 2003.

[84] G. d. O. Ramos, A. L. Bazzan, and B. C. da Silva. Analysing the impact of travel information for minimising the regret of route choice. *Transportation Research Part C: Emerging Technologies*, 88:257–271, 2018.

[85] C. Riley, P. Van Hentenryck, and E. Yuan. Real-time dispatching of large-scale ride-sharing systems: Integrating optimization, machine learning, and model predictive control. *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence (IJCAI)*, 2020.

[86] H. Rong, X. Zhou, C. Yang, Z. Shafiq, and A. Liu. The rich and the poor: A markov decision process approach to optimizing taxi driver revenue efficiency. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 2329–2334, 2016.

[87] S. Schmoll and M. Schubert. Semi-markov reinforcement learning for stochastic resource collection. *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence (IJCAI)*, 2020.

[88] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[89] S. Shah, M. Lowalekar, and P. Varakantham. Neural approximate dynamic programming for on-demand ride-pooling. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 507–515, 2020.

[90] W. Shang, Y. Yu, Q. Li, Z. Qin, Y. Meng, and J. Ye. Environment reconstruction with hidden confounders for reinforcement learning based recommendation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 566–576, 2019.

[91] W. Shen, X. He, C. Zhang, Q. Ni, W. Dou, and Y. Wang. Auxiliary-task based deep reinforcement learning for participant selection problem in mobile crowdsourcing. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 1355–1364, 2020.

[92] J. Shi, Y. Gao, W. Wang, N. Yu, and P. A. Ioannou. Operating electric vehicle fleet for ride-hailing services with reinforcement learning. *IEEE Transactions on Intelligent Transportation Systems*, 21(11):4822–4834, 2019.

[93] Z. Shou and X. Di. Multi-agent reinforcement learning for dynamic routing games: A unified paradigm. *arXiv preprint arXiv:2011.10915*, 2020.

[94] Z. Shou and X. Di. Reward design for driver repositioning using multi-agent reinforcement learning. *Transportation research part C: emerging technologies*, 119:102738, 2020.

[95] Z. Shou, X. Di, J. Ye, H. Zhu, H. Zhang, and R. Hampshire. Optimal passenger-seeking policies on e-hailing platforms using markov decision process and imitation learning. *Transportation Research Part C: Emerging Technologies*, 111:91–113, 2020.

[96] D. Silver and D. Hassabis. Alphago: Mastering the ancient game of go with machine learning. *Research Blog*, 9, 2016.

[97] H. P. Simao, J. Day, A. P. George, T. Gifford, J. Nienow, and W. B. Powell. An approximate dynamic programming algorithm for large-scale fleet management: A case application. *Transportation Science*, 43(2):178–197, 2009.

[98] A. Singh, A. Alabbasi, and V. Aggarwal. A distributed model-free algorithm for multi-hop ride-sharing using deep reinforcement learning. *arXiv preprint arXiv:1910.14002*, 2019.

[99] M. Smith. *Here?s how long you have to wait for an Uber or Lyft in DC*, 2019. `https://wtop.com/dc-transit/2019/12/how-long-you-have-to-wait-for-an-uber-or-lyft-in-d-c/`.

[100] J. Song, Y. J. Cho, M. H. Kang, and K. Y. Hwang. An application of reinforced learning-based dynamic pricing for improvement of ridesharing platform service in seoul. *Electronics*, 9(11):1818, 2020.

[101] R. S. Sutton. Learning to predict by the methods of temporal differences. *Machine learning*, 3(1):9–44, 1988.

[102] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[103] X. Tang, Z. Qin, F. Zhang, Z. Wang, Z. Xu, Y. Ma, H. Zhu, and J. Ye. 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*, pages 1780–1790, 2019.

[104] X. Tang, F. Zhang, Z. Qin, Y. Wang, D. Shi, B. Song, Y. Tong, H. Zhu, and J. Ye. Value function is all you need: A unified learning framework for ride hailing platforms. *arXiv e-prints*, pages arXiv–2105, 2021.

[105] TLC. Nyc taxi & limousine commission trip record data. 2020. "`https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page`".

[106] Y. Tong, Y. Zeng, Z. Zhou, L. Chen, J. Ye, and K. Xu. A unified approach to route planning for shared mobility. *Proceedings of the VLDB Endowment*, 11(11):1633, 2018.

[107] Y. Tong, Z. Zhou, Y. Zeng, L. Chen, and C. Shahabi. Spatial crowdsourcing: a survey. *The VLDB Journal*, 29(1):217–250, 2020.

[108] R. Traoré, H. Caselles-Dupré, T. Lesort, T. Sun, N. Díaz-Rodríguez, and D. Filliat. Continual reinforcement learning deployed in real-life using policy distillation and sim2real transfer. *arXiv preprint arXiv:1906.04452*, 2019.

[109] K. Tumer, Z. T. Welch, and A. Agogino. Aligning social welfare and agent preferences to alleviate traffic congestion. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems-Volume 2*, pages 655–662. Citeseer, 2008.

[110] B. Turan, R. Pedarsani, and M. Alizadeh. Dynamic pricing and fleet management for electric autonomous mobility on demand systems. *Transportation Research Part C: Emerging Technologies*, 121:102829, 2020.

[111] Uber. Uber movement. 2021. "`https://movement.uber.com/?lang=en-US`".

[112] M. W. Ulmer, J. C. Goodson, D. C. Mattfeld, and B. W. Thomas. On modeling stochastic dynamic vehicle routing problems. *EURO Journal on Transportation and Logistics*, 9(2):100008, 2020.

[113] T. Verma, P. Varakantham, S. Kraus, and H. C. Lau. Augmenting decisions of taxi drivers through reinforcement learning for improving revenues. In *Twenty-Seventh International Conference on Automated Planning and Scheduling*, 2017.

[114] O. Vinyals, M. Fortunato, and N. Jaitly. Pointer networks. In *Advances in Neural Information Processing Systems*, pages 2692–2700, 2015.

[115] H. Wang and H. Yang. Ridesourcing systems: A framework and review. *Transportation Research Part B: Methodological*, 129:122–155, 2019.

[116] Y. Wang, Y. Tong, C. Long, P. Xu, K. Xu, and W. Lv. Adaptive dynamic bipartite graph matching: A reinforcement learning approach. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*, pages 1478–1489. IEEE, 2019.

[117] Z. Wang, V. Bapst, N. Heess, V. Mnih, R. Munos, K. Kavukcuoglu, and N. de Freitas. Sample efficient actor-critic with experience replay. *arXiv preprint arXiv:1611.01224*, 2016.

[118] Z. Wang, Z. Qin, X. Tang, J. Ye, and H. Zhu. Deep reinforcement learning with knowledge transfer for online rides order dispatching. In *International Conference on Data Mining*. IEEE, 2018.

[119] C. J. Watkins and P. Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

[120] F. Wen, X. Wang, and X. Xu. Hierarchical sarsa learning based route guidance algorithm. *Journal of Advanced Transportation*, 2019, 2019.

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

[122] R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

[123] C. Wu, A. Kreidieh, K. Parvate, E. Vinitsky, and A. M. Bayen. Flow: Architecture and benchmarking for reinforcement learning in traffic control. *arXiv preprint arXiv:1710.05465*, page 10, 2017.

[124] T. Wu, A. D. Joseph, and S. J. Russell. Automated pricing agents in the on-demand economy. *University of California at Berkeley: Berkeley, CA, USA*, 2016.

[125] Y. Xu, Y. Tong, Y. Shi, Q. Tao, K. Xu, and W. Li. An efficient insertion operator in dynamic ridesharing services. *IEEE Transactions on Knowledge and Data Engineering*, 2020.

[126] Z. Xu, Z. Li, Q. Guan, D. Zhang, Q. Li, J. Nan, C. Liu, W. Bian, and J. Ye. 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*, pages 905–913. ACM, 2018.

[127] C. Yan, H. Zhu, N. Korolko, and D. Woodard. Dynamic pricing and matching in ride-hailing platforms. *Naval Research Logistics (NRL)*, 67(8):705–724, 2020.

[128] Y. Yang, R. Luo, M. Li, M. Zhou, W. Zhang, and J. Wang. Mean field multi-agent reinforcement learning. In *International Conference on Machine Learning*, pages 5571–5580. PMLR, 2018.

[129] R. Yao and S. Bekhor. A ridesharing simulation platform that considers dynamic supply-demand interactions. *arXiv preprint arXiv:2104.13463*, 2021.

[130] K.-L. A. Yau, J. Qadir, H. L. Khoo, M. H. Ling, and P. Komisarczuk. A survey on reinforcement learning models and algorithms for traffic signal control. *ACM Comput. Surv.*, 50(3), June 2017.

[131] S. Yu, J. Zhou, B. Li, S. Mabu, and K. Hirasawa. Q value-based dynamic programming with sarsa learning for real time route guidance in large scale road networks. In *The 2012 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7. IEEE, 2012.

[132] X. Yu, S. Gao, X. Hu, and H. Park. A markov decision process approach to vacant taxi routing with e-hailing. *Transportation Research Part B: Methodological*, 121:114–134, 2019.

[133] X. Yu and S. Shen. An integrated decomposition and approximate dynamic programming approach for on-demand ride pooling. *IEEE Transactions on Intelligent Transportation Systems*, 21(9):3811–3820, 2019.

[134] C. F. Yuen, A. P. Singh, S. Goyal, S. Ranu, and A. Bagchi. Beyond shortest paths: Route recommendations for ride-sharing. In *The World Wide Web Conference*, pages 2258–2269, 2019.

[135] K. Zhang, F. He, Z. Zhang, X. Lin, and M. Li. Multi-vehicle routing problems with soft time windows: A multi-agent reinforcement learning approach. *Transportation Research Part C: Emerging Technologies*, 121:102861, 2020.

[136] R. Zhang, F. Rossi, and M. Pavone. Model predictive control of autonomous mobility-on-demand systems. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1382–1389. IEEE, 2016.

[137] W. Zhang, Q. Wang, J. Li, and C. Xu. Dynamic fleet management with rewriting deep reinforcement learning. *IEEE Access*, 8:143333–143341, 2020.

[138] L. Zheng, L. Chen, and J. Ye. Order dispatch in price-aware ridesharing. *Proceedings of the VLDB Endowment*, 11(8):853–865, 2018.

[139] B. Zhou, Q. Song, Z. Zhao, and T. Liu. A reinforcement learning scheme for the equilibrium of the in-vehicle route choice problem based on congestion game. *Applied Mathematics and Computation*, 371:124895, 2020.

[140] M. Zhou, J. Jin, W. Zhang, Z. Qin, Y. Jiao, C. Wang, G. Wu, Y. Yu, and J. Ye. 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*, pages 2645–2653, 2019.

[141] M. Zhou, J. Luo, J. Villela, Y. Yang, D. Rusu, J. Miao, W. Zhang, M. Alban, I. Fadakar, Z. Chen, et al. Smarts: Scalable multi-agent reinforcement learning training school for autonomous driving. *arXiv preprint arXiv:2010.09776*, 2020.

[142] X. Zhou, H. Rong, C. Yang, Q. Zhang, A. V. Khezerlou, H. Zheng, Z. Shafiq, and A. X. Liu. Optimizing taxi driver profit efficiency: A spatial network-based markov decision process approach. *IEEE Transactions on Big Data*, 6(1):145–158, 2018.

[143] F. Zou, G. G. Yen, and C. Zhao. Dynamic multiobjective optimization driven by inverse reinforcement learning. *Information Sciences*, 575:468–484, 2021.