

An Integrated Reinforcement Learning and Centralized Programming Approach for Online Taxi Dispatching

Enming Liang, Kexin Wen, William H.K. Lam, Agachai Sumalee, and Renxin Zhong

Abstract—Balancing the supply and demand for ride-sourcing companies is a challenging issue, especially with real-time requests and stochastic traffic conditions of large-scale congested road networks. To tackle this challenge, this paper proposes a robust and scalable approach that integrates reinforcement learning (RL) and a centralized programming (CP) structure to promote real-time taxi operations. Both real-time order matching decisions and vehicle relocation decisions at the microscopic network scale are integrated within a Markov decision process framework. The RL component learns the decomposed state-value function, which represents the taxi drivers' experience, the off-line historical demand pattern, and the traffic network congestion. The CP component plans non-myopic decisions for drivers collectively under the prescribed system constraints to explicitly realize cooperation. Further, to circumvent sparse reward and sample imbalance problems over the microscopic road network, this paper proposed temporal-difference learning algorithm with prioritized gradient descent and adaptive exploration techniques.

A simulator is built and trained with the Manhattan road network and New York City yellow taxi data to simulate the real-time vehicle dispatching environment. Both centralized and decentralized taxi dispatching policies are examined with the simulator. This case study shows that the proposed approach can further improve taxi drivers' profits while reducing customers' waiting times compared to several existing vehicle dispatching algorithms.

Index Terms—Vehicle dispatching, Online vehicle routing, Multi-agent system, Deep reinforcement learning, Stochastic network traffic.

I. INTRODUCTION

TAXIS provide door-to-door transportation services for their passengers and serve as a common mode of transport around the world. Hong Kong has about 18,000 registered taxis for its population of 7.5 million, whereas other megacities like Beijing has almost 100,000 taxis in service. In traditional

taxi operations, vacant taxis cruise around the urban road network to search for customers, which wastes fuel and the taxi drivers' time, and causes additional congestion in the network. In addition, potential taxi passengers suffer from long waiting times in remote or less popular areas and/or during peak hours. Such spatially and temporally unbalanced demand and supply of taxi services has long been the key challenge for the taxi industry's ability to ensure the drivers' income and the passengers' service satisfaction. Owing to the recent advances in mobile sensing and wireless communication technology, on-demand ride-sourcing companies (e.g., Didi and Uber) have helped bridge taxi drivers (vehicles) and passengers to relieve this imbalance. Instead of the empirical decisions made by traditional taxi companies and taxi drivers, these platforms make centralized decisions to dispatch taxis to serve on-demand orders in a more efficient manner. However, given the highly stochastic nature of traffic conditions and real-time on-demand requests from potential customers, challenges confront the effective use of dynamic endogenous information for the design of strategies to operate these platforms, particularly in densely populated urban areas of megacities such as Beijing and Hong Kong.

A. Related works

The computer science and operations research communities are dedicated to settling this challenge in different ways. With complete/exact information (e.g., knowing departure time, origin and destination for every customer in the planning horizon), the classical vehicle routing problem (VRP) is formulated as an equivalent mathematical programming problem in the operations research community. Efforts are dedicated to the development of exact or heuristic algorithms to solve the mathematical programming models [19]. With well-calibrated or predicted near-future demand information (e.g., customers information in future 5 minutes), model predictive control or receding/rolling horizon control approaches are adopted to solve dispatching decisions in the planning horizon [20–22]. These works rely heavily on the accuracy of short-term prediction of travel demand, while its accuracy is another issue under highly dynamic traffic and the prediction can only be conducted in macroscopic regional level instead of specific road network [23].

While considering the dynamic supply and demand, especially on-demand customer requests, various approaches have been proposed to tackle the spatial-temporal distribution of

Manuscript received May 26, 2020; revised August 7, 2020, and January 21, 2021; accepted February 12, 2021. This work was jointly supported by National Natural Science Foundation of China (No. 72071214), the Research Grants Council of the Hong Kong Special Administrative Region, China (Project Nos. 152101/17E and R5029-18) and the CCF-DiDi Big-Data Joint Lab. (Corresponding author: Renxin Zhong.)

Renxin Zhong, Enming Liang, and Kexin Wen are with the Guangdong Key Laboratory of Intelligent Transportation Systems, School of Intelligent Systems Engineering, Sun Yat-sen University, Guangzhou 510275, China (e-mail: zhrenxin@mail.sysu.edu.cn).

William H.K. Lam is with Department of Civil and Environmental Engineering, The Hong Kong Polytechnic University, Hong Kong.

Agachai Sumalee is with the School of Integrated Innovation, Chulalongkorn University, Bangkok 10330, Thailand.

Digital Object Identifier

Table I: Relevant literature for the vehicle dispatching problem.

Literature	Agent	Scale	State	Action	Reward	Training Algorithm
[1]	vacant vehicles	Gr	T+L	assign orders	discounted fare	PE
[2]	vacant vehicles	Gr	T+L+local CF	assign orders	discounted fare	PE
[3]	vacant vehicles	Gr	T+L+local CF	assign orders	fare	DRL
[4]	unmatched orders	Gr(1 km)	T+L+local CF	delay orders	weighted reward	DRL
[5]	vacant vehicles	Gr(1.2 km)&Co	T+L+local CF	assign orders	weighted reward	DRL
[6]	vacant vehicles	Gr(1.2 km)	L+local CF	assign orders	fare	DRL
[7]	bipartite graph	Co	batch length	delay orders	edge weight	RL
[8]	vacant vehicles	Gr(1.2 km)	T+L+local CF	reposition	averaged revenue	DRL
[9]	dispatch center	Gr(150 m)	global CF	reposition	profit	DRL
[10]	vacant vehicles	Gr(irregular)	global counts	reposition	profit	DRL
[11]	vacant vehicles	Ne	T+L+arriving direction	reposition	profit	DP
[12]	single vehicle	Gr(5 km)	L	reposition	profit	DP
[13]	vacant vehicles	Gr(700 m)	T+L+indicator	reposition	profit	DP
[14]	single vehicle	Co	T+L+time budget	assign orders+routes	profit	ADP
[15]	vacant vehicles	Gr(0.5 mile)	T+L+battery level	assign orders+reposition+recharge	profit	ADP
[16]	all vehicles	Gr(5 mile)	T+L+local CF	price+assign orders+routes	profit	ADP
[17]	whole system	Gr	T+global CF	assign orders + reposition	fare	DRL
[18]	hierarchical grids	Gr(1.2 km)	hierarchical CF	assign orders + reposition	hierarchical reward	DRL
This work	vacant (almost) vehicles	Ne	vacant T+L	assign orders + reposition	profit	DRL+CP

Scale: Gr: Grid level; Co: Coordinate level; Ne: Network level

State: T: Timestamps; L: Location; CF: Contextual features

Algorithm: PE: Policy Evaluation; DRL: Deep Reinforcement Learning; DP: Dynamic Programming; ADP: Approximate Dynamic Programming; CP: Centralized Programming

travel demand and its uncertainty. Based on the dynamic programming (DP) framework [24, 25], the value of future demand uncertainty over a longer horizon is evaluated in the value function, which can be trained through approximate dynamic programming (ADP) [14–16, 26]. However, facing more complex real-life scenarios, learning-based methods seem to be promising (e.g., learning from demonstrations [27–30] and reinforcement learning (RL) [31, 32]). In particular, RL-based algorithms over data-driven simulation environments have been devised to solve various engineering problems such as robot control and navigation [33–35], smart grid management [36–38], and optimal power management for electric vehicles [39].

While for the taxi dispatching problem studied in this paper, two lines of research, order dispatching and vehicle relocation are conducted. For the order dispatching problem, the core is to matching vacant vehicles and real-time requested customers. Xu et al. [1] and Tang et al. [2] adopt policy evaluation to derive the region-specific time-dependent value function and verify its efficiency with an online A/B test¹. Although the online A/B test can verify the efficiency of the policy evaluation approaches developed in these works, the derived value function completely relied on historical drivers' decisions. While with a well-designed ride-sourcing simulator, different RL-based algorithms are proposed to solve the order dispatching decision [3, 5].

For the vehicle relocating problem, the core is to relocating vacant vehicles to promote future supply-demand balance. Under the single-agent view, the vehicle relocation problem is modeled as a Markov decision process (MDP) and solved by DP, which relies on explicitly fitted state transition probability

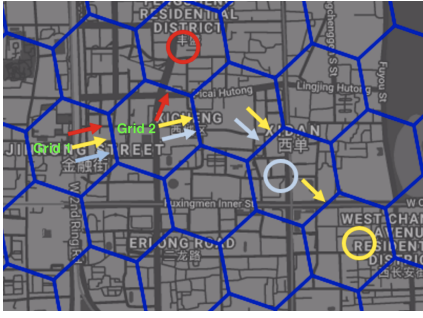
[11, 12]. Under the multi-agent view, the macroscopic vehicle relocation decisions for fleets are solved by various multi-agent algorithms, including deep Q-network (DQN) and advantage actor-critic (A2C) algorithms considering the contextual features [5], DQN-based framework that directly learns the value function from supply-demand heat maps [9], a count-based state representation framework [10] and hierarchical RL for both order dispatching and fleet management [18].

The aforementioned RL-based works are summarized in Table I. Although they are well-designed to tackle different challenges, there are still several unsolved issues.

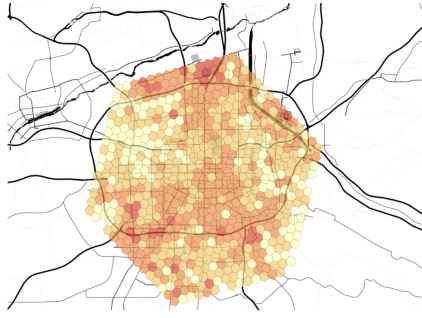
The first one is the joint optimization. As reviewed, most works regarded the order dispatching and vehicle relocating as two independent problems while investigate them separately. However, these two tasks are deeply coupled and mutually interacted. During the planning horizon, the dispatching and relocating decisions are alternately executed for different vehicles. It is still challenging to integrate them under real-time operational level.

The second one is the temporal-spatial operational level. Most previous reviewed studies model an urban road traffic network at an aggregated zone (or grid) level while ignoring the physical network topology and enforces the within-grid homogeneous assumption that all locations and vacant vehicles inside the grid are identical. For instance, as shown in Figure 1(a), three different origins in grid 1 would have equal chances to traverse to grid 2, but because these origins may lie on different road links, some would have no direct connection to the downstream grid. Moreover, the spatial distribution of the travel demand pattern may be significantly heterogeneous in a grid because of the urban network characteristics (e.g., Figure 1(b)). The larger grid then leads to a large decision interval. For instance, a large grid of 1 square kilometer would lead to a

¹ A/B testing is widely used to perform controlled experiments with two or more variants in real-world systems. Please refer to [1] for details



(a) Grid routing: it indicates the optimal passenger-seeking policies for vacant vehicles [13].



(b) Grid value: different color indicates different future potential earnings [2].



(c) Network topology: it indicates the online taxi-routing simulation in Manhattan road network [22].

Figure 1: Grid vs link-node network representations

longer decision interval (e.g., 10 minutes in [8]), which delays and aggregates the on-demand orders in an interval. However, impatient passengers may cancel their orders, which would lead to infeasible operation strategies in real-life scenarios.

The last one is the explicit and robust cooperation of vehicles. These multi-agent RL based works adopt decentralized execution approaches, which are implemented for thousands of vehicles via sequential decision-making. However, such a decentralized method may undermine system-level efficiency with limited and unexplicit cooperation among agents.

B. Contribution

To tackle previously mentioned challenging issues for efficient online taxi dispatching under stochastic traffic conditions and real-time on-demand requests, this study proposes an RL-based approach to optimize real-time taxi operations. The proposed framework for this study is shown in Figure 2, and its methodological contributions are summarized as follows.

- This paper proposes an efficient RL-based algorithm which combines the data-driven RL components with a centralized programming based planning module, to solve large-scale online taxi dispatching problem. To leverage the real-time level operation and heterogeneous traffic network topology, we reformulate the online vehicle dispatching problem using a link-node based micro-network representation and integrates both the order dispatching stage and the vehicle routing stage. The proposed algorithm can dispatch thousands of taxis on a real-time basis.
- Instead of decentralized/sequential decisions that lead to implicit and limited cooperation among taxi drivers (vehicles), we adopt a centralized mathematical programming model, where the system objective function is decomposed and approximated by the sum of the parameterized agent's action-value function. Two options of actions referring to order assignment and route selection are optimized collectively under system constraints to explicitly realize cooperation among agents and reduce the computational burden. This centralized decision approach better suits the centralization of taxi routing in the ride-sharing industry, as deemed by [1, 22].
- In the centralized programming/planning stage, unlike that reviewed works only plans for current vacant vehicles

[5] or all vehicles [22], the proposed algorithm considers vacant vehicles and those to be vacant very soon to improve the matching efficiency robustly and reduce computational burden.

- Furthermore, to tackle the sparse reward signal and the transition sample imbalance issues when agents interact with a large-scale micro-network environment, we propose temporal-difference (TD) learning with prioritized gradient descent using the adaptive exploration strategy to accelerate the learning process of agents.

The proposed algorithm is examined using the Manhattan road network and New York City yellow taxi data. This real-world case study involves thousands of taxis and tens of thousands of customers. Our experiments demonstrate the computational feasibility and scalability of the proposed algorithm in large-scale applications. Furthermore, compared with some existing multi-agent RL algorithms (e.g., the mean-field DQN algorithm) and pure optimization algorithms (e.g., Kuhn-Munkres algorithm), our experimental results confirm that the proposed algorithm not only improves the total platform revenue and the average driver profit but also reduces the platform mismatching rate, the matching delay, and the customers' total waiting time in various supply-demand scenarios.

The remainder of this paper is organized as follows. Section II presents the description and formulation for online taxi dispatching problem over network. Section III includes its MDP-based formulation and the RL-based dispatching algorithm. An empirical study is conducted in Section IV to validate the proposed algorithm in various supply-demand scenarios. Sensitivity analysis for key parameters is also included in the empirical study. Finally, Section V concludes the paper.

II. PROBLEM FORMULATION

We consider an online vehicle dispatching problem where a central platform manages a fleet of vehicles such as taxis to serve on-demand customers in order to maximize system profit. A central platform performs two functions: first, it collects real-time order requirements from passengers and matches them with available vacant vehicles; second, it plans routes for vehicles without matching orders to improve their probability of matching orders in the near future. In summary, the problem is how to dispatch orders and plan routes to

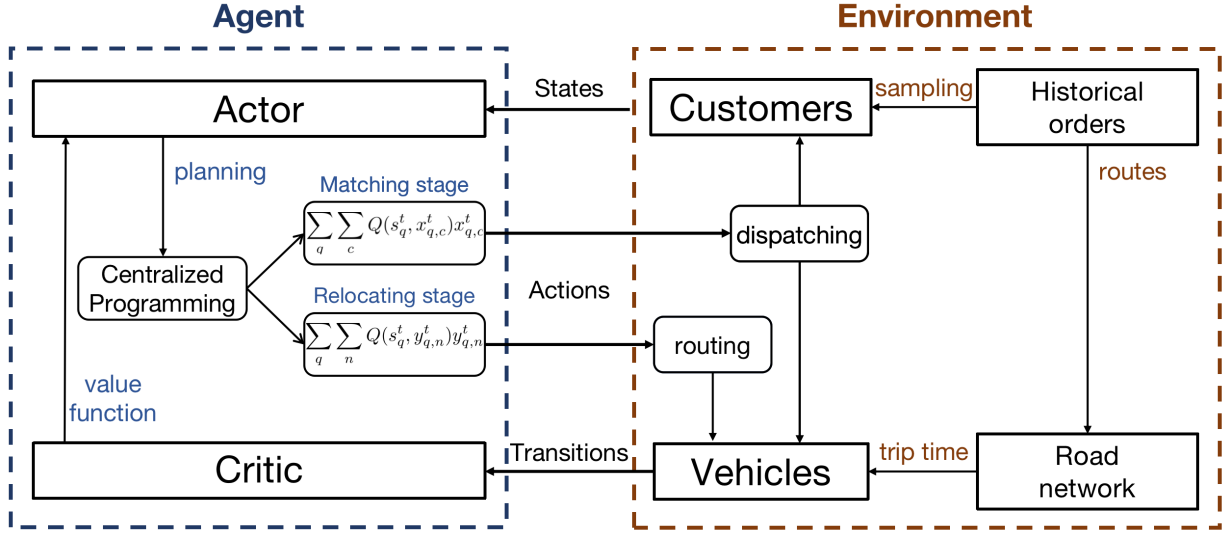


Figure 2: The proposed vehicle dispatching framework.

Table II: Nomenclature

Notation	Description
G	Directed graph of road network
N	Set of nodes or road intersections
E	Set of edges or road links
e	Distance function of nodes
σ	Estimated travel time function of nodes
Δt	Set time step
Q	Vehicle set: $q \in Q$
h_q^t	Estimated vacant time of a vehicle q at time t
d_q^t	Location of a vacant vehicle q at time t
C	Customer set: $c \in C$
o_c	Origin node of a customer c
d_c	Destination node of a customer c
t_c	Request time of a customer c
w_c	Maximum waiting time of a customer c
p_c	Trip fare paid by a customer c
H	Pre-planning horizon
e_s	Matching radius
$x_{q,c}$	1-0 variable indicating whether a vehicle q is matched with a customer c or not
$y_{q,n}$	1-0 variable indicating whether a vehicle q is routed to a node n or not
β_d	Unit time cost for driving of vehicles
β_w	Unit time cost for waiting of vehicles

maximize total system profit. In this section, we clarify the key settings in the online vehicle dispatching problem and key notations are summarized in Table II.

Environment. First, we abstract the real road network as a directed graph $G = \{N, E\}$, where N stands for the nodes (road intersections), and E denotes the edges (road links). The graph distance between node i and j is denoted as $e(i, j)$, and the corresponding estimated travel time is denoted as $\sigma(i, j)$.

Customer demand. Vehicles travel along the links to deliver customers from one node to another. In this problem settings, we simplify the origin and destination of an order are both nodes of the graph G [22]. Specifically, each customer's order information can be represented as a tuple $c = (o_c, d_c, t_c, w_c, p_c) \in C$, where $o_c \in N$ and $d_c \in N$ are the origin and destination node of the order, respectively, t_c is

the time the customer submits an order, w_c is the maximum time a customer is prepared to wait, and p_c is the paid fare for the trip. We further assume that customers do not cancel their orders after being assigned a vehicle. We consider a discrete sequential decision process in which customers waiting to be matched at time t requested their orders before t , which means that we consider the subset of customers such that $\{t_c \leq t | \forall c \in C\}$ [22].

Vehicle supply. Vehicles are assigned to pick up passengers and deliver them to destinations, or they are assigned new routes to relocate themselves for further supply-demand balance. Because we assume that each vehicle carries only one passenger² in a trip, each vehicle's information can be represented by a tuple: $q^t = (h_q^t, d_q^t) \in Q$, where h_q^t and d_q^t are the time and node, respectively, of a vehicle that will be vacant. For instance, if a vehicle is currently vacant, h_q^t is the current time step t and d_q^t is the current location; otherwise, if a vehicle is delivering customer c at time t , then h_q^t is the estimated time of arrival and $d_q^t = d_c$ is the order's destination.

Platform decisions. We consider a system profit maximization problem by controlling a fleet of vehicles to serve on-demand customers. In each decision interval $[t, t + \Delta t]$, the central platform matches available orders with the vacant vehicles, and additionally, it routes vacant vehicles without assigned orders to certain locations with a higher probability of being matched with orders. There are thus two kinds of decisions: $x_{q,c}^t$ and $y_{q,n}^t$, that represent the matching and routing decision variables in each time step, respectively.

In the matching stage, the set of matching decisions $x_{q,c}^t$ is defined as:

$$x_{q,c}^t \in Q^t \times C^t = \begin{cases} 1, & \text{if a vehicle } q \text{ is matched with a customer } c \\ 0, & \text{otherwise} \end{cases} \quad (1)$$

where Q^t represents the vehicles set at time t , $C^t = C_{new}^t \cup C_{wait}^{t-1}$ represents the on-demand customers at time t , including

²A group of fellow passengers is regarded as one customer as well.

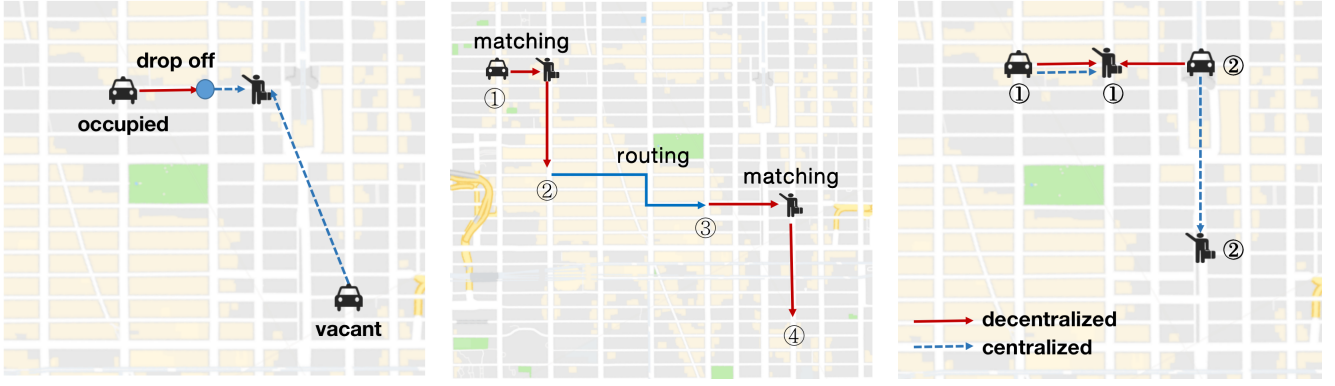


Figure 3: Pre-matching process, where the red line indicates the occupied vehicle's routes, and the blue dashed line indicates the matching between vehicles and customers. Figure 4: MDP definition: each agent represents a vehicle, and the action represents assigning the vehicle to serve a customer that vehicles make decisions for their own benefit, and a centralized decision indicates that vehicle routes are planned by a central platform.

newly requested customers C_{new}^t before t and previously requesting but still waiting customers C_{wait}^{t-1} , i.e.,

$$C_{new}^t = \{c | t - \Delta t \leq t_c \leq t\}, \quad \forall c \in C \quad (2)$$

$$C_{wait}^{t-1} = \{c | t_c \leq t - \Delta t, t - t_c \leq w_c\}, \quad \forall c \in C^{t-1} / C_{match}^{t-1} \quad (3)$$

where C^{t-1} is the requested customers in the previous time step, and $C_{match}^{t-1} = \{c | \sum_q x_{q,c}^{t-1} = 1\}$, $\forall c \in C^{t-1}$ is the matched customers in the previous time step.

$x_{q,c}^t$ indicates that a vehicle q is matched with a customer c . Furthermore, each possible matching pair satisfies the following constraints:

$$\sum_q x_{q,c}^t \leq 1 \quad (4)$$

$$\sum_c x_{q,c}^t \leq 1 \quad (5)$$

$$e(d_q^t, o_c) \leq e_s \quad (6)$$

$$h_q^t + \sigma(d_q^t, o_c) \leq t_c + w_c \quad (7)$$

$$h_q^t - t \leq H \quad (8)$$

where Eq.(4) indicates that only one customer can be picked by each vehicle, Eq.(5) indicates that each vehicle can at most be assigned only one customer, Eq.(6) indicates the matching radius e_s and limits the maximum pickup distance, and Eq.(7) indicates that the customer pickup time $h_q^t + \sigma(d_q^t, o_c) - t_c$ does not exceed the maximum waiting time w_c .

Then, in each decision interval, if the central platform only plans for vacant vehicles, this may bring about the matching failure³ during peak hours and undermine platform efficiency [41]. If the platform considers all vehicles, including occupied and vacant ones, this will add additional computational complexity for planning steps [22]. Here we introduce a flexible and robust approach that considers both vacant vehicles and occupied vehicles that will arrive at their destinations within

³Matching failure of ride-hailing platforms was first anticipated by Arnott [40]. When there are few vacant vehicles relative to requesting passengers, vehicles are quickly occupied, which leads to the dispersal of idle drivers throughout a city so that vacant vehicles must pick up distant customers, wasting the drivers' time and reducing earnings [41].

the pre-planning horizon H . Eq.(8) thus indicates that the planning vehicles are to be vacant within H , where $h_q^t - t$ indicates the remaining driving time before arriving at the destination. For example, Figure 3 shows one customer and two vehicles, one occupied and the other vacant. If the platform only plans for vacant vehicles and then assigns the vacant vehicles to pick up the distant customer, matching failure occurs, which wastes both the driver's and the customer's time. If the platform considers another occupied vehicle that is near its destination and the destination is near the customer's origin, then matching the occupied vehicle can reduce the vehicle's cost and the customer's waiting time, thus improving the platform's profit from the perspective of the system optimum.

Although there are many other approaches to seek more efficient matching, including surging pricing [41, 42], adjusting matching interval and radius [43, 44] and delaying orders [4], these methods must adjust the model parameters to adapt the dynamic supply-demand patterns. Improper tuning of parameters may lead to platform inefficiency. Our approach is robust to various situations because of those system constraints, such as the time window constraint Eq.(7) and the matching radius constraint Eq.(6), ensure the solution feasibility under different H . As H increases, the platform considers more vehicles to dispatch and potentially alleviate inefficient matching, albeit at the cost of a little more computational complexity. The trade-off between the computational burden and platform efficiency is shown in Section IV.

At the routing stage, unlike [22], which assumed that vehicles remain in place when vacant, we consider arranging routes for vacant vehicles to improve their future matching probability [12, 13]. The set of routing decisions $y_{q,n}^t$ is then defined as:

$$y_{q,n}^t \in Q^t \times N = \begin{cases} 1, & \text{if a vehicle } q \text{ is routed to a node } n \\ 0, & \text{otherwise} \end{cases} \quad (9)$$

where $y_{q,n}^t$ indicates that a vehicle q is routed toward a node n in the road network. To simplify the relocation problem, the

decision variable satisfies the following constraints:

$$\sum_n y_{q,n}^t = 1 \quad (10)$$

$$x_{q,c}^t + y_{q,n}^t \leq 1 \quad (11)$$

$$\sigma(d_q^t, n) \leq \Delta t \quad (12)$$

$$h_q^t = t \quad (13)$$

where Eq.(10) indicates that one vehicle can be routed to one location including waiting in its current location, Eq.(11) indicates that one vehicle can either be matched to an order or relocated, Eq.(12) indicates that the travel time to the routed node is within the decision time interval so that the routed vehicles can be re-planned in the next time step according to the future traffic situation, and Eq.(13) indicates that only vacant vehicles can be relocated.

Platform profits. After making decisions in each time interval, the platform gains profit r defined as:

$$r(x_{q,c}^t) = p_c - \beta_d[\sigma(d_q^t, o_c) + \sigma(o_c, d_c)] \quad (14)$$

$$r(y_{q,n}^t) = \begin{cases} -\beta_w \Delta t, & \text{if } n = d_q^t \\ -\beta_d \sigma(d_q^t, n), & \text{otherwise} \end{cases} \quad (15)$$

where Eq.(14) indicates the profit of matching a vehicle q with a customer c , including the paid trip fare p_c and vehicle driving cost $-\beta_d[\sigma(d_q^t, o_c) + \sigma(o_c, d_c)]$; and Eq.(15) indicates the profit of routing a vehicle q to a node n , which either equals to its driving cost $-\beta_d \sigma(d_q^t, n)$ or its waiting cost $-\beta_w \Delta t$.

III. THE MDP FORMULATION AND THE RL-BASED ALGORITHM

In this section, we model the platform decision process as an MDP and define its state, action, reward, and transition function. We then introduce an RL-based algorithm to tackle this problem.

MDP. An MDP is defined as $\langle \mathbf{S}, \mathbf{A}, \mathbf{R}, P, \gamma \rangle$ [25]. $\mathbf{S} = \{s_q\}_{q \in Q}$ is the joint state space for all vehicles, where the state of each vehicle is defined the same as before: $s_q^t = (h_q^t, d_q^t)$. $\mathbf{A} = \{a_q\}_{q \in Q}$ is the joint action space for all vehicles, wherein each vehicle's action $a_q^t \in \{c, n\}$ has two options, matching a customer $c \in C^t$ or routing to a node $n \in N$ as shown in Figure 4. In other words, the action is to choose a trip to a particular destination node, which could be either delivering a customer to his/her destination node $d_c \in C^t$ or be relocating without a customer to the routed node $n \in N$. Both types of actions can be represented as a tuple (origin, destination, fare, cost, etc) for the implementation in the Q network. $\mathbf{R} = \{r_q\}_{q \in Q} : \mathbf{S} \times \mathbf{A} \rightarrow \mathbb{R}$ is the reward space, where each vehicle's reward $r(s_q^t, a_q^t) \in R_q$ has been defined in Eq.(14) and Eq.(15).

$P = \Pi_{q \in Q}[p(s_q^{t'} | s_q^t, a_q^t)] : \mathbf{S} \times \mathbf{A} \times \mathbf{S} \rightarrow [0, 1]$ denotes the state transition probability. In our problem settings, the transition probability $p(s_q^{t'} | s_q^t, a_q^t)$ of a vehicle is represented as:

$$p(s_q^{t'} | s_q^t, a_q^t) = \begin{cases} p(c)p(o_c, d_c | d_q^t)p(h_q^{t'} | h_q^t), & \text{if } a_q^t = x_{q,c}^t \\ (1 - p(c))p(h_q^{t'} | h_q^t), & \text{otherwise} \end{cases} \quad (16)$$

where the first term denotes the transition probability after matching a customer c , which equals to the probability of

being matched with a customer c , multiplied by the probability of customer with origin o_c and destination d_c , and multiplied by the probability of traveling time $h_q^{t'} - h_q^t$ from vacant node d_q^t to the customer destination node $d_q^{t'} = d_c$; the second term denotes the probability after being routed to node n , which equals to the probability of no customers multiplied by the probability of traveling time $h_q^{t'} - h_q^t$ from vacant node d_q^t to the routed destination node $d_q^{t'} = n$. Instead of calculating these probabilities explicitly by fitting the historical data [12, 13], which is computationally intractable in large-scale network settings considering travel time probability, we adopt the data-driven approach to leverage the historical data to solve the optimal decision with implicit consideration of the transition probability.

The objective of an MDP is then to find a policy π to generate actions for all agents that maximizes the global cumulative discounted reward given the initial state as $V^\pi(S) = \mathbb{E}_{\pi,p}[\sum_t \gamma^t R^t | S]$, where $\gamma \in [0, 1]$ is the discounted factor and $R^t = \sum_q r_q^t$ is immediate reward for all agents. The value function can then be represented recursively as $V^\pi(S) = \mathbb{E}_{\pi,p}[R(S, A) + \gamma V^\pi(S')]$. The action-value function can be correspondingly denoted as $Q(S, A) = \mathbb{E}_p[R(S, A) + \gamma V^\pi(S')]$. Under the Bellman optimality [24, 45], the optimal value function satisfies: $V(S) = \max_A \mathbb{E}_p[R(S, A) + \gamma V(S')]$, which is equivalent to maximizing the action-value function $Q(S, A)$. Towards this, the long-horizon effect of decision and cooperation among agents have to be considered jointly. The global value function naturally evaluated the system's long-horizon value from the current state [45]. So we focus on how to achieve cooperative decisions.

Multi-agent Cooperation. The multi-agent RL community has developed various approaches to improve the performance of a sequential or decentralized decision-making process, including local observation embedding [46], value function decomposition [47, 48], learning to communication [49], and difference reward setting [50]. Most of those works focus on decentralized execution for agents. To evaluate the equivalence between decentralized and centralized decision, we firstly introduce IGM (Individual-Global-Max) principle [51] as:

$$\arg \max_A Q(S, A) = \begin{pmatrix} \arg \max_{a_1} Q_1(s_1, a_1) \\ \vdots \\ \arg \max_{a_n} Q_n(s_n, a_n) \end{pmatrix} \quad (17)$$

Eq.(17) indicates that the global maximization of all agents could be reached by individual maximization of each agent. In our problem settings, the IGM can not be guaranteed as follows:

Proposition 1. *With coupled action constraints of agents, such as $f(a_1, \dots, a_n) \leq 0$ and $g(a_1, \dots, a_n) = 0$, the IGM is not guaranteed.*

Proof for Proposition 1. Suppose the global Q value function has an additive form as $Q(S, A) = \sum_i Q_i(s_i, a_i)$ [47, 51]. The global maximization is then equal to a constrained opti-

mization problem as:

$$\max_A \sum_i Q(s_i, a_i) \quad (18)$$

$$s.t. \quad \mathbf{f}(a_1, \dots, a_n) \leq 0 \quad (19)$$

$$\mathbf{g}(a_1, \dots, a_n) = 0 \quad (20)$$

The solution for this constrained optimization (global maximization) problem depends on constraints Eq.(19) and Eq.(20). While the individual maximization $\max_{a_i} \{Q(s_i, a_i)\}$ is decentralized/sequentially solved without jointly considering those coupled constraints. Therefore, the equivalence between global maximization and individual maximization is not guaranteed. \square

As in our case, the coupled action constraint is Eq.(4), which indicates that one customer can only be matched with one vehicle. For example, as shown in Figure 5, for both vehicles 1 and 2, the individual maximization is to take customer 1 since it is with the nearest distance for both vehicles. However, the one-customer-one-vehicle constraint makes this individual maximization impossible. Therefore, the global maximization is vehicle 1 picks customer 1, and vehicle 2 takes customer 2.

Moreover, the centralized approach has the following benefits: 1) It is robust and explainable. The centralized programming model solves optimal decisions under a series of system constraints, which makes the results robust in various situations; 2) The per-step optimum is guaranteed. The output of sequential execution is unstable since it depends on the decision sequence. For example, in Figure 5, if vehicle 1 makes the decision first, then the system optimum is reached; otherwise, vehicle 2 makes the decision first, and the system profit is undermined. However, the centralized model guarantees the global optimal actions; 3) The computation is efficient because it is formulated as linear programming forms.

Based on the decomposition form of joint value function as $V(S^t) = \sum_q V(s_q^t)$ [47, 48, 51], the Bellman equation $V(S^t) = \max_{A^t} \mathbb{E}_p[r(S^t, A^t) + \gamma V(S^{t'})]$ can be reformulated as:

$$\sum_q V(s_q^t) = \max_{A^t} \mathbb{E}_p[\sum_q r(s_q^t, a_q^t) + \gamma \sum_q V(s_q^{t'})] \quad (21)$$

$$= \max_{A^t} \mathbb{E}_p[\sum_q Q(s_q^t, a_q^t)] \quad (22)$$

To maximize the joint value function $\sum_q Q(s_q^t, a_q^t)$, we gain different constrained programming problems in matching and routing stages.

In the matching stage, Eq.(22) can be interpreted as:

$$\max_{x_{q,c}^t} \sum_q \sum_c Q(s_q^t, a_q^t) x_{q,c}^t \quad (23)$$

$$s.t. \quad (1), (4), (5), (6), (7), (8) \quad (24)$$

In the routing stage, Eq.(22) can be interpreted as:

$$\max_{y_{q,n}^t} \sum_q \sum_n Q(s_q^t, a_q^t) y_{q,n}^t \quad (25)$$

$$s.t. \quad (9), (10), (11), (12), (13) \quad (26)$$

Compared with the standard optimal assignment problem [52], problems (23) and (25) are with more system constraints (e.g., matching radius, time window, etc.) apart from the standard one-to-one matching constraints. However, those system constraints can help filter out some infeasible matching pairs, which can be processed by the following steps: 1) find matching pairs dissatisfying those system constraints and change the weights of those infeasible pairs into a small negative number (e.g., -10,000); 2) solve the standard optimal assignment problem without system constraints by the K-M algorithm [52]; 3) filter out the infeasible pairs in the final assignment result.

Those system constraints also accelerate the K-M algorithm because most of the candidate matching pairs are infeasible due to system constraints, and the corresponding weight matrix is very sparse.

TD Learning. To acquire the optimal joint decision, the optimal Q-function or value function must be derived in advance. Here we adopt TD learning to tackle this problem [45]. Firstly, we adopt a neural network (NN) to parameterize and approximate the value function as $V(s|\theta)$, where θ represents the parameters of hidden layers.

The TD error is then defined as:

$$\delta_q^t = V(s_q^t|\theta) - [r(s_q^t, a_q^t) + \gamma V(s_q^{t'}|\theta)] \quad (27)$$

where the first term $V(s_q^t|\theta)$ is the estimated value function, and the second term $r(s_q^t, a_q^t) + \gamma V(s_q^{t'}|\theta)$ is the target value function.

Under optimal policy π , the TD error equals to 0. In other words, the TD error must be minimized to derive the optimal value function. Therefore, The problem is then to find optimal parameters θ such that the following loss function is minimized.

$$\min_{\theta} L(\theta) = E_p[(\delta_q^t)^2] \approx \sum_i (\delta_i)^2 \quad (28)$$

where δ_i is the TD error sample when agents interact with the dynamic environment.

By applying mini-batch stochastic gradient descent (SGD), we can update θ iteratively as:

$$\theta^{k+1} = \theta^k - \alpha \sum_{i=1}^m \delta_i \nabla V_{\theta^k} \quad (29)$$

where α is the learning rate and m is the batch size.

To make the TD learning algorithm practical in the environment with large-scale microscopic road network, several issues below must be addressed.

1) **Sample imbalance:** after interacting with the environment, to break the correlation among samples and to improve the stability of the learning process, the SGD is performed by a batch uniformly sampling from collected transition samples to update the NN parameters according to Eq.(29). However, our case includes two types of transition samples (matching and routing), which have different time scales. A routing transition lasts a typical time interval of 30 seconds to drive along with several links, while a matching transition may last more than 10 minutes to transport customers. This results in a huge imbalance between the two types of transition samples.

Conventional SGD ignores such imbalance and thus induce instability in the learning process. Prioritized experience replay (PER) techniques are adopted to tackle this problem [53], where instead of uniformly sampling, a batch is sampled according to their absolute TD error $|\sigma_i|$ (e.g., the larger $|\sigma_i|$, the larger probability of sampling this transition). Since the absolute TD error provides a way to measure the potential improvement from different transition samples [53], we sample a batch according to the rank of the absolute TD error of the transition samples. Note our algorithm is on-policy, where the collected samples are used once in each iteration. While the off-policy one with experience replay [54–56] is a worthy direction in future works.

2) Deep exploration⁴: in the training process, agents must explore the unknown environment to find potential states with high further reward. In our case, the vehicles are relocated along the routes when no customers are available. However, as in the large-scale network and long-horizon case, the exploration is extremely difficult. Generally adopted exploration strategies (e.g., ϵ -greedy, soft-max and gaussian noise), cannot achieve deep exploration [57]. For example, in the routing stage, it is possible that a vehicle will drive back and forth along with a road link instead of successively finding a route. To tackle this problem, the idea of pre-planning is again adopted, where the routed set is selected as $(\tau - 1)\Delta t \leq \sigma(d_{q,n}^t) \leq \tau\Delta t$, where τ is the parameter to adjust the planning window. Specifically, at the beginning of the iteration, τ is relatively large, so the agent engages in “far-sighted” exploration beyond the next time interval, whereas it finally falls to 1 as in Eq.(12) to reduce exploration with iterations. Action is then selected from the plan set according to ϵ -greedy.

The overall algorithm is stated in Algorithm 1.

IV. EMPIRICAL STUDY

Simulation Environment. To train the value function off-line and implement the online optimization procedures, we need to construct a comprehensive simulator to reproduce the real-world demand and supply dynamics via Monte Carlo simulation. We first extracted the Manhattan road network data from the open-source website OpenStreetMap⁵. We then extracted the strongly connected component of the simplified driveway network according to [58], which yields a strongly connected directed graph G with 4,486 nodes and 9,755 arcs. Furthermore, the adjacent matrix and the shortest path matrix can also be calculated easily.

The detailed customer order data was obtained from the yellow taxi data of the New York City Taxi and Limousine Commission, which is freely available online⁶. This data set includes the pickup and drop-off coordinates and time-stamps, travel distance, travel time, and trip fare of each trip. The customers’ origins and destinations were projected to the nearest intersections or nodes on the graph. For the routes

⁴Deep exploration means exploration which is directed over multiple time steps; it can also be called “planning to learn” or “far-sighted” exploration [57].

⁵OpenStreetMap: <https://www.openstreetmap.org>

⁶Dataset: <https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page>

Algorithm 1 TDCP: TD Learning with Centralized Programming.

Input: initial neural network parameters θ
Output: converged network parameters θ^*

- 1: Initial buffer D and parameter τ , historical demand C
- 2: **for** $k = 1$ to maximum iteration epoch **do**
- 3: Initialize vehicles states $s_q^{t_0} \forall q \in Q^{t_0}$.
- 4: **while** $t < T$ **do**
- 5: Solve actions for all agents collectively according to Eq.(22).
- 6: Execute simulator with input joint states and actions, then output state transition series $\{s_q^t, a_q^t, r_q^t, s_q^{t'}\}$ according to Algorithm 2.
- 7: Calculate TD error δ from transition series according to Eq.(27).
- 8: **end while**
- 9: Select and store D size samples with the highest absolute TD error $|\delta_q^t|$ in the buffer D .
- 10: **for** $j = 1$ to maximum update steps **do**
- 11: Sample m batch transition series from buffer D .
- 12: Update neural network parameters θ according to Eq.(29).
- 13: **end for**
- 14: **end for**

Algorithm 2 Ride-hailing simulator.

Input: road network $G = \{N, E\}$ and platform decisions policy: $a_q^t = \{x_{q,c}^t, y_{q,n}^t\} \forall q \in Q^t$
Output: vehicles’ state transition series $\{s_q^t, a_q^t, r_q^t, s_q^{t'}\}$

- 1: **for** each time step $\{t = 0, \dots, T\}$ during simulation **do**
- 2: **Order pool:** collect available customer demand $C^t = C_{new}^t \cup C_{wait}^{t-1}$.
- 3: **Vehicle pool:** collect current vehicles information $Q^t = \{q^t\}$.
- 4: **Matching stage:** match vehicles with orders according to platform decision $x_{q,c}^t$ under constraints (1), (4), (5), (6), (7) and (8).
- 5: **Routing stage:** route vehicles left vacant according to platform decisions $y_{q,n}^t$ under constraints (9), (10), (11), (12) and (13).
- 6: **State update:** update vehicles’ state according to samplers’ information.
- 7: **Environment update:** remove matched orders as in Eq. (3) and add newly requested orders C_{new}^{t+1} .
- 8: **end for**

or trips missing travel time records, we used a mean travel speed of 25 km/h along the shortest path. The order data in May 2016 was adopted to feed the simulation regarding the on-demand customers’ orders. Specifically, the data of May 1-25 was used for training, and that of May 26-31 was used for testing, see Figure 6. The demand pattern shows periodic characteristics but differences in each day. The simulation time is from 10:00 am to 10:00 pm at 30-s time intervals. We set 1,000 vehicles to simulate under 100,000 sampled order records, the maximum waiting time of 5 minutes, the

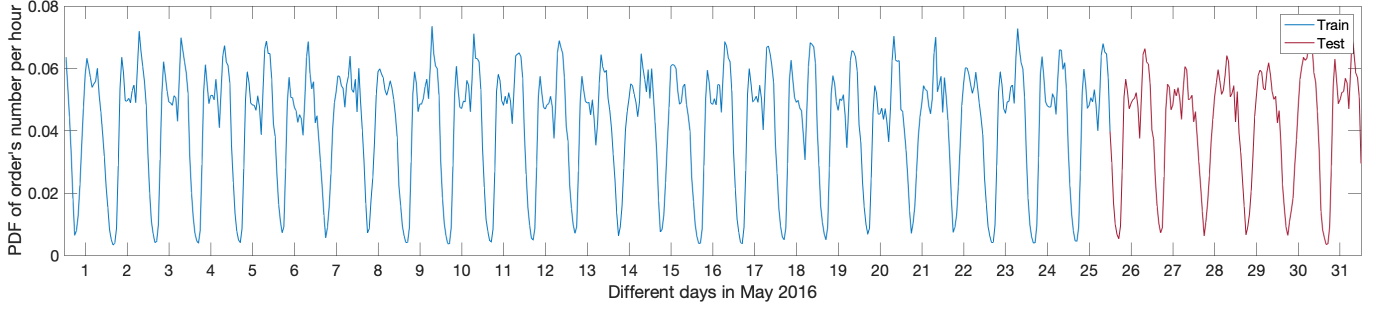


Figure 6: Historical demand pattern in Manhattan, New York. Blue curve represents sampled training data; red curve indicates sampled test data (PDF denotes the probability density function).

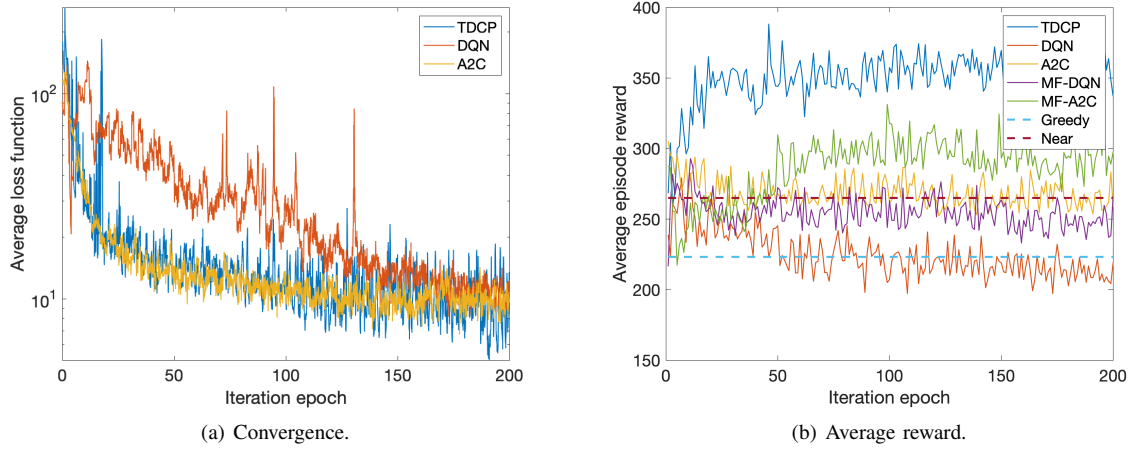


Figure 7: Average training performance for various RL algorithms; dashed lines indicate two pure optimization-based algorithms.

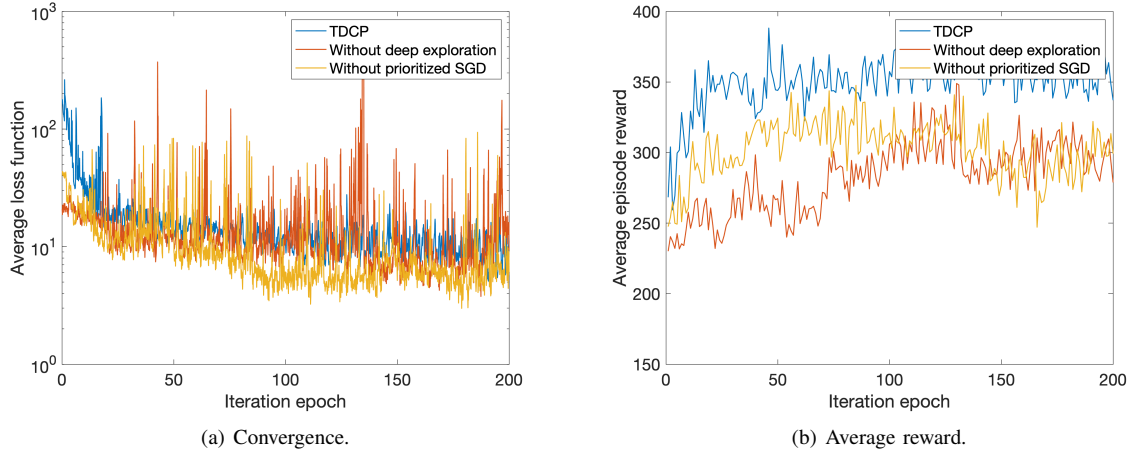


Figure 8: Average training performance for proposed prioritized replay and deep exploration techniques.

matching radius of 1km, and pre-planning horizon of 1 minute. The setting is used in training, testing process and sensitivity analysis.

The ride-hailing simulator and the algorithms proposed in the study were coded and run in MATLAB on a MacBook Pro with a 2.20-GHz Intel Core i7 CPU and 16 GB of memory.

Off-line Training. To approximate the value function under a stochastic dynamic traffic network for on-demand travel service, we adopted a deep neural network of four hidden layers composed respectively of 256, 128, 64, and 32 neurons, leaky

Relu activation function, and Xavier initialization method [59]. We set 200 iteration epochs, batch size of 1024, update steps of 10, and learning rate of 0.01 for the training process. In each iteration epoch, agents interact with the environment with randomized initial states, store transition samples, and update policy with prioritized SGD.

We applied several algorithms stated below, including RL-based as well as pure optimization-based algorithms, to solve the problem for comparison. All compared RL-based algorithms share the same state space, hyper-parameters, network

architecture, and exploration strategy.

- **Near**: in the matching stage, a centralized programming model with the nearest pickup distance objective is adopted, whereas in the routing stage, vehicles left vacant remain where they are.
- **Greedy**: in the matching stage, a centralized programming model with maximum profit objective is adopted, whereas in the routing stage, vehicles left vacant remain where they are.
- **DQN**: a DQN policy with independent learners is adopted in both stages [8].
- **A2C**: an A2C policy with independent learners is adopted in both stages [8].
- **MF-DQN**: a mean-field DQN policy with decentralized execution policy is adopted in both stages [5, 60].
- **MF-A2C**: a mean-field A2C policy with decentralized execution policy is adopted in both stages [5, 60].
- **TDCP**: as shown in Algorithm 1.

Figures 7(a) and 7(b) show that the proposed algorithm TDCP outperforms the other four RL-based algorithms in terms of the average episode reward. A similar type of learners (e.g., DQN and MF-DQN) have a similar performance of convergence, so we only plot DQN and A2C for a more clear illustration of Figure 7(a).

It needs to be mentioned that test RL-based algorithms are sensitive to the learning rate because of high-dimension input (state and action) of Q function/policy network. In other words, the high dimension input brings challenges to the training process. As a result, given a slow learning rate (e.g., 0.0001), thousands or even tens of thousands of iterations are required to converge and perform stably in the average episode reward. On the other hand, given a faster learning rate (e.g., 0.01), they are more likely to oscillate heavily. However, for state-value function estimation with the state as input, the input dimension is smaller and easy to train. Additionally, the corresponding Q function is calculated as the true reward signal plus the value function for the next state, which also decreases the variance of estimated Q function compared with the directly fitting approach. This enables the robustness against the noise of the proposed algorithm. Therefore, our algorithm can adopt a faster learning rate to accelerate convergence.

Furthermore, the decentralized nature of these four RL algorithms that use independent/sequential learners leads to competition, i.e., maximizing individual action-value function. Such non-cooperative learning may harm the system episode reward as shown in Figure 5. Since the proposed algorithm adopts the centralized mathematical programming formulation to solve the system profit maximization problem, it generates cooperative behaviors and leads to higher episode reward. Furthermore, the set of system constraints devised in this paper ensures the feasibility of decisions, which leads to better samples and thus computational efficiency. While the pure optimization baselines, which make the centralized decision in each decision time interval based on various objectives, do not utilize the future demand uncertainty. As a result, the performance of such kind of algorithms would not exceed that of the proposed TDCP, which considers the value function.

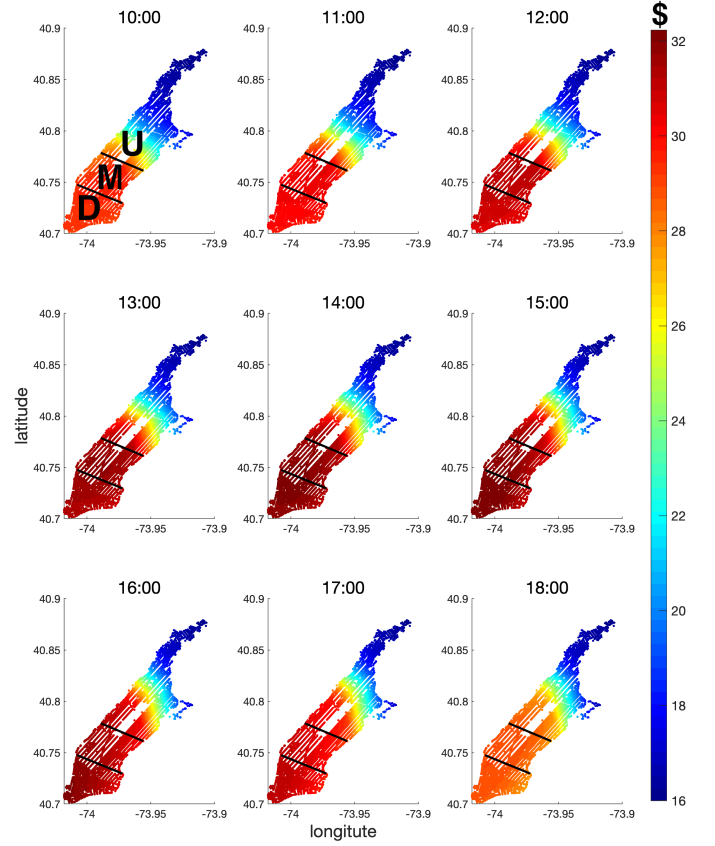


Figure 9: Value function visualization, where **D**, **M** and **U** denote the Downtown, Midtown and Uptown Manhattan, respectively.

To prove the benefits brought by the prioritized SGD and deep exploration, results of the proposed algorithm with and without those techniques are presented in Figures 8(a) and 8(b). Both techniques help the algorithm obtain a stable performance. As shown in Figure 8(a), the variance in the average TD error is greatly reduced with iterations, as a result of a full exploration of the environment in the beginning and updating with prioritized samples. Deep exploration helps agents find potential routes with a higher likelihood of picking up customers, and the decreasing rate of exploration with iterations helps agents focus more on the existing knowledge about the environment. Therefore, it balances environment uncertainty and policy stability. The prioritized SGD helps agents find important transition samples to update their policy, which accelerates and stabilizes the learning process. Figure 8(b) illustrates the average episode reward without deep exploration, which makes the collected episode reward improve slowly because it is difficult for agents to explore in such a large-scale network. The absence of prioritized SGD makes the reward improvement unstable because uniform sampling and updating result in useless samples, causing the learning process to oscillate, instead of identifying potential samples. Therefore, those techniques help our algorithm perform robustly and well in a large-scale network.

Figure 9 shows the spatial-temporal evolution of the trained value function. The value function indicates the expected dis-

counted future profit $\mathbb{E}_p[\sum_t \gamma^t r^t]$ in a certain state. Spatially, the Downtown area always takes the highest value because it is the busiest area of Manhattan and is correspondingly accompanied by a higher likelihood of a customer pickup. The likelihood decreases from Downtown to Midtown until Uptown. Temporally, the value function along time first increases due to the increasing demand during the peak noon hours, and it then decreases near the end of the simulation horizon because less time remains to pick up a customer. A video demonstration of value function evolution and on-demand customer visualization is provided online⁷.

Online Execution. After training the NN for value function, those tuned algorithms are compared with the use of the test dataset.

We choose following indices: 1) the order completion rate ($\frac{\text{the number of completed orders}}{\text{the number of all orders}}$), 2) the vehicle's average profit (average revenue - average cost), 3) the average customer matching delay (the matched timestamp - the requested timestamp), 4) the average customer waiting time (the pickup timestamp - the requested timestamp), and 5) the computation time (average computation time for decisions every 30s) to evaluate those algorithms. For simplicity, we denote 1) and 2) as revenue indices, 3) and 4) as customer indices, and 5) as the computation index. Aforementioned algorithms are then compared for serving the 100,000 orders with different fleet sizes of taxis to simulate various supply-demand situations.

Table III shows that the proposed TDCP algorithm performs the best, with the largest average profit in all three situations with different fleet sizes. First, when the demand exceeds the supply (1,000 vehicles) a lot, there are enough orders for vehicles to pick up. As a result, the competition among vehicles does not lead to weaker performance, as indicated by the fact that the independent A2C and MF-A2C algorithm performs similarly with TDCP in terms of the order completion rate. The mean-field based RL algorithms perform better over total customer waiting but not reward. Because they optimize decision considering local mean action (the ratio of local drivers and orders), and make decentralized action toward states with higher order-vehicle ratio. However, as mentioned before, due to the limited cooperation, decentralized RL-based algorithms do not yield a better average episode reward within the given iteration epochs as indicated in Figure 7(b) and Table III.

With a larger number of vehicles, it is found that the proposed TDCP outperforms all other RL-based policies in the revenue indices, which illustrates the superiority of the centralized decision process under competition among vehicles. For the optimization-based policies, only customer waiting time under the nearest distance-based objective performs better than the proposed algorithm. The reason for this is obviously that the platform assigns the vehicle nearest to the customer. The customer's waiting time is expected to be minimum. But it does not yield better performance in other indices because it assigns the nearest vehicles to customers in each decision interval without considering non-myopic assignments.

Under the scenario with the fiercest competition, the proposed algorithm outperforms all other policies in terms of both the revenue indices and the matching delay. The matching delay decreases due to the pre-planning techniques, because the platform matches customers with occupied but soon-to-be-vacant vehicles instead of delaying them to the next decision interval when there are no vacant vehicles. Such techniques are likely to reduce the anxiety of customers because they have been matched with a vehicle instead of merely waiting. In addition, the computation time for decision making stays at an acceptable level (below 1 second), compared to the 30s decision time interval, it is small enough. The computational burden comes mainly from two factors: the NN forward calculation time and the time required to solve the centralized programming. The first part can be distributed, whereas the second part is solved efficiently with the K-M algorithm, and both lead to computational efficiency.

Sensitive Analysis. In this section, we analyzed three key parameters in our simulation: the customer's maximum waiting time w_c , the matching radius e_s , and the pre-planning horizon H , to investigate their effects on the proposed policy performance. Figure 10 shows that an increase in the maximum waiting time, i.e., if customers are willing to wait longer for vehicles, leads to greater system profits and longer waiting time. However, its effectiveness is restricted due to the limited vehicle supply. In Figure 11, a larger matching radius implies that customers can be matched with distant vacant vehicles. However, the performance is also heavily bounded by the maximum waiting time. The computation time in Figure 10 and 11 decreases firstly because with the increase in the maximum waiting time w_c or the matching radius e_s , the number of available customers also increases, which makes more vacant vehicles occupied with customers and the platform needs less time for relocation decisions. The computation time then increases because more available customers result in more computation for matching decisions.

Figure 12 shows how the pre-planning horizon influences system performance. A larger pre-planning horizon indicates that the system considers more vehicles, including vacant and occupied vehicles, when making centralized decisions, to seek a better solution. As a result, with the increase in the pre-planning horizon, the system's revenue performance improves at the cost of the additional computational burden. It also leads to a shorter matching delay because more soon-to-be-vacant vehicles are planned. However, the computation time decreases gradually after increasing firstly because the pre-planning strategy improves the system efficiency and decreases the number of vacant vehicles in the planning horizon, which leads to less computation time for relocation decisions. Additionally, the computation time for decision making in all test scenarios stays at an acceptable level (far below 1 second) compared to the 30s decision time interval.

V. CONCLUSIONS

We investigated the online vehicle dispatching problem in practice with a platform to manage a fleet of thousands of taxis to serve tens of thousands of customers in a real-time operational level. To solve it, we then proposed an

⁷Video demonstration: <https://github.com/emliang/Simulation>

Table III: Online performance in various supply-demand scenarios. For a fair comparison, the random seeds for the sampling test data set are the same across all algorithms.

Policy	Completion Rate (%)	Average Profit (\$)	Matching Delay (min)	Waiting Time (min)	Computation Time (ms)
1,000 Vehicles					
Greedy	34.46	223.90	0.84	2.90	22.70
Near	54.24	255.83	1.40	2.76	19.30
DQN	33.08	204.40	0.86	3.02	83.40
A2C	70.07	276.81	0.67	2.89	56.80
MF-DQN	40.54	243.41	0.82	2.23	75.20
MF-A2C	71.71	284.44	0.70	2.22	62.9
TDCP	69.18	350.34	0.66	2.92	39.50
1,500 Vehicles					
Greedy	50.53	196.30	0.71	2.82	39.30
Near	70.51	220.18	1.14	2.54	36.20
DQN	50.58	188.98	0.72	2.88	126.20
A2C	83.47	236.16	0.55	2.73	112.50
MF-DQN	56.77	205.27	0.71	2.23	132.10
MF-A2C	84.96	240.83	0.55	2.17	121.1
TDCP	90.87	290.00	0.41	2.63	92.00
2,000 Vehicles					
Greedy	65.30	175.95	0.60	2.52	56.40
Near	80.65	188.19	0.90	2.30	52.20
DQN	67.44	172.23	0.59	2.73	192.90
A2C	91.07	200.61	0.44	2.53	174.10
MF-DQN	74.00	181.95	0.54	2.15	215.00
MF-A2C	92.30	203.46	0.42	2.06	203.60
TDCP	98.04	229.47	0.30	2.25	211.00

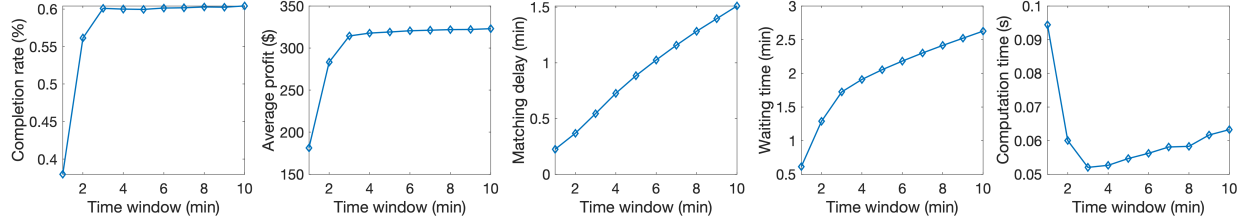


Figure 10: Maximum waiting time w_c .

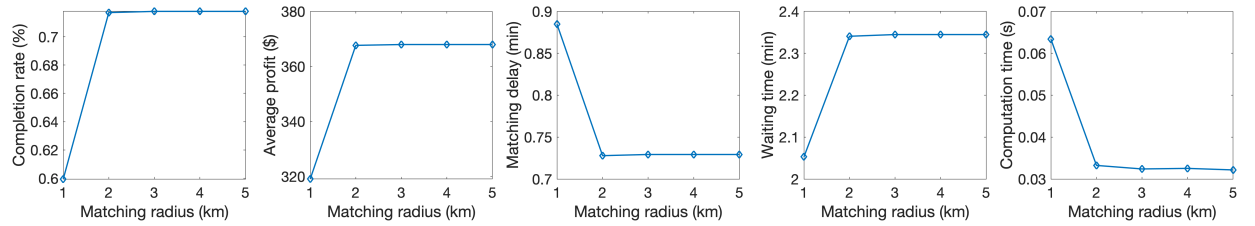


Figure 11: Matching radius e_s .

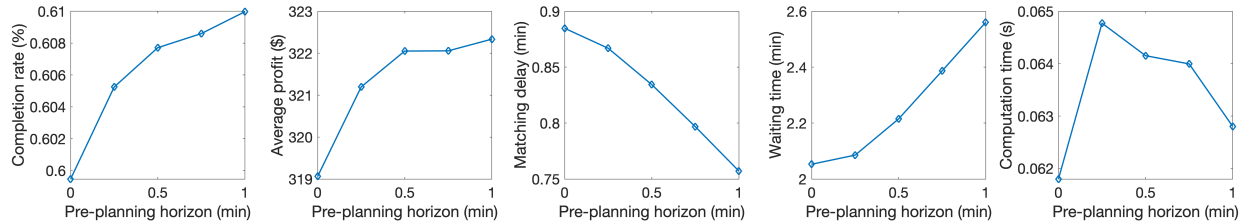


Figure 12: Pre-planning horizon H .

efficient RL-based algorithm which combines the data-driven RL components with a centralized programming based planning module. To leverage the real-time level operation and heterogeneous traffic network topology, we reformulate the online vehicle dispatching problem using a link-node based micro-network representation and integrates both the order dispatching stage and the vehicle routing stage. Instead of decentralized/sequential decisions that lead to implicit and limited cooperation among vehicles, we adopt a centralized mathematical programming model, where the system objective function is decomposed and approximated by the sum of the parameterized agent's action-value function. Two options of actions referring to order assignment and route selection are optimized collectively under system constraints to explicitly realize cooperation among agents and reduce the computational burden. Furthermore, to tackle the sparse reward signal and the transition sample imbalance issues when agents interact with a large-scale micro-network environment, we proposed TD learning with prioritized gradient descent using the adaptive exploration strategy to accelerate the learning process of agents.

Simulation experiments conducted on the Manhattan road network demonstrated the computational feasibility and scalability of the proposed algorithm in large-scale applications. Historical data were used to learn travel demand and to simulate dynamic traffic conditions over the network to support online vehicle routing. The learned information was also used to route the vacant taxis to areas of popular demand. Furthermore, compared with some existing decentralized RL algorithms and pure optimization algorithms, the experiment results confirm that the proposed algorithm not only improves the total platform revenue and average driver profit. It also reduces the platform's mismatching rate and matching delay and the customers' total waiting time.

REFERENCES

- [1] 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 *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*. ACM, 2018, pp. 905–913.
- [2] X. Tang, Z. T. 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 *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*. ACM, 2019, pp. 1780–1790.
- [3] Z. Wang, Z. Qin, X. Tang, J. Ye, and H. Zhu, "Deep reinforcement learning with knowledge transfer for online rides order dispatching," in *2018 IEEE Int. Conf. Data Mining (ICDM)*. IEEE, 2018, pp. 617–626.
- [4] J. Ke, F. Xiao, H. Yang, and J. Ye, "Learning to delay in ride-sourcing systems: a multi-agent deep reinforcement learning framework," *IEEE Trans. on Knowl. and Data Eng.*, 2020.
- [5] M. Li, Z. Qin, Y. Jiao, Y. Yang, J. Wang, C. Wang, G. Wu, and J. Ye, "Efficient ridesharing order dispatching with mean field multi-agent reinforcement learning," in *Proc. 28th Int. Conf. World Wide Web*. ACM, 2019, pp. 983–994.
- [6] 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 *Proc. 28th ACM Int. Conf. Inf. Knowl. Manage.* ACM, 2019, pp. 2645–2653.
- [7] Y. Wang, Y. Tong, C. Long, P. Xu, K. Xu, and W. Lv, "Adaptive dynamic bipartite graph matching: A reinforcement learning approach," in *Proc. IEEE 35th Int. Conf. Data Eng. ICDE*. IEEE, 2019, pp. 1478–1489.
- [8] K. Lin, R. Zhao, Z. Xu, and J. Zhou, "Efficient large-scale fleet management via multi-agent deep reinforcement learning," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*. ACM, 2018, pp. 1774–1783.
- [9] T. Oda and C. Joe-Wong, "Movi: A model-free approach to dynamic fleet management," in *IEEE INFOCOM*. IEEE, 2018, pp. 2708–2716.
- [10] D. T. Nguyen, A. Kumar, and H. C. Lau, "Credit assignment for collective multiagent rl with global rewards," in *Proc. NIPS*, 2018, pp. 8102–8113.
- [11] X. Zhou, H. Rong, C. Yang, Q. Zhang, A. V. Khezerlou, H. Zheng, M. Z. Shafiq, and A. X. Liu, "Optimizing taxi driver profit efficiency: A spatial network-based markov decision process approach," *IEEE Trans. Big Data*, 2018.
- [12] X. Yu, S. Gao, X. Hu, and H. Park, "A markov decision process approach to vacant taxi routing with e-hailing," *Transp. Res. B, Methodol.*, vol. 121, pp. 114–134, 2019.
- [13] 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," *Transp. Res. C, Emerg. Technol.*, vol. 111, pp. 91–113, 2020.
- [14] M. W. Ulmer, J. C. Goodson, D. C. Mattfeld, and M. Hennig, "Offline–online approximate dynamic programming for dynamic vehicle routing with stochastic requests," *Transp. Sci.*, vol. 53, no. 1, pp. 185–202, 2018.
- [15] L. Al-Kanj, J. Nascimento, and W. B. Powell, "Approximate dynamic programming for planning a ride-hailing system using autonomous fleets of electric vehicles," *Eur. J of Oper. Res.*, vol. 284, no. 3, pp. 1088–1106, 2020.
- [16] C. Lei, Z. Jiang, and Y. Ouyang, "Path-based dynamic pricing for vehicle allocation in ridesharing systems with fully compliant drivers," *Transp. Res. B, Methodol.*, vol. 53, pp. 60–75, 2019.
- [17] 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 *2019 IEEE Int. Conf. Data Mining (ICDM)*. IEEE, 2019, pp. 1090–1095.
- [18] 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 *Proc. 28th ACM Int. Conf. Inf. Knowl. Manage.* ACM, 2019, pp. 1983–1992.
- [19] B. L. Golden, S. Raghavan, and E. A. Wasil, *The vehicle routing problem: latest advances and new challenges*.

- Springer Science & Business Media, 2008, vol. 43.
- [20] R. Zhang, F. Rossi, and M. Pavone, "Model predictive control of autonomous mobility-on-demand systems," in *Proc. 2016 IEEE Int. Conf. on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 1382–1389.
 - [21] 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 Trans. Autom. Sci. Eng.*, vol. 13, no. 2, pp. 463–478, 2016.
 - [22] D. Bertsimas, P. Jaillet, and S. Martin, "Online vehicle routing: The edge of optimization in large-scale applications," *Oper. Res.*, vol. 67, no. 1, pp. 143–162, 2019.
 - [23] J. Ke, H. Yang, H. Zheng, X. Chen, Y. Jia, P. Gong, and J. Ye, "Hexagon-based convolutional neural network for supply-demand forecasting of ride-sourcing services," *IEEE trans. Intell. Transp. Syst.*, vol. 20, no. 11, pp. 4160–4173, 2019.
 - [24] R. Bellman, "Dynamic programming," *Science*, vol. 153, no. 3731, pp. 34–37, 1966.
 - [25] M. L. Puterman, *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.
 - [26] W. B. Powell, *Approximate Dynamic Programming: Solving the curses of dimensionality*. John Wiley & Sons, 2007, vol. 703.
 - [27] S. Schaal *et al.*, "Learning from demonstration," in *Proc. NIPS*, 1997, pp. 1040–1046.
 - [28] S. Schaal, "Is imitation learning the route to humanoid robots?" *Trends in cognitive sciences*, vol. 3, no. 6, pp. 233–242, 1999.
 - [29] A. Y. Ng, S. J. Russell *et al.*, "Algorithms for inverse reinforcement learning," in *Proc. ICML*, vol. 1, 2000, p. 2.
 - [30] K.-F. Chu, A. Y. Lam, C. Fan, and V. O. Li, "Disturbance-aware neuro-optimal system control using generative adversarial control networks," *IEEE Trans. Neural Netw. Learn. Syst.*, 2020.
 - [31] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.
 - [32] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.*, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, p. 484, 2016.
 - [33] H. Hu, S. Song, and C. L. P. Chen, "Plume tracing via model-free reinforcement learning method," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 8, pp. 2515–2527, 2019.
 - [34] Y. Hu, W. Wang, H. Liu, and L. Liu, "Reinforcement learning tracking control for robotic manipulator with kernel-based dynamic model," *IEEE Trans. Neural Netw. Learn. Syst.*, pp. 1–9, 2020.
 - [35] L. Xie, Y. Miao, S. Wang, P. Blunsom, Z. Wang, C. Chen, A. Markham, and N. Trigoni, "Learning with stochastic guidance for robot navigation," *IEEE Trans. Neural Netw. Learn. Syst.*, pp. 1–11, 2020.
 - [36] W. Liu, P. Zhuang, H. Liang, J. Peng, and Z. Huang, "Distributed economic dispatch in microgrids based on cooperative reinforcement learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 6, pp. 2192–2203, 2018.
 - [37] Z. Ni and S. Paul, "A multistage game in smart grid security: A reinforcement learning solution," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 9, pp. 2684–2695, 2019.
 - [38] S. Paul, Z. Ni, and C. Mu, "A learning-based solution for an adversarial repeated game in cyber-physical power systems," *IEEE Trans. Neural Netw. Learn. Syst.*, pp. 1–12, 2020.
 - [39] C. Liu and Y. L. Murphey, "Optimal power management based on q-learning and neuro-dynamic programming for plug-in hybrid electric vehicles," *IEEE Trans. Neural Netw. Learn. Syst.*, pp. 1–13, 2020.
 - [40] R. Arnott, "Taxi travel should be subsidized," *Journal of Urban Economics*, vol. 40, no. 3, pp. 316–333, 1996.
 - [41] J. C. Castillo, D. Knoepfle, and G. Weyl, "Surge pricing solves the wild goose chase," in *Proc. 2017 ACM Conf. on Economics and Computation*. ACM, 2017, pp. 241–242.
 - [42] K. Bimpikis, O. Candogan, and D. Saban, "Spatial pricing in ride-sharing networks," *Oper. Res.*, vol. 67, no. 3, pp. 744–769, 2019.
 - [43] Z. Xu, Y. Yin, and J. Ye, "On the supply curve of ride-hailing systems," *Transp. Res. B, Methodol.*, 2019.
 - [44] H. Yang, X. Qin, J. Ke, and J. Ye, "Optimizing matching time interval and matching radius in on-demand ride-sourcing markets," *Transp. Res. B, Methodol.*, vol. 131, pp. 84–105, 2020.
 - [45] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.
 - [46] M. Hüttenrauch, S. Adrian, G. Neumann *et al.*, "Deep reinforcement learning for swarm systems," *Journal of Machine Learning Research*, vol. 20, no. 54, pp. 1–31, 2019.
 - [47] P. Sunehag, G. Lever, A. Gruslys, W. M. Czarnecki, V. Zambaldi, M. Jaderberg, M. Lanctot, N. Sonnerat, J. Z. Leibo, K. Tuyls *et al.*, "Value-decomposition networks for cooperative multi-agent learning based on team reward," in *Proc. AAMAS*, 2018, pp. 2085–2087.
 - [48] T. Rashid, M. Samvelyan, C. S. De Witt, G. Farquhar, J. Foerster, and S. Whiteson, "Qmix: monotonic value function factorisation for deep multi-agent reinforcement learning," in *Proc. ICML*, 2018, pp. 4295–4304.
 - [49] J. Foerster, I. A. Assael, N. de Freitas, and S. Whiteson, "Learning to communicate with deep multi-agent reinforcement learning," in *Proc. NIPS*, 2016, pp. 2137–2145.
 - [50] J. N. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson, "Counterfactual multi-agent policy gradients," in *Proc. AAAI*, vol. 32, no. 1, 2018.
 - [51] K. Son, D. Kim, W. J. Kang, D. E. Hostallero, and Y. Yi, "Qtran: Learning to factorize with transformation for cooperative multi-agent reinforcement learning," in

Proc. ICML, 2019, pp. 5887–5896.

- [52] H. Kuhn, “The hungarian method for the assignment problem,” in *50 Years of Integer Programming*, 2010.
- [53] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, “Prioritized experience replay,” in *Proc. ICLR*, 2016.
- [54] A. Vahidi-Moghaddam, M. Mazouchi, and H. Modares, “Memory-augmented system identification with finite-time convergence,” *IEEE Control Systems Letters*, vol. 5, no. 2, pp. 571–576, 2020.
- [55] H. Modares, F. L. Lewis, and M.-B. Naghibi-Sistani, “Adaptive optimal control of unknown constrained-input systems using policy iteration and neural networks,” *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 24, no. 10, pp. 1513–1525, 2013.
- [56] G. Chowdhary and E. Johnson, “Concurrent learning for convergence in adaptive control without persistency of excitation,” in *49th IEEE Conference on Decision and Control (CDC)*. IEEE, 2010, pp. 3674–3679.
- [57] I. Osband, C. Blundell, A. Pritzel, and B. Van Roy, “Deep exploration via bootstrapped DQN,” in *Proc. NIPS*, 2016, pp. 4026–4034.
- [58] G. Boeing, “Osmnx: New methods for acquiring, constructing, analyzing, and visualizing complex street networks,” *Computers, Environment and Urban Systems*, vol. 65, pp. 126–139, 2017.
- [59] I. Goodfellow, Y. Bengio, and A. C. Courville, “Deep learning,” *Nature*, vol. 521, pp. 436–444, 2015.
- [60] Y. Yang, R. Luo, M. Li, M. Zhou, W. Zhang, and J. Wang, “Mean field multi-agent reinforcement learning,” in *Proc. ICML*, 2018, pp. 5571–5580.



William H.K. Lam is a Chair Professor of Civil and Transportation Engineering and the Head of the Department of Civil and Environmental Engineering at The Hong Kong Polytechnic University, P.R. China. Prof. Lam has over 35-year professional experience and is the author of about 300 SCI journal papers and book chapters, together with more than 70 consultancy reports. In the past 10-year period, he was ranked within the top 30 in the world in terms of the research publications in the ISI Web of Science categories (Transportation Science Technology OR Transportation). Prof. Lam is the one of the Editors-in-Chief of *Transportmetrica A*. He is also the current Convenor of the International Advisory Committee of the International Symposium on Transportation and Traffic Theory (ISTTT), the member of the International Scientific Committee of the International Symposium on Transportation Network Reliability (INSTR) and the member of the Board of the Eastern Asia Society for Transportation Studies (EASTS).



Agachai Sumalee was a Full Professor in transportation engineering with The Hong Kong Polytechnic University. He is currently a Chair Professor with the School of Integrated Innovation, Chulalongkorn University. His research areas are transit planning, intelligent transport system, network modeling, and optimization and transport economics. He has published more than 100 research articles in top peer-reviewed journals on these research topics. He has received several prizes and awards for his research works, including the APEC Science Prize for Innovation, Research and Education, and the National Innovation Awards of Thailand. He is also the Editor-in-Chief of *Transportmetrica B*, an Associate Editor of *Networks and Spatial Economics*, and an editorial board member of several prestigious journals.



Enming Liang received the B.Eng. degree in traffic engineering from Sun Yat-sen University, Guangzhou, China in 2020. His research interests include reinforcement learning, multi-agent system and intelligent transportation systems.



Kexin Wen received the B.S. degree in logistics engineering in 2019 from Dalian Maritime University, Dalian, China. She is currently pursuing the M.S. degree in transportation engineering at Sun Yat-sen University, Guangzhou, China. Her research interests include applications of operations research to transportation systems, port and maritime logistics.



Renxin Zhong is currently an Associate Professor with the School of Intelligent Systems Engineering, Sun Yat-sen University. He is also the Deputy Director of the Guangdong Key Laboratory of Intelligent Transportation Systems. His main research interests include traffic incident detection and management strategies, machine learning and data mining for transportation big data analysis, and optimal and nonlinear control theory with applications in transportation engineering. He received the Outstanding Dissertation Paper Award and the Gordon Newell

Memorial Prize at the 17th HKSTS International Conference, as well as the First Runner-up of the HKSTS Outstanding Student Paper Award at the 14th HKSTS International Conference. His article was shortlisted for the Best Paper Award at the IEEE ITSC 2018. His team won third place in the KDD Cup 2017 (freeway travel time and traffic flow estimations) and KDD Cup 2018 (urban pollutant prediction). Dr. Zhong serves as associate editors and guest editors for *Transportmetrica A* and *Transportmetrica B*.