

# Optimizing Long-Term Efficiency and Fairness in Ride-Hailing Under Budget Constraint via Joint Order Dispatching and Driver Repositioning

Jiahui Sun<sup>1b</sup>, Haiming Jin<sup>1b</sup>, *Member, IEEE*, Zhaoxing Yang<sup>1b</sup>, and Lu Su<sup>1b</sup>, *Member, IEEE*

**Abstract**—Ride-hailing platforms (e.g., Uber and Didi Chuxing) have become increasingly popular in recent years. *Efficiency* has always been an important metric for such platforms. However, only focusing on efficiency inevitably ignores the *fairness* of driver incomes, which could impair the sustainability of ride-hailing systems. To optimize such two essential objectives, *order dispatching* and *driver repositioning* play an important role, as they impact not only the immediate, but also the future order-serving outcomes of drivers. In practice, the platform offers monetary incentives to drivers for completing the repositioning and has a budget for the repositioning cost. Therefore, in this paper, we aim to exploit joint order dispatching and driver repositioning to optimize both long-term efficiency and fairness in ride-hailing under the budget constraint. To this end, we propose JDRCL, a novel multi-agent reinforcement learning framework, which integrates a group-based action representation that copes with the variable action space, and a primal-dual iterative training algorithm to learn a constraint-satisfying policy that maximizes both the worst and the overall incomes of drivers. Furthermore, we prove the asymptotic convergence rate of our training algorithm. Extensive experiments based on three real-world ride-hailing order datasets show that JDRCL outperforms state-of-the-art baselines on both efficiency and fairness.

**Index Terms**—Ride-hailing, long-term efficiency and fairness, joint order dispatching and driver repositioning, budget constraint.

## I. INTRODUCTION

RECENT years have witnessed a rapid development of on-demand ride-hailing platforms, such as Uber and Didi Chuxing. Rather than hailing a taxi by the road, users of the ride-hailing service submit their trip requests to the platform via mobile applications, and the platform dispatches the orders to specific drivers that are registered in the platform. Such ride-hailing service has greatly facilitated people's traveling and

commuting, and become increasingly popular. Reportedly,<sup>1,2</sup> Didi Chuxing and Uber completed 10 billion and 6.9 billion ride-hailing orders in 2019, respectively, and even under the impact of COVID-19, the number of ride-hailing orders completed by them still reached 8 billion and 5 billion, respectively, in 2020.

From the perspective of the ride-hailing platform, it is essential to optimize the *efficiency* which is often evaluated by the gross merchandise volume of the platform, and recent works [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13] have proposed a series of approaches to achieve such an objective. However, only focusing on the efficiency inevitably ignores the *fairness* of driver incomes, and may discriminate against some drivers. Consequently, the discriminated drivers will be deterred from registering in the platform, which could impair the sustainability of the overall ride-hailing system in the long run. Therefore, efficiency and fairness are both important metrics for the ride-hailing platform.

To optimize such two metrics, *order dispatching* that matches drivers with orders plays an important role, since it not only directly determines the immediate order-serving outcome, but also causes changes to the spatial distribution of drivers due to order delivery, which in turn impacts the future order-serving outcome. *Driver repositioning*, on the other hand, proactively redistributes drivers to specific regions with potentially high future demands, which greatly influences drivers' future potential of serving orders. Clearly, both of the above two operations have critical effects on the long-term efficiency and fairness. In practice, the platform offers monetary incentives to drivers for completing the repositioning [10], [14] and has a budget for the overall repositioning costs. Therefore, in this paper, we aim to address the imperative problem of optimizing the platform's efficiency and the fairness among drivers in the long run under the repositioning budget constraint via joint order dispatching and driver repositioning.

An intuitive approach for such a sequential decision-making problem is to consider the platform as an agent, who makes centralized order dispatching and repositioning decisions for all drivers. However, a centralized decision-making framework, such as single-agent reinforcement learning, faces a number of issues. One of them is the risk of "single point of failure", i.e., the

Manuscript received 12 December 2022; revised 18 November 2023; accepted 13 December 2023. Date of publication 1 January 2024; date of current version 10 June 2024. This work was supported in part by the NSF China under Grants U21A20519, U20A20181, and 62372288, and in part by DiDi GAIA Research Collaboration Plan. An extended version of this paper is presented at the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2022 [DOI: 10.1145/3534678.3539060]. Recommended for acceptance by K. Zheng. (Corresponding author: Haiming Jin.)

Jiahui Sun, Haiming Jin, and Zhaoxing Yang are with the Shanghai Jiao Tong University, Shanghai 200240, China (e-mail: jhsun1997@sjtu.edu.cn; jinhaiming@sjtu.edu.cn; yiannis@sjtu.edu.cn).

Lu Su is with Purdue University, West Lafayette, IN 47907 USA (e-mail: lusu@purdue.edu).

Our supplemental material is at <https://drive.google.com/file/d/1oTdp-BtyTShUPm4WjvEdDagNVjMYp3ZF/view?usp=sharing>  
Digital Object Identifier 10.1109/TKDE.2023.3348491

<sup>1</sup>[Online]. Available: <https://www.businessofapps.com/>

<sup>2</sup>[Online]. Available: <https://www.statista.com/>

failure of the central agent will fail the whole system. Another is the issue of exponential explosion in the decision space due to the large scale of drivers in a real-world ride-hailing system. Instead, we propose JDRCL, a novel multi-agent reinforcement learning (MARL) framework that views each driver as an agent, and trains a distributed policy for each agent.

Designing such an MARL framework faces the challenge of variable action space, since the set of candidate orders of each agent constantly changes over time. As a result, a typical policy network whose output dimension is fixed and equal to the size of the action space is thus problematic. JDRCL resolves such a challenge by segmenting each agent's dynamic action space into a fixed number of action groups, where actions in the same group can be seen as homogeneous. Then, JDRCL fixes the policy output dimension for order selection as the number of action groups, which is invariable to the size of the action space, and uses each dimension of the output to represent a possible group. In this way, the policy network of JDRCL not only overcomes the variable action space problem, but is also efficient, as it only needs one forward pass in one execution.

In terms of the fairness criterion, we adopt the max-min fairness that aims to maximize the worst driver income. As a result, our problem of solving agents' policies that jointly optimize the efficiency and fairness under budget constraint falls into the category of constrained max-min optimization, which is intractable for existing MARL frameworks. JDRCL addresses this issue by augmenting the vanilla policy gradient to a primal-dual iterative training algorithm, which alternates between updating the policy parameters and the dual variable. Specifically, each agent's policy is updated towards improving both the worst and the overall performance of agents, and the dual variable is learned to reduce the constraint violation until finding a constraint-satisfying solution. Theoretically, we prove that the training algorithm of JDRCL converges even under non-convex policy networks and stochastic gradient updating.

Overall, this paper makes the following contributions.

- To the best of our knowledge, this is the first work that exploits joint order dispatching and driver repositioning to optimize both the platform's efficiency and the fairness among drivers in the long run under the repositioning budget constraint.
- Technically, we propose JDRCL, a novel distributed MARL framework, which integrates (i) a group-based action representation that copes with the variable action space, and (ii) a primal-dual iterative training algorithm to learn a constraint-satisfying policy that maximizes both the worst and the overall performance of agents. Furthermore, we prove a sub-linear asymptotic convergence rate of our JDRCL training algorithm.
- We conduct extensive experiments to evaluate JDRCL with three public real-world ride-hailing order datasets, including over 2 million orders in Haikou, China, over 5 million orders in Chengdu, China, and over 6 million orders in New York City, USA. Our experimental results show that JDRCL demonstrates a consistent advantage compared to state-of-the-art baselines in terms of both the efficiency and fairness.

TABLE I  
SUMMARY OF MOST FREQUENTLY USED NOTATIONS

Notations	Descriptions
$\mathcal{N}, \mathcal{N}_t$	set of $N$ drivers, set of idle drivers in time slot $t$
$\mathcal{T}, \mathcal{G}$	set of $T$ time slots, set of $G$ grids
$s_t, \mathbf{o}_t$	global state and joint observation in time slot $t$
$\mathbf{o}_t^i$	agent $i$ 's observation in time slot $t$
$\mathcal{A}_t^i$	agent $i$ 's action space in time slot $t$
$\mathbf{a}_t, \mathbf{a}_t^i$	joint action and agent $i$ 's action in time slot $t$
$\pi, \pi^i$	agents' joint policy and agent $i$ 's policy
$P(s_{t+1} s_t, \mathbf{a}_t)$	state transition probability
$r_t^i$	agent $i$ 's immediate reward in time slot $t$
$c_t^i$	agent $i$ 's repositioning cost in time slot $t$
$B$	platform's repositioning budget
$J_r^i(\pi)$	agent $i$ 's expected cumulative reward
$\hat{J}_r^i(\pi)$	agent $i$ 's empirical mean cumulative reward
$J_c^i(\pi)$	agent $i$ 's expected cumulative cost
$\hat{J}_c^i(\pi)$	agent $i$ 's empirical mean cumulative cost
$\alpha$	coefficient to balance efficiency and fairness
$\mathbf{W}_c$	weight matrix of the convolution layer
$\mathbf{W}_i$	weight matrix of the linear layer
$V_r^i$	agent $i$ 's state value function of the reward
$V_c^i$	agent $i$ 's state value function of the cost
$A_r^i$	agent $i$ 's advantage function of the reward
$A_c^i$	agent $i$ 's advantage function of the cost
$\lambda, \lambda_{\max}$	dual variable, maximum dual variable
$\eta, \rho$	step sizes of dual variable and primal variable
$\theta$	parameters of agents' policies
$\theta^i$	parameter of agent $i$ 's policy
$\varphi$	parameters of agents' reward critics
$\varphi^i$	parameters of agent $i$ 's reward critic
$\xi$	parameters of agents' cost critics
$\xi^i$	parameters of agent $i$ 's cost critic
$\mathcal{D}, b_t$	replay buffer, one experience in replay buffer
$K$	the number of training epochs
$M$	the number of sampled episodes in each epoch

In the rest of this paper, we first introduce the preliminaries in Section II, and then present our formulation in Section III, as well as our solution method in Section IV. After describing the experimental results in Section V and discussing the related works in Section VI, we conclude our paper in Section VII.

## II. PRELIMINARIES

In this paper, we consider a ride-hailing system in a city, which consists of a set  $\mathcal{N} = \{1, 2, \dots, N\}$  of drivers to serve ride-hailing orders submitted by passengers. The status of each driver is either on-service, if she is serving orders at present, or idle, otherwise. As an industrial common practice [8], [9], we discretize both the time horizon and the geographical area. On one hand, the time horizon is discretized into  $T$  equal-length time slots, denoted as  $\mathcal{T} = \{1, 2, \dots, T\}$ . The ride-hailing platform receives orders at any time, but only dispatches orders at the end of each time slot. In addition to order dispatching, the platform also performs repositioning for idle drivers. On the other hand, to simplify the latitude-longitude representation of locations in a ride-hailing system, we discretize the geographical area into equal-size grids, denoted as  $\mathcal{G} = \{1, 2, \dots, G\}$ .

In this paper, we aim to optimize both the platform's efficiency and the fairness among drivers in the long run via joint order dispatching and driver repositioning, subject to the repositioning budget. To this end, we first introduce the *driver accumulative*

income (DAI) in Definition 1 and the gross merchandise volume (GMV) in Definition 2.

**Definition 1 (DAI):** Let  $\mathcal{O}_t^i$  be the set of orders that have been served by driver  $i \in \mathcal{N}$  before the end of time slot  $t$ . The driver accumulative income  $u_t^i$  of driver  $i$  at the end of time slot  $t$  is defined as  $\sum_{j \in \mathcal{O}_t^i} p_j$ , where  $p_j$  denotes the price of order  $j \in \mathcal{O}_t^i$ .

**Definition 2 (GMV):** Given  $u_T^i$  of each driver  $i \in \mathcal{N}$ , the gross merchandise volume of the platform is defined as  $\sum_{i \in \mathcal{N}} u_T^i$ , which is the sum of all drivers' DAIs at the end of the time horizon.

We use GMV to measure the platform's long-term efficiency. In terms of the criterion for fairness, we adopt the max-min fairness [15], which aims to maximize the worst DAI of all drivers at the end of the time horizon. In practice, the repositioning of idle drivers to specific regions will incur a non-negative cost proportional to the cruising distance, and the platform will set up a budget for the system repositioning cost (SRC), which is defined in Definition 3.

**Definition 3 (SRC):** Let  $c_t^i$  denote the repositioning cost of each driver  $i \in \mathcal{N}$  in each time slot  $t$ . Then, the system repositioning cost is defined as  $\sum_{i=1}^N \sum_{t=1}^T c_t^i$ , which is the sum of all drivers' repositioning costs along the whole time horizon.

Next, we present our formulation that takes the long-term efficiency, fairness, as well as the budget constraint into account in the following Section III.

### III. FORMULATION

In a real-world ride-hailing system, the joint order dispatching and driver repositioning problem is naturally a sequential decision-making problem. A typical way to resolve such a problem is to consider the platform as an agent, who makes centralized decisions for all drivers. However, such a method faces the exponential explosion problem in the state and action spaces due to the large scale of drivers that typically exist in a real-world ride-hailing system, and is thus unscalable. Therefore, we adopt a decentralized multi-agent decision-making framework. Specifically, we formulate such a problem with efficiency and fairness objectives as a *constrained multi-objective Markov game* (referred to as CMOMG), which contains the following elements.

- **Agent:** We treat each driver in the ride-hailing system as an agent in our CMOMG, and use  $\mathcal{N}$  to denote the set of agents.
- **State:** At the end of each time slot  $t$ , the state  $s_t$  consists of the numbers of idle drivers, on-service drivers, and orders waiting to be served in each grid, as well as  $u_t^i$  of each agent  $i$  and the current time slot index  $t$ .
- **Observation:** At the end of each time slot  $t$ , each agent  $i$  receives an observation  $o_t^i = [x_t^i, d_t^i]$ , which contains the information in agent  $i$ 's neighborhood set  $\mathcal{G}_t^i$  that includes the  $3 \times 3$  grids centered at agent  $i$ 's current grid. Specifically,  $x_t^i$  consists of the numbers of idle drivers, on-service drivers, and orders waiting to be served in each grid within  $\mathcal{G}_t^i$ ;  $d_t^i$  consists of agent  $i$ 's DAI at the end of time slot  $t$ , as well as the average and minimal DAI of the agents located in grids within  $\mathcal{G}_t^i$ .

- **Action:** At the end of each time slot  $t$ , the set of idle agents, denoted as  $\mathcal{N}_t$ , take actions, and the action  $a_t^i$  of each agent  $i \in \mathcal{N}_t$  indicates either choosing an order from a set  $\mathcal{U}_t^i$  of orders within her current grid, or repositioning to a grid in  $\mathcal{G}_t^i$ . We denote the action space of each agent  $i$  at time slot  $t$  as  $\mathcal{A}_t^i = \mathcal{U}_t^i \cup \mathcal{G}_t^i$ , agents' joint action at time slot  $t$  as  $\mathbf{a}_t = [a_t^i]_{i \in \mathcal{N}_t}$ , and agents' joint observation at time slot  $t$  as  $\mathbf{o}_t = [o_t^i]_{i \in \mathcal{N}_t}$ .
- **Policy:** Each agent  $i$ 's policy  $\pi^i$  specifies the probability  $\pi^i(a_t^i | o_t^i)$  that agent  $i$  takes each action  $a_t^i$  given observation  $o_t^i$  at time slot  $t$ . The joint policy of the agents are denoted as  $\pi = [\pi^i]_{i \in \mathcal{N}}$ .
- **Transition function:** Given the state  $s_t$  and joint action  $\mathbf{a}_t$ , the environment transits to the next state  $s_{t+1}$  with probability  $P(s_{t+1} | s_t, \mathbf{a}_t)$ .
- **Reward:** After taking actions, each agent  $i \in \mathcal{N}_t$  receives an instantaneous reward  $r_t^i$ , which is the order price if the action is to choose an order, and zero if the action is to reposition.
- **Cost:** In addition to the reward, each agent  $i \in \mathcal{N}_t$  also receives an immediate cost  $c_t^i$ , which is its repositioning cost if the action is to reposition, and zero, otherwise.
- **Budget:** Initially, the platform has a repositioning budget  $B$  on SRC.

In our CMOMG, the expected cumulative reward of each agent  $i$  is defined as  $J_r^i(\pi) = \mathbb{E}_{P, \pi}[\sum_{t=0}^T r_t^i]$ , which depends on the agents' joint policy  $\pi$ , and is in fact the expected DAI of agent  $i$  at the end of the time horizon. The expected cumulative cost of each agent  $i$  is defined as  $J_c^i(\pi) = \mathbb{E}_{P, \pi}[\sum_{t=0}^T c_t^i]$ , which is the expected repositioning cost of agent  $i$  over the entire time horizon. The CMOMG maximizes two objectives. The first objective is the platform's efficiency, defined as  $\sum_{j \in \mathcal{N}} J_r^j(\pi)$ . The second objective is the max-min fairness metric in terms of agents' expected DAIs, i.e.,  $\min_{i \in \mathcal{N}} J_r^i(\pi)$ . In addition, CMOMG is subject to the repositioning budget  $B$ . Formally, we formulate the problem of jointly optimizing the long-term efficiency and fairness under the budget constraint as

$$\begin{aligned} \max_{\pi} \quad & \left\{ \min_{i \in \mathcal{N}} (1 - \alpha) J_r^i(\pi) + \frac{\alpha}{N} \sum_{j \in \mathcal{N}} J_r^j(\pi) \right\} \\ \text{s.t.} \quad & \sum_{j \in \mathcal{N}} J_c^j(\pi) \leq B, \end{aligned} \quad (1)$$

where  $\alpha \in [0, 1]$  is a coefficient to trade off between the efficiency and fairness objectives.

As we consider the practical scenario where the transition function is unknown a priori, we take the approach of learning the optimal policy of our CMOMG by proposing a novel MARL framework, which will be elaborated in the following Section IV.

### IV. SOLUTION METHOD

To solve our CMOMG, we propose an MARL framework, referred to as JDRCL<sup>3</sup>, based on the actor-critic method. In

<sup>3</sup>The name JDRCL comes from Joint Order Dispatching and Driver Repositioning with Constrained MARL.

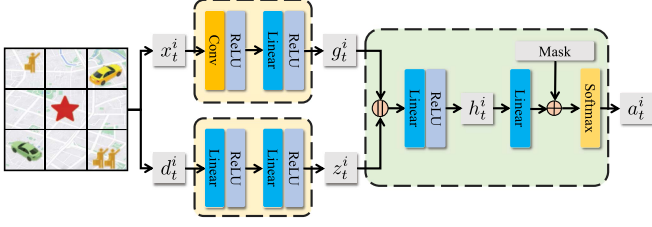


Fig. 1. Actor network of JDRCL. The star indicates the location of agent  $i$ , the orange person represents the ride-hailing order, the yellow car represents the on-service driver, the green car represents the idle driver,  $\oplus$  denotes element-wise addition, and  $\parallel$  denotes the concatenation operation.

this section, we first describe the design details of the actor and critic networks in Section IV-A, propose the training algorithm in Section IV-B, and then analyze its asymptotic convergence rate in Section IV-C. Finally, we describe the training details in Section IV-D.

#### A. Actor and Critic Network Structures

1) *Actor*: In our CMOMG, the action space  $\mathcal{A}_t^i$  of each agent  $i$  varies over time. A typical actor network whose output dimension is fixed and equal to the size of the action space is thus problematic. Existing works [11], [12] tackle this problem by taking  $a_t^i$  as an input of the actor in addition to  $o_t^i$ , and evaluating all available actions for decision making. However, such a method is inefficient, as it needs  $|\mathcal{A}_t^i|$  forward passes in a single execution.

To deal with the variable action space and improve the execution efficiency, we propose the following method. As aforementioned, each agent  $i$ 's action space consists of  $\mathcal{U}_t^i$  and  $\mathcal{G}_t^i$ . We represent each order in  $\mathcal{U}_t^i$  by its destination grid, and fix the actor's output dimension for order selection as  $G$  with each dimension representing a possible destination grid. When agent  $i$  makes an order selection decision, it chooses a grid  $g \in \mathcal{G}$  from the aforementioned  $G$  dimensions of the actor's output, and then uniformly selects one of the orders in  $\mathcal{U}_t^i$  whose destinations locate in the grid  $g$ . Our rationales for such a way of order representation and selection are as follows. On one hand, the destination of an order is a crucial metric that determines the future location of the agent that serves it, and thus the agent's future potential of serving orders. On the other hand, orders with the same origin and destination grids can be seen as homogeneous. In addition to the above  $G$  dimensions, we also introduce 9 dimensions corresponding to the 9 grids in  $\mathcal{G}_t^i$  for repositioning. Then, the output dimension of each agent  $i$ 's actor is  $G + 9$  in total, which is invariable to the size of  $\mathcal{A}_t^i$ . Such an actor network only needs one forward pass in one execution, which is described as follows.

As shown in Fig. 1, at the end of each time slot  $t$ , the input of each agent  $i$ 's actor consists of  $x_t^i$  and  $d_t^i$ .  $x_t^i$  is a tensor and fed into a convolution layer, followed by a linear layer and a non-linear activation function to output the demand-supply feature  $g_t^i$  as

$$g_t^i = f(\mathbf{W}_1 f(\mathbf{W}_c * x_t^i)),$$

where  $f$  is the ReLU activation function,  $\mathbf{W}_1$  is the weight matrix of the linear layer,  $\mathbf{W}_c$  is the convolution kernel, and  $*$  denotes the convolution operation.  $d_t^i$  is a vector and fed into a series of linear layers and non-linear activation functions to obtain the DAI feature  $z_t^i$  as

$$z_t^i = f(\mathbf{W}_3 f(\mathbf{W}_2 d_t^i)),$$

where  $\mathbf{W}_2$  and  $\mathbf{W}_3$  are weight matrices of the linear layers. Then,  $g_t^i$  and  $z_t^i$  are concatenated and further fed into a linear layer and a non-linear activation function to obtain an intermediate representation  $h_t^i$  as

$$h_t^i = f(\mathbf{W}_4 [g_t^i \parallel z_t^i]),$$

where  $\mathbf{W}_4$  is the weight matrix of the linear layer, and  $\parallel$  denotes the concatenation operation. Afterwards, the actor maps  $h_t^i$  to a  $(G + 9)$ -dimensional vector, and masks out the invalid elements of such a vector by negative infinity, including the grids that are not the destination of any order in  $\mathcal{U}_t^i$ , as well as the grids in  $\mathcal{G}_t^i$  that agent  $i$  cannot reposition to. We use  $m_t^i$  to denote such a mask. Lastly, the masked vector is fed into a Softmax layer to generate the action distribution  $\pi^i(\cdot | o_t^i)$  as

$$\pi^i(\cdot | o_t^i) = \text{Softmax}([\mathbf{W}_5 h_t^i] \oplus m_t^i),$$

where  $\mathbf{W}_5$  is the weight matrix of the linear layer, and  $\oplus$  denotes element-wise addition.

2) *Reward Critic and Cost Critic*: Each agent owns a reward critic and a cost critic to estimate its state value functions of the reward and cost, respectively, which is only utilized in the training stage for variance reduction. The structures of the actor and critic are the same except for the last layer and the input. Since we adopt the *centralized training with decentralized execution (CTDE)* paradigm, the reward and cost critics can access the global state during training, which enables a better estimation of the state value function. Thus, we use the global state  $s_t$  as the input of each agent  $i$ 's critic networks. The last layer of the reward critic network maps the intermediate representation  $h_t^i$  to a scalar to represent the state value function of the reward as

$$V_r^i(s_t) = \mathbf{W}_6 h_t^i,$$

where  $\mathbf{W}_6$  is the weight matrix of the linear layer. Similarly, the last layer of the cost critic network maps  $h_t^i$  to estimate the state value function of the cost as

$$V_c^i(s_t) = \mathbf{W}_7 h_t^i,$$

where  $\mathbf{W}_7$  is the weight matrix of the linear layer.

#### B. Training Algorithm

1) *Lagrange Transformation*: First, we reformulate Problem (1) as

$$\begin{aligned} \max_{\theta} \min_{\mathbf{w} \in \Delta} (1 - \alpha) \sum_{i \in \mathcal{N}} w_i J_r^i(\pi) + \frac{\alpha}{N} \sum_{i \in \mathcal{N}} J_c^i(\pi) \\ \text{s.t. } \sum_{i \in \mathcal{N}} J_c^i(\pi) \leq B, \end{aligned} \quad (2)$$

where  $\Delta = \{\mathbf{w} | \sum_{i \in \mathcal{N}} w_i = 1, w_i \geq 0, \forall i \in \mathcal{N}\}$  is the probability simplex. Then, with the method of Lagrange multipliers,

**Algorithm 1:** JDRCL Training Algorithm.

---

```

// Initialization.
1 Randomly initialize parameters  $\theta_0 = [\theta_0^i]_{i \in \mathcal{N}}$  of
  policies, parameters  $\varphi_0 = [\varphi_0^i]_{i \in \mathcal{N}}$  of reward
  critics, and parameters  $\xi_0 = [\xi_0^i]_{i \in \mathcal{N}}$  of cost critics;
2 Initialize dual variable  $\lambda_0$  as 0;
// Iterative training process.
3 foreach epoch  $k = 1, \dots, K$  do
  // Experience collection phase.
  4  $\mathcal{D} \leftarrow \emptyset$ ;
  5 foreach episode  $m = 1, \dots, M$  do
    6 foreach time slot  $t = 1, \dots, T$  do
      7 Each agent  $i \in \mathcal{N}_t$  observes  $o_t^i$ , takes an
        action  $a_t^i$ , and receives a reward  $r_t^i$  and a
        cost  $c_t^i$ ;
      8 Observe the next state  $s_{t+1}$  and next joint
        observation  $\mathbf{o}_{t+1}$ ;
      9 Store experience
         $(s_t, \mathbf{o}_t, \mathbf{a}_t, \mathbf{r}_t, \mathbf{c}_t, s_{t+1}, \mathbf{o}_{t+1})$  in  $\mathcal{D}$ ;
  // Parameter updating phase.
  10  $\hat{J}_r^i(\pi_{k-1}) \leftarrow \frac{1}{M} \sum_{m=1}^M \sum_{t=1}^T r_t^i, \forall i \in \mathcal{N}$ ;
  11  $\hat{J}_c^i(\pi_{k-1}) \leftarrow \frac{1}{M} \sum_{m=1}^M \sum_{t=1}^T c_t^i, \forall i \in \mathcal{N}$ ;
  12 Update  $\lambda_{k-1}$  to  $\lambda_k$  by Equation (4) with
     $\{\hat{J}_c^i(\pi_{k-1})\}_{i \in \mathcal{N}}$  as input;
  13 Solve  $\mathbf{w}_k$  by Equation (5) with  $\{\hat{J}_r^i(\pi_{k-1})\}_{i \in \mathcal{N}}$  as
    input;
  14 Update  $\theta_{k-1}$  to  $\theta_k$  via stochastic gradient ascent
    using Equation (6) with  $\lambda_k$  and  $\mathbf{w}_k$  as input;
  15 Update  $\varphi_{k-1}$  to  $\varphi_k$  by minimizing Equation (7);
  16 Update  $\xi_{k-1}$  to  $\xi_k$  by minimizing Equation (8);

```

---

we further transform Problem (2) into the following max-min problem.

$$\max_{\theta} \min_{\substack{\mathbf{w} \in \Delta \\ \lambda \geq 0}} (1 - \alpha) \sum_{i \in \mathcal{N}} w_i J_r^i(\pi) + \frac{\alpha}{N} \sum_{i \in \mathcal{N}} J_r^i(\pi) - \lambda \left( \sum_{i \in \mathcal{N}} J_c^i(\pi) - B \right), \quad (3)$$

where  $\theta = [\theta^i]_{i \in \mathcal{N}}$  and  $\mathbf{w}$  are the primal variables, and  $\lambda$  is the non-negative dual variable. We let  $L(\pi, \mathbf{w}, \lambda)$  denote

$$(1 - \alpha) \sum_{i \in \mathcal{N}} w_i J_r^i(\pi) + \frac{\alpha}{N} \sum_{i \in \mathcal{N}} J_r^i(\pi) - \lambda \left( \sum_{i \in \mathcal{N}} J_c^i(\pi) - B \right).$$

Thus, Problem (3) is equivalent to  $\max_{\pi} \min_{\mathbf{w}, \lambda} L(\pi, \mathbf{w}, \lambda)$ . Based on the primal-dual training framework [16], [17], [18], [19], [20], [21], [22], [23], [24], [25], [26], we design an iterative training algorithm that alternates between a gradient ascent step on  $\pi$ , a minimization step on  $\mathbf{w}$ , and a gradient descent step on  $\lambda$ . As the transition function is unknown, we cannot obtain the exact gradient of  $L(\pi, \mathbf{w}, \lambda)$  w.r.t. either the primal variables or the dual variable. We thus design a model-free algorithm which estimates such gradients only using samples. In the following, we add the epoch index  $k$  as a subscript to indicate the variables at the  $k$ -th iteration.

2) *Algorithm Overview:* We provide the primal-dual training algorithm in Algorithm 1. First, Algorithm 1 initializes the parameter  $\theta_0$ ,  $\varphi_0$ , and  $\xi_0$  of each agent  $i$ 's policy  $\pi^i$ , reward critic  $V_r^i$ , and cost critic  $V_c^i$ , respectively (line 1), and also initializes the dual variable  $\lambda_0$  as 0 (line 2). Then, the algorithm enters the iterative training process (line 3–16). Each iteration consists of the experience collection phase (line 4–9) and the parameter updating phase (line 10–16). In the experience collection phase, the replay buffer  $\mathcal{D}$  that is used to store experiences is first set as empty (line 4). Then, the algorithm runs the current policy  $\pi_{k-1}$  for  $M$  episodes (line 5–9). In each episode, agents interact with the environment, and the experiences of such interactions are collected into  $\mathcal{D}$  (line 6–9).

In the parameter updating phase, the algorithm first calculates  $\hat{J}_r^i(\pi_{k-1})$  and  $\hat{J}_c^i(\pi_{k-1})$  for each agent  $i$  based on the collected experiences, which is the average cumulative reward and average cumulative cost over  $M$  episodes, respectively (line 10–11). Given  $\{\hat{J}_c^i(\pi_{k-1})\}_{i \in \mathcal{N}}$ , Algorithm 1 updates the dual variable  $\lambda_{k-1}$  by projected gradient descent according to (4) (line 12). Given  $\{\hat{J}_r^i(\pi_{k-1})\}_{i \in \mathcal{N}}$ , Algorithm 1 solves  $\mathbf{w}_k$  in (5) by finding the maximum element of  $\{\hat{J}_r^i(\pi_{k-1})\}_{i \in \mathcal{N}}$  (line 13), which can be achieved by the sequential traversal. Based on the dual variable  $\lambda_k$  and weight  $\mathbf{w}_k$ , Algorithm 1 updates agents' policies by stochastic gradient ascent (line 14) according to (6), and updates agents' critics by minimizing the TD error according to (7) and (8) (line 15–16). In what follows, we present our method of updating the dual variable, as well as agents' policies and critics in detail.

3) *Updating Dual Variable:* In each epoch  $k$ , the dual variable is updated by projected gradient descent as

$$\lambda_k = \Gamma \left[ \lambda_{k-1} + \eta \left( \sum_{i \in \mathcal{N}} \hat{J}_c^i(\pi_{k-1}) - B \right) \right], \quad (4)$$

where  $\eta$  is the step size, and  $\Gamma[\cdot]$  is a projection operator that projects the dual variable into the range  $[0, \lambda_{\max}]$ .

4) *Updating Policies:* In each epoch  $k$ , the weight  $\mathbf{w}_k$  is obtained by solving the following linear program.

$$\begin{aligned} \min_{\mathbf{w}} \quad & \sum_{i \in \mathcal{N}} w_i \hat{J}_r^i(\pi_{k-1}) \\ \text{s.t.} \quad & \sum_{i \in \mathcal{N}} w_i = 1, \\ & w_i \geq 0, \forall i \in \mathcal{N}. \end{aligned} \quad (5)$$

Given the weight  $\mathbf{w}_k$  and dual variable  $\lambda_k$ , Algorithm 1 updates the parameters of  $\pi_{k-1}$  from  $\theta_{k-1}$  to  $\theta_k$  via stochastic gradient ascent as

$$\theta_k^i = \theta_{k-1}^i + \rho \nabla_{\theta_{k-1}^i} \hat{L}(\pi_{k-1}, \mathbf{w}_k, \lambda_k), \forall i \in \mathcal{N}, \quad (6)$$

where  $\rho$  is the step size, and  $\nabla_{\theta_{k-1}^i} \hat{L}(\pi_{k-1}, \mathbf{w}_k, \lambda_k)$  is the sample-based estimate of the exact gradient  $\nabla_{\theta_{k-1}^i} L(\pi_{k-1}, \mathbf{w}_k, \lambda_k)$ , as presented in Theorem 1.

*Theorem 1:* In each epoch  $k$  of Algorithm 1, given  $\mathbf{w}_k$  and  $\lambda_k$ , the sample-based estimate of the exact gradient

$\nabla_{\theta_{k-1}^i} L(\pi_{k-1}, \mathbf{w}_k, \lambda_k), \forall i \in \mathcal{N}$ , satisfies

$$\begin{aligned} & \nabla_{\theta_{k-1}^i} \widehat{L}(\pi_{k-1}, \mathbf{w}_k, \lambda_k) \\ &= \frac{1}{|\mathcal{D}|} \sum_{b_t \in \mathcal{D}} \nabla_{\theta_{k-1}^i} \log \pi_{k-1}^i(a_t^i | o_t^i) \widetilde{A}_{k-1}(s_t, \mathbf{a}_t), \end{aligned}$$

where  $b_t$  denotes one of the experiences in the replay buffer  $\mathcal{D}$ , and

$$\begin{aligned} \widetilde{A}_{k-1}(s_t, \mathbf{a}_t) &= \sum_{j \in \mathcal{N}} \left[ (1 - \alpha) w_{j,k} + \frac{\alpha}{N} \right] A_{r,k-1}^j(s_t, \mathbf{a}_t) \\ &\quad - \lambda_k \sum_{j \in \mathcal{N}} A_{c,k-1}^j(s_t, \mathbf{a}_t), \end{aligned}$$

with  $A_{r,k-1}^j(s_t, \mathbf{a}_t) = r_t^j + V_{r,k-1}^j(s_{t+1}) - V_{r,k-1}^j(s_t)$  and  $A_{c,k-1}^j(s_t, \mathbf{a}_t) = c_t^j + V_{c,k-1}^j(s_{t+1}) - V_{c,k-1}^j(s_t)$  denoting the advantage functions associated with the reward and cost of each agent  $j \in \mathcal{N}$ , respectively.

Please refer to our supplemental material, available online, for the detailed proof of Theorem 1. This theorem indicates that, each agent's policy is updated towards improving the probabilities of the actions that can increase the worst and the overall long-term rewards of agents, and decrease the overall long-term costs.

5) *Updating Critics*: In each epoch  $k$ , Algorithm 1 updates the parameters of agents' reward critics from  $\varphi_{k-1}$  to  $\varphi_k$  by minimizing the TD error over the collected experiences, given in the following (7),

$$\mathcal{L}(\varphi_{k-1}) = \sum_{b_t \in \mathcal{D}} \sum_{i \in \mathcal{N}} (r_t^i + V_{r,k-1}^i(s_{t+1}) - V_{r,k-1}^i(s_t))^2, \quad (7)$$

where  $V_{r,k-1}^i$  is the agent  $i$ 's state value function of the reward. Similarly, Algorithm 1 updates the parameters of agents' cost critics from  $\xi_{k-1}$  to  $\xi_k$  by minimizing

$$\mathcal{L}(\xi_{k-1}) = \sum_{b_t \in \mathcal{D}} \sum_{i \in \mathcal{N}} (c_t^i + V_{c,k-1}^i(s_{t+1}) - V_{c,k-1}^i(s_t))^2, \quad (8)$$

where  $V_{c,k-1}^i$  is the agent  $i$ 's state value function of the cost.

### C. Convergence Analysis

It is worth mentioning that  $L(\pi, \mathbf{w}, \lambda)$  is non-convex w.r.t. the policy parameters  $\theta$ , and the exact gradient of  $L(\pi, \mathbf{w}, \lambda)$  cannot be accessed. These reasons make the convergence rate analysis of Algorithm 1 challenging. To establish the convergence rate of Algorithm 1, we introduce two assumptions that are commonly adopted in previous works [27], [28] as follows.

*Assumption 1*:  $L(\pi, \mathbf{w}, \lambda)$  is  $l$ -smooth and  $L$ -Lipschitz with respect to the parameter  $\theta$  of  $\pi$ .

*Assumption 2*: The variance of the stochastic approximation of  $\nabla_{\theta} L(\pi, \mathbf{w}, \lambda)$  given by Theorem 1 is upper bounded by  $\sigma^2$ .

Let  $\Phi(\pi) = \max_{\mathbf{w}, \lambda} -L(\pi, \mathbf{w}, \lambda)$ . According to [27], [28], when  $\Phi(\pi)$  is differentiable,  $\pi$  is a stationary point of  $L(\pi, \mathbf{w}, \lambda)$ , if it satisfies  $\|\nabla_{\theta} \Phi(\pi)\|_2 = 0$ , and when  $\Phi(\pi)$  is non-differentiable but  $\ell$ -weakly convex,  $\pi$  is a stationary point of  $L(\pi, \mathbf{w}, \lambda)$ , if it satisfies  $\|\nabla_{\theta} \Phi_{1/2\ell}(\pi)\|_2 = 0$ , where  $\Phi_{1/2\ell}(\pi)$  is the Moreau envelope of  $\Phi(\pi)$ .

Naturally, the immediate reward  $r_t^i$ , which is the order price, and the immediate cost  $c_t^i$ , which is the repositioning cost, are both non-negative and bounded for any agent  $i$  in any time slot  $t$ . We use  $R_{\max}$  and  $C_{\max}$  to denote the largest values of the sum of agents' immediate rewards and immediate costs in one time slot, respectively.

We start our analysis on the convergence rate of Algorithm 1 by first introducing Lemma 1, which states that the inner minimization problem of (3) can be solved with a bounded error.

*Lemma 1*: Given  $\pi_{k-1}$  and a constant  $\delta \in (0, 1)$ ,  $\mathbf{w}_k$  and  $\lambda_k$  solved in each epoch  $k$  of Algorithm 1 achieves

$$L(\pi_{k-1}, \mathbf{w}_k, \lambda_k) \leq \min_{\mathbf{w}, \lambda} L(\pi_{k-1}, \mathbf{w}, \lambda) + \epsilon,$$

with probability at least  $1 - \delta$ , where

$$\epsilon = \frac{(R_{\max} + \lambda_{\max} N C_{\max}) \sqrt{2 \ln 2 / \delta}}{\sqrt{M}}.$$

Please refer to our supplemental material, available online, for the detailed proof of Lemma 1. We are in position to present the asymptotic convergence rate of Algorithm 1 in Theorem 2.

*Theorem 2*: Given  $\delta \in (0, 1)$ , we set  $\rho = \frac{1}{\sqrt{K}}$  and

$$\eta = \begin{cases} \frac{\lambda_{\max}}{|\sum_{i \in \mathcal{N}} \widehat{J}_c^i(\pi_{k-1}) - B|}, & \text{if } \sum_{i \in \mathcal{N}} \widehat{J}_c^i(\pi_{k-1}) - B \neq 0, \\ 0, & \text{otherwise.} \end{cases}$$

With probability at least  $1 - \delta$ , the output of Algorithm 1 satisfies,

$$\begin{aligned} & \frac{1}{K} \sum_{k=1}^K \|\nabla_{\theta_{k-1}} \Phi_{1/2\ell}(\pi_{k-1})\|^2 \\ & \leq \frac{2(\Delta + \ell L^2)}{\sqrt{K}} + \frac{2\ell\sigma^2}{\sqrt{K}} + \frac{4\ell(R_{\max} + \lambda_{\max} N C_{\max}) \sqrt{2 \ln \frac{2}{\delta}}}{\sqrt{M}}. \end{aligned}$$

where  $\Delta = \Phi_{1/2\ell}(\pi_0) - \min_{\pi} \Phi_{1/2\ell}(\pi)$ . That is, Algorithm 1 has an asymptotic convergence rate  $O(\frac{1}{\sqrt{K}} + \frac{1}{\sqrt{M}})$  to a stationary point.

Please refer to our supplemental material, available online, for the detailed proof of Theorem 2. Theorem 2 provides a desirable theoretical guarantee that Algorithm 1 ends with finding an approximate stationary point, where the gap is composed of three parts. The first part is caused by the distance between the initial point  $\Phi_{1/2\ell}(\pi_0)$  and the stationary point  $\min_{\pi} \Phi_{1/2\ell}(\pi)$ , and decreases at a rate of  $\frac{1}{\sqrt{K}}$ . The second part is caused by the variance of the stochastic gradient and also decreases at a rate of  $\frac{1}{\sqrt{K}}$ . The last part is due to the empirical estimates of  $J_r^i(\pi)$  and  $J_c^i(\pi)$ , and decreases at a rate of  $\frac{1}{\sqrt{M}}$ .

### D. Training Details

*Budget-Agnostic Pre-Training*: Given a budget, training agents' policies from scratch often results in a poor performance in our environment, or requires numerous training epochs to gain good performance. The reason is as follows. At the beginning, the randomly initialized policy shows great randomness and selects the repositioning actions frequently, leading to an extremely large cumulative cost that far exceeds a practical budget

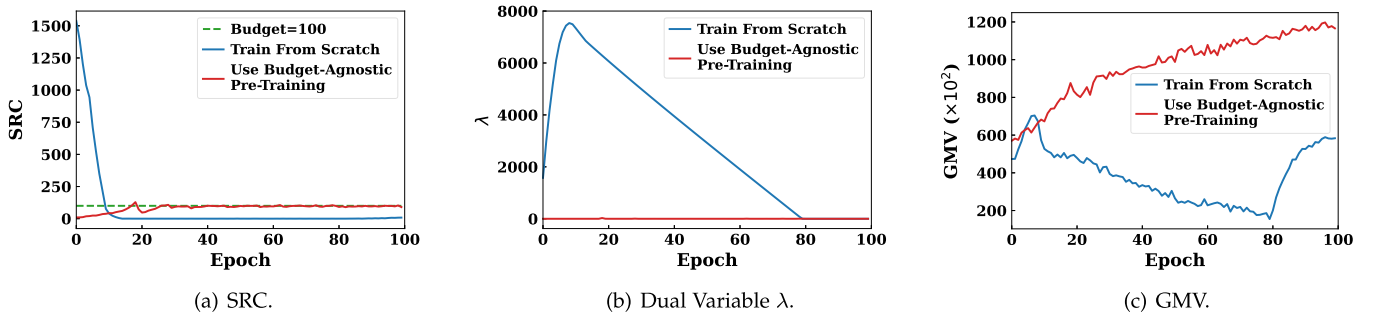


Fig. 2. Learning curves of SRC, dual variable  $\lambda$ , and GMV w.r.t. the epoch. Compared with training from scratch, our budget-agnostic pre-training technique significantly speeds up the learning process.

of the platform, as shown by Epoch 0 in Fig. 2(a). According to (4), dual variable  $\lambda$  gradually increases to a large value, so that the policy is forced to decrease the cumulative cost until the cumulative cost is below the budget (see Epoch 0 to 10 of Fig. 2(a) and (b)). After that,  $\lambda$  gradually decreases. However, due to the extremely large constraint violation at the beginning, it takes a long time for the dual variable  $\lambda$  to drop from the highest value to zero (see Epoch 10 to 80 in Fig. 2(b)). As a result, the policy remains conservative for a long time whose cumulative reward barely improves, as shown in Fig. 2(c). To overcome the problem of the large initial constraint violation, we pre-train a policy that minimizes the cumulative cost to zero, and use it as the initial policy for any given budget. Essentially, such a pre-trained policy guarantees that Algorithm 1 always starts from a feasible solution of Problem (1) for any budget, rather than an initial policy that violates the constraints severely, which significantly speeds up the training process.

**Parameter Sharing.** Due to the large scale of agents, learning a separate set of actor and critic networks for each agent is computationally expensive, and hard to scale when more agents come. We tackle this problem by letting agents share the same actor and critic networks. To distinguish different agents, a binary encoding of the agent ID is concatenated into each agent's inputs of the actor and critic networks. During the execution, each agent keeps a copy of the learned policy network.

**Worst-Off Driver First.** Different agents in the same grid might choose the same order. To avoid order selection collisions, we adopt the *worst-off driver first* principle as in [29], where the agent with the lowest DAI will choose first.

## V. EXPERIMENTS

### A. Experimental Setups

**1) Dataset:** Our experiments are based on three public real-world ride-hailing order datasets, two of which are released by Didi Chuxing GAIA Initiative,<sup>4</sup> and one of which is released by New York City Taxi and Limousine Commission.<sup>5</sup> The first dataset contains over 2 million orders in Haikou, China, from September 1st to September 30th, 2017. The second dataset

contains over 5 million orders in Chengdu, China, from November 1st to November 30th, 2016. The last dataset contains over 6 million orders in New York City, USA, from September 1st to September 30th, 2019. Each piece of data in the datasets consists of a number of features, such as the order ID, departure time, origin, destination, price and so on, which help us simulate and reproduce the behaviors of orders and drivers in the simulator. We filter out the incorrect order samples in datasets, which are the ones with missing or wrong features, such as the orders that have negative fees or missing departure times.

**2) Simulator:** To support the training and evaluation of our JDRCL, we design a ride-hailing simulator, which simulates the generation of orders and state transitions of drivers. We use OpenStreetMap<sup>6</sup> to access the geographical area of each city. As an industrial common practice [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], we discretize the city area into square grids to form a grid-based map. Such a grid-based map is used to reflect the real-time demand and supply conditions, i.e., the numbers of drivers and orders waiting to be served in each grid. In addition, a grid in the grid-based map is served as the basic unit of the destination of driver repositioning. The detailed simulation parameters are as follows. The simulator will load the correct order samples within a day, about 96K, 200K, and 230K orders in Haikou, Chengdu, and New York City, respectively, and generate up to 5K drivers online at the same time to serve orders. Such the number of orders and drivers exceeds or is competitive with the vast majority works [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13]. Orders appear at their real-happened time and locations, and the maximum waiting time of orders is 10 minutes. Drivers are randomly initialized from a discrete uniform distribution defined over grids. The speed of drivers in the simulation is between 18 km/h–40 km/h, and the maximum pickup distance of drivers is 3 km. At the end of each time slot, each idle driver uses her policy to choose an order to serve, or choose one of the neighboring grids to reposition to. After the order dispatching and driver repositioning operations, drivers who are on service drive towards order destinations through the shortest routes, and others will remain in the current grid or reposition to specific grids.

**3) Baselines:** Based on the simulator, we compare JDRCL with the following strong baselines.

<sup>4</sup>[Online]. Available: <https://gaia.didichuxing.com>

<sup>5</sup>[Online]. Available: <https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page>

<sup>6</sup>[Online]. Available: <https://www.openstreetmap.org/>

- **KM**: KM is a myopic order dispatching method that finds the maximum weight matching of the driver-order bipartite graph constructed in every time slot. We associate each driver-order pair with the order's price.
- **REA**: REA [30] is also a myopic order dispatching method. The difference between REA and KM is that REA optimizes both the platform's efficiency and the max-min fairness among drivers in one time slot. It starts from an existing matching of driver-order pairs and repeatedly revises such a matching to satisfy a given fairness threshold.
- **MFAC**: MFAC [11] formulates the order dispatching as a Markov game and proposes a mean-field actor-critic framework to simplify the agent interactions by taking the average action of neighbors into consideration. We adapt MFAC to make joint order dispatching and driver repositioning decisions. Agents cannot take repositioning actions after the repositioning budget is reached.
- **CoRide**: CoRide [13] is a hierarchical reinforcement learning framework for joint order dispatching and driver repositioning. At the lower level, each grid is treated as a worker agent and takes actions to follow the goal from the manager agent. At the upper level, a region that consists multiple grids is treated as a manager agent and generates goals for worker agents to maximize the platform's long-term efficiency.
- **CVNet**: CVNet [9] adopts a learning and planning approach for order dispatching. In the training stage, it learns a state value function via offline reinforcement learning. In the planning stage, CVNet constructs the driver-order bipartite graph, where each driver-order pair is valued as the sum of the immediate reward and the future state value, and then uses KM to find a maximum weight matching. Repositioning is treated as matching a driver to a virtual order whose fee is zero and destination is the repositioning grid. Agents cannot take repositioning actions after the budget is reached.
- **V1D3**: V1D3 [7] augments CVNet by performing both the offline learning and online learning. Specifically, V1D3 learns a state value function with online experiences in real time and ensembles such a state value function with the offline-trained one.
- **LAF**: LAF [31] is similar to CVNet, and the differences are as follows. In the learning stage, LAF trains a state value function via online learning. In the planning stage, LAF revises KM by deleting unfair driver-order pairs to optimize the fairness in a heuristic manner.
- **JDRL-SL**: JDRL [1] jointly optimizes the long-term efficiency and fairness without considering the budget constraint. We augment JDRL with a safety layer that prohibits agents to take repositioning actions when the budget is reached, which is referred to as JDRL-SL.
- **JDRL-FP**: JDRL-FP augments JDRL [1] by penalizing the objective with the constraint as

$$\max_{\pi} \min_{i \in \mathcal{N}} (1 - \alpha) J_r^i(\pi) + \frac{\alpha}{N} \sum_{j \in \mathcal{N}} J_r^j(\pi) - \gamma \sum_{j \in \mathcal{N}} J_c^j(\pi),$$

where  $\gamma$  is a penalty coefficient. In the experiments, we use grid-search for  $\gamma$  in  $[0, 14]$  and report the best result whose cumulative cost is below the given budget.

4) **Metrics**: We adopt the following metrics to evaluate all methods.

- **GMV**: GMV is defined in Definition 1 to evaluate the platform's long-term efficiency.
- **ORR**: ORR refers to order response rate, which is the number of the served orders divided by the total number of orders. ORR evaluates the platform's efficiency.
- **Worst**: Worst is a fairness metric that equals to the average of the worst 20% of all drivers' DAIs at the end of the time horizon.
- **Entropy**: Entropy refers to the *temporal earning fairness* criterion defined in [31], which reflects the variance among drivers' incomes. A larger Entropy indicates a higher earning unfairness.
- **DIR**: The average of all drivers' idle ratios. The idle ratio of a driver is the ratio of the driver's idle cruising time to the total time.
- **KL**: The average of KL divergences of the supply and demand distributions at all time slots.

To evaluate JDRL under different spatio-temporal order distributions, we report the metrics of three time periods within a day respectively, i.e., the morning period from 6 am to 12 pm, the afternoon period from 12 pm to 6 pm, and the evening period from 6 pm to 0 am. For the Haikou dataset, the morning, afternoon, and evening periods have about 21K, 37K, and 28K orders, respectively. For the Chengdu dataset, the morning, afternoon, and evening periods have about 58K, 77K, and 60K orders, respectively. For the New York City dataset, the morning, afternoon, and evening periods have about 60K, 75K, and 85K orders, respectively.

5) **Detailed Parameter Settings**: The hyper-parameters are shared in all experiments. Specifically, we run  $K = 150$  epochs for each method, and sample  $M = 2$  episodes in each epoch to strike a balance between the training complexity and the estimation accuracy. We use the Adam optimizer with the learning rate 0.0003 for actor networks and 0.001 for critic networks, and clip the actor loss and the critic loss with the ratio 0.2 that is commonly used in PPO [32]. The dimensions of  $g_t^i$ ,  $z_t^i$ , and  $h_t^i$  are set as 128, 128, and 256, respectively. We evaluate each method with 7 different seeds and report the average and standard deviation of the aforementioned metrics in the following Section V-B.

## B. Experimental Results

1) **Comparison With Baseline Methods**: In this section, we set the coefficient  $\alpha$  in JDRL-SL, JDRL-FP, and JDRL as 0.5, set the repositioning budget of each dataset as 3000, and set the number of drivers as 3000. Tables II, III, and IV summarize the efficiency and fairness results of JDRL and the baseline methods on three datasets. Overall, JDRL demonstrates a consistent advantage compared to baselines in all three datasets. Next, we will discuss some observations of the results and the rationale behind them.

TABLE II  
METRIC COMPARISON FOR THE HAIKOU DATASET

Method	Morning				Afternoon				Evening			
	GMV	ORR	Worst	Entropy	GMV	ORR	Worst	Entropy	GMV	ORR	Worst	Entropy
KM	372 ± 1	82.8 ± 0.2	4 ± 1	23.7 ± 0.3	594 ± 1	85.2 ± 0.2	3 ± 1	18.9 ± 0.2	586 ± 1	82.5 ± 0.1	6 ± 1	17.7 ± 0.3
REA	372 ± 1	82.7 ± 0.3	8 ± 1	21.7 ± 0.3	592 ± 2	85.1 ± 0.3	14 ± 2	15.5 ± 0.4	587 ± 2	82.7 ± 0.2	24 ± 2	13.7 ± 0.4
MFAC	348 ± 1	81.2 ± 0.2	10 ± 1	18.9 ± 0.3	579 ± 1	85.3 ± 0.2	24 ± 2	13.2 ± 0.4	555 ± 1	80.3 ± 0.1	80 ± 1	9.6 ± 0.2
CoRide	348 ± 3	81.3 ± 0.5	2 ± 1	24.5 ± 1.2	545 ± 11	81.7 ± 1.2	6 ± 1	19.3 ± 0.2	569 ± 1	81.8 ± 0.1	9 ± 1	20.3 ± 0.2
CVNet	360 ± 5	83.5 ± 0.9	7 ± 5	16.3 ± 1.3	577 ± 9	84.8 ± 1.2	42 ± 5	16.7 ± 2.0	597 ± 3	84.1 ± 0.3	121 ± 20	4.5 ± 1.7
V1D3	362 ± 6	83.7 ± 1.1	22 ± 3	16.3 ± 2.8	575 ± 18	84.7 ± 2.2	38 ± 3	16.6 ± 3.8	596 ± 4	83.9 ± 0.6	122 ± 13	4.1 ± 1.1
LAF	357 ± 6	83.3 ± 1.3	4 ± 2	21.2 ± 4.9	571 ± 7	83.5 ± 0.7	14 ± 8	16.0 ± 1.6	573 ± 4	81.1 ± 0.5	12 ± 7	16.7 ± 2.5
JDRL-SL	387 ± 1	87.4 ± 0.4	33 ± 3	11.3 ± 0.4	600 ± 1	88.1 ± 0.1	32 ± 3	14.5 ± 0.2	620 ± 1	88.3 ± 0.1	152 ± 2	4.7 ± 0.4
JDRL-FP	394 ± 2	90.9 ± 0.4	21 ± 5	13.7 ± 0.9	570 ± 2	83.7 ± 0.3	45 ± 2	11.6 ± 0.3	615 ± 1	87.1 ± 0.1	124 ± 3	4.5 ± 0.2
JDRCL	<b>404 ± 1</b>	<b>91.8 ± 0.2</b>	<b>63 ± 5</b>	<b>7.1 ± 0.5</b>	<b>603 ± 1</b>	<b>88.2 ± 0.1</b>	<b>74 ± 6</b>	<b>10.5 ± 0.5</b>	<b>628 ± 1</b>	<b>89.8 ± 0.1</b>	<b>156 ± 3</b>	<b>4.0 ± 0.2</b>

GMV, ORR, and Entropy are written in scientific notations that omit  $\times 10^3$ ,  $\times 10^{-2}$ , and  $\times 10^3$ , respectively. Each value is the average of 7 runs. Bold number denotes the best in each column.

TABLE III  
METRIC COMPARISON FOR THE CHENGDU DATASET

Method	Morning				Afternoon				Evening			
	GMV	ORR	Worst	Entropy	GMV	ORR	Worst	Entropy	GMV	ORR	Worst	Entropy
KM	156 ± 1	72.1 ± 0.3	15 ± 1	14.6 ± 0.4	223 ± 1	67.6 ± 0.3	33 ± 1	10.6 ± 0.3	192 ± 1	68.7 ± 0.2	25 ± 1	12.1 ± 0.3
REA	157 ± 1	72.2 ± 0.2	17 ± 1	13.7 ± 0.3	223 ± 1	67.7 ± 0.2	35 ± 1	9.8 ± 0.3	192 ± 1	68.7 ± 0.4	28 ± 1	11.1 ± 0.3
MFAC	120 ± 1	62.9 ± 0.2	2 ± 1	22.5 ± 0.2	163 ± 1	59.2 ± 0.2	9 ± 1	20.0 ± 0.1	171 ± 1	68.7 ± 0.3	30 ± 1	10.5 ± 0.3
CoRide	151 ± 1	75.2 ± 0.2	13 ± 1	17.7 ± 0.4	146 ± 3	53.9 ± 0.8	4 ± 1	26.3 ± 0.6	171 ± 2	68.0 ± 0.5	39 ± 1	4.5 ± 0.1
CVNet	149 ± 1	73.1 ± 0.4	10 ± 1	18.9 ± 0.4	206 ± 2	69.0 ± 0.5	39 ± 2	6.0 ± 0.8	159 ± 3	61.8 ± 0.9	18 ± 6	14.2 ± 3.4
V1D3	154 ± 2	74.8 ± 1.0	20 ± 5	14.8 ± 2.0	206 ± 1	69.0 ± 0.3	40 ± 1	5.7 ± 0.3	172 ± 5	66.5 ± 1.9	36 ± 1	6.6 ± 0.9
LAF	120 ± 2	60.9 ± 0.5	1 ± 1	25.5 ± 0.7	161 ± 2	54.5 ± 0.5	2 ± 1	23.6 ± 0.3	163 ± 2	63.4 ± 0.5	27 ± 3	9.9 ± 1.2
JDRL-SL	159 ± 1	74.4 ± 0.6	28 ± 1	10.5 ± 0.6	221 ± 4	69.7 ± 1.6	46 ± 4	4.9 ± 1.4	190 ± 1	71.4 ± 0.1	52 ± 1	3.3 ± 0.3
JDRL-FP	176 ± 1	85.2 ± 0.2	48 ± 1	4.4 ± 0.5	231 ± 1	73.9 ± 0.1	53 ± 1	6.3 ± 0.3	182 ± 1	69.1 ± 0.2	23 ± 1	16.4 ± 0.2
JDRCL	<b>178 ± 1</b>	<b>85.4 ± 0.2</b>	<b>51 ± 1</b>	<b>2.3 ± 0.2</b>	<b>238 ± 1</b>	<b>74.8 ± 0.2</b>	<b>59 ± 1</b>	<b>2.0 ± 0.2</b>	<b>200 ± 1</b>	<b>75.5 ± 0.1</b>	<b>55 ± 1</b>	<b>2.9 ± 0.3</b>

GMV, ORR, and Entropy are written in scientific notations that omit  $\times 10^3$ ,  $\times 10^{-2}$ , and  $\times 10^3$ , respectively. Each value is the average of 7 runs. Bold number denotes the best in each column.

TABLE IV  
METRIC COMPARISON FOR THE NEW YORK CITY DATASET

Method	Morning				Afternoon				Evening			
	GMV	ORR	Worst	Entropy	GMV	ORR	Worst	Entropy	GMV	ORR	Worst	Entropy
KM	630 ± 2	72.9 ± 0.2	24 ± 1	7.5 ± 0.2	682 ± 3	60.1 ± 0.2	11 ± 2	9.8 ± 0.3	555 ± 3	49.5 ± 0.3	35 ± 1	7.3 ± 0.3
REA	634 ± 3	73.4 ± 0.4	53 ± 1	4.3 ± 0.2	686 ± 4	60.8 ± 0.4	13 ± 2	9.5 ± 0.3	558 ± 2	49.8 ± 0.2	35 ± 2	6.9 ± 0.4
MFAC	554 ± 2	62.9 ± 0.2	27 ± 2	13.4 ± 0.2	577 ± 4	57.4 ± 0.3	11 ± 2	17.8 ± 0.4	461 ± 4	47.6 ± 0.3	4 ± 1	18.7 ± 0.2
CoRide	511 ± 5	61.7 ± 0.6	3 ± 2	21.2 ± 0.4	668 ± 7	61.9 ± 0.6	44 ± 5	8.6 ± 0.3	410 ± 4	40.6 ± 0.5	3 ± 1	26.8 ± 0.4
CVNet	634 ± 7	74.0 ± 0.7	58 ± 10	7.4 ± 1.4	642 ± 8	61.4 ± 0.5	21 ± 8	10.3 ± 0.7	561 ± 9	50.4 ± 0.9	18 ± 6	10.1 ± 0.9
V1D3	634 ± 6	74.1 ± 0.5	49 ± 13	8.1 ± 0.9	662 ± 20	62.9 ± 1.6	58 ± 10	8.5 ± 1.6	593 ± 18	53.3 ± 1.5	52 ± 18	7.3 ± 1.4
LAF	618 ± 6	73.8 ± 0.7	42 ± 12	6.9 ± 1.1	643 ± 28	59.6 ± 2.6	22 ± 10	11.9 ± 4.8	546 ± 9	49.2 ± 0.9	11 ± 7	10.8 ± 0.8
JDRL-SL	686 ± 1	81.2 ± 0.1	152 ± 3	3.1 ± 0.1	791 ± 2	72.6 ± 0.1	208 ± 5	2.4 ± 0.5	635 ± 1	58.8 ± 0.1	166 ± 2	2.9 ± 0.1
JDRL-FP	693 ± 1	83.4 ± 0.2	159 ± 3	3.4 ± 0.1	802 ± 1	72.7 ± 0.1	216 ± 1	<b>2.0 ± 0.1</b>	659 ± 1	63.2 ± 0.1	156 ± 2	3.0 ± 0.5
JDRCL	<b>694 ± 1</b>	<b>83.5 ± 0.1</b>	<b>161 ± 3</b>	<b>2.8 ± 0.2</b>	<b>807 ± 1</b>	<b>73.4 ± 0.1</b>	<b>223 ± 3</b>	2.8 ± 0.2	<b>694 ± 3</b>	<b>64.0 ± 0.3</b>	<b>170 ± 4</b>	<b>2.1 ± 0.6</b>

GMV, ORR, and Entropy are written in scientific notations that omit  $\times 10^3$ ,  $\times 10^{-2}$ , and  $\times 10^3$ , respectively. Each value is the average of 7 runs. Bold number denotes the best in each column.

Tables II, III, and IV show that JDRL-SL, JDRL-FP, and JDRCL outperform greedy methods, including KM and REA, in all datasets. This is because KM and REA are myopic methods that only account for the order-serving revenue in each time slot, instead of the whole time horizon, which may harm GMV in the long run. Compared with KM, REA achieves higher Worst and lower Entropy than KM in most scenarios, and is not much different from KM on GMV. However, as REA only optimizes fairness within each time slot, it performs less satisfactorily than JDRCL that considers the long-term fairness. Such results validate the importance of optimizing the fairness among drivers in the long run.

Among all the RL-based methods, MFAC and CoRide achieve the inferior performance in our experiments. Agents in MFAC

aim to maximize their own cumulative rewards, which incurs unnecessary competition among agents that decreases both the efficiency and fairness. CoRide trains the worker agent with the intrinsic reward generated by its manager agent, which is difficult to learn. Compared with JDRCL, CVNet, V1D3 and LAF formulate the order dispatching and driver repositioning problem in a single-agent setting, and assume all drivers are independent, which inevitably ignores the interactions among agents. Due to the online fine-tuning, V1D3 is better than CVNet in most cases. JDRCL considers the interaction between agents by incorporating the global state and sharing agents' state value functions during centralized training. Compared with JDRCL, LAF optimizes the fairness by deleting the so-called unfair driver-order pairs, which is a heuristic approach and balances

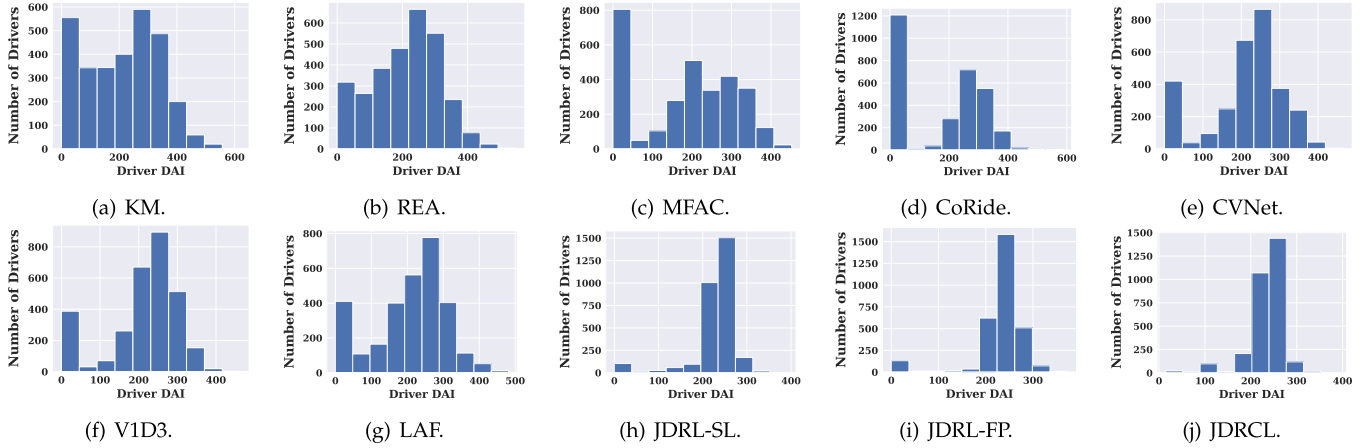


Fig. 3. Distribution of drivers' DAIs at the end of the time horizon in the Morning period of the New York City dataset.

the efficiency and fairness implicitly. Instead, JDRCL directly optimizes the efficiency and fairness objectives and also enjoys convergence guarantee.

In order not to overspend the repositioning budget, agents in JDRL-SL cannot take repositioning actions after the budget is reached. Essentially, such a hard rule restricts agents' actions and is treated as a part of the environment. On the contrary, we explicitly consider the budget constraint in our formulation and design a primal-dual training algorithm, which is shown to outperform JDRL-SL that combines MARL with hard rules. JDRL-FP is a comparable baseline if with a proper penalty coefficient. However, JDRL-FP requires a time-consuming process of hyper-parameter tuning to find such a proper penalty coefficient that satisfies the budget constraint. In contrast, our JDRCL learns the dual variables automatically during training, leading eventually to a constraint-satisfying solution.

2) *Driver Income Visualization*: In this section, we use the Morning period of the New York City dataset as an example to take a deeper look at the distribution of drivers' DAIs at the end of the time horizon. Fig. 3 shows the number of drivers with respect to the driver DAI under different methods. From this figure, we can observe that the drivers' DAIs produced by KM is quite unfair. Specifically, Fig. 3(a) shows that, although the highest driver's DAI is over 600, 40% of the drivers' DAIs are lower than 200. From Fig. 3(b), the DAI distribution of REA, which also belongs to greedy methods as KM, is obviously fairer than that of KM, since it considers the max-min fairness on drivers' DAI in every time slot. But due to the myopic nature of REA, the number of drivers whose DAIs are larger than 200 in REA is much smaller than those of JDRL-SL, JDRL-FP, and JDRCL, as shown in Fig. 3(h)–(j). Compared with all the baselines, JDRCL has the least number of drivers whose DAIs are below 100, and most drivers' DAIs are concentrated between 200 and 300. The DAI distribution of JDRCL is approximately a normal distribution with a small variance. Based on the above observations shown in Fig. 3, we can conclude that JDRCL achieves the best performance on fairness among all the evaluated methods.

3) *Correlation Between Efficiency and Fairness*: In this section, we fix the repositioning budget as 3000 and evaluate the

impact of the coefficient  $\alpha$  of JDRCL on both efficiency and fairness metrics. The results of JDRCL when  $\alpha$  changes are shown in Tables V, VI, and VII, from which we have the following interesting observations. In Table V, we can observe that as  $\alpha$  decreases, GMV and ORR gradually decrease, and Worst gradually increases, which indicates that the efficiency and fairness conflict with each other. However, in Tables VI and VII, GMV and Worst do not demonstrate such a conflicting phenomenon. In particular, GMV, ORR and Worst achieve maximal at the same time when  $\alpha = 0.5$  in the Morning period of the Chengdu dataset, as well as when  $\alpha = 0.5$  in the Afternoon period of the New York City dataset. Our reasoning of the above results is as follows. The Chengdu and New York City dataset have significantly more orders than the Haikou dataset, such that each driver in Chengdu or Haikou has a large number of orders to serve. Thus, both GMV and Worst could be fairly large.

4) *Impact of the Budget*: In this section, we use the Evening period of the Chengdu dataset as an example to show the impact of the budget in Fig. 4. The coefficient  $\alpha$  in JDRL-SL, JDRL-FP, and JDRCL is set as 0.5. As shown in Fig. 4(e) and (f), with the increase of the repositioning budget, both KL and DIR gradually decrease, indicating that the supply and demand distributions become more aligned and the idle cruising of drivers becomes less. Meanwhile, the overall efficiency and fairness of a ride-hailing system gradually become better, as shown in Fig. 4(a)–(d). Such results are in line with our expectations, as the higher the platform's budget, the more repositioning decisions the platform can make. Moreover, Fig. 4 shows that JDRCL demonstrates a consistent advantage over the baselines in all the budgets.

In this experiment, we also show the budget satisfaction of our JDRCL in Table VIII. As each action has a certain probability of being chosen, there is a probability of the cumulative cost exceeding the budget in one episode. However, our formulation aims to ensure that the expected cumulative cost over episodes is no larger than the budget, not the cumulative cost of a single episode. Policies trained by JDRCL achieve such an effect, which can be validated from Table VIII. For each budget, we execute the learned policy for 7 episodes and average the cumulative

TABLE V  
METRIC COMPARISON FOR THE HAIKOU DATASET

JDRCL	Morning				Afternoon				Evening			
	GMV	ORR	Worst	Entropy	GMV	ORR	Worst	Entropy	GMV	ORR	Worst	Entropy
$\alpha = 1.0$	<b>410 ± 1</b>	<b>93.0 ± 0.1</b>	16 ± 1	15.3 ± 0.3	<b>619 ± 1</b>	<b>90.1 ± 0.1</b>	58 ± 3	9.4 ± 0.5	<b>633 ± 1</b>	<b>89.9 ± 0.1</b>	144 ± 2	5.1 ± 0.4
$\alpha = 0.7$	408 ± 1	92.4 ± 0.3	36 ± 6	10.8 ± 0.6	614 ± 1	89.6 ± 0.1	68 ± 3	9.5 ± 0.4	631 ± 1	89.2 ± 0.1	150 ± 4	5.1 ± 0.3
$\alpha = 0.5$	404 ± 1	91.8 ± 0.2	63 ± 5	7.1 ± 0.5	603 ± 1	88.2 ± 0.1	74 ± 6	10.5 ± 0.5	628 ± 1	89.8 ± 0.1	156 ± 3	4.0 ± 0.2
$\alpha = 0.3$	395 ± 1	89.8 ± 0.2	65 ± 3	6.8 ± 0.5	610 ± 2	89.3 ± 0.2	90 ± 5	<b>7.4 ± 0.3</b>	625 ± 1	88.9 ± 0.1	161 ± 1	4.7 ± 0.4
$\alpha = 0.1$	398 ± 1	90.7 ± 0.2	<b>76 ± 1</b>	<b>6.1 ± 0.3</b>	591 ± 5	87.3 ± 0.5	<b>95 ± 4</b>	9.2 ± 0.4	620 ± 1	88.3 ± 0.1	<b>167 ± 2</b>	<b>3.6 ± 0.3</b>

GMV, ORR, and Entropy are written in scientific notations that omit  $\times 10^3$ ,  $\times 10^{-2}$ , and  $\times 10^3$ , respectively. Each value is the average of 7 runs. Bold number denotes the best in each column.

TABLE VI  
METRIC COMPARISON FOR THE CHENGDU DATASET

JDRCL	Morning				Afternoon				Evening			
	GMV	ORR	Worst	Entropy	GMV	ORR	Worst	Entropy	GMV	ORR	Worst	Entropy
$\alpha = 1.0$	172 ± 1	80.9 ± 0.2	46 ± 1	4.2 ± 0.3	<b>242 ± 1</b>	76.6 ± 0.1	<b>62 ± 1</b>	2.9 ± 0.2	<b>201 ± 0.7</b>	75.1 ± 0.2	<b>56 ± 1</b>	3.5 ± 0.2
$\alpha = 0.7$	173 ± 1	81.3 ± 0.1	49 ± 1	3.4 ± 0.4	241 ± 1	<b>76.8 ± 0.1</b>	59 ± 1	2.1 ± 0.1	199 ± 0.4	74.7 ± 0.1	54 ± 1	<b>2.7 ± 0.1</b>
$\alpha = 0.5$	<b>178 ± 1</b>	<b>85.4 ± 0.2</b>	<b>51 ± 1</b>	<b>2.3 ± 0.2</b>	238 ± 1	74.8 ± 0.2	59 ± 1	<b>2.0 ± 0.2</b>	200 ± 0.2	<b>75.5 ± 0.1</b>	55 ± 1	2.9 ± 0.3
$\alpha = 0.3$	169 ± 1	80.3 ± 0.2	42 ± 1	4.4 ± 0.3	227 ± 1	73.5 ± 0.2	48 ± 1	4.4 ± 0.3	188 ± 0.9	71.8 ± 0.3	50 ± 1	2.9 ± 0.3
$\alpha = 0.1$	164 ± 1	78.5 ± 0.2	42 ± 1	5.6 ± 0.3	218 ± 1	71.2 ± 0.2	42 ± 1	7.0 ± 0.5	181 ± 0.5	69.7 ± 0.2	44 ± 1	4.1 ± 0.2

GMV, ORR, and Entropy are written in scientific notations that omit  $\times 10^3$ ,  $\times 10^{-2}$ , and  $\times 10^3$ , respectively. Each value is the average of 7 runs. Bold number denotes the best in each column.

TABLE VII  
METRIC COMPARISON FOR THE NEW YORK CITY DATASET

JDRCL	Morning				Afternoon				Evening			
	GMV	ORR	Worst	Entropy	GMV	ORR	Worst	Entropy	GMV	ORR	Worst	Entropy
$\alpha = 1.0$	693 ± 1	82.5 ± 0.1	172 ± 4	2.4 ± 0.3	793 ± 1	70.4 ± 0.1	206 ± 3	2.6 ± 0.1	<b>724 ± 1</b>	<b>66.1 ± 0.2</b>	160 ± 3	3.2 ± 0.2
$\alpha = 0.7$	693 ± 1	83.0 ± 0.1	<b>178 ± 3</b>	2.6 ± 0.2	788 ± 1	70.7 ± 0.1	173 ± 4	4.8 ± 0.2	678 ± 1	63.6 ± 0.1	163 ± 1	3.5 ± 0.1
$\alpha = 0.5$	<b>694 ± 1</b>	<b>83.5 ± 0.1</b>	161 ± 3	2.8 ± 0.2	<b>807 ± 1</b>	<b>73.4 ± 0.1</b>	<b>223 ± 3</b>	2.8 ± 0.2	694 ± 3	64.0 ± 0.3	<b>170 ± 4</b>	<b>2.1 ± 0.6</b>
$\alpha = 0.3$	688 ± 2	82.1 ± 0.1	169 ± 4	<b>2.3 ± 0.2</b>	791 ± 3	72.2 ± 0.2	206 ± 4	3.1 ± 0.2	701 ± 3	64.6 ± 0.2	152 ± 4	2.9 ± 0.2
$\alpha = 0.1$	672 ± 1	80.1 ± 0.1	149 ± 4	3.0 ± 0.4	742 ± 3	69.4 ± 0.3	154 ± 6	<b>2.5 ± 0.3</b>	677 ± 4	62.6 ± 0.3	153 ± 2	2.3 ± 0.3

GMV, ORR, and Entropy are written in scientific notations that omit  $\times 10^3$ ,  $\times 10^{-2}$ , and  $\times 10^3$ , respectively. Each value is the average of 7 runs. Bold number denotes the best in each column.

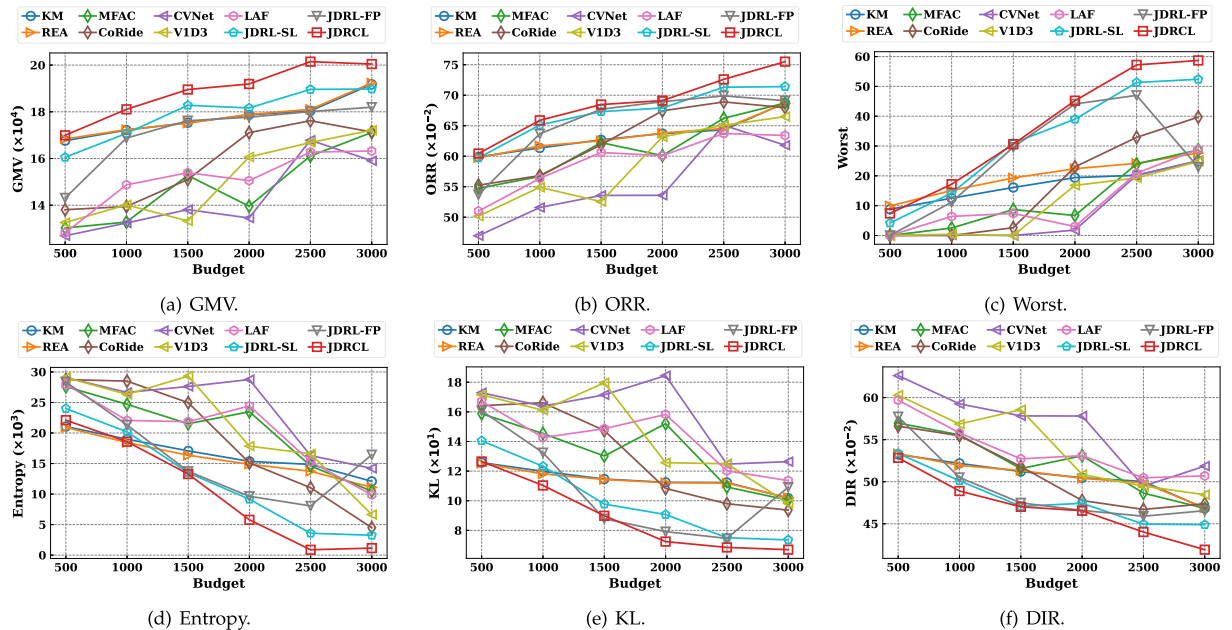


Fig. 4. Impact of the budget in the Evening period of the Chengdu dataset. JDRCL is superior in all the budgets.

TABLE VIII  
AVERAGED SRC OF JDRCL OVER 7 RUNS UNDER DIFFERENT BUDGETS IN  
THE EVENING PERIOD OF THE CHENGDU DATASET

$B$	500	1000	1500	2000	2500	3000
SRC	$453 \pm 8$	$808 \pm 25$	$1310 \pm 36$	$1831 \pm 29$	$2342 \pm 41$	$2547 \pm 82$

costs over episodes. The experimental results demonstrates that JDRCL satisfies the budget constraints consistently.

5) *Impact of the Number of Drivers*: In this section, we vary the number of drivers from 1K to 5K to test its effect on the results of our JDRCL in the New York City dataset. The coefficient  $\alpha$  is set as 0.5, and the budget is set as 3000. The results are shown in Table IX. As the number of drivers increases, both GMV and ORR gradually increase, but the rate of such increasement gradually decreases. Worst and Entropy become worse significantly as the number of drivers increases. Our reasoning of the above results is as follows. When the number of drivers is small, each driver has a large number of orders to serve, and thus Worst can be fairly large. With the increase of the number of drivers, although the total number of served orders increases, the number of served orders of some drivers decreases, resulting in a decline in Worst.

6) *Impact of the Grid Size*: In this section, we use the Afternoon period of the New York City dataset as an example to show the impact of the size of the grid on the results of our JDRCL. The coefficient  $\alpha$ , the budget, and the number of drivers are fixed as 0.5, 3000, and 3000, respectively. As shown in Table X, the policy performs poorly when the side length of the grid is too small (e.g., 1 km) or too large (e.g., 5 km). The reason is as follows. On the one hand, the size of the grid affects the resolution of the state representation, as the state in each time slot consists of the numbers of idle drivers, on-service drivers, and orders waiting to be served in each grid. If the grid is too large, the resolution of the state representation will be too low to contain enough information. If the grid is too small, the dimension of the state representation will be rather high, which increases the computational complexity. On the other hand, the size of the grid also affects the spatial granularity of driver repositioning. If the grid is too large, it is infeasible to finely control the drivers' positions. If the grid is too small, the action space of driver repositioning will be too large, making the repositioning decision overcomplicated. Therefore, we set the side length of the grid as 3 km in our experiments to strike a balance between the performance and computational complexity according to Table X.

## VI. RELATED WORK

With the rapid development of online ride-hailing systems, solving their unique problems has attracted growing attentions nowadays. For example, [33], [34], [35], [36] design ride-sharing mechanisms to improve the utilization of vehicles among multiple demands. [37], [38], [39] propose route planning and recommendation algorithms for drivers. [40], [41] integrate spatial-temporal information to predict the time of arrivals of

vehicles or passenger trip orders. Apart from serving passengers, [42], [43], [44] also propose algorithms to enable vehicles for other urban crowdsourcing tasks, such as urban sensing and food delivery. In contrast, our work focuses on making joint order dispatching and driver repositioning decisions in ride-hailing platforms. Table XI summarizes recent works [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [29], [30], [31], [45], [46], [47], [48], [49], [50] related to optimizing the aforementioned two decision-making tasks.

Specifically, [2], [3], [8], [9] adopt a learning and planning approach for order dispatching. In the learning stage, they train a state value function. In the planning stage, they construct the driver-order bipartite graph whose edges' weights are valued as the sum of the immediate reward and the future state value, and then find a maximum weight matching to optimize the long-term efficiency. [5] also adopts the learning and planning approach as [2], [3], [8], [9], but it learns the delayed time for each passenger request in the learning stage. [4] advocates a federated order dispatching for cross-platform ride-hailing to boost the total order-serving revenue. [6], [11], [12], [45] design MARL algorithms to dispatch orders in a decentralized way, where [11], [45] aim to achieve Nash equilibrium among agents, and [6], [12] focus on aligning the distributions of drivers and orders. For the driver repositioning task, [10] proposes an MARL framework, and [49] introduces the driver interaction design with a modularized algorithm to reposition drivers coordinately to appropriate regions with the goal of mitigating the demand-supply gap. [13] proposes a hierarchical MARL framework to make joint order dispatching and driver repositioning decisions, aiming to maximize the platform's long-term efficiency. [7] achieves such an objective by extending the method in [9], and learns a unified value function for both order dispatching and driver repositioning tasks. However, all works mentioned above [2], [3], [7], [8], [9], [10], [11], [12], [13], [45], [49] only focus on improving the efficiency and ignore the fairness of driver incomes. In contrast, we consider to optimize both the efficiency and fairness in the ride-hailing system.

Similar to our work, a line of recent works [29], [30], [31], [46], [47], [48], [50] also take both efficiency and fairness as objectives. Our work differs from them in various ways. Specifically, [30], [46], [47], [48] aim to optimize myopic fairness when dispatching orders in each time slot. However, our work focuses on optimizing long-term fairness, which is essential to the sustainability of the ride-hailing system in the long run. Moreover, all the aforementioned works do not consider the repositioning budget of a ride-hailing platform. As indicated by Table XI, our work is the first one that exploits joint order dispatching and driver repositioning to optimize both the long-term efficiency and fairness under the repositioning budget constraint in the ride-hailing system.

To handle the constraints, many reinforcement learning works [16], [17], [18], [19], [20], [21], [22], [23], [24], [25], [26] utilize the canonical Lagrange transformation and primal-dual training techniques in their methods. Among them, [16] first proves that the primal-dual training has a zero duality gap. [17], [18], [19], [20], [21] are a series of works to pursue tighter bounds on the optimality gap and constraint violation.

TABLE IX  
IMPACT OF THE NUMBER OF DRIVERS IN NEW YORK CITY DATASET

Number of Drivers	Morning				Afternoon				Evening			
	GMV	ORR	Worst	Entropy	GMV	ORR	Worst	Entropy	GMV	ORR	Worst	Entropy
1000	330 ± 2	33.9 ± 0.1	<b>285 ± 2</b>	<b>1.1 ± 0.1</b>	345 ± 4	26.9 ± 0.2	<b>284 ± 4</b>	<b>1.5 ± 0.2</b>	292 ± 2	24.5 ± 0.2	<b>254 ± 3</b>	<b>1.0 ± 0.1</b>
2000	534 ± 5	65.5 ± 0.6	179 ± 6	2.8 ± 0.3	610 ± 3	57.0 ± 0.1	220 ± 3	2.3 ± 0.2	489 ± 3	46.3 ± 0.3	138 ± 4	3.9 ± 0.2
3000	694 ± 1	83.5 ± 0.1	161 ± 3	2.8 ± 0.2	807 ± 1	73.4 ± 0.1	223 ± 3	2.8 ± 0.2	694 ± 3	64.0 ± 0.3	170 ± 4	2.1 ± 0.6
4000	731 ± 1	<b>86.6 ± 0.1</b>	101 ± 6	4.2 ± 0.2	852 ± 2	76.8 ± 0.3	154 ± 6	4.0 ± 0.2	709 ± 5	63.5 ± 0.4	76 ± 9	9.0 ± 0.6
5000	<b>733 ± 1</b>	86.4 ± 0.0	20 ± 3	9.9 ± 0.3	<b>866 ± 3</b>	<b>77.9 ± 0.2</b>	12 ± 4	15.8 ± 1.0	<b>728 ± 2</b>	<b>66.6 ± 0.2</b>	30 ± 4	12.0 ± 0.3

GMV, ORR, and Entropy are written in scientific notations that omit  $\times 10^3$ ,  $\times 10^{-2}$ , and  $\times 10^3$ , respectively. Each value is the average of 7 runs. Bold number denotes the best in each column.

TABLE X  
IMPACT OF THE SIZE OF THE GRID IN NEW YORK CITY DATASET

Grid Size	GMV	ORR	Worst	Entropy
1 km	698 ± 3	62.7 ± 0.2	23 ± 4	10.2 ± 0.6
2 km	<b>821 ± 2</b>	73.0 ± 0.1	95 ± 3	5.8 ± 0.3
3 km	807 ± 1	<b>73.4 ± 0.1</b>	<b>219 ± 1</b>	<b>2.8 ± 0.2</b>
4 km	768 ± 1	70.7 ± 0.1	42 ± 3	9.9 ± 0.3
5 km	699 ± 5	60.8 ± 0.4	2 ± 2	16.3 ± 0.5

GMV, ORR, and Entropy are written in scientific notations that omit  $\times 10^3$ ,  $\times 10^{-2}$ , and  $\times 10^3$ , respectively. Each value is the average of 7 runs. Bold number denotes the best in each column.

TABLE XI  
CLASSIFICATION OF RECENT WORKS RELATED TO ORDER DISPATCHING (OD)  
AND DRIVER REPOSITIONING (DR)

Decisions	Efficiency	Efficiency & Fairness
OD	[2–6, 8, 9, 11, 12, 46]	[30–32, 47–49]
DR	[10, 50]	[51]
OD & DR	[7, 13]	<b>JDRCL</b>

Compared with [16], [17], [18], [19], [20], [21], we do not have the assumption of tabular or linear MDPs and prove the asymptotic convergence guarantee of our algorithm even under non-convex policy networks and stochastic gradient updating scheme. [22] first combines the primal-dual method with the actor-critic framework. [23] leverages the PID control to improve the updating of dual variables to make the training process more stable. [24] introduces the model-based RL to the primal-dual method to improve sample efficiency. Compared with the above works, we focus on the multi-agent setting where agents learn to optimize the team efficiency and individual fairness under the team budget. [25] is the first work that applies the primal-dual training in the MARL setting, and [26] studies the fully decentralized MARL with constraints. Compared with all the aforementioned works, we analyze the reason of the inefficiency of the primal-dual training and propose a budget-agnostic pre-training technique, which significantly speeds up the primal-dual training process.

## VII. CONCLUSION

In this paper, we study the problem of optimizing the platform's efficiency and the fairness among drivers in the long run under the budget constraint via joint order dispatching and driver repositioning. To this end, we formulate such a problem as a constrained multi-objective Markov game that explicitly incorporates the long-term efficiency, the max-min fairness, and

the repositioning budget, and propose a novel MARL framework, referred to as JDRCL, to help drivers make distributed order selection and repositioning decisions. Specifically, JDRCL contains the following two key designs. First, JDRCL segments the dynamic action space that exists in the order dispatching process into a fixed number of action groups, where actions in the same group can be seen as homogeneous. Then, JDRCL fixes the policy output dimension for order selection as the number of groups to overcome the variable action space problem in an efficient manner. Second, JDRCL proposes a primal-dual iterative training algorithm to solve agents' policies that jointly optimize the efficiency and fairness under the budget constraint. Such an algorithm alternates between reducing the constraint violation and improving the worst as well as the overall performance of agents. We prove the asymptotic convergence rate of our JDRCL training algorithm even under non-convex policy networks and stochastic gradient updating. Extensive experiments across three public real-world ride-hailing order datasets demonstrate that JDRCL consistently outperforms state-of-the-art baselines in terms of both the efficiency and fairness.

According to the report,<sup>7</sup> there are about 13K Yellow Taxis in New York City. Admittedly, the number of drivers in our simulation is less than that in reality. What our experiments verified is the performance of our method and baselines if there are up to 5K drivers online at the same time to serve the Yellow Taxis' orders. Adding more drivers requires higher computational time and resources for simulation and training. We leave it for our future work.

## REFERENCES

- [1] J. Sun, H. Jin, Z. Yang, L. Su, and X. Wang, "Optimizing long-term efficiency and fairness in ride-hailing via joint order dispatching and driver repositioning," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2022, pp. 3950–3960.
- [2] B. Han, H. Lee, and S. Martin, "Real-time rideshare driver supply values using online reinforcement learning," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2022, pp. 2968–2976.
- [3] S. S. Eshkevari et al., "Reinforcement learning in the wild: Scalable RL dispatching algorithm deployed in ridehailing marketplace," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2022, pp. 3838–3848.
- [4] Y. Wang et al., "Fed-LTD: Towards cross-platform ride hailing via federated learning to dispatch," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2022, pp. 4079–4089.
- [5] 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. Knowl. Data Eng.*, vol. 34, no. 5, pp. 2280–2292, May 2022.

<sup>7</sup>[Online]. Available: [www.nyc.gov/assets/tlc/downloads/pdf/annual\\_report\\_2021.pdf](http://www.nyc.gov/assets/tlc/downloads/pdf/annual_report_2021.pdf)

- [6] F. Zhou et al., "Multi-objective distributional reinforcement learning for large-scale order dispatching," in *Proc. IEEE Int. Conf. Data Mining*, 2021, pp. 1541–1546.
- [7] X. Tang et al., "Value function is all you need: A unified learning framework for ride-hailing platforms," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2021, pp. 3605–3615.
- [8] Z. Xu et al., "Large-scale order dispatch in on-demand ride-hailing platforms: A learning and planning approach," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2018, pp. 905–913.
- [9] X. Tang et al., "A deep value-network based approach for multi-driver order dispatching," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2019, pp. 1780–1790.
- [10] K. Lin, R. Zhao, Z. Xu, and J. Zhou, "Efficient large-scale fleet management via multi-agent deep reinforcement learning," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2018, pp. 1774–1783.
- [11] M. Li et al., "Efficient ridesharing order dispatching with mean field multi-agent reinforcement learning," in *Proc. World Wide Web Conf.*, 2019, pp. 983–994.
- [12] M. Zhou et al., "Multi-agent reinforcement learning for order-dispatching via order-vehicle distribution matching," in *Proc. ACM Int. Conf. Inf. Knowl. Manage.*, 2019, pp. 2645–2653.
- [13] J. Jin et al., "CoRide: Joint order dispatching and fleet management for multi-scale ride-hailing platforms," in *Proc. ACM Int. Conf. Inf. Knowl. Manage.*, 2019, pp. 1983–1992.
- [14] Z. T. Qin, H. Zhu, and X. Ye, "Reinforcement learning for ridesharing: An extended survey," *Transp. Res. Part C: Emerg. Technol.*, vol. 144, 2022, Art. no. 103852.
- [15] J. Rawls, *A Theory of Justice: Revised Edition*. Cambridge, MA, USA: Harvard Univ. Press, 1999.
- [16] S. Paternain, L. F. O. Chamon, M. Calvo-Fullana, and A. Ribeiro, "Constrained reinforcement learning has zero duality gap," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2019, pp. 7553–7563.
- [17] Y. Efroni, S. Mannor, and M. Pirotta, "Exploration-exploitation in constrained MDPs," 2020, *arXiv: 2003.02189*.
- [18] S. Miryosefi and C. Jin, "A simple reward-free approach to constrained reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2022, pp. 15666–15698.
- [19] T. Liu, R. Zhou, D. Kalathil, P. R. Kumar, and C. Tian, "Learning policies with zero or bounded constraint violation for constrained MDPs," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2021, pp. 17183–17193.
- [20] S. Vaswani, L. Yang, and C. Szepesvári, "Near-optimal sample complexity bounds for constrained MDPs," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2022, pp. 3110–3122.
- [21] A. Bura, A. HasanadeZonuz, D. Kalathil, S. Shakkottai, and J. Chamberland, "DOPE: Doubly optimistic and pessimistic exploration for safe reinforcement learning," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2022, pp. 1047–1059.
- [22] C. Tessler, D. J. Mankowitz, and S. Mannor, "Reward constrained policy optimization," in *Proc. Int. Conf. Learn. Representations*, 2019.
- [23] A. Stooke, J. Achiam, and P. Abbeel, "Responsive safety in reinforcement learning by PID Lagrangian methods," in *Proc. Int. Conf. Mach. Learn.*, 2020, Art. no. 847.
- [24] A. K. Jayant and S. Bhatnagar, "Model-based safe deep reinforcement learning via a constrained proximal policy optimization algorithm," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2022, pp. 24432–24445.
- [25] S. Gu et al., "Multi-agent constrained policy optimisation," 2021, *arXiv:2110.02793*.
- [26] S. Lu, K. Zhang, T. Chen, T. Basar, and L. Horesh, "Decentralized policy gradient descent ascent for safe multi-agent reinforcement learning," in *Proc. AAAI Conf. Artif. Intell.*, 2021, pp. 8767–8775.
- [27] C. Jin, P. Netrapalli, and M. I. Jordan, "What is local optimality in nonconvex-nonconcave minimax optimization?," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 4880–4889.
- [28] K. K. Thekumparampil, P. Jain, P. Netrapalli, and S. Oh, "Efficient algorithms for smooth minimax optimization," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2019, pp. 12659–12670.
- [29] T. Sühr, A. J. Biega, M. Zehlike, K. P. Gummedi, and A. Chakraborty, "Two-sided fairness for repeated matchings in two-sided markets: A case study of a ride-hailing platform," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2019, pp. 3082–3092.
- [30] N. S. Lesmana, X. Zhang, and X. Bei, "Balancing efficiency and fairness in on-demand ridesourcing," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2019, pp. 5310–5320.
- [31] D. Shi, Y. Tong, Z. Zhou, B. Song, W. Lv, and Q. Yang, "Learning to assign: Towards fair task assignment in large-scale ride hailing," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2021, pp. 3549–3557.
- [32] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017, *arXiv: 1707.06347*.
- [33] Y. Xu, Y. Tong, Y. Shi, Q. Tao, K. Xu, and W. Li, "An efficient insertion operator in dynamic ridesharing services," *IEEE Trans. Knowl. Data Eng.*, vol. 34, no. 8, pp. 3583–3596, Aug. 2022.
- [34] H. Luo, Z. Bao, F. M. Choudhury, and J. S. Culpepper, "Dynamic ridesharing in peak travel periods," *IEEE Trans. Knowl. Data Eng.*, vol. 33, no. 7, pp. 2888–2902, Jul. 2021.
- [35] Y. Li et al., "Top- $k$  vehicle matching in social ridesharing: A price-aware approach," *IEEE Trans. Knowl. Data Eng.*, vol. 33, no. 3, pp. 1251–1263, Mar. 2021.
- [36] Z. Liu, Z. Gong, J. Li, and K. Wu, "Mobility-aware dynamic taxi ridesharing," in *Proc. IEEE Int. Conf. Data Eng.*, 2020, pp. 961–972.
- [37] L. Duan et al., "Efficiently solving the practical vehicle routing problem: A novel joint learning approach," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2020, pp. 3054–3063.
- [38] J. Wang et al., "Demand-aware route planning for shared mobility services," in *Proc. VLDB Endowment*, vol. 13, pp. 979–991, 2020.
- [39] D. Chen, Y. Yuan, W. Du, Y. Cheng, and G. Wang, "Online route planning over time-dependent road networks," in *Proc. IEEE Int. Conf. Data Eng.*, 2021, pp. 325–335.
- [40] H. Hong et al., "HetETA: Heterogeneous information network embedding for estimating time of arrival," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2020, pp. 2444–2454.
- [41] X. Fang, J. Huang, F. Wang, L. Liu, Y. Sun, and H. Wang, "SSML: Self-supervised meta-learner for En route travel time estimation at Baidu maps," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2021, pp. 2840–2848.
- [42] Y. Tong, Y. Zeng, B. Ding, L. Wang, and L. Chen, "Two-sided online micro-task assignment in spatial crowdsourcing," *IEEE Trans. Knowl. Data Eng.*, vol. 33, no. 5, pp. 2295–2309, May 2021.
- [43] Y. Zhao, K. Zheng, H. Yin, G. Liu, J. Fang, and X. Zhou, "Preference-aware task assignment in spatial crowdsourcing: From individuals to groups," *IEEE Trans. Knowl. Data Eng.*, vol. 34, no. 7, pp. 3461–3477, Jul. 2022.
- [44] B. Zheng et al., "Online trichromatic pickup and delivery scheduling in spatial crowdsourcing," in *Proc. IEEE Int. Conf. Data Eng.*, 2020, pp. 973–984.
- [45] T. Oda, "Equilibrium inverse reinforcement learning for ride-hailing vehicle network," in *Proc. World Wide Web Conf.*, 2021, pp. 2281–2290.
- [46] Y. Xu and P. Xu, "Trade the system efficiency for the income equality of drivers in rideshare," in *Proc. Int. Joint Conf. Artif. Intell.*, 2020, pp. 4199–4205.
- [47] N. Raman, S. Shah, and J. P. Dickerson, "Data-driven methods for balancing fairness and efficiency in ride-pooling," in *Proc. Int. Joint Conf. Artif. Intell.*, 2021, pp. 363–369.
- [48] V. Nanda, P. Xu, K. A. Sankararaman, J. Dickerson, and A. Srinivasan, "Balancing the tradeoff between profit and fairness in rideshare platforms during high-demand hours," in *Proc. AAAI Conf. Artif. Intell.*, 2020, pp. 2210–2217.
- [49] Z. Xu et al., "When recommender systems meet fleet management: Practical study in online driver repositioning system," in *Proc. World Wide Web Conf.*, 2020, pp. 2220–2229.
- [50] G. Wang, S. Zhong, S. Wang, F. Miao, Z. Dong, and D. Zhang, "Data-driven fairness-aware vehicle displacement for large-scale electric taxi fleets," in *Proc. IEEE Int. Conf. Data Eng.*, 2021, pp. 1200–1211.



**Jiahui Sun** received the BS degree in electrical and information engineering from Tianjin University, Tianjin, China, in 2019. He is currently working toward the PhD degree in information and communication engineering with Shanghai Jiao Tong University, Shanghai, China. His current research interests include deep reinforcement learning and sensing.



interests include wireless sensing and reinforcement learning.

**Haiming Jin** (Member, IEEE) is currently a tenure-track associate professor with the Department of Computer Science and Engineering, Shanghai Jiao Tong University (SJTU). From 2021 to 2022, he was a tenure-track associate professor with the John Hopcroft Center (JHC) for Computer Science, SJTU. From 2018 to 2021, he was an assistant professor with JHC, SJTU. From 2017 to 2018, he was a postdoctoral research associate with the Coordinated Science Laboratory (CSL), University of Illinois at Urbana-Champaign (UIUC). His current research in-



recipient of the NSF CAREER Award, the University at Buffalo Young Investigator Award, the ICCPS'17 Best Paper Award, and the ICDCS'17 Best Student Paper Award. He is a member of the ACM.

**Lu Su** (Member, IEEE) received the MS degree in statistics and the PhD degree in computer science from the University of Illinois at Urbana-Champaign, in 2012 and 2013, respectively. He is currently an associate professor with the School of Electrical and Computer Engineering, Purdue University. His research interests include the general areas of mobile and crowd sensing systems, Internet of Things, and cyber-physical systems. He has also worked with the IBM T. J. Watson Research Center and National Center for Supercomputing Applications. He is the



**Zhaoxing Yang** received the MS degree in computer science and technology from Shanghai Jiao Tong University, Shanghai, China, in 2022, where he is currently working toward the PhD degree. His research interests include reinforcement learning algorithms, multi-agent system, and applications.