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

Jiahui Sun
Shanghai Jiao Tong University
Shanghai, China
jhsun1997@sjtu.edu.cn

Haiming Jin*
Shanghai Jiao Tong University
Shanghai, China
jinhaiming@sjtu.edu.cn

Zhaoxing Yang
Shanghai Jiao Tong University
Shanghai, China
yiannis@sjtu.edu.cn

Lu Su
Purdue University
Indiana, USA
lusu@purdue.edu

Xinbing Wang
Shanghai Jiao Tong University
Shanghai, China
xwang8@sjtu.edu.cn

ABSTRACT

The ride-hailing service offered by mobility-on-demand platforms, such as Uber and Didi Chuxing, has greatly facilitated people's traveling and commuting, and become increasingly popular in recent years. *Efficiency* (e.g., gross merchandise volume) has always been an important metric for such platforms. However, only focusing on the efficiency inevitably ignores the *fairness* of driver incomes, which could impair the sustainability of the overall ride-hailing system in the long run. To optimize the aforementioned two essential metrics, *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. Thus, in this paper, we aim to exploit joint order dispatching and driver repositioning to optimize both the long-term efficiency and fairness for ride-hailing platforms. To address this problem, we propose a novel multi-agent reinforcement learning framework, referred to as JDRL, to help drivers make distributed order selection and repositioning decisions. Specifically, to cope with the variable action space, JDRL segments the action space into a fixed number of action groups, and fixes the policy output dimension for order selection as the number of action groups. In terms of the fairness criterion, JDRL adopts the max-min fairness, and augments the vanilla policy gradient to an iterative training algorithm that alternates between a minimization step and a policy improvement step to maximize both the worst and the overall performance of agents. In addition, we provide the theoretical convergence guarantee of our JDRL training algorithm even under non-convex policy networks and stochastic gradient updating. Extensive experiments are conducted 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. Experimental results show that JDRL demonstrates a consistent advantage compared to state-of-the-art baselines in terms of both efficiency and fairness. To the best of our knowledge, this is the first work that exploits joint order dispatching and driver repositioning to optimize both the long-term efficiency and fairness in a ride-hailing system.

CCS CONCEPTS

• **Computing methodologies** → **Multi-agent reinforcement learning**; • **Applied computing** → **Transportation**.

KEYWORDS

Ride-Hailing; Long-Term Efficiency and Fairness; Joint Order Dispatching and Driver Repositioning

ACM Reference Format:
Jiahui Sun, Haiming Jin, Zhaoxing Yang, Lu Su, Xinbing Wang. 2022. Optimizing Long-Term Efficiency and Fairness in Ride-Hailing via Joint Order Dispatching and Driver Repositioning. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '22), August 14–18, 2022, Washington, DC, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3534678.3539060

1 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[1,2], 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 [1–7] have proposed a series of approaches to achieve such 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

*Corresponding author.

KDD '22, August 14–18, 2022, Washington, DC, USA

ACM ISBN 978-1-4503-9385-0/22/08…$15.00
https://doi.org/10.1145/3534678.3539060

[1]https://www.businessofapps.com/
[2]https://www.statista.com/

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 operations have critical effects on long-term efficiency and fairness. 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 via joint order dispatching and driver repositioning.

An intuitive approach for such 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 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 JDRL, 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 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. JDRL resolves such 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, JDRL 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 JDRL 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. Thus, solving agents' policies that jointly optimize the efficiency and fairness falls into the category of max-min optimization, which is intractable for existing MARL frameworks. JDRL addresses this issue by augmenting the vanilla policy gradient to an iterative training algorithm, which alternates between a minimization step that finds the least-advantaged agent, and a policy improvement step that updates agents' policies towards increasing both the worst and the overall performance of agents. Theoretically, we prove that the training algorithm of JDRL converges even under non-convex policy networks and stochastic gradient updating.

In summary, 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.
- Technically, we propose JDRL, a novel distributed MARL framework, which integrates (i) a group-based action representation that copes with the variable action space, and (ii) a training algorithm that alternates between a minimization step and a policy improvement step to maximize both the worst and the overall performance of agents. Furthermore, we provide the theoretical results on the convergence guarantee of our JDRL training algorithm.
- We conduct extensive experiments to evaluate JDRL 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 JDRL demonstrates a consistent advantage compared to state-of-the-art baselines on the aforementioned three datasets in terms of both efficiency and fairness.

In the rest of this paper, we first introduce the preliminaries in Section 2, and then present our formulation in Section 3, as well as our solution method in Section 4. After describing the experimental results in Section 5 and discussing the related works in Section 6, we conclude our paper in Section 7.

2 PRELIMINARIES

In this paper, we consider a ride-hailing system in a city, which consists of a set $\mathcal{N} = \{1, 2, \cdots, N\}$ 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 [2, 3], 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 $\mathcal{T} = \{1, 2, \cdots, T\}$. 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 $\mathcal{G} = \{1, 2, \cdots, G\}$.

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. 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 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 $\sum_{j \in O_t^i} p_j$, where p_j denotes the price of order $j \in O_t^i$.*

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 [8], which aims to maximize the worst DAI of all drivers at the end of the time horizon. We present our formulation that takes both efficiency and fairness into account in the following Section 3.

3 FORMULATION

In a real-world ride-hailing system, the joint order dispatching and driver repositioning (JDR[3]) problem is naturally a sequential decision-making problem. Thus, a typical way to resolve such problem is to consider the platform as a single agent, who makes centralized decisions for all drivers. However, such 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 the JDR problem with efficiency and fairness objectives as a *multi-objective Markov game* (referred to as JDR-MOMG), which contains the following elements.

- **Agent**: We treat each driver in the ride-hailing system as an agent in our JDR-MOMG, 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 $o_t^i = [x_t^i, c_t^i]$, which contains the information in agent i's neighborhood set $\mathcal{G}_t^i$ that includes the 3×3 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 $\mathcal{G}_t^i$; c_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 the 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 $\mathcal{A}_t^i = \mathcal{U}_t^i \bigcup \mathcal{G}_t^i$, 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 π^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 $\boldsymbol{\pi} = [\pi^i]_{i \in \mathcal{N}}$.
- **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.

In our JDR-MOMG, the expected cumulative reward of each agent i is defined as $J^i(\boldsymbol{\pi}) = \mathbb{E}_{P,\boldsymbol{\pi}}\left[\sum_{t=0}^{T} r_t^i\right]$, which depends on agents' joint policy $\boldsymbol{\pi}$, and is in fact the expected DAI of agent i at the end of the time horizon. The JDR-MOMG maximizes two objectives. The first objective is the platform's efficiency, defined as $\sum_{j \in \mathcal{N}} J^j(\boldsymbol{\pi})$. The second objective is the max-min fairness metric in terms of agents' expected DAIs, i.e., $\min_{i \in \mathcal{N}} J^i(\boldsymbol{\pi})$. We formulate the problem of jointly optimizing the long-term efficiency and fairness of JDR-MOMG as

$$\max_{\boldsymbol{\pi}} \left\{ \min_{i \in \mathcal{N}} (1-\lambda) J^i(\boldsymbol{\pi}) + \lambda \sum_{j \in \mathcal{N}} J^j(\boldsymbol{\pi}) \right\}, \tag{1}$$

where $\lambda \in [0, 1]$ 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 JDR-MOMG by proposing a novel MARL framework, which will be elaborated in the following Section 4.

[3]The name JDR comes from the first letters of Joint, Dispatching, and Repositioning.

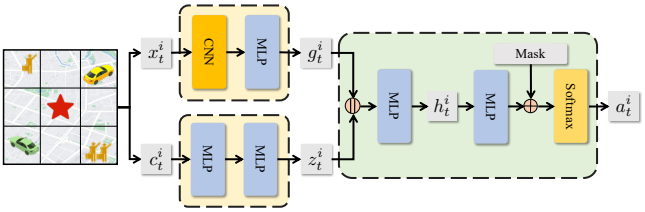


Figure 1: The actor network of JDRL. The star indicates the location of agent i, the orange person represents the ride-hailing order, the yellow car represents the on-service driver, green car represents the idle driver, $\oplus$ denotes element-wise addition, and $\Vert$ denotes the concatenation operation.

4 SOLUTION METHOD

To solve our JDR-MOMG, we propose an MARL framework, referred to as JDRL[4], based on the actor-critic method. In this section, we first describe the design details of the actor and critic networks in Section 4.1, propose the training algorithm in Section 4.2, and then analyze its convergence property in Section 4.3.

4.1 Actor and Critic Network Structures

4.1.1 Actor. In our JDR-MOMG, the action space $\mathcal{A}_t^i$ of each agent i varies over time. A typical actor whose output dimension is fixed and equal to the size of the action space is thus problematic. Existing works [4, 5] 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 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 way of order representation and selection are as follows. On one hand, the destination of an order is a crucial

[4]The name JDRL comes from Joint Order Dispatching and Driver Repositioning with MARL.

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 actor network only needs one forward pass in one execution, which is described as follows.

As shown in Figure 1, at the end of each time slot t, the input of each agent i's actor consists of x_t^i and c_t^i. x_t^i is a tensor and fed into a convolution layer, followed by an MLP layer to output the demand-supply feature g_t^i as

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

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

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

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

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

where $\mathbf{W}_4$ is the weight matrix of the MLP layer, and $||$ 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 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 mask. Lastly, the masked vector is fed into a Softmax layer to generate the action distribution $\pi^i(a_t^i|o_t^i)$ as

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

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

4.1.2 Critic. Each agent owns a critic to estimate its state value function, 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 critic 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, and the last layer of the critic maps the intermediate representation h_t^i to a scalar to represent the state value function as

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

where $\mathbf{W}_6$ is the weight matrix of the MLP layer.

4.2 Training Algorithm

4.2.1 Algorithm Overview. We let

$$J(\boldsymbol{\pi}, i) = \lambda J^i(\boldsymbol{\pi}) + (1-\lambda)\sum_{j\in\mathcal{N}} J^j(\boldsymbol{\pi}), \forall i \in \mathcal{N}.$$

Algorithm 1: JDRL Training Algorithm

```
// Initialization.
1  Initialize parameters θ_1 = [θ_1^i]_{i∈N} of policies, and
     parameters φ_1 = [φ_1^i]_{i∈N} of critics;
   // Iterative training process.
2  foreach epoch k = 1, ··· , K do
       // Experience collection phase.
3      D ← ∅;
4      foreach episode m = 1, ··· , M do
5          foreach time slot t = 1, ··· , T do
6              Each agent i ∈ N_t observes o_t^i, takes an action
                 a_t^i, and receives a reward r_t^i;
7              Observe the next state s_{t+1} and next joint
                 observation o_{t+1};
8              Store experience (s_t, o_t, a_t, r_t, s_{t+1}, o_{t+1}) in D;
       // Parameter updating phase.
9      Ĵ^i(π_k) ← (1/M) Σ_{m=1}^{M} Σ_{t=1}^{T} r_t^i, ∀i ∈ N;
10     ξ_k ← arg min_{i∈N} Ĵ^i(π_k); // Minimization step.
11     Update θ_k to θ_{k+1} via stochastic gradient ascent using
         Equation (2) with ξ_k as input; // Gradient ascent step.
12     Update φ_k to φ_{k+1} by minimizing Equation (3);
```

Then, Equation (1) is equivalent to $\max_{\boldsymbol{\pi}} \min_{i\in\mathcal{N}} J(\boldsymbol{\pi}, i)$. To solve this max-min problem, we design an iterative algorithm that alternates between a gradient ascent step on $\boldsymbol{\pi}$ and a minimization step on i. We provide the overall training algorithm in Algorithm 1.

Firstly, Algorithm 1 initializes the parameter θ_1^i of each agent i's policy π^i, and the parameter ϕ_1^i of each agent i's critic V^i (line 1). Then, the algorithm enters the iterative training process (line 2-12). Each iteration consists of the experience collection phase (line 3-8) and the parameter updating phase (line 9-12). In the experience collection phase, the replay buffer $\mathcal{D}$ that is used to store experiences is first set as empty (line 3). Then, the algorithm runs the current policy $\boldsymbol{\pi}_k$ for M episodes (line 4-8). In each episode, agents interact with the environment, and the experiences of such interactions are collected into $\mathcal{D}$ (line 5-8). In the parameter updating phase, the algorithm first calculates $\hat{J}^i(\boldsymbol{\pi}_k)$ for each agent i based on the collected experiences, which is the average DAI at the end of the time horizon over M episodes (line 9). Then, the algorithm finds the agent ξ_k with the least average DAI at the end of the time horizon (line 10). Based on the collected experiences, Algorithm 1 updates agents' joint policy $\boldsymbol{\pi}_k$ by stochastic gradient ascent (line 11), and updates agents' critics by minimizing the TD error (line 12). In what follows, we present our method of updating the agents' policies and critics in detail.

4.2.2 Updating Policies. When updating policies in each epoch k, we first solve the inner min problem of $\max_{\boldsymbol{\pi}} \min_{i\in\mathcal{N}} J(\boldsymbol{\pi}, i)$ given $\boldsymbol{\pi}_k$ by finding the agent ξ_k with the least average DAI at the end of the time horizon. Then, given ξ_k, we update the parameters of $\boldsymbol{\pi}_k$ from $\boldsymbol{\theta}_k$ to $\boldsymbol{\theta}_{k+1}$ via stochastic gradient ascent. Next, we present the stochastic approximation of the gradient of $J(\boldsymbol{\pi}_k, \xi_k)$ with respect to each agent's policy parameters in Theorem 1.

Theorem 1. *In each epoch k of Algorithm 1, given $\boldsymbol{\pi}_k$ and ξ_k, the gradient of $J(\boldsymbol{\pi}_k, \xi_k)$ w.r.t. $\theta_k^i, \forall i \in \mathcal{N}$, satisfies*

$$\nabla_{\theta_k^i} J(\boldsymbol{\pi}_k, \xi_k) \approx \frac{1}{|\mathcal{D}|} \sum_{b_t \in \mathcal{D}} \nabla_{\theta_k^i} \log \pi^i(a_t^i | o_t^i) \tilde{A}_k(s_t, \mathbf{a}_t), \quad (2)$$

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

$$\tilde{A}_k(s_t, \mathbf{a}_t) = (1-\lambda) A_k^{\xi_k}(s_t, \mathbf{a}_t) + \lambda \sum_{j \in \mathcal{N}} A_k^j(s_t, \mathbf{a}_t),$$

where $A_k^j(s_t, \mathbf{a}_t) = r_t^j + V_k^j(s_{t+1}) - V_k^j(s_t)$ is the advantage function of each agent $j \in \mathcal{N}$.

Please refer to Appendix A for the detailed proof of Theorem 1. Such theorem indicates that, in Algorithm 1, 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.

4.2.3 Updating Critics. In each epoch k of Algorithm 1, we update the parameters of agents' critics from $\boldsymbol{\phi}_k$ to $\boldsymbol{\phi}_{k+1}$ by minimizing the TD error over the collected experiences, given in the following Equation (3),

$$L(\boldsymbol{\phi}_k) = \frac{1}{N|\mathcal{D}|} \sum_{b_t \in \mathcal{D}} \sum_{i \in \mathcal{N}} \left(r_t^i + V_k^i(s_{t+1}) - V_k^i(s_t) \right)^2, \quad (3)$$

where V_k^i is agent i's state value function with parameter ϕ_k^i.

4.3 Convergence Analysis

Solving the optimal policy of our JDR-MOMG falls into the category of max-min optimization. A well-known stationarity concept is the Nash equilibrium that views the max-min problem as a two-player zero-sum game. However, such a concept is not applicable to our problem, since our objective function $J(\boldsymbol{\pi}, i)$ is non-convex, where the minimization and maximization cannot interchange, i.e., $\max_{\boldsymbol{\pi}} \min_{i \in \mathcal{N}} J(\boldsymbol{\pi}, i) \neq \min_{i \in \mathcal{N}} \max_{\boldsymbol{\pi}} J(\boldsymbol{\pi}, i)$. Therefore, in this paper, we adopt the optimization stationarity [9] concept defined as follows. First, we rewrite Equation (1) as $\min_{\boldsymbol{\pi}} \Phi(\boldsymbol{\pi})$, where $\Phi(\boldsymbol{\pi}) = \max_{i \in \mathcal{N}} -J(\boldsymbol{\pi}, i)$. Then, the optimization stationarity defines $\boldsymbol{\pi}$ as a stationary point of $J(\boldsymbol{\pi}, i)$, if it satisfies $||\nabla_{\boldsymbol{\theta}} \Phi(\boldsymbol{\pi})|| = 0$.

As $J(\boldsymbol{\pi}, i)$ is non-convex, and the exact gradient of $J(\boldsymbol{\pi}, i)$ cannot be obtained during training, we introduce Assumptions 1 and 2 for convergence analysis that are commonly adopted in previous works [9, 10].

Assumption 1. $J(\boldsymbol{\pi}, i)$ is $l-$smooth and $L-$Lipschitz with respect to the parameter $\boldsymbol{\theta}$ of $\boldsymbol{\pi}$.

Assumption 2. The variance of the stochastic approximation of $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\pi}, i)$ given by Theorem 1 is upper bounded by σ^2.

Naturally, the order price r_t^i is non-negative and bounded for any agent i in any time slot t, and thus we use R_{max} to denote the largest possible DAI of agents at the end of the time horizon. Now, we present the convergence property of Algorithm 1 in Theorem 2.

Theorem 2. *Given $\delta \in (0,1), \gamma > 0$, and the step size $\eta = \frac{\gamma}{\sqrt{K}}$ for policy update, Algorithm 1 satisfies*

$$\frac{1}{K} \sum_{k=1}^{K} ||\nabla_{\boldsymbol{\theta}_k} \Phi_{1/2l}(\boldsymbol{\pi}_k)||^2 \leq \frac{2(\Delta + l\gamma^2 L^2)}{\gamma \sqrt{K}} + \frac{2l\gamma\sigma^2}{\sqrt{K}} + \frac{4lR_{max}\sqrt{2\ln\frac{2}{\delta}}}{\sqrt{M}},$$

with probability $1-\delta$, where $\Delta = \Phi_{1/2l}(\boldsymbol{\pi}_1) - \min_{\boldsymbol{\pi}} \Phi_{1/2l}(\boldsymbol{\pi})$, and $\Phi_{1/2l}(\boldsymbol{\pi})$ is the Moreau envelope of $\Phi(\boldsymbol{\pi})$. That is, Algorithm 1 converges to an approximate stationary point with probability $1-\delta$.

Please refer to Appendix B for a detailed proof of Theorem 2. Theorem 2 provides a desirable theoretical guarantee that Algorithm 1 converges to an approximate stationary point almost surely, where the gap is composed of three parts. The first part is caused by the distance between the initial point $\Phi_{1/2l}(\boldsymbol{\pi}_1)$ and the stationary point $\min_{\boldsymbol{\pi}} \Phi_{1/2l}(\boldsymbol{\pi})$, 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 solving the inner min problem approximately, which converges to zero as M approaches infinity. According to Theorem 2, increasing the number of epochs and the number of episodes in each epoch help Algorithm 1 converge to a point closer to the stationary point.

5 EXPERIMENTS

5.1 Practical Implementations

We now introduce practical implementations in the following three aspects that help boost the performance of JDRL.

(i) Due to the large scale of the agents, learning a separate set of actor and critic networks for different agents 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. During the execution, each agent keeps a copy of the learned policy network. (ii) As in [11, 12], we utilize the experiences of the worst 10% agents in terms of agents' DAIs at the end of the time horizon to update agents' policies rather than the experiences of only the agent with the worst DAI. This is because the experiences of the least-advantaged agent often have large variance, leading to an unstable policy training process. (iii) 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 [13], where the agent with the lowest DAI will choose first.

5.2 Experimental Setups

5.2.1 Dataset. Our experiments are based on three public real-world ride-hailing order datasets, two of which are released by Didi Chuxing GAIA Initiative[5], and one of which is released by New York City Taxi and Limousine Commission[6]. 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.

[5] https://gaia.didichuxing.com
[6] https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page

5.2.2 Simulator. To support the training and evaluation of our JDRL, we design a ride-hailing simulator, which simulates the generation of orders and state transitions of drivers. We use OpenStreetMap[7] to access the geographical area of each city, and divide the city area into square grids whose lengths are approximately 3 kilometers. After cleaning the invalid grids, e.g., the grids located in the sea or mountains, Haikou is covered by 98 grids, Chengdu is covered by 196 grids, and New York City is covered by 110 grids. The length of each time slot is set as 2 minutes. Similar to [2], we assume orders will get cancelled, if not served for more than 3 time slots since they are submitted.

In our simulation, there are 500 drivers, whose initial locations are randomly sampled from a discrete uniform distribution defined over the grids. During the simulation, orders are generated at the time and locations that are bootstrapped from the datasets. 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.

5.2.3 Baselines. Based on the simulator, we compare JDRL with the following baselines.

- **KM**: KM algorithm is a myopic order dispatching method that finds a maximum weight matching of the driver-order bipartite graph constructed in every time slot. Since we associate each driver-order pair with the order's price, KM maximizes the platform's efficiency in one time slot.
- **REA**: REA [14] is a myopic order dispatching method that 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 matching to satisfy a given fairness threshold.
- **MFAC**: MFAC [4] formulates the order dispatching as a Markov game, where each agent aims to maximize its own long-term order-serving revenue. MFAC proposes a mean-field actor-critic framework to simplify the agent interactions by taking the average action of neighbors into consideration.
- **CVNet**: CVNet [3] adopts a learning and planning approach for order dispatching. In the learning stage, it trains a state value function represented by a neural network via offline 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 uses KM algorithm to find a maximum weight matching to optimize the long-term efficiency.
- **LAF**: LAF [15] 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 the KM algorithm by deleting unfair driver-order pairs to optimize its fairness criterion in a heuristic manner. This is the state-of-the-art method that optimizes both fairness and efficiency in ride-hailing.

[7] https://www.openstreetmap.org/

To fairly compare JDRL with the baselines, we add the random repositioning strategy to KM and REA, and adapt RL-based baseline methods, including MFAC, CVNet, and LAF, to make joint order dispatching and driver repositioning decisions as JDRL.

5.2.4 Metrics. We adopt three metrics to evaluate all the methods, namely GMV, Worst, and Entropy. GMV is defined in Definition 1 to evaluate the platform's efficiency. Worst is a fairness metric that equals to the average of the worst 10% of all drivers' DAIs at the end of the time horizon. Entropy refers to the *temporal earning fairness* criterion defined in [15], which reflects the variance among drivers' DAIs, and a larger Entropy indicates a higher earning unfairness.

To evaluate our JDRL under different spatio-temporal order distributions, we conduct experiments in three time periods, i.e., the morning period from 7 a.m. to 11 a.m., the noon period from 11 a.m. to 3 p.m., and the evening period from 5 p.m. to 9 p.m.. All experiments are conducted on an Ubuntu 18.04 server with four GeForce RTX 3090 GPUs and one AMD EPYC 7302 16-Core CPU. We evaluate each method with 5 different seeds, and report the average and standard deviation of the aforementioned metrics in the following Section 5.3.

5.2.5 Detailed Parameter Settings. The hyper-parameters are shared in all experiments. Specifically, we run $K = 150$ epochs for each dataset, 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, and clip the critic loss and the policy loss with the ratio 0.2 that is commonly used in PPO [16]. The dimensions of g_t^i, z_t^i, and h_t^i are set as 128, 128, and 256, respectively.

5.3 Experimental Results

5.3.1 Metrics Comparison. The results of three metrics on three datasets are presented in Tables 1-3. Overall, JDRL demonstrates a consistent advantage compared to baselines in all three datasets.

First, we analyze the results from the efficiency perspective. In Tables 1 and 3, RL-based methods, including MFAC, CVNet, LAF and JDRL, consistently outperform greedy methods, including KM and REA. However, in Table 2, KM and REA achieve higher GMV than those of MFAC and CVNet, but less than those of LAF and JDRL. This is because, in Haikou and New York City, orders are distributed unevenly over the geographical area. KM and REA only optimize the order dispatching revenue in each time slot, instead of the whole time horizon, which may harm GMV in the long run. The Chengdu dataset has significantly larger number of orders than the other datasets, and the orders are relatively evenly distributed. Thus, the advantage of RL-based methods is not as obvious as in Haikou and New York City.

Among all RL-based methods evaluated in our experiments, JDRL achieves the best performance on GMV in all settings. Specifically, our reasoning of JDRL's superior performance compared to MFAC is that agents in MFAC aim to maximize their own long-term rewards, which incurs unnecessary competition among agents that decreases the platform's efficiency. Compared with JDRL, CVNet and LAF formulate the order dispatching and driver repositioning problem with the single-agent setting, and assume all drivers are independent, which inevitably ignores the interactions among agents.

Table 1: Metric comparison for the Haikou dataset. Each value denotes the mean and standard deviation over 5 runs with different seeds. Bold number is the best in each column.

Method	Morning			Noon			Evening		
	GMV	Worst	Entropy	GMV	Worst	Entropy	GMV	Worst	Entropy
KM	222399 ± 1502	50.2 ± 9.6	712 ± 69	205475 ± 2114	38.4 ± 7.9	793 ± 56	271557 ± 3828	56.8 ± 17.1	726 ± 81
REA	222516 ± 1376	97.6 ± 16.8	526 ± 40	206135 ± 3210	77.4 ± 13.1	606 ± 69	271226 ± 2654	110.9 ± 16.2	560 ± 53
MFAC	222923 ± 1665	81.6 ± 9.5	523 ± 41	223960 ± 1072	71.6 ± 7.0	583 ± 43	299264 ± 915	192.3 ± 12.3	322 ± 30
CVNet	227635 ± 7769	47.8 ± 6.3	561 ± 43	236313 ± 2620	52.3 ± 10.8	594 ± 32	306196 ± 7817	194.1 ± 33.2	306 ± 39
LAF	234175 ± 4297	75.7 ± 16.2	404 ± 50	239771 ± 4228	95.1 ± 14.3	511 ± 53	316801 ± 1307	260.2 ± 8.1	253 ± 17
JDRL ($\lambda = 1$)	**243525 ± 354**	101.8 ± 7.7	396 ± 30	**258023 ± 379**	123.0 ± 9.6	379 ± 18	**325092 ± 698**	272.6 ± 11.3	275 ± 30
JDRL ($\lambda = 0.7$)	241559 ± 303	106.6 ± 4.0	365 ± 14	256801 ± 306	112.9 ± 5.2	428 ± 61	324791 ± 660	287.3 ± 11.2	252 ± 21
JDRL ($\lambda = 0.3$)	242183 ± 453	124.2 ± 2.3	**342 ± 6**	253976 ± 459	134.2 ± 5.6	390 ± 32	319890 ± 564	284.7 ± 7.3	**251 ± 16**
JDRL ($\lambda = 0$)	241542 ± 372	**130.7 ± 5.6**	360 ± 19	251528 ± 851	**155.6 ± 4.8**	**374 ± 23**	321846 ± 678	**316.9 ± 10.7**	254 ± 21

Table 2: Metric comparison for the Chengdu dataset. Each value denotes the mean and standard deviation over 5 runs with different seeds. Bold number is the best in each column.

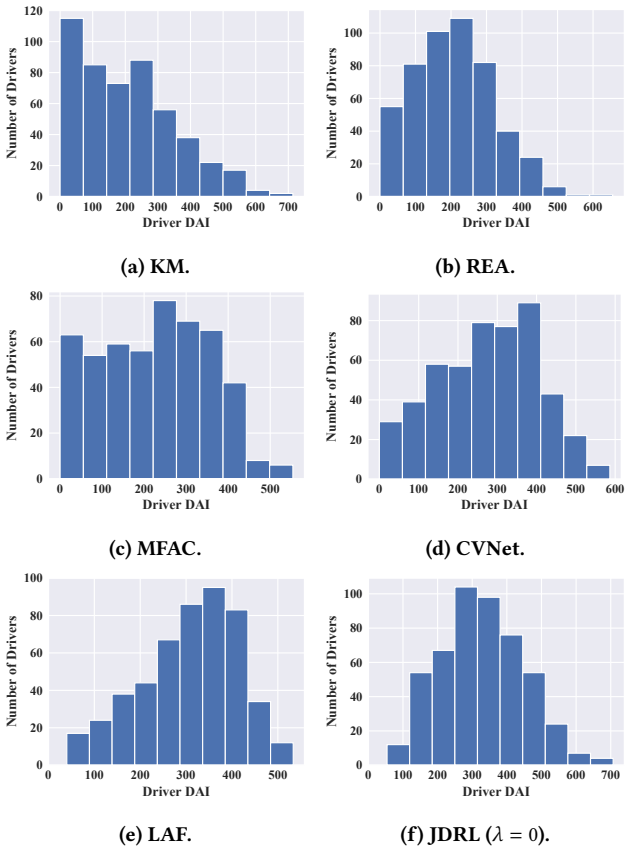
Method	Morning			Noon			Evening		
	GMV	Worst	Entropy	GMV	Worst	Entropy	GMV	Worst	Entropy
KM	62826 ± 431	30.6 ± 4.7	547 ± 73	63571 ± 739	23.4 ± 5.6	587 ± 86	59384 ± 535	12.0 ± 1.9	793 ± 53
REA	61689 ± 729	35.6 ± 5.1	471 ± 69	63827 ± 744	35.6 ± 5.5	458 ± 70	59324 ± 546	16.9 ± 3.5	691 ± 59
MFAC	55149 ± 464	41.6 ± 4.8	444 ± 54	55232 ± 502	38.7 ± 4.9	479 ± 44	53563 ± 724	18.2 ± 5.5	660 ± 75
CVNet	60785 ± 1760	74.2 ± 4.5	233 ± 43	62482 ± 2670	75.8 ± 22.0	299 ± 123	60697 ± 2764	78.2 ± 14.1	260 ± 50
LAF	63554 ± 177	98.3 ± 2.9	**212 ± 44**	65457 ± 1641	93.2 ± 3.6	249 ± 16	61216 ± 882	90.4 ± 1.5	**190 ± 9**
JDRL ($\lambda = 1$)	66915 ± 132	100.5 ± 0.5	237 ± 15	**67766 ± 150**	103.4 ± 1.0	212 ± 21	64983 ± 133	98.6 ± 0.5	238 ± 21
JDRL ($\lambda = 0.7$)	66215 ± 87	102.0 ± 0.6	254 ± 33	67609 ± 156	**106.9 ± 0.8**	**192 ± 25**	65206 ± 115	100.2 ± 0.6	232 ± 32
JDRL ($\lambda = 0.3$)	66208 ± 114	102.7 ± 0.5	264 ± 57	66708 ± 115	100.4 ± 0.9	226 ± 22	**65745 ± 110**	102.1 ± 0.6	213 ± 12
JDRL ($\lambda = 0$)	**67110 ± 138**	**104.9 ± 0.7**	236 ± 12	67634 ± 142	104.2 ± 1.0	199 ± 11	65366 ± 114	**102.3 ± 0.8**	216 ± 21

Table 3: Metric comparison for the New York City dataset. Each value denotes the mean and standard deviation over 5 runs with different seeds. Bold number is the best in each column.

Method	Morning			Noon			Evening		
	GMV	Worst	Entropy	GMV	Worst	Entropy	GMV	Worst	Entropy
KM	104179 ± 1090	5.8 ± 2.2	1205 ± 89	150468 ± 2035	40.1 ± 4.1	623 ± 23	170527 ± 1378	77.9 ± 7.0	460 ± 40
REA	103894 ± 952	33.2 ± 9.2	737 ± 94	147593 ± 1768	67.6 ± 7.7	491 ± 27	170229 ± 1624	108.7 ± 6.5	389 ± 29
MFAC	112455 ± 1271	16.2 ± 5.6	913 ± 79	187561 ± 1084	185.6 ± 7.1	372 ± 31	200677 ± 926	219.4 ± 4.7	374 ± 31
CVNet	134303 ± 5056	30.3 ± 16.8	788 ± 189	197948 ± 5132	200.0 ± 11.9	366 ± 63	218542 ± 3078	277.7 ± 11.5	436 ± 171
LAF	149952 ± 7179	92.4 ± 3.6	540 ± 70	203632 ± 7279	241.2 ± 20.1	385 ± 80	218567 ± 1523	296.1 ± 2.7	402 ± 137
JDRL ($\lambda = 1$)	**165328 ± 248**	103.4 ± 3.1	423 ± 29	219502 ± 237	276.8 ± 5.7	301 ± 41	226866 ± 230	305.9 ± 4.7	291 ± 28
JDRL ($\lambda = 0.7$)	164297 ± 245	106.0 ± 3.3	430 ± 31	220254 ± 396	272.3 ± 4.9	312 ± 29	225898 ± 326	312.3 ± 4.8	**274 ± 31**
JDRL ($\lambda = 0.3$)	163581 ± 370	120.1 ± 6.8	**406 ± 54**	**220760 ± 409**	279.2 ± 4.2	**286 ± 30**	**230438 ± 256**	312.7 ± 3.8	395 ± 30
JDRL ($\lambda = 0$)	163339 ± 448	**130.1 ± 2.9**	417 ± 9	214968 ± 326	**279.6 ± 3.8**	319 ± 31	228105 ± 369	**320.0 ± 3.2**	310 ± 29

JDRL method considers the interaction between agents by incorporating the global state and sharing agents' state value functions during centralized training.

Then, we analyze the results from the fairness perspective. In addition to efficiency, REA, LAF, and JDRL also optimize the fairness among drivers. Compared with KM, REA achieves higher Worst than KM, and is not much different with KM on GMV. However, REA only considers fairness in one time slot, which makes it perform less satisfactorily than LAF and JDRL. Such results validate the importance of optimizing fairness in the long run. Compared with JDRL, LAF optimizes fairness by deleting unfair driver-order pairs, which is a heuristic method and has no convergence guarantee. Among all the baselines, JDRL achieves the highest performance on Worst, which indicates JDRL ensures fairness better than baselines.



(a) KM. (b) REA.

(c) MFAC. (d) CVNet.

(e) LAF. (f) JDRL ($\lambda = 0$).

Figure 2: The distribution of drivers' DAIs at the end of the time horizon in the Morning period of the New York City dataset.

Furthermore, we notice that the standard deviations of RL-based baselines are much larger than that of JDRL. Our reasoning for such observation is that the implementation of our algorithm is based on PPO, which updates the policy more safely than A2C used by MFAC and the value-based methods used by CVNet and LAF.

5.3.2 Driver Income Visualization. We take a deeper look at the distribution of drivers' DAIs at the end of the time horizon under different methods. We use the Morning period of the New York City dataset as an example, and show the distributions of drivers' DAIs in Figure 2. From this figure, we observe that, among all the evaluated methods, KM generates the most unfair DAI distribution. Specifically, Figure 2a shows that, in the DAI distribution generated by KM, although the highest DAI of drivers is over 700, 70% of the drivers have DAIs lower than 300. As shown by Figure 2b, the DAI distribution of REA that also belongs to greedy methods as KM, is obviously better than that of KM, since it considers the max-min fairness in terms of DAI in every time slot. But due to the myopic nature of REA, the number of drivers with DAIs larger than 300 in REA is much smaller than those of the RL-based methods shown in Figures 2c-2f. From Figures 2c-2f, we observe that the worst DAIs of MFAC and CVNet are smaller than those of LAF and JDRL, and, compared to LAF, JDRL has more drivers whose DAIs are concentrated around the median. Furthermore, we could also observe that the worst DAI of JDRL is larger than those of the other methods, and the DAI distribution of JDRL is approximately a normal distribution with a small variance. Based on the above observations on Figure 2, we can conclude that JDRL achieves the best performance on fairness among all the evaluated methods.

6 RELATED WORK

With the rapid development of online ride-hailing systems, solving their unique problems has attracted growing attentions nowadays. For example, [17–20] designed ride-sharing mechanisms to improve the utilization of vehicles among multiple demands. [21–24] proposed route planning and recommendation algorithms for drivers. [25–28] integrated spatial-temporal information and designed deep models to predict the time of arrivals of vehicles or passenger trip orders. Apart from serving passengers, [29–31] also proposed algorithms to enable vehicles for other urban spatial crowdsourcing tasks, such as urban sensing and food delivery. In contrast, our work focuses on making joint order dispatching and driver repositioning decisions for ride-hailing platforms. Table 4 summarizes recent works [1–7, 13–15, 32–38] related to optimizing the aforementioned two decision-making tasks.

Table 4: Recent works related to order dispatching (OD) and driver repositioning (DR).

Decisions	Efficiency	Efficiency & Fairness
OD	[2–5, 32, 33]	[13–15, 34–36]
DR	[1, 37]	[38]
OD & DR	[6, 7]	**JDRL**

Specifically, authors in [2, 3, 33] formulated the order dispatching problem as a constrained optimization problem and solved it with specific greedy algorithms (i.e., KM algorithm). [4, 5, 32] designed MARL algorithms to dispatch orders in a decentralized way, where [4, 32] aim to achieve Nash equilibrium among agents, and [5] focuses on aligning the distributions of drivers and orders. For the driver repositioning task, [1] proposed an MARL framework, and [37] introduced the driver interaction design with a modularized algorithm, to reposition drivers coordinately to appropriate regions with the goal of mitigating the demand-supply gap. Furthermore, [6] 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 objective by extending the method in [3], and learns a unified value function for both order dispatching and driver repositioning tasks. However, all works mentioned above [1–7, 32, 33, 37] focused on improving the efficiency only, and ignored the fairness of driver incomes. Therefore, different from [1–7, 32, 33, 37], we consider to optimize both the efficiency and fairness.

Similar to our work, a line of recent works [13–15, 34–36, 38] also take both the efficiency and fairness as objectives. Our work differs from them in various ways. Specifically, [14, 34–36] aim to guarantee myopic fairness when dispatching orders at 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. Although long-term fairness has been considered by [13, 15, 38], these works designed centralized decision-making frameworks which are less scalable than our distributed framework, and suffer from the "single point of failure" issue. Furthermore, as indicated by Table 4, our work is the first one that exploits joint order dispatching and driver repositioning to optimize both the long-term efficiency and fairness in a ride-hailing system.

7 CONCLUSIONS

In this paper, we deal with the problem of optimizing the platform's efficiency and the fairness among drivers in the long run via joint order dispatching and driver repositioning. To this end, we formulate such a problem as a multi-objective Markov game that explicitly incorporates the overall efficiency and the max-min fairness, and propose a novel MARL framework, referred to as JDRL, to help drivers make distributed order selection and repositioning decisions. Specifically, JDRL contains the following two key designs. First, JDRL segments the dynamic action space that exits in the order dispatching process into a fixed number of action groups, where actions in the same group can be seen as homogeneous. Then, JDRL 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, JDRL proposes an iterative training algorithm to solve agents' policies that jointly optimize the efficiency and fairness. Such an algorithm alternates between a minimization step that finds the least-advantaged agent, and a policy improvement step that updates agents' policies toward increasing both the worst and the overall performance of agents. We provide the theoretical convergence guarantee of our JDRL 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 JDRL consistently outperforms state-of-the-art baselines in terms of three efficiency and fairness metrics.

ACKNOWLEDGMENTS

This work was supported in part by NSF China (No. U21A20519, U20A20181, 61902244, 62061146002, 42050105), and in part by NSF grant CNS-1737590. We would also like to thank DiDi Chuxing for the long-term support and cooperation.

REFERENCES

[1] Kaixiang Lin, Renyu Zhao, Zhe Xu, and Jiayu Zhou. Efficient large-scale fleet management via multi-agent deep reinforcement learning. *KDD*, 2018.
[2] Zhe Xu, Zhixin Li, Qingwen Guan, Dingshui Zhang, Qiang Li, Junxiao Nan, Chunyang Liu, Wei Bian, and Jieping Ye. Large-scale order dispatch in on-demand ride-hailing platforms: A learning and planning approach. *KDD*, 2018.
[3] Xiaocheng Tang, Zhiwei Qin, Fan Zhang, Zhaodong Wang, Zhe Xu, Yintai Ma, Hongtu Zhu, and Jieping Ye. A deep value-network based approach for multi-driver order dispatching. *KDD*, 2019.
[4] Minne Li, Zhiwei Qin, Yan Jiao, Yaodong Yang, Jun Wang, Chenxi Wang, Guobin Wu, and Jieping Ye. Efficient ridesharing order dispatching with mean field multi-agent reinforcement learning. *WWW*, 2019.
[5] Ming Zhou, Jiarui Jin, Weinan Zhang, Zhiwei Qin, Yan Jiao, Chenxi Wang, Guobin Wu, Yong Yu, and Jieping Ye. Multi-agent reinforcement learning for order-dispatching via order-vehicle distribution matching. *CIKM*, 2019.
[6] Jiarui Jin, Ming Zhou, Weinan Zhang, Minne Li, Zilong Guo, Zhiwei Qin, Yan Jiao, Xiaocheng Tang, Chenxi Wang, Jun Wang, et al. Coride: joint order dispatching and fleet management for multi-scale ride-hailing platforms. *CIKM*, 2019.
[7] Xiaocheng Tang, Fan Zhang, Zhiwei Qin, Yansheng Wang, Dingyuan Shi, Yongxin Tong, Hongtu Zhu, and Jieping Ye. Value function is all you need: A unified learning framework for ride hailing platforms. *KDD*, 2021.
[8] John Rawls. *A theory of justice: Revised edition*. Harvard university press, 1999.
[9] Chi Jin, Praneeth Netrapalli, and Michael I. Jordan. What is local optimality in nonconvex-nonconcave minimax optimization? *ICML*, 2020.
[10] Kiran Koshy Thekumparampil, Prateek Jain, Praneeth Netrapalli, and Sewoong Oh. Efficient algorithms for smooth minimax optimization. *NIPS*, 2019.
[11] Aravind Rajeswaran, Sarvjeet Ghotra, Balaraman Ravindran, and Sergey Levine. Epopt: Learning robust neural network policies using model ensembles. *ICLR*, 2017.
[12] Yuankun Jiang, Chenglin Li, Wenrui Dai, Junni Zou, and Hongkai Xiong. Monotonic robust policy optimization with model discrepancy. *ICML*, 2021.
[13] Tom Sühr, Asia J. Biega, Meike Zehlike, Krishna P. Gummadi, and Abhijnan Chakraborty. Two-sided fairness for repeated matchings in two-sided markets: A case study of a ride-hailing platform. *KDD*, 2019.
[14] Nixie S Lesmana, Xuan Zhang, and Xiaohui Bei. Balancing efficiency and fairness in on-demand ridesourcing. *NIPS*, 2019.
[15] Dingyuan Shi, Yongxin Tong, Zimu Zhou, Bingchen Song, Weifeng Lv, and Qiang Yang. Learning to assign: Towards fair task assignment in large-scale ride hailing. *KDD*, 2021.
[16] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv*, 2017.
[17] Libin Zheng, Peng Cheng, and Lei Chen. Auction-based order dispatch and pricing in ridesharing. *ICDE*, 2019.
[18] Yi Xu, Yongxin Tong, Yexuan Shi, Qian Tao, Ke Xu, and Wei Li. An efficient insertion operator in dynamic ridesharing services. *ICDE*, 2019.
[19] Zhidan Liu, Zengyang Gong, Jiangzhou Li, and Kaishun Wu. Mobility-aware dynamic taxi ridesharing. *ICDE*, 2020.
[20] Yifang Liu, Will Skinner, and Chongyuan Xiang. Globally-optimized realtime supply-demand matching in on-demand ridesharing. *WWW*, 2019.
[21] Chak Fai Yuen, Abhishek Pratap Singh, Sagar Goyal, Sayan Ranu, and Amitabha Bagchi. Beyond shortest paths: Route recommendations for ride-sharing. *WWW*, 2019.
[22] Lu Duan, Yang Zhan, Haoyuan Hu, Yu Gong, Jiangwen Wei, Xiaodong Zhang, and Yinghui Xu. Efficiently solving the practical vehicle routing problem: A novel joint learning approach. *KDD*, 2020.
[23] Jiachuan Wang, Peng Cheng, Libin Zheng, Chao Feng, Lei Chen, Xuemin Lin, and Zheng Wang. Demand-aware route planning for shared mobility services. *VLDB*, 2020.
[24] Di Chen, Ye Yuan, Wenjin Du, Yurong Cheng, and Guoren Wang. Online route planning over time-dependent road networks. *ICDE*, 2021.
[25] Xiaomin Fang, Jizhou Huang, Fan Wang, Lingke Zeng, Haijin Liang, and Haifeng Wang. Constgat: Contextual spatial-temporal graph attention network for travel time estimation at baidu maps. *KDD*, 2020.
[26] Huiting Hong, Yucheng Lin, Xiaoqing Yang, Zang Li, Kung Fu, Zheng Wang, Xiaohu Qie, and Jieping Ye. Heteta: Heterogeneous information network embedding for estimating time of arrival. *KDD*, 2020.
[27] Junchen Ye, Leilei Sun, Bowen Du, Yanjie Fu, Xinran Tong, and Hui Xiong. Co-prediction of multiple transportation demands based on deep spatio-temporal neural network. *KDD*, 2019.
[28] Xiaomin Fang, Jizhou Huang, Fan Wang, Lihang Liu, Yibo Sun, and Haifeng Wang. Ssml: Self-supervised meta-learner for en route travel time estimation at baidu maps. *KDD*, 2021.
[29] Hao Wang, Chi Harold Liu, Zipeng Dai, Jian Tang, and Guoren Wang. Energy-efficient 3d vehicular crowdsourcing for disaster response by distributed deep reinforcement learning. *KDD*, 2021.
[30] Chi Harold Liu, Yinuo Zhao, Zipeng Dai, Ye Yuan, Guoren Wang, Dapeng Wu, and Kin K. Leung. Curiosity-driven energy-efficient worker scheduling in vehicular crowdsourcing: A deep reinforcement learning approach. *ICDE*, 2020.
[31] Bolong Zheng, Chenze Huang, Christian S. Jensen, Lu Chen, Nguyen Quoc Viet Hung, Guanfeng Liu, Guohui Li, and Kai Zheng. Online trichromatic pickup and delivery scheduling in spatial crowdsourcing. *ICDE*, 2020.
[32] Takuma Oda. Equilibrium inverse reinforcement learning for ride-hailing vehicle network. *WWW*, 2021.
[33] Peng Cheng, Chao Feng, Lei Chen, and Zheng Wang. A queueing-theoretic framework for vehicle dispatching in dynamic car-hailing. *ICDE*, 2019.
[34] Yifan Xu and Pan Xu. Trade the system efficiency for the income equality of drivers in rideshare. *IJCAI*, 2020.
[35] Naveen Raman, Sanket Shah, and John P. Dickerson. Data-driven methods for balancing fairness and efficiency in ride-pooling. *IJCAI*, 2021.
[36] Vedant Nanda, Pan Xu, Karthik Abhinav Sankararaman, John Dickerson, and Aravind Srinivasan. Balancing the tradeoff between profit and fairness in rideshare platforms during high-demand hours. *AAAI*, 2020.
[37] Zhe Xu, Chang Men, Peng Li, Bicheng Jin, Ge Li, Yue Yang, Chunyang Liu, Ben Wang, and Xiaohu Qie. When recommender systems meet fleet management: Practical study in online driver repositioning system. *WWW*, 2020.
[38] Guang Wang, Shuxin Zhong, Shuai Wang, Fei Miao, Zheng Dong, and Desheng Zhang. Data-driven fairness-aware vehicle displacement for large-scale electric taxi fleets. *ICDE*, 2021.

A PROOF OF THEOREM 1

Given $\boldsymbol{\pi}_k, \xi_k$, we rewrite our objective as

$$J(\boldsymbol{\pi}_k, \xi_k) = (1-\lambda)J^{\xi_k}(\boldsymbol{\pi}_k) + \lambda \sum_{j=1}^{N} J^j(\boldsymbol{\pi}_k).$$

Let τ denote a trajectory $(s_0, \mathbf{a}_0, \cdots, s_T)$, the probability of τ is

$$P(\tau) = \rho(s_0) \prod_{t=0}^{T-1} P(s_{t+1}|s_t, \mathbf{a}_t)\boldsymbol{\pi}_k(\mathbf{a}_t|s_t).$$

Then, the log-probability of τ and its derivative w.r.t. $\theta_k^i, i \in \mathcal{N}$ is

$$\log P(\tau) = \log \rho(s_0) + \sum_{t=0}^{T-1} \big(\log P(s_{t+1}|s_t, \mathbf{a}_t) + \log \boldsymbol{\pi}_k(\mathbf{a}_t|s_t)\big)$$

$$\begin{aligned}
\nabla_{\theta_k^i} \log P(\tau) &= \sum_{t=0}^{T-1} \nabla_{\theta_k^i} \log \boldsymbol{\pi}_k(\mathbf{a}_t|s_t) \\
&= \sum_{t=0}^{T-1} \nabla_{\theta_k^i} \log \prod_{j=1}^{N} \pi_k^j(a_t^j|o_t^j) \\
&= \sum_{t=0}^{T-1} \nabla_{\theta_k^i} \log \pi_k^i(a_t^i|o_t^i).
\end{aligned}$$

We denote $R_k^j(\tau)$ as the cumulative reward of agent j's trajectory τ under policy $\boldsymbol{\pi}_k$. Then, we obtain the analytical policy gradient of each agent j's $J^j(\boldsymbol{\pi}_k)$ with respect to $\theta_k^i, i \in \mathcal{N}$ as

$$\begin{aligned}
J^j(\boldsymbol{\pi}_k) &= \mathbb{E}_\tau[R_k^j(\tau)] = \int_\tau P(\tau) R_k^j(\tau) d\tau \\
\nabla_{\theta_k^i} J^j(\boldsymbol{\pi}_k) &= \int_\tau \nabla_{\theta_k^i} P(\tau) R_k^j(\tau) d\tau \\
&= \int_\tau P(\tau) \nabla_{\theta_k^i} \log P(\tau) R_k^j(\tau) d\tau \\
&= \mathbb{E}_\tau[\nabla_{\theta_k^i} \log P(\tau) R_k^j(\tau)] \\
&= \mathbb{E}_\tau[\sum_{t=0}^{T-1} \nabla_{\theta_k^i} \log \pi_k^i(a_t^i|o_t^i) R_k^j(\tau)].
\end{aligned}$$

To reduce variance, we use the advantage function $A_k^j(s_t, \mathbf{a}_t)$ to replace $R_k^j(\tau)$, and obtain

$$\nabla_{\theta_k^i} J^j(\boldsymbol{\pi}_k) = \mathbb{E}_\tau[\sum_{t=0}^{T-1} \nabla_{\theta_k^i} \log \pi_k^i(a_t^i|o_t^i) A_k^j(s_t, \mathbf{a}_t)].$$

Now, we can derive the gradient of $J(\boldsymbol{\pi}_k, \xi_k)$ w.r.t. $\theta_k^i, i \in \mathcal{N}$ as

$$\begin{aligned}
&\nabla_{\theta_k^i} J(\boldsymbol{\pi}_k, \xi_k) \\
=&(1-\lambda)\nabla_{\theta_k^i} J^{\xi_k}(\boldsymbol{\pi}_k) + \lambda \sum_{i=1}^{N} \nabla_{\theta_k^i} J^j(\boldsymbol{\pi}_k) \\
=&\mathbb{E}_\tau\Big[\sum_{t=0}^{T-1} \nabla_{\theta_k^i} \log \pi_k^i(a_t^i|o_t^i)\big[(1-\lambda)A_k^{\xi_k}(s_t, \mathbf{a}_t) + \lambda \sum_{j=1}^{N} A_k^j(s_t, \mathbf{a}_t)\big]\Big] \\
=&\mathbb{E}_\tau[\sum_{t=0}^{T-1} \nabla_{\theta_k^i} \log \pi_k^i(a_t^i|o_t^i) \widetilde{A}_k(s_t, \mathbf{a}_t)],
\end{aligned}$$

where $\widetilde{A}_k(s_t, \mathbf{a}_t) = (1-\lambda)A_k^{\xi_k}(s_t, \mathbf{a}_t) + \lambda \sum_{j=1}^{N} A_k^j(s_t, \mathbf{a}_t)$. This is an expectation, which means that we can estimate it with a sample mean as

$$\nabla_{\theta_k^i} J(\boldsymbol{\pi}_k, \xi_k) = \frac{1}{|\mathcal{D}|} \sum_{b_t \in \mathcal{D}} \nabla_{\theta_k^i} \log \pi_k^i(a_t^i|o_t^i) \widetilde{A}_k(s_t, \mathbf{a}_t),$$

which ends the proof of Theorem 1.

B PROOF OF THEOREM 2

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

Lemma 1. *Given $\boldsymbol{\pi}_k$ and a fixed $\delta \in (0,1)$, ξ_k solved in each epoch of Algorithm 1 achieves*

$$J(\boldsymbol{\pi}_k, \xi_k) \le \min_{i \in \mathcal{N}} J(\boldsymbol{\pi}_k, i) + \epsilon,$$

with probability $1-\delta$, where ϵ is bounded as

$$\epsilon \le \frac{R_{max}\sqrt{2\ln 2/\delta}}{\sqrt{M}}.$$

Proof. The expected total reward $J^i(\boldsymbol{\pi}_k)$ of agent i is estimated by $\hat{J}^i(\boldsymbol{\pi}_k) = \frac{1}{M}\sum_{m=1}^{M}\sum_{t=1}^{T} r_t^i$. Then, according to Hoeffding's inequality,

$$P\big(|\hat{J}^i(\boldsymbol{\pi}_k) - J^i(\boldsymbol{\pi}_k)| \ge \frac{\epsilon}{2}\big) \le 2\exp\big(\frac{-2L(\epsilon/2)^2}{R_{max}^2}\big).$$

Let the probability be $\delta \in (0,1)$, then we have

$$\epsilon \le \frac{R_{max}\sqrt{2\ln 2/\delta}}{\sqrt{M}}.$$

Thus, $|\hat{J}^i(\boldsymbol{\pi}_k) - J^i(\boldsymbol{\pi}_k)| \le \frac{\epsilon}{2}$ holds with probability $1-\delta$.

We now bound the difference between $J(\boldsymbol{\pi}_k, \xi_k)$ and the real objective $\min_{i\in\mathcal{N}} J(\boldsymbol{\pi}_k, i)$. Suppose $j = \arg\min_{i\in\mathcal{N}} J(\boldsymbol{\pi}, i)$, we have $\hat{J}^{\xi_k}(\boldsymbol{\pi}_k) < \hat{J}^j(\boldsymbol{\pi}_k)$. Now, we calculate the gap between the real minimal $J(\boldsymbol{\pi}_k, j)$ with the estimated minimal value $J(\boldsymbol{\pi}_k, \xi_k)$ as

$$\begin{aligned}
J(\boldsymbol{\pi}_k, \xi_k) - J(\boldsymbol{\pi}_k, j) &= J^{\xi_k}(\boldsymbol{\pi}_k) - J^j(\boldsymbol{\pi}_k) \\
&= J^{\xi_k}(\boldsymbol{\pi}_k) - \hat{J}^j(\boldsymbol{\pi}_k) + \hat{J}^j(\boldsymbol{\pi}_k) - J^j(\boldsymbol{\pi}_k) \\
&\le J^{\xi_k}(\boldsymbol{\pi}_k) - \hat{J}^{\xi_k}(\boldsymbol{\pi}_k) + \hat{J}^j(\boldsymbol{\pi}_k) - J^j(\boldsymbol{\pi}_k) \\
&\le |J^{\xi_k}(\boldsymbol{\pi}_k) - \hat{J}^{\xi_k}(\boldsymbol{\pi}_k)| + |\hat{J}^j(\boldsymbol{\pi}_k) - J^j(\boldsymbol{\pi}_k)| \\
&\le \frac{\epsilon}{2} + \frac{\epsilon}{2} \\
&= \epsilon,
\end{aligned}$$

which thus proves that $J(\boldsymbol{\pi}_k, \xi_k) \le \min_{i\in\mathcal{N}} J(\boldsymbol{\pi}_k, i) + \epsilon$ with probability $1-\delta$, and ends the proof. □

Lemma 1 states that, by estimating $J(\boldsymbol{\pi}_k, i), \forall i \in \mathcal{N}$ with samples, the inner min problem can be solved approximately, and the error ϵ decreases at a rate of $\frac{1}{\sqrt{M}}$. Next, based on Assumption 1, we prove that $\Phi(\boldsymbol{\pi})$ is $l-$weakly convex in Lemma 2.

Lemma 2. *$\Phi(\boldsymbol{\pi})$ is $l-$weakly convex. That is, for any $\boldsymbol{\pi}_1, \boldsymbol{\pi}_2$,*

$$\Phi(\boldsymbol{\pi}_1) \ge \Phi(\boldsymbol{\pi}_2) + < \nabla_{\boldsymbol{\pi}_2}\Phi(\boldsymbol{\pi}_2), \boldsymbol{\pi}_1 - \boldsymbol{\pi}_2 > -\frac{l}{2}||\boldsymbol{\pi}_1 - \boldsymbol{\pi}_2||^2.$$

Proof. For any $\boldsymbol{\pi}_1, \boldsymbol{\pi}_2$, suppose $i_2 = \arg\max_i -J(\boldsymbol{\pi}_2, i)$, then we have

$$\begin{aligned}\Phi(\boldsymbol{\pi}_1) &= \max_i -J(\boldsymbol{\pi}_1, i)\\ &\geq -J(\boldsymbol{\pi}_1, i_2)\\ &\geq -J(\boldsymbol{\pi}_2, i_2) + < -\nabla_{\boldsymbol{\pi}_2} J(\boldsymbol{\pi}_2, i_2), \boldsymbol{\pi}_1 - \boldsymbol{\pi}_2 > -\frac{l}{2}||\boldsymbol{\pi}_1 - \boldsymbol{\pi}_2||^2\\ &\geq \Phi(\boldsymbol{\pi}_2) + < \nabla_{\boldsymbol{\pi}_2}\Phi(\boldsymbol{\pi}_2), \boldsymbol{\pi}_1 - \boldsymbol{\pi}_2 > -\frac{l}{2}||\boldsymbol{\pi}_1 - \boldsymbol{\pi}_2||^2,\end{aligned}$$

which proves that $\Phi(\boldsymbol{\pi})$ is $l-$weakly convex. □

Moreau envelope [9] is widely used to analyze the stationarity for the weakly convex function. The definition of Moreau envelope and its property are presented in Definition 3.

Definition 3 (Moreau Envelope). *We say $\Phi_{1/2l}(\boldsymbol{\pi})$ is Moreau envelope of function $\Phi(\boldsymbol{\pi})$ with parameter $1/2l$ if*

$$\Phi_{1/2l}(\boldsymbol{\pi}) = \min_z \Phi(z) + l||\boldsymbol{\pi} - z||^2.$$

Let $\widetilde{\boldsymbol{\pi}} = \arg\min_z \Phi(z) + l||\boldsymbol{\pi} - z||^2$, then the following statements holds, (1) $\nabla_{\boldsymbol{\pi}}\Phi_{1/2l}(\boldsymbol{\pi}) = 2l(\boldsymbol{\pi} - \widetilde{\boldsymbol{\pi}})$, (2) $\nabla_{\widetilde{\boldsymbol{\pi}}}\Phi_{1/2l}(\boldsymbol{\pi}) = 0$.

Based on Lemma 1 and Definition 3, we can obtain the following two lemmas that are needed for the convergence proof of Theorem 2.

Lemma 3. *Given $\boldsymbol{\pi}, \widetilde{\boldsymbol{\pi}}$, suppose $i = \arg\max_i -J(\boldsymbol{\pi}, i)$, we have*

$$\begin{aligned}\Phi(\widetilde{\boldsymbol{\pi}}) &= \arg\max_{\tilde{i}} -J(\widetilde{\boldsymbol{\pi}}, \tilde{i}) \geq -J(\widetilde{\boldsymbol{\pi}}, i)\\ &\geq -J(\boldsymbol{\pi}, i) + < -\nabla_{\boldsymbol{\pi}} J(\boldsymbol{\pi}, i), \widetilde{\boldsymbol{\pi}} - \boldsymbol{\pi} > -\frac{l}{2}||\widetilde{\boldsymbol{\pi}} - \boldsymbol{\pi}||^2\\ &\geq \Phi(\boldsymbol{\pi}) - \epsilon + < -\nabla_{\boldsymbol{\pi}} J(\boldsymbol{\pi}, i), \widetilde{\boldsymbol{\pi}} - \boldsymbol{\pi} > -\frac{l}{2}||\widetilde{\boldsymbol{\pi}} - \boldsymbol{\pi}||^2.\end{aligned}$$

Lemma 4. *Given $\boldsymbol{\pi}, \widetilde{\boldsymbol{\pi}}$, we have*

$$\begin{aligned}&\Phi(\boldsymbol{\pi}) - \Phi(\widetilde{\boldsymbol{\pi}}) - \frac{l}{2}||\boldsymbol{\pi} - \widetilde{\boldsymbol{\pi}}||^2\\ =&\Phi(\boldsymbol{\pi}) + l||\boldsymbol{\pi} - \boldsymbol{\pi}||^2 - \Phi(\widetilde{\boldsymbol{\pi}}) - ||\boldsymbol{\pi} - \widetilde{\boldsymbol{\pi}}||^2 + \frac{l}{2}||\boldsymbol{\pi} - \widetilde{\boldsymbol{\pi}}||^2\\ \geq&\frac{l}{2}||\boldsymbol{\pi} - \widetilde{\boldsymbol{\pi}}||^2 + \frac{l}{2}||\boldsymbol{\pi} - \widetilde{\boldsymbol{\pi}}||^2\\ =&l||\boldsymbol{\pi} - \widetilde{\boldsymbol{\pi}}||^2\\ =&\frac{1}{4l}||\nabla_{\boldsymbol{\pi}}\Phi_{1/2l}(\boldsymbol{\pi})||^2.\end{aligned}$$

Based on Lemma 1, 2, 3 and 4, we are now in position to prove Theorem 2.

$$\begin{aligned}\mathbb{E}\big[\Phi_{1/2l}(\boldsymbol{\pi}_{k+1})\big] &= \mathbb{E}\big[\min_z \Phi(z) + l||\boldsymbol{\pi}_{k+1} - z||^2\big]\\ &\leq \mathbb{E}\big[\Phi(\widetilde{\boldsymbol{\pi}}_k) + l||\boldsymbol{\pi}_{k+1} - \widetilde{\boldsymbol{\pi}}_k||^2\big]\\ &= \mathbb{E}\big[\Phi(\widetilde{\boldsymbol{\pi}}_k) + l||\boldsymbol{\pi}_k - \eta\nabla_{\boldsymbol{\pi}_k}\hat{J}(\boldsymbol{\pi}_k, i_k) - \widetilde{\boldsymbol{\pi}}_k||^2\big]\\ &= \mathbb{E}\big[\Phi(\widetilde{\boldsymbol{\pi}}_k) + l||\boldsymbol{\pi}_k - \widetilde{\boldsymbol{\pi}}_k||^2 + l\eta^2||\nabla_{\boldsymbol{\pi}_k}\hat{J}(\boldsymbol{\pi}_k, i_k)||^2\\ &\quad + 2l\eta < \nabla_{\boldsymbol{\pi}_k}\hat{J}(\boldsymbol{\pi}_k, i_k), \boldsymbol{\pi}_k - \widetilde{\boldsymbol{\pi}}_k > \big]\\ &= \mathbb{E}\big[\Phi_{1/2l}(\boldsymbol{\pi}_k)\big] + l\eta^2\mathbb{E}\big[||\nabla_{\boldsymbol{\pi}_k}\hat{J}(\boldsymbol{\pi}_k, i_k)||^2\big]\\ &\quad + 2l\eta < \nabla_{\boldsymbol{\pi}_k} J(\boldsymbol{\pi}_k, i_k), \boldsymbol{\pi}_k - \widetilde{\boldsymbol{\pi}}_k >\\ &= \mathbb{E}\big[\Phi_{1/2l}(\boldsymbol{\pi}_k)\big] + 2l\eta < \nabla_{\boldsymbol{\pi}_k} J(\boldsymbol{\pi}_k, i_k), \boldsymbol{\pi}_k - \widetilde{\boldsymbol{\pi}}_k >\\ &\quad + l\eta^2\mathbb{E}\Big[\Big\|\nabla_{\boldsymbol{\pi}_k}\hat{J}(\boldsymbol{\pi}_k, i_k) - E\big[\nabla_{\boldsymbol{\pi}_k}\hat{J}(\boldsymbol{\pi}_k, i_k)\big]\\ &\quad + E\big[\nabla_{\boldsymbol{\pi}_k}\hat{J}(\boldsymbol{\pi}_k, i_k)\big]\Big\|^2\Big]\\ &\leq \mathbb{E}\big[\Phi_{1/2l}(\boldsymbol{\pi}_k)\big] + 2l\eta < \nabla_{\boldsymbol{\pi}_k} J(\boldsymbol{\pi}_k, i_k), \boldsymbol{\pi}_k - \widetilde{\boldsymbol{\pi}}_k >\\ &\quad + l\eta^2(\sigma^2 + L^2)\\ &\leq \mathbb{E}\big[\Phi_{1/2l}(\boldsymbol{\pi}_k)\big] + l\eta^2(\sigma^2 + L^2)\\ &\quad + 2l\eta\big[\Phi(\widetilde{\boldsymbol{\pi}}_k) - \Phi(\boldsymbol{\pi}_k) + \frac{l}{2}||\widetilde{\boldsymbol{\pi}}_k - \boldsymbol{\pi}_k||^2 + \epsilon\big].\end{aligned}$$

Taking a telescopic sum over k, we obtain

$$\begin{aligned}\mathbb{E}\big[\Phi_{1/2l}(\boldsymbol{\pi}_{K+1})\big] \leq\ &\mathbb{E}\big[\Phi_{1/2l}(\boldsymbol{\pi}_1)\big] + Kl\eta^2(\sigma^2 + L^2)\\ &+ 2l\eta\sum_{k=1}^{K}\big[\Phi(\widetilde{\boldsymbol{\pi}}_k) - \Phi(\boldsymbol{\pi}_k) + \frac{l}{2}||\widetilde{\boldsymbol{\pi}}_k - \boldsymbol{\pi}_k||^2 + \epsilon\big].\end{aligned}$$

Rearranging this, we obtain

$$\begin{aligned}&\frac{1}{K}\sum_{k=1}^{K}\big[\Phi(\boldsymbol{\pi}_k) - \Phi(\widetilde{\boldsymbol{\pi}}_k) - \frac{l}{2}||\widetilde{\boldsymbol{\pi}}_k - \boldsymbol{\pi}_k||^2\big]\\ \leq&\frac{\mathbb{E}\big[\Phi_{1/2l}(\boldsymbol{\pi}_1)\big] - \mathbb{E}\big[\Phi_{1/2l}(\boldsymbol{\pi}_{K+1})\big]}{2l\eta K} + \epsilon + \frac{\eta(\sigma^2 + L^2)}{2}\\ \leq&\frac{\Phi_{1/2l}(\boldsymbol{\pi}_1) - \min_{\boldsymbol{\pi}}\Phi_{1/2l}(\boldsymbol{\pi})}{2l\eta K} + \epsilon + \frac{\eta(\sigma^2 + L^2)}{2}.\end{aligned}$$

Plugging Lemma 4 into left side, we obtain

$$\begin{aligned}&\frac{1}{K}\sum_{k=1}^{K}||\nabla_{\boldsymbol{\pi}_k}\Phi_{1/2l}(\boldsymbol{\pi}_k)||^2\\ \leq&2\cdot\frac{\Phi_{1/2l}(\boldsymbol{\pi}_1) - \min_{\boldsymbol{\pi}}\Phi_{1/2l}(\boldsymbol{\pi})}{\eta K} + 4l\epsilon + 2l\eta(\sigma^2 + L^2).\end{aligned}$$

Let $\Delta = \Phi_{1/2l}(\boldsymbol{\pi}_1) - \min_{\boldsymbol{\pi}}\Phi_{1/2l}(\boldsymbol{\pi})$, $\eta = \frac{\gamma}{\sqrt{K}}$, we obtain the desired result

$$\begin{aligned}&\frac{1}{K}\sum_{k=1}^{K}||\nabla_{\boldsymbol{\pi}_k}\Phi_{1/2l}(\boldsymbol{\pi}_k)||^2\\ \leq&\frac{2\Delta + 2l\gamma^2L^2}{\gamma\sqrt{K}} + \frac{2l\gamma\sigma^2}{\sqrt{K}} + \frac{4lR_{max}\sqrt{2\ln 2/\delta}}{\sqrt{M}}.\end{aligned}$$

This ends the proof of Theorem 2.