RESEARCH-ARTICLE

CoopRide: Cooperate All Grids in City-Scale Ride-Hailing Dispatching with Multi-Agent Reinforcement Learning

JINGWEI WANG, Beijing National Research Center for Information Science and Technology, Beijing, China

QIANYUE HAO, Beijing National Research Center for Information Science and Technology, Beijing, China

WENZHEN HUANG, Beijing National Research Center for Information Science and Technology, Beijing, China

XIAOCHEN FAN, Tsinghua University, Beijing, China

QIN ZHANG, Shenzhen University, Shenzhen, Guangdong, China

ZHENTAO TANG, Huawei Technologies Noah's Ark Lab, Hong Kong, Hong Kong

View all

Open Access Support provided by:

Beijing National Research Center for Information Science and Technology

Huawei Technologies Noah's Ark Lab

Shenzhen University

Tianjin University

Tsinghua University

PDF Download
3690624.3709205.pdf
14 January 2026
Total Citations: 1
Total Downloads: 494

Published: 20 July 2025

Citation in BibTeX format

KDD '25: The 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining
August 3 - 7, 2025
Toronto ON, Canada

Conference Sponsors:
SIGMOD
SIGKDD

CoopRide: Cooperate All Grids in City-Scale Ride-Hailing Dispatching with Multi-Agent Reinforcement Learning

Jingwei Wang*
Department of EE, BNRist,
Tsinghua University
Beijing, China

Qianyue Hao*
Department of EE, BNRist,
Tsinghua University
Beijing, China

Wenzhen Huang
Department of EE, BNRist,
Tsinghua University
Beijing, China

Xiaochen Fan
IEIT in Tianjin,
Department of EE, BNRist,
Tsinghua University
Beijing, China

Qin Zhang
College of Computer Science and
Software Engineering,
Shenzhen University
Shenzhen, China

Zhentao Tang
Huawei Noah's Ark Lab
Beijing, China

Bin Wang
Huawei Noah's Ark Lab
Beijing, China

Jianye Hao
Tianjin University
Tianjin, China
Huawei Noah's Ark Lab
Beijing, China

Yong Li
Department of EE, BNRist,
Tsinghua University
Beijing, China
liyong07@tsinghua.edu.cn

Abstract

Ride-hailing services offer convenient travel options in urban transportation. To improve passengers' experience and platforms' revenue, plentiful studies use multi-agent reinforcement learning (MARL) for efficient order dispatching, controlling each grid with one agent to balance the supply-demand (drivers-orders) distribution. However, despite the critical role of cooperation among grids for efficient dispatching strategies, existing works neglect it or limit it within neighboring grids. There exist three key challenges in scaling the cooperation to the whole city: (1) cooperative strategies cause complex interactions among grids, making the grids' states coupled and complicating the information extraction from the states for decision-making; (2) cooperation among grids requires both within- and cross-grid dispatching, where the priorities of these two types of actions are difficult to balance; (3) the value of cooperation is not only heterogeneous over different pairs of grids, but also varies temporally, adding difficulty to dynamically determine the intensities of cooperation for each pair of grids and obtain the global cooperation rewards. In this paper, we propose the CoopRide framework to solve the above challenges. We model the interactions among agents with graphs and utilize graph neural network (GNN) for efficient information extraction. We uniformly encode both within- and cross-grid dispatching, enabling flexible choice of both types of actions in the embedding space. We also design to automatically learn the cooperation intensities among grids, thereby obtaining the cooperative rewards to drive the learning of global cooperation actions. We conduct experiments in three real-world datasets with millions of orders, and extensive results demonstrate the superior performance of CoopRide, outperforming the state-of-the-art baselines by up to 12.4%. Our source codes are available at https://github.com/tsinghua-fib-lab/CoopRide.

CCS Concepts

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

Keywords

Multi-agent reinforcement learning, ride-hailing dispatching, cooperative decision-making

ACM Reference Format:
Jingwei Wang*, Qianyue Hao*, Wenzhen Huang, Xiaochen Fan, Qin Zhang, Zhentao Tang, Bin Wang, Jianye Hao, and Yong Li. 2025. CoopRide: Cooperate All Grids in City-Scale Ride-Hailing Dispatching with Multi-Agent Reinforcement Learning. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.1 (KDD '25), August 3–7, 2025, Toronto, ON, Canada.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3690624.3709205

*Equal contribution.

1 Introduction

Ride-hailing platforms, e.g., Uber, Lyft, and Didi, have revolutionized urban transportation and created enormous commercial value by providing flexible, convenient, and personalized travel options for passengers [34]. The ride-hailing platforms involve a large number of drivers (supply) and orders (demand), which tend to exhibit an unbalanced spatial distribution [27]. Plentiful of works have been conducted to find appropriate and efficient dispatching strategies, i.e., matching the drivers with orders, through rule-based

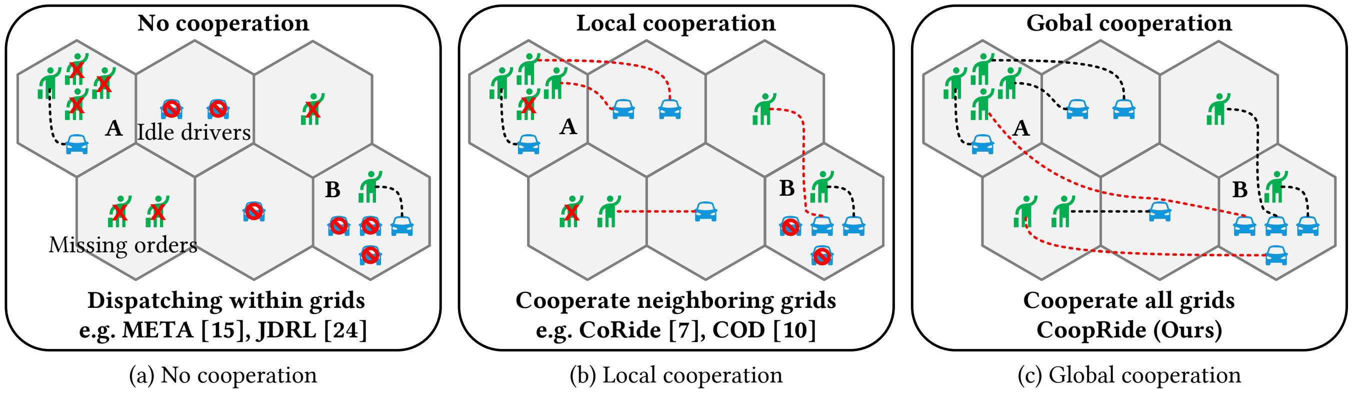


Figure 1: Comparison among ride-hailing dispatching methods with different cooperation strategies among grids. (a) No cooperation. Drivers are only dispatched to orders within the same grids. (b) Local cooperation. Drivers can be dispatched to orders within neighboring grids. (c) Global cooperation (proposed CoopRide). Drivers can be dispatched to orders within the whole city, satisfying passengers' demands.

approaches [9, 12, 23], which are beneficial in balancing the supply and demand, thereby maximizing passengers' satisfaction and increasing platforms' revenue [16]. On the other hand, recent advancements in reinforcement learning (RL) inspire researchers to model ride-hailing dispatching as a sequential decision problem and solve it within the framework of Markov decision process (MDP) [6, 7, 10, 13, 15, 24, 32, 39], enhancing the strategies' performance. In a number of RL works, researchers divide the city into hexagonal grids and control the dispatching in each grid with one agent, following multi-agent reinforcement learning (MARL) framework. However, due to the spatial heterogeneity and temporal variation in supply-demand distribution across the grids and the great complexity of city-scale mobility, finding efficient dispatching strategies for satisfying the on-demand needs of ride-hailing platforms remains a challenging problem.

Intuitively, cooperation among grids is one critical approach to balance the supply-demand distribution. As illustrated in Figure 1, some grids have excessive orders but insufficient drivers, e.g., grid *A*, while others have redundant drivers but lack orders, e.g., grid *B*. Basic RL solutions merely consider dispatching drivers to orders within the same grids, excluding reward signals that motivate cross-grid cooperation [6, 15, 24]. As a result, each RL agent learns to maximize the individual benefit of its own grid. Thus, agents controlling grids with redundant drivers tend to keep the drivers in their own grids despite their idleness rather than dispatching them to help other grids. In contrast, agents controlling grids that lack sufficient drivers cannot serve all orders within their grids in a timely manner (Figure 1a). To solve this dilemma, researchers develop local cooperation strategies [7, 10, 13, 39], which reward the agents to cooperate with their neighboring grids. Still, such a limited extent of cooperation fails to fully balance the supply-demand distribution and maintains the problem of missing orders and idle drivers, especially when clusters of neighboring grids face the same driver or order redundancy (Figure 1b).

To improve the effectiveness of grid cooperation, we propose extending the scale of cooperation from neighboring grids to all grids within the whole city (Figure 1c). However, such global cooperation faces several major challenges:

- **State representation.** When learning cooperative strategies, each agent needs to be informed of the states of its correlated agents. Also, the actions of each agent affect the states of its correlated agents, leading to complex coupling among agent states. Compared to local cooperation methods, our global cooperation design correlates each agents with all the rest $N-1$ agents, resulting in higher dimension and greater coupling complexity of agents' states, adding difficulty to effectively extract valuable information from the states to support the agents' decision-making.
- **Action generation.** The cooperative strategy requires agents to simultaneously dispatch drivers to serve orders within their grids and move across grid to help other grids. Unlike typical local-cooperation agents that prior serving orders within their grids and then consider cross-grid movement [10, 13, 39], the global cooperation design requires more flexible cross-grid dispatching. Therefore, uniformly generating these two kinds of actions and balancing their priority is critical yet challenging.
- **Reward calculation.** Among all grids across the whole city, the potential value of cooperation is not only heterogeneous among different grids but also varies over time. Thus, it is difficult to dynamically determine the intensities of cooperation and use them to weigh the individual benefits of all grids to calculate the global cooperation rewards, which motivate the agents to learn global cooperation strategies.

Facing these challenges, we propose the CoopRide framework, which includes joint designs covering state representation, action generation, and reward calculation, empowering global cooperation among grids. To model the relationships among grids, we design a graph model of the grids, where node features include the state of each grid and edge weights reflect the strength of inflow and outflow among the grids. We employ a graph neural network (GNN) as the Cooperative State Representation module, which extracts information from the graph model and passes it into RL agents. Inside the agents, we propose a Cooperative Action

Generation mechanism, which projects the tasks of serving orders within their own grids and moving across grids to help others into a uniform embedding space. Then, we calculate the priorities of each task via inner product within the embedding space and dispatch drivers to tasks sampled according to the priorities. Furthermore, we design a Cooperative Reward Learning module to automatically capture the varying intensities of cooperation through the technique of meta-gradient RL [2, 33, 38], dynamically calculating the global cooperation rewards to drive the policy-learning. We evaluate CoopRide using three real-world ride-hailing datasets with millions of orders through extensive experiments. Our results demonstrate CoopRide's superior performance, outperforming the state-of-the-art baselines by up to 12.4%. Our in-depth analyses reveal the validity and importance of our global cooperation design.

In summary, our major contributions in this work include:

- We recognize the critical role of cooperation among grids to balance supply-demand distribution in the ride-hailing system and thus propose CoopRide, a ride-hailing dispatching framework that enables global cooperation among all grids in a city.
- We design the components of state representation, action generation, and reward calculation in the framework of RL, empowering the efficient learning process of dispatching strategies with global cooperation.
- We evaluate our method through extensive experiments on real-world datasets. The results show that our method surpasses the state-of-the-art baselines, and our in-depth analyses prove the validity and importance of our global cooperation design.

2 Preliminaries

2.1 Problem Formulation

In this paper, we focus on city-scale ride-hailing dispatching problem. Adhering to industry conventions [7, 29, 30, 32], we employ a discretized time horizon and geographical area setting. First, the day is divided into T time slots, denoted as $\mathcal{T} = \{1, 2, ..., T\}$, where orders can be submitted to the ride-hailing platform at any time, but the platform only dispatches orders to drivers at the end of each time slot. Second, the city is partitioned into N hexagonal grids, labeled as $\mathcal{G} = \{1, 2, ..., N\}$, and the locations of drivers and orders are aligned to the centers of corresponding grids. If matched with an order, the driver will transport the passengers to the destination within specified time slots, determined by the distance, and accrue income based on the order price. Here, we provide a mathematical definition of the ride-hailing dispatching problem.

Definition 1 (Ride-Hailing Dispatching). *Given orders with price and destination features, denoted as $O_{i,t} = \{o_{i,t}^1, ..., o_{i,t}^K\}$, and drivers $\mathcal{D}_{i,t}$ in grid $i \in \mathcal{G}$ at time slot $t \in \mathcal{T}$, the ride-hailing strategy aims to match the drivers in $\mathcal{D}_{i,t}$ with orders in $O_{i,t}$ to maximize the revenue derived from order servicing over all grids and all time slots.*

2.2 Multi-Agent Markov Decision Process

The ride-hailing dispatching problem can be regarded as a sequential decision and solved following the framework of multi-agent Markov decision process (MAMDP) [14]. A MAMDP is defined by the tuple $(N, \mathcal{S}, \mathcal{A}, P, R, \gamma)$, where N denotes the total number of agents, and in this work, we control the dispatching in each of the N hexagonal grids with one agent. At time step t, the global state $s_t = (s_{1,t}, \cdots, s_{N,t}) \in \mathcal{S} = \mathcal{S}^1 \times \cdots \times \mathcal{S}^N$ includes local state of each agent. The joint action $a_t = (a_{1,t}, \cdots, a_{N,t}) \in \mathcal{A} = \mathcal{A}^1 \times \cdots \times \mathcal{A}^N$ consists of the local action of each agent and is obtained from $\pi_\theta : \mathcal{S} \mapsto \mathcal{A}$, the global policy, which is parameterized with $\theta = (\theta_1, \cdots, \theta_N)$ and each agent i owns a local policy π_{θ_i} for local action $a_{i,t}$. Given a state and a joint action, the state transition probability $P : \mathcal{S} \times \mathcal{A} \mapsto \mathcal{S}'$ outputs a distribution over successor states. The global one-step reward is the sum of local one-step rewards from each agent, i.e., $R_t(s_t, a_t) = \sum_{i=1}^N r_i(s_{i,t}, a_{i,t})$. The objective is to find a global policy π_θ that maximizes the discounted return G: $G = \sum_{t=1}^T \gamma^t R_t(s_t, a_t)$, where γ is the discount factor.

3 Methods

3.1 System Overview

We illustrate the architecture of CoopRide in Figure 2. We treat each grid as an agent, and the basic elements of our MARL-based global-cooperative ride-hailing dispatching are as follows:

State. At time t, $s_{i,t} = [O_{i,t}, g_{i,t}]$ denotes the state of agent i, consisting of orders' features $O_{i,t}$ and statistical features of the grid $g_{i,t}$, where the latter include the number of drivers and orders in grid i, as well as statistics of orders.

Action. At time t, agent i selects $D = min(\|\mathcal{D}_{i,t}\|, \|O_{i,t}\|)$ orders from $O_{i,t}$ as the action $a_{i,t} = \{o_{i,t}^{k_1}, ..., o_{i,t}^{k_D}\}$. The selected orders are dispatched to drivers currently in grid i. Following common assumption [13, 15, 24, 30], drivers in the same grid are regarded as identical and are randomly matched with selected orders.

Reward. At time t, agent i gets immediate reward $r_{i,t}$, which is the total price of orders successfully dispatched. $r_{i,t} = \sum_{d=1}^D \times p_{i,t}^{k_d}$, where $p_{i,t}^{k_d}$ denotes the price of order $o_{i,t}^{k_d}$.

Transition. In the next time slot, new orders arrive, and the drivers dispatched with orders in previous time slots move to destination grids specified by the orders within a specified number of time slots, determined by the distance.

To cooperate all grids within the city and maximize the performance of the learned dispatching strategies, we jointly design a Cooperative State Representation, a Cooperative Action Generation, and a Cooperative Reward Learning module. First, the Cooperative State Representation module models the complicated coupling relationships among the cooperative grids with graphs and utilizes GNN to encode the states of the grids $g_{i,t}$ into representations $h_{i,t}$, which contain refined information from the states to support the decision-making (Section 3.2). Second, the Cooperative Action Generation module takes in orders' features $O_{i,t}$ and $h_{i,t}$, uniformly encoding both within- and cross-grid dispatching requirements, and thereby flexibly generate cooperative actions among grids(Section 3.3). Third, the Cooperative Reward Learning module automatically learns and calculates the cooperation intensities among grids based on $h_{i,t}$ via the meta-gradient technique. We use these intensities to weigh the local rewards of each grid, thereby dynamically obtaining the cooperative rewards, which drive the learning of global cooperation strategies (Section 3.4).

3.2 Cooperative State Representation

To effectively extract information from cooperative relationships among grids, we design a graph-based representation method.

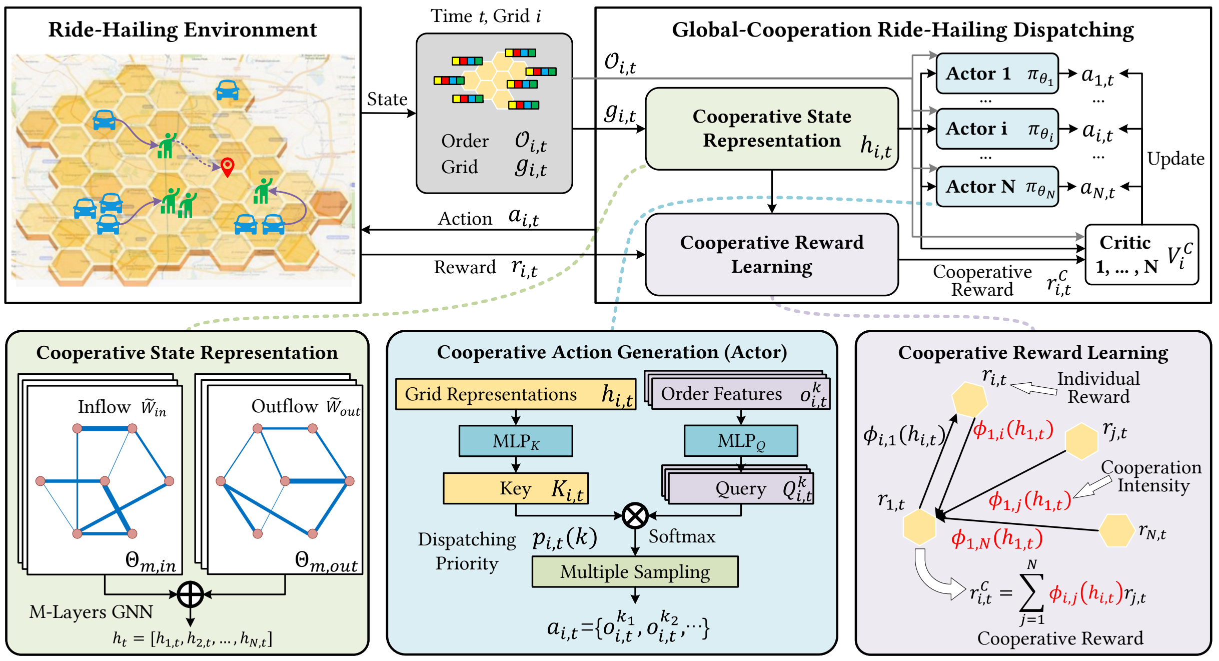

Ride-Hailing Environment
Time t, Grid i
State
Order $O_{i,t}$
Grid $g_{i,t}$
Global-Cooperation Ride-Hailing Dispatching
$O_{i,t}$
$g_{i,t}$
Cooperative State Representation $h_{i,t}$
Actor 1 π_{θ_1} $a_{1,t}$
Actor i π_{θ_i} $a_{i,t}$
Actor N π_{θ_N} $a_{N,t}$
Update
Action $a_{i,t}$
Reward $r_{i,t}$
Cooperative Reward Learning
Cooperative Reward $r^C_{i,t}$
Critic 1, ..., N V^C_i
Cooperative State Representation
Inflow $\tilde{W}_{in}$
Outflow $\tilde{W}_{out}$
$\Theta_{m,in}$
$\Theta_{m,out}$
M-Layers GNN
$h_t = [h_{1,t}, h_{2,t}, ..., h_{N,t}]$
Cooperative Action Generation (Actor)
Grid Representations $h_{i,t}$
Order Features $o^k_{i,t}$
MLP$_K$
MLP$_Q$
Key $K_{i,t}$
Query $Q^k_{i,t}$
Dispatching Priority $p_{i,t}(k)$
Softmax
Multiple Sampling
$a_{i,t}=\{o^{k_1}_{i,t}, o^{k_2}_{i,t}, \cdots\}$
Cooperative Reward Learning
$r_{i,t}$ Individual Reward
$\phi_{i,1}(h_{i,t})$
$\phi_{1,i}(h_{1,t})$
$r_{j,t}$
Cooperation Intensity
$r_{1,t}$
$\phi_{1,j}(h_{1,t})$
$\phi_{1,N}(h_{1,t})$
$r_{N,t}$
$r^C_{i,t} = \sum_{j=1}^{N} \phi_{i,j}(h_{i,t}) r_{j,t}$
Cooperative Reward

Figure 2: Overview of the CoopRide framework. The actors take in state representations incorporating grids cooperation information and generate globally cooperative actions. The actions are updated under the guidance of cooperative rewards containing dynamically learned cooperation intensity.

Recognizing that interactions among grids arise from cross-grid movements, we construct grid interaction graphs, including inflow and outflow graphs, based on the origin-destination (OD) connections among grids. In both inflow and outflow graphs, each grid is treated as a node, and the representations of grid features, $h_{i,t}$, are set to be node features. We utilize OD of historical orders to construct weighted adjacency matrix for the graphs, denoted as $\mathbf{W}_{in}, \mathbf{W}_{out} \in \mathbb{R}^{N\times N}$, respectively. We apply standard normalization in graph convolutional network (GCN) [8] to the adjacency matrix and obtain $\tilde{\mathbf{W}}_{in}$ and $\tilde{\mathbf{W}}_{out}$.

Inspired by [11], we employ a bidirectional diffusion graph convolutional network to incorporate directed views of both inflow to the grids and outflow from the grids. We stack M layers in the network and denote the parameters of layer M as $\Theta_{m,in}$ and $\Theta_{m,out}$. Finally, the association among grids can be extracted as follows:

$$\mathbf{h}_t = \text{ReLU}\left(\sum_{m=0}^{M-1}\left(\left(\tilde{\mathbf{W}}_{in}\right)^m \mathbf{h}_t\Theta_{m,in} + \left(\tilde{\mathbf{W}}_{out}\right)^m \mathbf{h}_t\Theta_{m,out}\right)\right), \quad (1)$$

where $\mathbf{h}_t = \{h_{1,t}, ...h_{N,t}\}$ is the joint node feature.

3.3 Cooperative Action Generation

In the cooperation context, there exist two circumstances of cross-grid movement. First, drivers complete the dispatched orders that require transporting passengers to different grids and stay in the destination grids after arrival. Second, to balance the supply-demand distribution, the platform requests drivers to move to different grids without passengers. We treat the latter as virtual orders with zero prices and destinations representing relocation targets. If matched with virtual orders, the drivers and the platform will not earn from the relocation themselves but may earn more in the future by serving more orders in the new grids.

In order to flexibly balance the priorities of both real and virtual orders, we design to uniformly encode them and determine the priorities in the embedding space via attention mechanism. On the one hand, we project $h_{i,t}$ of each grid into d-dimensional vectors with an MLP. On the other hand, we project the features of all orders in each grid into d-dimensional vectors with another MLP. The vectors respectively serve as keys and queries, denoted as:

$$K_{i,t} = MLP_K(h_{i,t}) \quad Q^k_{i,t} = MLP_Q(o^k_{i,t}). \quad (2)$$

Then, the priority of order k is computed via attention:

$$p_{i,t}(k) = \tanh\left(\frac{\left(K_{i,t}\right)^T Q^k_{i,t}}{\sqrt{d}}\right). \quad (3)$$

Considering that the grids' actions involve choosing multiple orders, we design a multiple sampling approach to select orders based on $p_{i,t}(k)$. Moreover, a real order can be selected only once, while a virtual order can be selected repeatedly, i.e., relocate numerous drivers to the same grid. Therefore, we design a mixed approach that combines both with and without replacement sampling, where the sampling probabilities of orders are computed as:

$$\tilde{p}_{i,t}(k) = \frac{\text{Mask}(k)\exp(p_{i,t}(k))}{\sum_{k=1}^{\|O_{i,t}\|}\text{Mask}(k)\exp(p_{i,t}(k))}. \quad (4)$$

Mask(k) are all initialized to 1, and when $o^k_{i,t}$ is sampled, Mask(k) is set to be 0 if $o^k_{i,t}$ is a real order, otherwise it remains unchanged.

3.4 Cooperative Reward Learning

In order to dynamically determine the cooperative reward and drive the learning process of global cooperation strategies, we consider the cooperation intensities among agents. The cooperative reward for agent i is obtained by weighing the local reward of each agent with the intensities and summing them up as follows:

$$r_{i,t}^{C} = \sum_{j=1}^{N} \left[\Phi_{i,j}(h_{i,t}) r_{j,t} \right], \tag{5}$$

where Φ are MLPs with learnable parameters. Then we train agent i with $r_{i,t}^{C}$, which dynamically changes during training. The optimization objective for agent i is:

$$J_i^C(\theta_i, \Phi) = \mathbb{E}\left[\sum_{t=1}^{T} r_{i,t}^{C} \right], \tag{6}$$

where π_{θ_i} denotes the policy network of agent i. At time step t, we compute the advantage function as:

$$A_{i,t}^{C} = r_{i,t}^{C} + \gamma V_i^C(s_{i,t+1}) - V_i^C(s_{i,t}), \tag{7}$$

where V_i^C is the value network and we denote $s_{i,t} = [O_{i,t}, h_{i,t}]$. Finally, we employ policy gradient [26]:

$$\nabla_{\theta_i} J_i^C(\theta_i, \Phi) = \mathbb{E}\left[\left[\nabla_{\theta_i} \log \pi_{\theta_i}(a_{i,t}|s_{i,t}) \right] A_{i,t}^{C} \right], \tag{8}$$

and update the parameters with learning rate α:

$$\theta_i' = \theta_i + \alpha \nabla_{\theta_i} J_i^C(\theta_i, \Phi). \tag{9}$$

We use standard PPO manner [22] to approximate the gradient and optimize π_{θ_i}. Mentioning that we share the parameters of π_{θ_i} and V_i^C among grid agents to reduce computational consumption, where we denote them as π_θ and V^C for simplicity.

To automatically learn the cooperation intensities, we optimize Φ to maximize the global return via meta-gradient method [2, 19, 33, 38]. The global return is the summation of all agents' rewards:

$$r_t^G = \sum_{i=1}^{N} r_{i,t} \tag{10}$$

The optimization objective for the cooperation intensities is:

$$J^G(\theta_1, \theta_2, ..., \theta_N; \Phi) = \mathbb{E}\left[\sum_{t=1}^{T} r_t^G \right]. \tag{11}$$

Noticing that J^G does not explicitly contain the parameters of Φ, we take θ_i as a bridge, calculating the gradient of J^G with respect to Φ with the chain rule:

$$\nabla_\Phi J^G = \sum_{i=1}^{N} \left[\nabla_{\theta_i} J^G \nabla_\Phi \theta_i \right]. \tag{12}$$

For the first term, we simply apply the policy gradient formula:

$$\nabla_{\theta_i} J^G = \mathop{\mathbb{E}}_{(\mathbf{s}_t, \mathbf{a}_t) \sim \theta_i} \left[\left[\nabla_{\theta_i} \log \pi_{\theta_i}(a_{i,t}|s_{i,t}) \right] A_t^G \right], \tag{13}$$

where $\mathbf{s}_t = \{s_{1,t}, ...s_{N,t}\}$, $\mathbf{a}_t = \{a_{1,t}, ...a_{N,t}\}$ and $\mathbf{r}_t = \{r_{1,t}, ...r_{N,t}\}$. We compute the global advantage as:

$$A_t^G = r_t^G + \gamma V^G(\mathbf{s}_{t+1}) - V^G(\mathbf{s}_t), \tag{14}$$

where V^G is the global value network. We still use standard PPO to approximate this gradient. For the second term in the chain rule, $\nabla_\Phi \theta_i$, since θ_i again does not explicitly contain the parameters of Φ, we utilize the updating of θ_i in (9) as a bridge, following:

$$\begin{aligned} \nabla_\Phi \theta_i &= \nabla_\Phi \left[\theta_i^{old} + \alpha \nabla_{\theta_i^{old}} J_i^C \left(\theta_i^{old}, \Phi \right) \right] \\ &= \alpha \nabla_\Phi \left[\nabla_{\theta_i^{old}} J_i^C \left(\theta_i^{old}, \Phi \right) \right] \\ &= \alpha \mathop{\mathbb{E}}_{(\mathbf{s}_t, \mathbf{a}_t) \sim \theta_i} \left[\left[\nabla_{\theta_i^{old}} \log \pi_{\theta_i^{old}}(a_{i,t}|s_{i,t}) \right] A_{i,t}^{C} \right]. \end{aligned} \tag{15}$$

Thereby, we can combine the two terms to update Φ as:

$$\Phi' = \Phi + \alpha \nabla_\Phi J^G(\theta_1, \theta_2, ..., \theta_N; \Phi). \tag{16}$$

3.5 Training Algorithm

We summarize the training process of CoopRide in Algorithm 1, which is detailed in Appendix A.1. In each episode, the algorithm starts by generating a trajectory and gathering the corresponding transitions. Next, we update the policy and value networks of the grid agents, i.e., π_{θ_i} and V_i^C. Finally, we update Φ and V^G to learn the changing cooperation intensities. The process iterates until the cooperation strategies and cooperation intensities both reach convergence. It is worth mentioning that we synchronously update parameters of the GNN in Section 3.2, treating it as part of any cascade network that takes $\mathbf{h}_t$ as input. All the hyper-parameters used in our implementation are summarized in Appendix A.2 for reproducibility.

4 Experiments

4.1 Experimental Settings

To thoroughly verify the effectiveness of CoopRide, we conduct extensive experiments in simulation environment for ride-hailing dispatching, which is built based on various large-scale real-world datasets, and we evaluate the performance with widely used metrics in ride-hailing dispatching task.

4.1.1 Dataset. We collect real-world datasets of three cities from two ride-hailing platforms, Didi Chuxing[1] and NYC Taxi and Limousine Commission (TLC)[2] for experiments, which all contain features of massive orders, including order start time, price, origin, destination, duration, and route distance. These datasets cover different cities and ride-hailing platforms around the world, reflecting various patterns of global ride-hailing services, and thus are representative enough to ensure the reliability and comprehensiveness of our experimental results.

Table 1: Statistics of the datasets.

City	Haikou	Chengdu	New York
# Grids	100	121	143
# Drivers	1,000	1,500	2,000
# Orders	1,905,810	2,781,660	2,675,520
Time	Sept. 2017	Nov. 2019	Sept. 2019
Source	Didi Chuxing	Didi Chuxing	NYC TCL

Following previous works [7, 32], we select the central area of each city and divide them into hexagonal grids with a radius of

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

Table 2: Performance comparison. Each value denotes the mean and standard deviation over 5 repeated runs with different seeds. In each column, the best results are highlighted in bold and the second best results are underlined.

City	Haikou		Chengdu		New York	
Metric	Norm. GMV	ORR	Norm. GMV	ORR	Norm. GMV	ORR
KM	113.67 ± 0.14	57.12% ± 0.12%	103.69 ± 0.03	60.23% ± 0.01%	103.29 ± 0.11	63.58% ± 0.11%
V1D3	119.74 ± 0.12	60.52% ± 0.09%	106.84 ± 0.11	61.57% ± 0.07%	104.84 ± 0.06	64.35% ± 0.08%
RLW	120.62 ± 0.14	61.49% ± 0.10%	107.21 ± 0.08	62.05% ± 0.06%	105.10 ± 0.04	63.55% ± 0.07%
COD	116.80 ± 0.14	59.78% ± 0.08%	105.54 ± 0.12	61.23% ± 0.07%	103.98 ± 0.08	64.25% ± 0.06%
CoRide	118.87 ± 0.09	61.48% ± 0.06%	107.42 ± 0.09	61.89% ± 0.06%	104.70 ± 0.07	63.89% ± 0.07%
CoopRide	**124.71 ± 0.03**	**67.92% ± 0.02%**	**108.01 ± 0.08**	**69.75% ± 0.05%**	**110.97 ± 0.07**	**71.97% ± 0.07%**

approximately 1.4 kilometers. We show detailed statistics about the datasets in Table 1. Notice that in the real world, one vehicle typically serves 24h, where numbers of drivers working in turn, but we regard them as a single 'driver' in the statistics. Thereby, we can calculate that the serving time is approximately half an hour for each order, which is reasonable based on real-world experience.

4.1.2 Simulation Environment. We develop a ride-hailing simulator following previous works [7, 13], supporting the training and testing of dispatching strategies. The simulator operates in 10-minute time slots, during which orders are generated by bootstrapping from the aforementioned datasets. If an order is successfully matched with a driver, the driver will transport the passenger to the destination within a few time steps based on the distance. However, if an order fails to be successfully matched with a driver, it will either continue waiting to the next time slot for a potential match or be canceled by probabilities derived from real-world statistics.

4.1.3 Metrics. We evaluate the performance of dispatching strategies with two commonly used criterias [7, 10, 15, 24, 39]:

- **Gross Merchandise Volume (GMV)**: GMV characterize the total daily income of all drivers. We present the normalized GMV, which is normalized by the Random dispatching.
- **Order Response Rate (ORR)**: ORR is the percentage of passengers successfully matched with drivers.

4.2 Baselines

We compare various baseline methods to evaluate the performance of our method. First, we include rule-based methods:

- **KM** [32]: This baseline defines the orders' prices as the weights of the order-driver bipartite graph and uses KM algorithm [18] to maximize the platform's revenue.

Also, we compare methods that utilize RL to enhance graph-matching-based order dispatching strategies:

- **V1D3** [30]: This baseline method combines bipartite graph matching and probability sampling to learn dispatching strategies with a value function.
- **RLW** [3]: This is the SOTA method in this category. On the basis of V1D3, RLW adds reward smoothing, edge standardization, and feedback loop based on a UCB algorithm.

Furthermore, we also compare MARL methods for order dispatching, which consider cooperation among neighboring grids:

- **COD** [10]: This method uses mean-field approximation to simplify the local interactions by taking an average over action among neighboring grids.
- **CoRide** [7]: This SOTA MARL baseline treats each grid as one worker agent and cooperates workers by setting local managers to control workers. It uses the attention mechanism to represent the interaction among workers and managers.

4.3 Overall Performance

We train the global cooperation dispatching strategies with CoopRide, where the training process takes approximately one day on a single NVIDIA RTX 3090 GPU, and we show the changing and converging of normalized GMV during the training process, in Appendix A.3. We show the performance comparison with baselines Table 2. The results illustrate that our method consistently outperforms all baselines, achieving a maximum 5.5% increase regarding GMV in the New York dataset and a maximum 12.4% increase regarding ORR in the Chengdu dataset.

We can observe that all categories of baselines fail to dispatch orders efficiently enough. On the one hand, some baselines perform well regarding OOR but perform bad regarding GMV. This indicates that they fail to serve long-range orders, which are relatively rare but expensive, and merely focus on short-range orders that are common but cheap. Therefore, they serve more orders, improving the OOR, but miss orders with high income, leading to a bad GMV. On the other hand, other baselines can improve the GMV but perform mediocrely on OOR. It is intuitive that they focus too much on the expensive long-range orders to earn more, but miss a large number of cheap but common short-range orders. In contrast, our method reaches supreme performance regarding both GMV and ORR, suggesting the validity of our global cooperation design, which enables the model to balance all types of orders.

4.4 Ablation Study

To provide a comprehensive understanding of the key components of our method, we conducted a series of ablation experiments to investigate the effects of different components:

- **w/o Coop. State Representation**: We substitute the GNN with simple MLP for encoding the grid features.
- **w/o Coop. Action Generation**: We substitute our action generation design with the approach used in CoRide [7], where the actors only learn order preferences of the grids without embedding orders' features into a uniform space.

Table 3: Ablation studies. Each value denotes the mean and standard deviation over 5 runs with different seeds. The best results in each column are highlighted in bold.

City	Haikou		Chengdu		New York	
Metric	Norm. GMV	ORR	Norm. GMV	ORR	Norm. GMV	ORR
w/o Coop. State Representation	96.77 ± 0.15	58.88% ± 0.06%	90.30 ± 0.07	58.31% ± 0.08%	89.81 ± 0.13	62.26% ± 0.06%
w/o Coop. Action Generation	121.84 ± 0.13	65.86% ± 0.05%	106.02 ± 0.12	66.51% ± 0.06%	108.42 ± 0.11	69.43% ± 0.06%
w/o Coop. Reward Learning	120.49 ± 0.09	65.36% ± 0.04%	105.92 ± 0.06	65.73% ± 0.05%	106.75 ± 0.06	68.45% ± 0.05%
Full Design	**124.71 ± 0.03**	**67.92% ± 0.02%**	**108.01 ± 0.08**	**69.75% ± 0.05%**	**110.97 ± 0.07**	**71.97% ± 0.07%**

- **w/o Coop. Reward Learning**: We neglect the variation of cooperation intensities and directly train all grid agents with the global return, which turns out to be standard MAPPO [37].

We report the evaluation results on three datasets in Table 3. It is evident that the absence of any component leads to performance degradation while removing the Cooperative State Representation module makes agents suffer the most. This proves the necessities of each component of our algorithm, and ultimately, the full version yields the best performance.

4.5 Effectiveness of Cooperation

In this section, we provide in-depth analyses of the effectiveness of our global cooperation design. Taking the dataset of Haikou as an example, we first present changes in the mean cooperation intensity of all agents throughout the training process in Figure 3. We aggregate the cooperation intensity based on the distance among grids, where y-hop refers to grids that can be reached by passing through at least y other grids. For example, 0-hop is the grid itself, and 1-hop contains its direct neighbors. We find that the training process can be roughly divided into several stages according to the changes in the mean cooperation intensity. In the initial stage, the agents focus on learning strategies that benefit themselves (0-hop). As the training proceeds to the next stage, the cooperation intensity of 1-hop, i.e., cooperation among neighbors, becomes dominant. Continuing the training process, the weight of selfishness decreases monotonically while the cooperation intensities among distant grids gradually increase. Eventually, all cooperation intensities converge, establishing a stable global cooperation that covers various extents. This indicates that our method can automatically learn cooperation from local to global, progressively learning strategies from simple to complex.

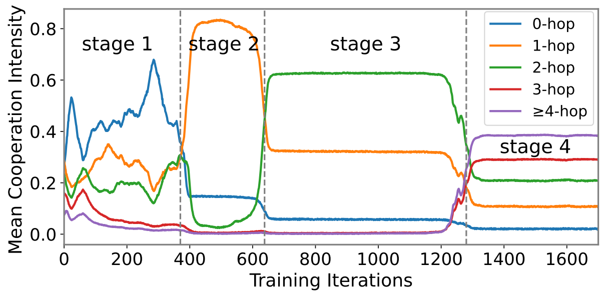
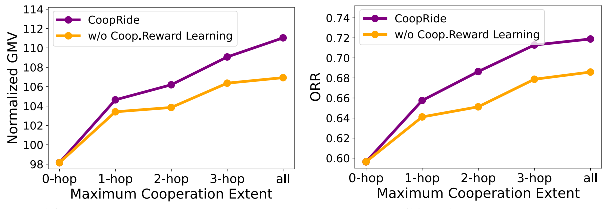


Figure 3: Changes in mean cooperation intensity among agents during the training process.

Besides, we look into the effect of the maximum cooperation extent. In figure 4, we use the dataset of New York as an example and limit the maximum cooperation extent, observing an evident performance degradation. This proves the significant superiority of our global cooperation design over the local cooperation solutions. Also, we neglect the variation of cooperation intensities as in Section 4.4 and also change the maximum cooperation extent. The results again demonstrate the significance of global cooperation compared to local ones and echo the importance of the dynamic learning of cooperation intensities. It is worth mentioning that when the maximum cooperation extent is limited to 0-hop, both curves degenerate to the IPPO [1] algorithm.



(a) Effect on Normalized GMV (b) Effect on ORR

Figure 4: Effect of the maximum cooperation extent.

4.6 Scalability

To assess the capability of our algorithm in scaling up to scenarios with more drivers and orders, we conduct scalability experiments using the Haikou dataset. We construct the 'double' and 'triple' datasets by sampling drivers and orders with replacements from the original dataset. We compare our method with rule-based methods and RLW, the best baseline in the Haikou dataset. As shown in Table 4, our algorithm consistently outperforms the baseline across different scales of drivers and orders. Therefore, our algorithm demonstrates the capability in scaling up, indicating great potential for deployment in real-world city-scale ride-hailing platforms.

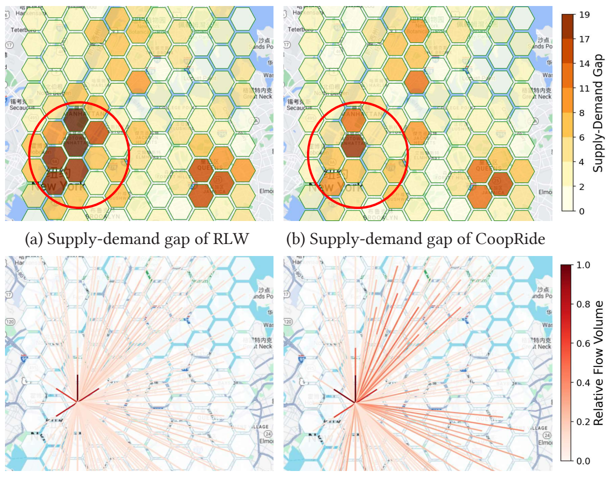
4.7 Strategy Visualization

To intuitively understand the learned strategies, we visualize the situations in New York after dispatching by RLM and CoopRide in Figure 5. It illustrates that RLM results in a substantial supply-demand gap, identified as the dark region in the lower left corner, especially. Also, the dispatching strategies lack systematic cooperation among grids, where the correlations only concentrate on neighboring grids. In contrast, our approach reduces this gap by global cooperation, which correlates both nearby and distant grids,

Table 4: Scalability performance. Each value denotes the mean and standard deviation over 5 repeated runs with different seeds. In each column, the best results are highlighted in bold.

Dataset Scale	Origin		Double		Triple	
Metric	Norm. GMV	ORR	Norm. GMV	ORR	Norm. GMV	ORR
KM	113.67 ± 0.14	57.12% ± 0.12%	114.35 ± 0.08	53.14% ± 0.05%	114.49 ± 0.07	56.09% ± 0.06%
RLW	120.62 ± 0.14	61.49% ± 0.10%	124.88 ± 0.08	60.68% ± 0.05%	121.70 ± 0.03	62.74% ± 0.03%
CoopRide	**124.71 ± 0.03**	**67.92% ± 0.02%**	**127.29 ± 0.06**	**69.09% ± 0.06%**	**124.24 ± 0.03**	**70.09% ± 0.01%**

balancing the distribution of drivers and orders. For detailed visualizations regarding more cities, please refer to Appendix A.4.



(a) Supply-demand gap of RLW (b) Supply-demand gap of CoopRide
(c) Cross-grid movements of RLW (d) Cross-grid movements of CoopRide

Figure 5: Visualizations in New York after dispatching by RLM and CoopRide. (a) (b) Gaps between supply and demand, where grids with larger gaps exhibit darker color. (c) (d) Cross-grid movements related to a representative grid, where lines with darker color indicate larger volumes of movements.

5 Related Works

5.1 Ride-Hailing Dispatching

Ride-hailing dispatching is a long-standing research problem. Conventional rule-based solutions [9, 12, 23] in ride-hailing systems tackle the problem via expert experience, leading to sub-optimal performance. Recently, the optimization of order dispatching has been explored by applying RL methods [6, 7, 10, 13, 15, 24]. On the one hand, some researchers apply RL for order dispatching by treating each driver as an independent agent [6]. On the other hand, it is more typical to divide the city into grids and control the dispatching in each grid with one grid agent, reducing the number of agents and accelerating the learning progress [13, 15, 24]. However, the above methods exclude reward signals that motivate the learning of cooperation among agents, leading to greedy choices of each driver or grid, which hinders the global benefit. To cope with this issue, other researchers develop grid-based MARL approaches to associate cooperation among neighboring agents [7, 10]. Different from these existing studies, we investigate global cooperation among all grid-agents in city-scale order dispatching, which maximizes the effectiveness of the learned strategy.

5.2 Multi-Agent Reinforcement Learning

Initial MARL methods directly put together multiple individual agents to adapt to multi-agent settings, such as independent Q-learning [28] and independent PPO [1]. The advanced MARL methods can be mainly categorized into two streams [36], value decomposition (VD) and centralized training and decentralized execution (CTDE). The VD methods [20, 25, 31] decompose the joint value function to each individual agent, addressing the credit assignment problem in multi-agent tasks, e.g. SMAC [21]. The CTDE framework [4, 17] learns a centralized critic to optimize the decentralized actors, which generate actions based on local observations.

When considering cooperation among agents, conventional MARL algorithms simply sum up the agents' local rewards to obtain the cooperative reward [4]. Besides, various existing works utilize the mean-field algorithm, which approximates high-dimensional states with averaged vectors, to simplify the interactions among agents [5, 35]. Different from them, we hire GNN to model the interactions among agents, and we employ the meta-gradient technique [2, 33, 38] to automatically learn the cooperation intensity among agents from a global view, thereby dynamically calculating the cooperative reward. Our designs ensure a more efficient learning of global cooperation strategies.

6 Conclusions

In this paper, we propose the CoopRide framework that empowers global cooperation among all grids in city-scale ride-hailing dispatching. We model the cooperative interactions among agents using graphs and design a GNN to extract valuable information for decision-making. We create a uniform encoding mechanism for within- and cross-grid dispatching, enabling flexible cooperation balancing both types of actions. We also design a automatic learning mechanism for the cooperation intensities among grids, based on which we dynamically calculate the cooperative rewards that drive the learning of global cooperation. We conducted extensive experiments and in-depth analyses on various real-world datasets, proving the effectiveness of our method, which achieves superior performance compared with baselines. Our work illustrates CoopRide's great potential in practical applications, providing optimized dispatching strategies for ride-hailing platforms in the real world.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China under U23B2030, 62272260, China Postdoctoral Science Foundation under Grant Number 2023M742034, and Tsinghua University-Toyota Research Center.

References

[1] Christian Schroeder De Witt, Tarun Gupta, Denys Makoviichuk, Viktor Makoviychuk, Philip HS Torr, Mingfei Sun, and Shimon Whiteson. 2020. Is independent learning all you need in the starcraft multi-agent challenge? *arXiv preprint arXiv:2011.09533* (2020).

[2] Yali Du, Lei Han, Meng Fang, Ji Liu, Tianhong Dai, and Dacheng Tao. 2019. Liir: Learning individual intrinsic reward in multi-agent reinforcement learning. *Advances in Neural Information Processing Systems* 32 (2019).

[3] Soheil Sadeghi Eshkevari, Xiaocheng Tang, Zhiwei Qin, Jinhan Mei, Cheng Zhang, Qianying Meng, and Jia Xu. 2022. Reinforcement Learning in the Wild: Scalable RL Dispatching Algorithm Deployed in Ridehailing Marketplace. *arXiv preprint arXiv:2202.05118* (2022).

[4] Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. 2018. Counterfactual multi-agent policy gradients. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 32.

[5] Qianyue Hao, Wenzhen Huang, Tao Feng, Jian Yuan, and Yong Li. 2023. GAT-MF: Graph Attention Mean Field for Very Large Scale Multi-Agent Reinforcement Learning. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 685–697.

[6] John Holler, Risto Vuorio, Zhiwei Qin, Xiaocheng Tang, Yan Jiao, Tiancheng Jin, Satinder Singh, Chenxi Wang, and Jieping Ye. 2019. Deep reinforcement learning for multi-driver vehicle dispatching and repositioning problem. In *2019 IEEE International Conference on Data Mining (ICDM)*. IEEE, 1090–1095.

[7] Jiarui Jin, Ming Zhou, Weinan Zhang, Minne Li, Zilong Guo, Zhiwei Qin, Yan Jiao, Xiaocheng Tang, Chenxi Wang, Jun Wang, et al. 2019. Coride: joint order dispatching and fleet management for multi-scale ride-hailing platforms. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. 1983–1992.

[8] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).

[9] Der-Horng Lee, Hao Wang, Ruey Long Cheu, and Siew Hoon Teo. 2004. Taxi dispatch system based on current demands and real-time traffic conditions. *Transportation Research Record* 1882, 1 (2004), 193–200.

[10] Minne Li, Zhiwei Qin, Yan Jiao, Yaodong Yang, Jun Wang, Chenxi Wang, Guobin Wu, and Jieping Ye. 2019. Efficient ridesharing order dispatching with mean field multi-agent reinforcement learning. In *The world wide web conference*. 983–994.

[11] Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. 2017. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. *arXiv preprint arXiv:1707.01926* (2017).

[12] Ziqi Liao. 2003. Real-time taxi dispatching using global positioning systems. *Commun. ACM* 46, 5 (2003), 81–83.

[13] Kaixiang Lin, Renyu Zhao, Zhe Xu, and Jiayu Zhou. 2018. Efficient large-scale fleet management via multi-agent deep reinforcement learning. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*. 1774–1783.

[14] Michael L Littman. 1994. Markov games as a framework for multi-agent reinforcement learning. In *Machine learning proceedings 1994*. Elsevier, 157–163.

[15] Chenxi Liu, Chao-Xiong Chen, and Chao Chen. 2021. META: A city-wide taxi repositioning framework based on multi-agent reinforcement learning. *IEEE Transactions on Intelligent Transportation Systems* 23, 8 (2021), 13890–13895.

[16] Yang Liu, Ruo Jia, Jieping Ye, and Xiaobo Qu. 2022. How machine learning informs ride-hailing services: A survey. *Communications in Transportation Research* 2 (2022), 100075.

[17] Ryan Lowe, Yi I Wu, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. 2017. Multi-agent actor-critic for mixed cooperative-competitive environments. *Advances in neural information processing systems* 30 (2017).

[18] James Munkres. 1957. Algorithms for the assignment and transportation problems. *Journal of the society for industrial and applied mathematics* 5, 1 (1957), 32–38.

[19] Zhenghao Peng, Quanyi Li, Ka Ming Hui, Chunxiao Liu, and Bolei Zhou. 2021. Learning to simulate self-driven particles system with coordinated policy optimization. *Advances in Neural Information Processing Systems* 34 (2021), 10784–10797.

[20] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder De Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. 2020. Monotonic value function factorisation for deep multi-agent reinforcement learning. *The Journal of Machine Learning Research* 21, 1 (2020), 7234–7284.

[21] Mikayel Samvelyan, Tabish Rashid, Christian Schroeder De Witt, Gregory Farquhar, Nantas Nardelli, Tim GJ Rudner, Chia-Man Hung, Philip HS Torr, Jakob Foerster, and Shimon Whiteson. 2019. The starcraft multi-agent challenge. *arXiv preprint arXiv:1902.04043* (2019).

[22] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* (2017).

[23] Kiam Tian Seow, Nam Hai Dang, and Der-Horng Lee. 2009. A collaborative multiagent taxi-dispatch system. *IEEE Transactions on Automation science and engineering* 7, 3 (2009), 607–616.

[24] Jiahui Sun, Haiming Jin, Zhaoxing Yang, Lu Su, and 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*. 3950–3960.

[25] Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z Leibo, Karl Tuyls, et al. 2017. Value-decomposition networks for cooperative multi-agent learning. *arXiv preprint arXiv:1706.05296* (2017).

[26] Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. 1999. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems* 12 (1999).

[27] Remi Tachet, Oleguer Sagarra, Paolo Santi, Giovanni Resta, Michael Szell, Steven H Strogatz, and Carlo Ratti. 2017. Scaling law of urban ride sharing. *Scientific reports* 7, 1 (2017), 1–6.

[28] Ardi Tampuu, Tambet Matiisen, Dorian Kodelja, Ilya Kuzovkin, Kristjan Korjus, Juhan Aru, Jaan Aru, and Raul Vicente. 2017. Multiagent cooperation and competition with deep reinforcement learning. *PloS one* 12, 4 (2017), e0172395.

[29] Xiaocheng Tang, Zhiwei Qin, Fan Zhang, Zhaodong Wang, Zhe Xu, Yintai Ma, Hongtu Zhu, and Jieping Ye. 2019. A deep value-network based approach for multi-driver order dispatching. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*. 1780–1790.

[30] Xiaocheng Tang, Fan Zhang, Zhiwei Qin, Yansheng Wang, Dingyuan Shi, Bingchen Song, Yongxin Tong, Hongtu Zhu, and Jieping Ye. 2021. Value Function is All You Need: A Unified Learning Framework for Ride Hailing Platforms. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 3605–3615.

[31] Jianhao Wang, Zhizhou Ren, Terry Liu, Yang Yu, and Chongjie Zhang. 2020. Qplex: Duplex dueling multi-agent q-learning. *arXiv preprint arXiv:2008.01062* (2020).

[32] Zhe Xu, Zhixin Li, Qingwen Guan, Dingshui Zhang, Qiang Li, Junxiao Nan, Chunyang Liu, Wei Bian, and Jieping Ye. 2018. Large-scale order dispatch in on-demand ride-hailing platforms: A learning and planning approach. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 905–913.

[33] Zhongwen Xu, Hado P van Hasselt, and David Silver. 2018. Meta-gradient reinforcement learning. *Advances in neural information processing systems* 31 (2018).

[34] Chiwei Yan, Helin Zhu, Nikita Korolko, and Dawn Woodard. 2020. Dynamic pricing and matching in ride-hailing platforms. *Naval Research Logistics (NRL)* 67, 8 (2020), 705–724.

[35] Yaodong Yang, Rui Luo, Minne Li, Ming Zhou, Weinan Zhang, and Jun Wang. 2018. Mean field multi-agent reinforcement learning. In *International conference on machine learning*. PMLR, 5571–5580.

[36] Yaodong Yang and Jun Wang. 2020. An overview of multi-agent reinforcement learning from game theoretical perspective. *arXiv preprint arXiv:2011.00583* (2020).

[37] Chao Yu, Akash Velu, Eugene Vinitsky, Jiaxuan Gao, Yu Wang, Alexandre Bayen, and Yi Wu. 2022. The surprising effectiveness of ppo in cooperative multi-agent games. *Advances in Neural Information Processing Systems* 35 (2022), 24611–24624.

[38] Zeyu Zheng, Junhyuk Oh, and Satinder Singh. 2018. On learning intrinsic rewards for policy gradient methods. *Advances in Neural Information Processing Systems* 31 (2018).

[39] Ming Zhou, Jiarui Jin, Weinan Zhang, Zhiwei Qin, Yan Jiao, Chenxi Wang, Guobin Wu, Yong Yu, and Jieping Ye. 2019. Multi-agent reinforcement learning for order-dispatching via order-vehicle distribution matching. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. 2645–2653.

A Appendix

A.1 Training Algorithm

The training process of CoopRide is summarized in Algorithm 1.

Algorithm 1 Training algorithm of CoopRide.

```
Require: Environment, data buffer $D$, grid policy network $\pi_\theta$ and value network $V^C$, cooperation intensity network $\Phi$ and global value network $V^G$.
1: for iteration= 1, ..., $ITER$ do
2:   Clear buffer $D$.
3:   Store current policy $\pi_{\theta old} \leftarrow \pi_\theta$.
4:   for $t = 1, ..., T$ do
5:     Sample $\mathbf{a}_t$ from $\pi_\theta$, observe $\mathbf{r}_t$ and next states $\mathbf{s}_{t+1}$.
6:     Store $\langle \mathbf{s}_t, \mathbf{a}_t, \mathbf{r}_t, \mathbf{s}_{t+1} \rangle$ into buffer $D$.
7:   end for
8:   Compute $A^C_{i,t}, A^G$ following (7) and (14) for all transitions.
9:   Compute $r^C_{i,t}$ for all transitions in $D$ with $\Phi$.
10:  for epoch= 1, ..., $K_p$ do
11:    Shuffle buffer $D$ and slice it into mini-batches.
12:    for $i = 1, ..., N$ do
13:      Update $\pi_\theta$ using the mini-batches following (9).
14:      Update $V^C$ minimizing $\|r^C_{i,t} + \gamma V^C(s_{i,t+1}) - V^C(s_{i,t})\|$.
15:    end for
16:  end for
17:  for epoch = 1, ..., $K_\Phi$ do
18:    Shuffle buffer $D$ and slice it into mini-batches.
19:    Update $\Phi$ using the mini-batches following (16).
20:    Update $V^G$ minimizing $\|r^G_t + \gamma V^G(\mathbf{s}_{t+1}) - V^G(\mathbf{s}_t)\|$.
21:  end for
22: end for
```

A.2 Implementation Details for Reproducibility

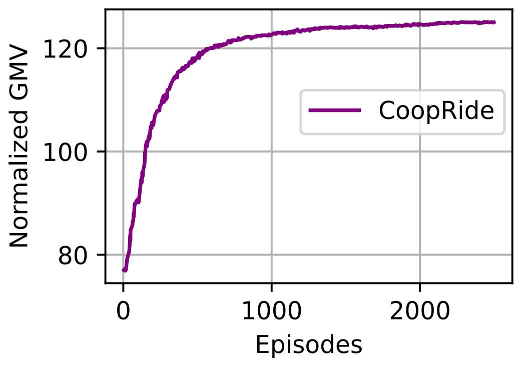
We perform experiments using Python 3.9 and Pytorch 2.1 with NVIDIA GeForce RTX 3090 GPUs. Here, we provide detailed values of the hyper-parameters used in the experiments for reproducibility in Table 5. For more details, please refer to our open-source code at https://github.com/tsinghua-fib-lab/CoopRide.

Table 5: Implementation Details.

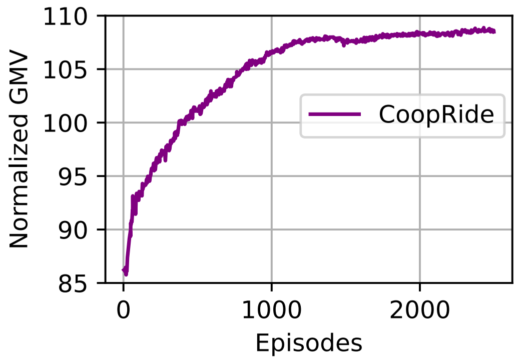
Hyper-parameter	Notation	Value
Network initialization	–	Orthogonal
Number of layers in GNN	M	2
Hidden state dimension of GNN	–	64
Dimension of grid state representations	–	64
Hidden state dimensions in MLP_K	–	[128,128]
Number of layers in MLP_Q	–	2
Hidden state dimensions in MLP_K	–	[128,128]
Number of layers in MLP_Q	–	2
Hidden state dimensions in Φ	–	[128,128]
Number of layers in Φ	–	2
Dimension of uniform embeddings	d	128
Learning rate of grid policy network π_θ	α	1e-3
Learning rate of grid value network V_C	α	1e-3
Learning rate of Φ	α	1e-3
Learning rate of global value network V_G	α	1e-3
Mini batch size	B	1000
Grid actors optimization epoch	K_p	1
Cooperative intensities optimization epoch	K_Φ	1
Optimizer	–	Adam
Optimizer epsilon	–	1e-5
Number of training steps	–	0.2M
Discount factor	γ	0.97
Gradient clip norm in PPO	–	10
GAE in PPO	λ	0.95
Clip ratio in PPO	ϵ	0.2
Entropy coefficient in PPO	–	0.005

A.3 Learning Curves

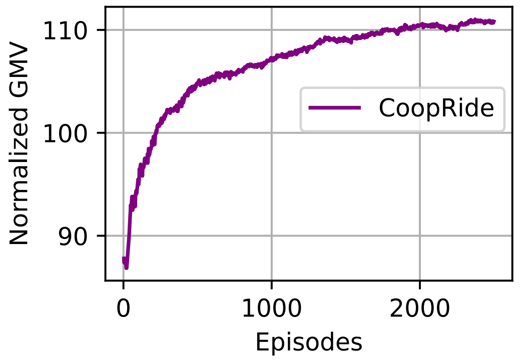
We show the changing and converging of normalized GMV during the training process on different datasets in Figure 6. The curves illustrate stable training process and good convergence of our method.



(a) Haikou



(b) Chengdu



(c) New York

Figure 6: Changing and converging of normalized GMV during the training process in (a) Haikou, (b) Chengdu, and (c) New York.

A.4 Extended Visualizations

In Figure 7, we visualize the situations in all three cities after dispatching by baseline methods and CoopRide. The visualizations illustrate that the baseline methods all result in a substantial supply-demand gap, identified as the dark regions. In contrast, our approach reduces this gap by global cooperation.

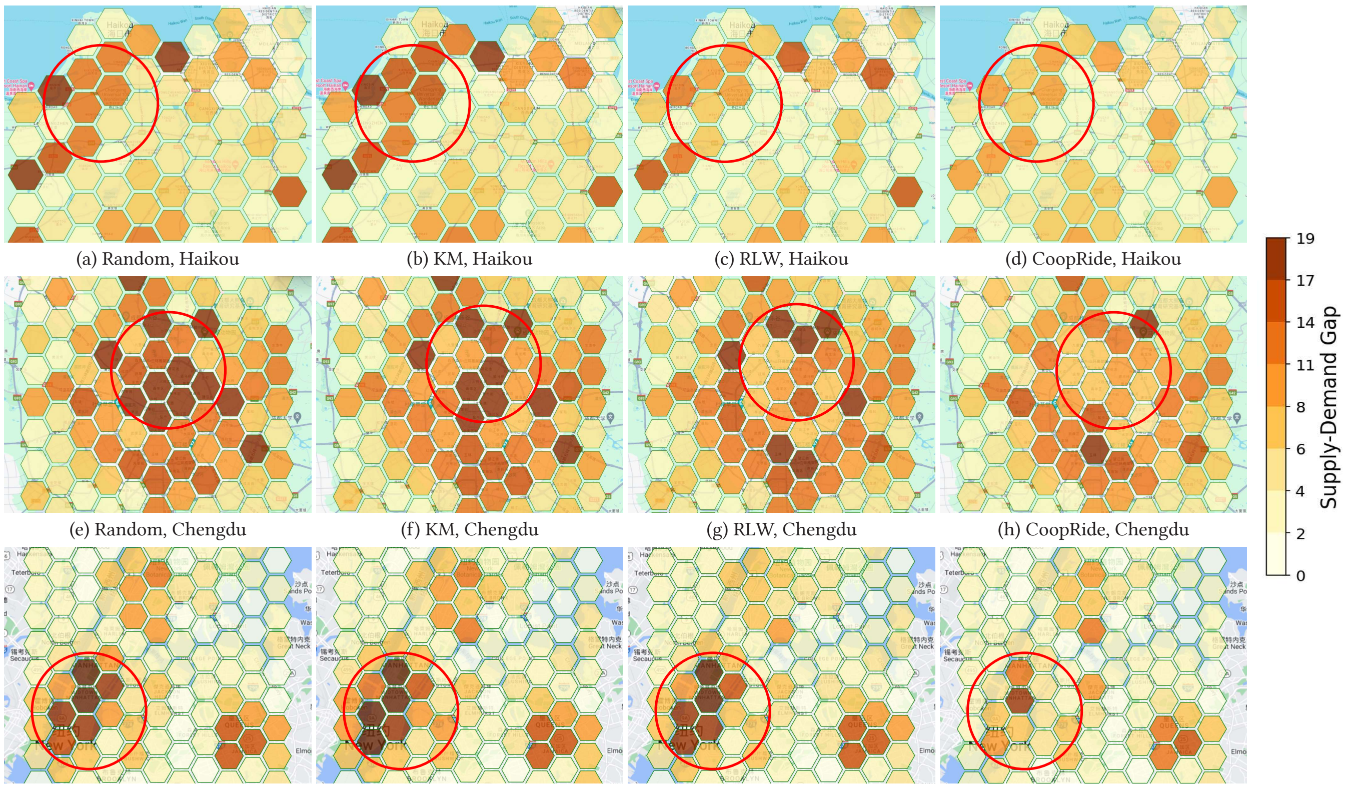


Figure 7: Gaps between supply and demand. (a)-(d) Haikou, (e)-(h) Chengdu, (i)-(l) New York. Grids with larger gaps exhibit darker color.