Rethinking Order Dispatching in Online Ride-Hailing Platforms

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

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

Guiyun Fan*
Shanghai Jiao Tong University
Shanghai, China
fgy726@sjtu.edu.cn

Min Lu
Didi Chuxing
Beijing, China
lumin@didiglobal.com

Yiran Liu
Didi Chuxing
Beijing, China
yiranliu@didiglobal.com

Xinlang Yue
Didi Chuxing
Beijing, China
rumyue@didiglobal.com

Hao Pan
Didi Chuxing
Beijing, China
hoytpan@didiglobal.com

Zhe Xu
Didi Chuxing
Beijing, China
xuzhejesse@didiglobal.com

Guobin Wu
Didi Chuxing
Beijing, China
wuguobin@didiglobal.com

Qun Li
Didi Chuxing
Beijing, China
liquntracy@didiglobal.com

Xiaotong Wang
Didi Chuxing
Beijing, China
xiaotongwang@didiglobal.com

Jiecheng Guo
Didi Chuxing
Beijing, China
jasonguo@didiglobal.com

Abstract

Achieving optimal order dispatching has been a long-standing challenge for online ride-hailing platforms. Early methods would make shortsighted matchings as they only consider order prices alone as the edge weights in the driver-order bipartite graph, thus harming the platform's revenue. To address this problem, recent works evaluate the value of the order's destination region to be the long-term income a driver could obtain in average in such region and incorporate it into the order's edge weight to influence the matching results. However, they often result in insufficient driver supplies in many regions, as the values evaluated in different regions vary greatly, mainly because the impact of one region's value on the future number of drivers and revenue in other regions is overlooked. This paper models such impact within a cooperative Markov game, which involves each value's impact over the platform's revenue with the goal to find the optimal region values for revenue maximization. To solve this game, our work proposes a novel *goal-reaching collaboration (GRC)* algorithm that realizes credit assignment from a novel goal-reaching perspective, addressing the difficulty for accurate credit assignment with large-scale agents of previous methods and resolving the conflict between credit assignment and offline reinforcement learning. Specifically, during training, GRC predicts the city's future state through an environment model and utilizes a scoring model to rate the predicted states to judge their levels of profitability, where high-scoring states are regarded as the goal states. Then, the policies in the game are updated to promote the city to stay in the goal states for as long as possible. To evaluate GRC, we deploy a baseline policy online in several cities for three weeks to collect real-world dataset. Training and testing results on the collected dataset indicate that our GRC consistently outperforms the baselines in different cities and peak periods.

*Corresponding authors.

CCS Concepts

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

Keywords

Order Dispatching, Online Ride-Hailing, Multi-Agent Reinforcement Learning

ACM Reference Format:
Zhaoxing Yang, Haiming Jin, Guiyun Fan, Min Lu, Yiran Liu, Xinlang Yue, Hao Pan, Zhe Xu, Guobin Wu, Qun Li, Xiaotong Wang, and Jiecheng Guo. 2024. Rethinking Order Dispatching in Online Ride-Hailing Platforms. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '24), August 25–29, 2024, Barcelona, Spain.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3637528.3672028

1 Introduction

The advent of online ride-hailing platforms (e.g., Didi Chuxing, Uber), providing on-demand car service, has greatly facilitated people's daily transportation. One of the most critical decisions for a ride-hailing platform is *order dispatching (OD)*, which aims to match drivers and orders at each decision moment to maximize the revenue of the platform.

In practice, a ride-hailing platform usually makes OD decisions by performing matching on a constructed driver-order bipartite

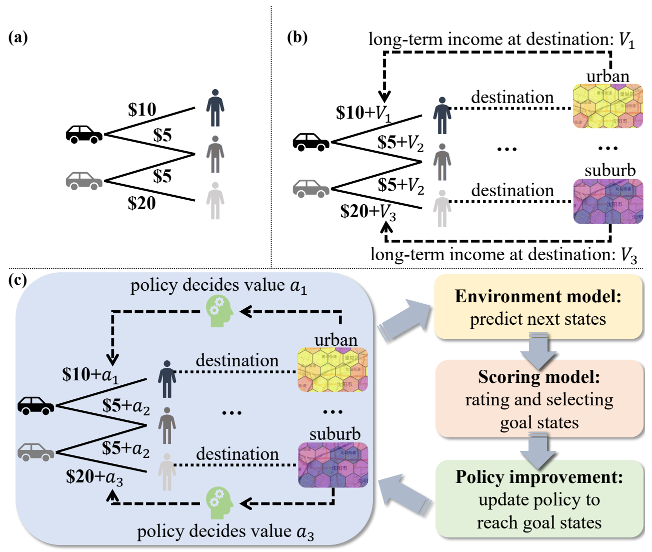
(a)
$10
$5
$5
$20
(b)
long-term income at destination: V_1
urban
destination
$10+$V_1$
$5+$V_2$
$5+$V_2$
$20+$V_3$
suburb
destination
long-term income at destination: V_3
(c)
policy decides value a_1
urban
destination
$10+$a_1$
$5+$a_2$
$5+$a_2$
$20+$a_3$
suburb
destination
policy decides value a_3
Environment model: predict next states
Scoring model: rating and selecting goal states
Policy improvement: update policy to reach goal states

Figure 1: Comparisons of the edge weights used in different methods. In each subfigure, the passenger orders from top to bottom have price \$10, \$5, \$20, respectively. (a) Methods only take the price of the order as the edge weight. (b) Recent works [22, 24, 25, 27] add a value that reflects the long-term income a driver could obtain at the destination into the edge weight. (c) Our work regards the value as the decision, and utilizes the proposed GRC algorithm to incentivize the policy to reach the goal states as possible. The figure and the descriptions in Sec. 1 utilize KM as the underlying matching algorithm, while other algorithms, such as the GS algorithm, can also be integrated in a similar way. Discount is not shown in (b-c) for the sake of simplicity and clarity.

graph using Kuhn-Munkres (KM) [17] or Gale–Shapley (GS) [6] algorithm. Specifically, the platform treats the available drivers and orders to be served at each decision moment in a city as the nodes of a bipartite graph, and constructs an edge between a driver node and an order node if the driver is within a predefined distance from the order. Naturally, the edge weights of such bipartite graph would significantly influence the matching results. Early methods adopted by the ride-hailing industry set the weight of each edge as the price of the order on it, as shown in Fig. 1(a). However, such methods do not take into account the future demands in each order's destination, and might match drivers to suburbs for short-term gains, potentially reducing the platform's future revenue.

To avoid making such myopic decisions, recent works [22, 24, 25, 27] define the destination value of an order to be the long-term income a driver could obtain in average in the order's destination region, and add the discounted value into the edge weights, as illustrated in Fig. 1(b). However, these works tend to reduce the number of vehicles in the cold regions within the city, leading to a shortage of drivers available to serve passengers, and consequently, a loss in revenue. This is because under these works, a few hot regions will have exceptionally high destination values, significantly surpassing other regions. The result of this immense disparity is that when such values are included in the edge weights of the KM algorithm, nearly all drivers will be matched to orders heading for the hot regions, whereas very few or even no drivers will get orders going to the cold regions. Given that real-world cities often contain a large number of cold regions, the cumulative loss from each region can result in significant revenue loss for the platform.

We argue that the primary reason for this result is that these works evaluate the value of different regions at the same time step independently of each other. However, the value of each region can have significant impacts on the future revenue of other regions. This is because when using these values for matching, the value of one region will affect the edge weights of all orders heading to that region. Furthermore, given the combinatorial nature of bipartite matching, the value will affect the matching of orders heading to other regions, thereby affecting the future number of drivers and revenue of those regions, and thus the platform. As such, each region must take into account the potential impacts of its value on the revenue of other regions when setting its value, in order to maximize the total platform's revenue. Otherwise, as shown by the aforementioned result, the excessively high value of a few regions severely compromises the supply and revenue of other regions.

Based on the above analysis, this study aims to solve the problem FOVIR: *Find the Optimal Value of each region, considering the Impacts of each region's value on other regions, in order to optimize the platform's Revenue.* In order to consider such impacts to model problem FOVIR, we use the cooperative Markov game, and we design a cooperative *multi-agent reinforcement learning (MARL)* algorithm to solve the game.

More specifically, in our game, each region is regarded as an agent, and it will output actions based on the policy. The objective of the game is to find the optimal policy to maximize total revenue. In particular, such action output of each agent will be considered as the value of each region. After discounting for future uncertainty, the action will be added to the edge weight of orders heading to that region, influencing the matching and future revenue. However, directly using existing cooperative MARL algorithms to optimize such policy presents two significant challenges.

- *Difficulty for accurate credit assignment among large-scale agents.* Efficient and accurate credit assignment is at the core of cooperative MARL algorithms, as it assesses each agent's contribution to the overall revenue and policies are optimized based on such estimated contribution. However, previous researches [10, 26] have found that in tasks involving a large number of agents, the credit assignment based on the central critic is a challenging endeavor. This is because the vast number of states and actions of multiple agents will overwhelm the contribution of an individual agent's decision, making it difficult to estimate the latter effectively and accurately.
- *Conflict between credit assignment and offline reinforcement learning.* Given the complex and dynamic nature of urban traffic conditions, creating a simulator that is both highly accurate and fast for generating large volume of samples for training online reinforcement learning algorithms is a challenging task. Consequently, we have opted for an offline reinforcement learning training approach. However, such method raises conflict with credit assignment: the former seeks to avoid sampling of actions outside the dataset distribution (i.e., out-of-distribution actions

[14]), while the latter relies on evaluating all possible actions and calculating the advantage of a specific action [5, 28].

To address these challenges, our inspiration originates from the following findings. Specifically, we find that in OD, in addition to the straightforward performance indicator revenue, there are several other indicators that can indirectly reflect the profitability of the platform. For instance, a very low passenger cancellation rate or a small supply-demand gap can indicate that a large number of orders within the city are being served, thereby suggesting that the platform is operating at a high level of profitability. In light of this, we can consider other indicators such as supply-demand gap, cancellation rate, and pick-up distance, in addition to revenue. By encouraging the city to consistently maintain a high level in these various indicators, we can optimize the overall revenue of the platform. Based on this, we propose the *goal-reaching collaboration (GRC)* algorithm to realize this idea, and it achieves credit assignment from a completely new perspective of achieving goals, overcoming the aforementioned challenges.

In GRC, we first train a scoring model, capable of determining whether a city state performs well on various indicators, where a city state at a certain time step refers to the vector that can fully describe all drivers and orders in the city. Next, we train an environment model that can combine the current city state and the action decisions of agents to predict possible future states of the city. Subsequently, the predicted states are fed into the scoring model for rating, where a small batch of high-scoring states is selected as the goal states, and the other states are regarded as ordinary states. Finally, we optimize the policy by minimizing the gap between ordinary and goal states, promoting the city to reach the goal states in the future as possible and continuously maintain a high level on various indicators.

Notably, GRC does not require a centralized critic. More importantly, the city goal states contain the local goal states of all region agents. Therefore, when each region agent updates its policy to reach its local goal state, the city's overall goal state is consequently achieved. Essentially, the negative of gap between each agent's local ordinary state and its local goal state represents the credit assigned by the GRC. Furthermore, such credit is accurate, as GRC only predicts the city state of the next time step. Such a relatively simple task allows the environment model to provide accurate credits and update signals to the policy. Moreover, such credit assignment avoids the need to sample and evaluate all possible actions, thus is compatible with offline reinforcement learning.

To evaluate GRC, we firstly deploy the CVnet model [24] online in several cities in China for 3 consecutive weeks during April and May in 2023 to collect datasets for subsequent training and testing. After a careful data cleaning procedure, the filtered dataset has approximately 6M transitions. Then, we train our policy and baselines upon the dataset, and test them with off-policy evaluation and simulations. Through comparisons with various baselines and validations via ablation studies, we verified the superiority of our proposed algorithm across different cities and peak periods. In summary, the main contributions of this paper can be summarized as follows:

- Through an in-depth analysis of existing works for OD, we find that they lack consideration of the impacts of the value in one region to other regions, which leads to revenue loss. Instead, we model such impacts from a brand-new perspective relying on cooperative Markov games.
- Given that traditional credit assignments are difficult to accurately estimate with a large number of agents, and their conflicts with offline reinforcement learning, we propose the GRC algorithm to solve the game, which achieves credit assignment from a novel goal-reaching perspective.
- To support the training and testing of algorithms, we deployed baseline method online in several cities in China for 3 weeks to collect abundant datasets. In performance comparisons across different cities and peak periods, our proposed GRC algorithm demonstrated a consistent improvement over other baseline algorithms.

2 Workflow and Formulation

2.1 Order Dispatching Workflow

In this study, the order dispatching is accomplished by a combination of two modules, namely the decision and the matching module.

Specifically, at each decision moment, each region agent will output its action based on the policy. At each matching moment, the platform collects the available drivers and unmatched orders to construct a bipartite graph. For each order, the price of the order, coupled with the discounted action of its destination region, is used as the weight. Following this, either the KM or GS algorithm is employed to match drivers and passengers based on the weights. Subsequently, the successfully matched drivers will head to the departure point of the order to pick up the passengers and transport them to the order's destination. Drivers who are not matched might randomly reposition themselves in the city or temporarily go offline. Unmatched passenger orders may continue to wait for the next matching step or get cancelled by the passengers. In the following sections, we will introduce the details of the Markov game describing the decision module and the details of the matching algorithm used in the matching module.

2.2 Problem Formulation for FOVIR

In this section, we present our OD-oriented Cooperative Markov Game (ODCMG) for FOVIR, which is defined by tuple $\langle \mathcal{T}, \mathcal{N}, \mathcal{S}, \mathcal{L}, \mathcal{A}, P, r, \gamma \rangle$, and each item is defined as follows.

- $\mathcal{T}$ denotes the time step (i.e., decision moment) set, where $\mathcal{T} = \{1, \cdots, T\}$ and T is the number of time steps in OD. The geographic space of the city is divided into hexagon grids, with each hexagon being considered as an agent in ODCMG. Set $\mathcal{N}$ includes all such agents.
- $\mathcal{S}$ denotes the global state space. For each $s_t \in \mathcal{S}, \forall t \in \mathcal{T}$, we have $s_t = \{l_t^i\}_{i \in \mathcal{N}}$, where $l_t^i \in \mathcal{L}$ denotes the local state of agent i at step t. In this work, l_t^i contains the following aspects of grid: id, the number of orders, the number of idle drivers, the number of passenger cancellations and other statistics in the past three time steps and the same time steps over the past two days.
- $\mathcal{A}$ is the joint action space of agents. For each agent $i \in \mathcal{N}$, it will decide $a_t^i \sim \pi^i(\cdot|l_t^i)$, where π^i is the policy of agent i.

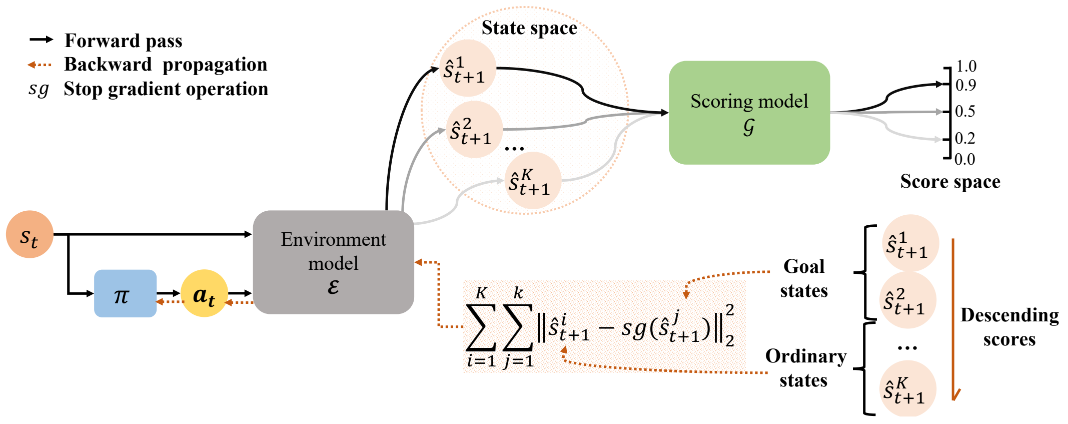

Figure 2: The goal-reaching collaboration (GRC) framework. We emphasize that each agent takes the action independently in ODCMG, while such procedure is simplified for clear illustration in the figure to a single path, which originates from state s_t through policy π to generate the joint action a_t.

- At each step t, each agent $i \in \mathcal{N}$ gets local reward r^i_t that equals the total price of orders that finish at grid i at step t, and the summation of r^i_t over all agents constitute the global reward.
- Transition function P determines the probability of state transitions given the chosen actions, but it is unknown a priori. $\gamma < 1$ is the discount factor.

ODCMG aims to find the optimal joint policy $\boldsymbol{\pi}^*$ that maximizes the long-term global reward, i.e., the revenue of the platform. That is, $\boldsymbol{\pi}^* = \arg\max_{\boldsymbol{\pi}} \mathbb{E}_{\boldsymbol{\pi}}[\sum_{t=1}^{T}\sum_{i\in\mathcal{N}} r^i_t]$ is the solution for FOVIR.

It seems that one can directly apply cooperative MARL algorithms [5, 16, 21, 28] to solve it and find the optimal policy $\boldsymbol{\pi}^*$. However, through an in-depth understanding of the online-hailing platform, two major challenges limit the feasibility of such algorithms. As discussed previously, when a large number of agents are involved, it is difficult to obtain accurate credit assignment based on a single reward signal. Secondly, credit assignment also raises conflict with offline reinforcement learning algorithms. In this work, we propose GRC to address these challenges and efficiently solve ODCMG, which will be introduced in the next section.

2.3 Matching Module

The outputted actions of each region agent will influence the subsequent matching module as follows. For action a^i_t of agent i and time step t, the weight for each order o that is heading to region i is defined as $\tau_o = R^o + \gamma^{\triangle t_o} a^i_t$, where R^o is the price of order o, $\triangle t_o$ is its duration. Then, when the underlying matching algorithm is KM, order weight τ_o will serve as the weight for each edge connecting to order o, and KM will output the maximal weighted matching of the bipartite graph. When the underlying matching algorithm is GS, each driver's preference list over the set of orders it is connected to is obtained by decreasing the order weights. Similarly, each order's preference list over the set of drivers it is connected to is obtained by increasing the pick-up distances. Then, GS is guaranteed to output the stable matching that there is no driver-order pair where both participants prefer each other to their matched ones.

We note that adopting weight τ_o here is due to its simplicity and comprehensiveness. That is, such design of weight takes into account both the price of the order and the duration of the order. The longer the order lasts, the more uncertain the future is, so we use exponential decay to lessen the impact of action on the weight when the order lasts too long. We emphasize that how to integrate the action into the weight is not the primary focus of this study, and our ODCMG formulation and GRC algorithm can be adapted to accommodate any alternative weight designs.

3 Method

In this section, we firstly take an overview of GRC. Then, the GRC will be introduced in detail by elaborating its key components. Finally, we conclude by presenting the policy training algorithm.

3.1 Overview of GRC Framework

The main idea underneath our framework comes from the specific properties of OD. Typically, the platform runs monitoring programs in each city to track real-time changes in multiple crucial indicators, making it easier for administrators to manage and operate. These indicators could indirectly reflect the platform's revenue. For example, when only small demand-supply gap exists in the city, orders will be served in time and drivers rarely remain idle, since the spatial distributions of drivers and orders are well-balanced. When the cancellation rate from passengers keeps at a low level, it represents that the pick-up distances between drivers and passengers are little, which improves the user experience and avoids the loss of revenue. In other words, when a city performs well on these crucial indicators, it reflects that the platform is in a high-revenue state. Therefore, by optimizing policies to encourage cities to sustain consistent high performance on these indicators, the platform can ensure long-term revenue growth. Formally, we refer to the states that have well-performing indicators as goal states, and we design the GRC framework to generate such goal states and incentivize and optimize the policies to reach them.

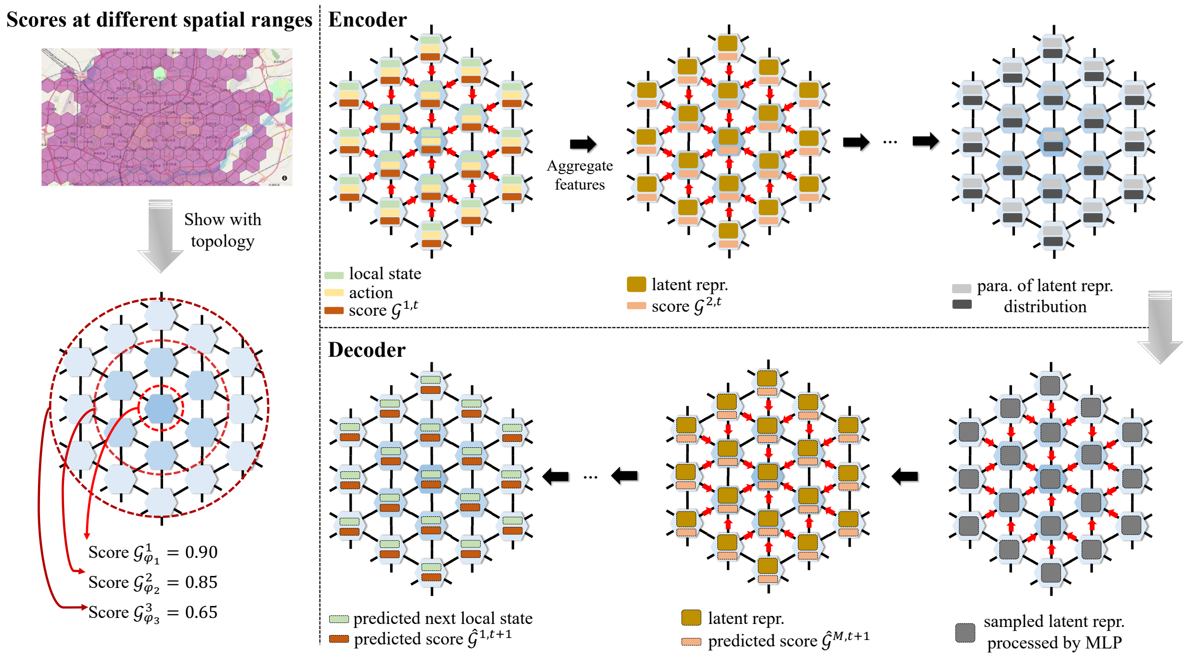
Scores at different spatial ranges
Show with topology
Score $\mathcal{G}^1_{\varphi_1} = 0.90$
Score $\mathcal{G}^2_{\varphi_2} = 0.85$
Score $\mathcal{G}^3_{\varphi_3} = 0.65$
Encoder
Aggregate features
local state
action
score $\mathcal{G}^{1,t}$
latent repr.
score $\mathcal{G}^{2,t}$
para. of latent repr. distribution
Decoder
predicted next local state
predicted score $\hat{\mathcal{G}}^{1,t+1}$
latent repr.
predicted score $\hat{\mathcal{G}}^{M,t+1}$
sampled latent repr. processed by MLP

Figure 3: Scores at different spatial ranges; illustrations of the encoder and decoder in environment model $\mathcal{E}$.

The whole GRC framework is shown in Fig. 2. At any state s_t, each policy π^i takes local state l^i_t and generates action a^i_t for each agent i, and the actions are stacked into vector $\boldsymbol{a}_t$ to ease presentation. Then, the environment model $\mathcal{E} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ takes both s_t and $\boldsymbol{a}_t$ as inputs and predicts K possible next states $s^j_{t+1}, \forall j \in [1, \cdots, K]$. Further, the scoring model $\mathcal{G} : \mathcal{S} \to [0, 1]$ rates each predicted states with a score. By ranking the predictions with descending scores, we select the top $k \ll K$ predictions as the goal states and leave others as the ordinary states. With such discrimination, we optimize the policy to let those ordinary states get closer to the goal states as possible. Hence, loss is defined as the squared differences between ordinary states and goal states, and the gradient signal is back propagated through the environment model $\mathcal{E}$ to update the policy π. The stop gradient operation is added to remove the gradients from the goal states for stability concerns.

In summary, by predicting multiple next states and ranking them with scores, GRC finds and annotates the goal states. To reach such states, GRC optimizes the policy to close the gap between ordinary and goal states. It's worth noting that such global goal state contains the local goal states of all agents. Therefore, when each agent updates its policy to achieve its own local goal state, the global goal state is also reached as a result. In this way, GRC realizes credit assignment and incentivizes the cooperation among all the agents. Compared to a single centralized reward signal, GRC provides each agent with a separate updating signal, which is much denser to estimate accurately and facilitates the optimization of policies. Furthermore, as such credit is essentially the gap between the goal states and ordinary states, it is free of sampling all the possible actions, thus avoiding the OOD issues.

3.2 Scoring Model $\mathcal{G}$

Namely, the mission of scoring model $\mathcal{G}$ is to rate each state by a score within $[0, 1]$ to show how good the state is. One intuitive idea is to rate the state directly based on the indicators. For example, a state has score 1 if and only if it has 0 cancelling rate and 0 demand-supply gap. However, such method is impractical as it ignores the real-world limitations, mainly coming from the transportation infrastructures. A more reasonable and practical method is to compare the states in a city at the same time steps across different days. By labelling the better state with a higher score and the worse state with a lower one, model $\mathcal{G}$ can be trained to discriminate the states and assign them with appropriate scores. In fact, such recipe coincides with the reward model training procedure in the recent success of large language models, such as InstructGPT [19] and ChatGPT [18]. However, our recipe is much more easier as we could utilize various application-specific rules to compare the states, without collecting the expensive human comparison data.

Formally, we pre-select a set of application-specific crucial indicators $\mathcal{D}$, including the revenue, the cancelling rate, the demand-supply gap and many others. In the experiment, we select 11 indicators for $\mathcal{D}$. Given any state pair (s, s'), we firstly use each indicator within set $\mathcal{D}$ to compare the state pair. For example, s is scored 1 if s has lower cancelling rate than s', and 0 otherwise; s is scored 1 if s has smaller absolute demand-supply gap than s', and 0 otherwise. After traversing all the indicators in $\mathcal{D}$, we sum their scores and denote the aggregated scores as (g, g') for pair (s, s'). Finally, the scoring model $\mathcal{G}$ is trained on a batch of state pairs and aggregated scores using the following loss:

$$\text{loss}(\theta) = \frac{1}{|\mathcal{D}|}\mathbb{E}_{s,s'\sim\mathcal{S}}[(g' - g)\mathcal{G}_\theta(s) + (g - g')\mathcal{G}_\theta(s')],$$

where θ is the parameters of model $\mathcal{G}$, and the gap between g and g' is used as the weight to strengthen the difference in scores.

3.3 Environment Model $\mathcal{E}$

The environment model $\mathcal{E}$ takes the current global state and the chosen actions as input to predict the possible next states. In this work, we further enhance the representation ability of $\mathcal{E}$ by exploiting the spatial properties of scores.

In fact, the scoring model $\mathcal{G}$ introduced in Sec. 3.2 is defined in the city-scale range, while scoring models of other spatial range, such as grid-wise range, can also be defined. The left figure of Fig. 3 shows the scores when different spatial range is considered. Formally, we denote $\mathcal{G}^1_{\phi_1}$ as the scoring model of each grid's local state, and $\mathcal{G}^2_{\phi_2}$ as the scoring model of each grid's aggregated local state within 1-hop neighbors... ϕ_1 and ϕ_2 are the parameters, and they are trained similarly as $\mathcal{G}$. As each grid has possibly distinct local state, the scores of the aggregated local state within different spatial range may be entirely different. Hence, integrating such hierarchical and informative scores into model $\mathcal{E}$ will be greatly helpful for the model to understand the spatial causality for goodness of a state and how its components are related to the scores. With such understanding, model $\mathcal{E}$ masters the ability to discriminate the states and predict them accordingly.

The architecture of model $\mathcal{E}$ is shown in the right figure of Fig. 3. For each step t, the encoder firstly takes the local states $\{l^i_t\}_{i\in\mathcal{N}}$, actions $\boldsymbol{a}_t$ and the scores of the local states $\boldsymbol{\mathcal{G}}^{1,t}$ as input[1], and generates the latent representation x_1 by a graph neural network (GNN). Later, the scores of each grid's aggregated local state within 1-hop neighbors $\boldsymbol{\mathcal{G}}^{2,t}$ are further combined with the latent representation x_1 to generate the second-layer latent representation x_2 by GNN... After M layers' processing, we obtain the parameters of the distribution over x_M. The decoder takes samples from the distribution, and decodes to the latent representation d_M and the predicted score $\widehat{\boldsymbol{\mathcal{G}}}^{M,t+1}$. Then, d_M is used for the next decoding to generate d_{M-1} and the predicted score $\widehat{\boldsymbol{\mathcal{G}}}^{M-1,t+1}$... After M layers' decoding, we obtain the predicted local states $\{\hat{l}^i_{t+1}\}_{i\in\mathcal{N}}$ and the predicted score $\widehat{\boldsymbol{\mathcal{G}}}^{1,t+1}$. Given a batch of sampled steps $\mathcal{B}$, model $\mathcal{E}_\chi$ is trained with the following loss:

$$\text{loss}(\chi) = \frac{1}{|\mathcal{B}|}\sum_{t\in\mathcal{B}}\Big(\sum_{i\in\mathcal{N}} \| l^i_{t+1} - \hat{l}^i_{t+1} \|_2^2 + \sum_{m=1}^{M}(\boldsymbol{\mathcal{G}}^{m,t+1} - \widehat{\boldsymbol{\mathcal{G}}}^{m,t+1})^2 + \text{KL}(\text{latent dist.} \,\|\, \mathcal{N}(0,\mathrm{I}))\Big),$$

where each item represents the state prediction error, the score prediction error and the regularization of the latent space, respectively.

When updating policies, we will sample K latent representations for each input s_t and $\boldsymbol{a}_t$, and decode them to K predicted next states. One might argue that the K predicted next states could become too similar to each other when model $\mathcal{E}_\chi$ is trained with high precision, which could potentially compromise the subsequent policy optimization. However, our findings suggest that the states derived from the real-world dataset exhibit a high degree of variance. This ensures that an accurately trained model $\mathcal{E}_\chi$ will not merely converge on predicting identical states.

[1]For each step t and scoring model $\mathcal{G}^j_{\phi_j}$, we use notation $\boldsymbol{\mathcal{G}}^{j,t}$ to stack the scores of all the grids.

3.4 Training Policy

With the trained scoring model $\mathcal{G}_\theta$ and environment model $\mathcal{E}_\chi$, policy π_β is optimized with a batch of sampled steps $\mathcal{B}$ by loss:

$$\text{loss}(\beta) = \frac{1}{|\mathcal{B}|}\sum_{t\in\mathcal{B}}\Big(\lambda_1 \sum_{i=1}^{K}\sum_{j=1}^{k} \| \hat{s}^i_{t+1} - \text{sg}(\hat{s}^j_{t+1}) \|_2^2 - \lambda_2 \sum_{i\in\mathcal{N}} \log \boldsymbol{\pi}_\beta(a^i_t|l^i_t) Q^h_i(l^i_t, a^i_t)\Big). \tag{1}$$

In the loss function, β denotes the parameters of $\boldsymbol{\pi}$, K is the number of predictions from model $\mathcal{E}_\chi$, and the top-k predictions (w.r.t. the scores rated by model $\mathcal{G}_\theta$) are served as the goal states. The remaining predictions are ordinary states, and their gaps are shown as the first item of the loss. sg is the stop gradient operation. The second item denotes the traditional policy gradient loss, where $Q^h_i(l^i_t, a^i_t) = \mathbb{E}[\sum_{t'=t}^{T} \gamma^{t'} \sum_{i\in\mathcal{N}^h_i} r^i_{t'}]$ evaluates the long-term return of agent i in h-hop range, and $\mathcal{N}^h_i$ denotes the set of agents in h-hop range of agent i. λ_1 and λ_2 are weights for the two items. We highlight that the second item only optimizes the local return of an agent and is treated as a regularizer, which has a distinct difference with the global objective in ODCMG. In experiments, we show the effectiveness when optimizing policy with $h \le 3$, while larger h may hinder the learning process.

Another noteworthy point is that GRC does not involve iterative optimization of policies. In other words, once the policy is optimized, it does not undergo retraining of a new $\mathcal{E}_\chi$ for another round of policy optimization. We adopted this single-round training approach primarily inspired by a class of one-step algorithms in recent offline reinforcement learning [2, 13]. These algorithms found that iterative policy updates require off-policy evaluation steps, which may introduce bias and harm the training of policies. However, it's worth noting that our GRC algorithm can also be combined with other offline reinforcement learning algorithms, while this is beyond the focus of this work.

4 Experiments

4.1 Dataset Collection

To support the training of GRC, we first deployed the CVnet [24] model as the behavior policy online to collect a real-world dataset. After training of CVnet, we deployed it in several cities in China for a period of three weeks during April and May in 2023. During this time, we recorded the local states, actions, and rewards of each grid at each time step in real-time. After data cleaning, filtering out holidays, and other outliers, our offline dataset contains approximately 6 million transitions. Subsequently, we partitioned the dataset for the training and testing of algorithms.

4.2 Evaluation Methods

To evaluate the policies over the real-world dataset and the generalization ability over the bootstrapped dataset, we employ two different testing methods. The first one is Off-Policy Evaluation (OPE) [14, 20], a type of method used for assessing the performance

Table 1: OPE results in city A and B. Testing is conducted on a 5-day dataset, and each entry represents the mean and standard deviation of these tests. IORR and IGMV are the improved ratios in ORR and GMV respectively, when compared to the behavior policy estimated from the dataset.

City	Algorithm	All Day		Morning Peak (06:30-09:30)		Evening Peak (16:30-19:30)	
		IORR (%)	IGMV (%)	IORR (%)	IGMV (%)	IORR (%)	IGMV (%)
A	IPPO	-4.02±4.99	-3.73±1.42	-4.85±5.08	-5.72±4.68	0.22±15.09	2.62±12.18
	MAPPO	-3.85±5.61	-3.75±1.33	-4.59±5.30	-6.41±4.51	0.42±14.51	0.47±11.79
	IL	3.43±4.99	2.72±1.28	0.11±3.84	0.01±4.25	1.99±16.31	-1.56±11.93
	LC-l1	2.59±5.18	2.07±1.28	-0.03±3.93	-1.60±3.61	0.65±14.64	-0.73±11.25
	LC-l2	3.92±5.37	2.20±1.18	0.28±3.66	-0.90±4.13	1.13±15.45	0.62±12.20
	LC-l3	4.25±5.24	1.94±1.21	0.35±3.87	-1.17±4.26	1.33±16.84	1.47±12.10
	GRCS	4.63±5.49	**3.69**±1.37	2.06±3.69	2.89±5.02	2.88±15.96	2.20±12.63
	GRC	**5.19**±5.03	3.32±1.17	**2.62**±4.16	**2.90**±4.68	**4.15**±16.91	**4.10**±13.22
B	IPPO	-4.89±1.63	-4.89±2.92	0.87±7.89	0.45±9.42	-0.34±2.10	0.29±2.84
	MAPPO	-5.14±1.54	-4.28±2.78	0.33±7.82	0.58±9.84	-0.10±2.14	0.45±2.96
	IL	5.11±1.18	4.24±2.61	0.18±7.53	0.11±9.48	0.09±2.35	-0.29±3.08
	LC-l1	5.69±1.14	4.63±2.55	0.35±7.70	0.52±9.75	0.33±2.24	-0.13±3.28
	LC-l2	5.59±1.25	4.86±2.59	0.12±7.54	0.75±10.01	0.24±2.28	0.33±3.31
	LC-l3	5.66±1.24	4.81±2.59	0.30±7.57	0.40±9.70	0.09±2.03	0.33±2.9
	GRCS	**5.97**±1.22	4.97±2.58	**1.50**±7.54	0.89±9.72	**0.40**±2.46	**0.69**±3.55
	GRC	5.71±1.23	**5.01**±2.21	**1.50**±7.64	**1.34**±9.77	0.33±2.30	0.57±2.97

of policies over the datasets obtained with a different policy. We use the widely adopted Weighted Importance Sampling (WIS) algorithm here as its simplicity and variance-reduction property. The second one is a simulation system. We have developed a simulation system that can mimic the online order dispatching workflow, and through careful parameter tuning and calibration, the simulation fits the revenue of the real online system well. Note that the simulation uses GS for matching module as it is adopted online in platforms such as Didi Chuxing [29]. As shown in Fig. 4, we plot the normalized GMV curves of the simulator and real world in two peak periods of a day for comparisons. The r^2 between the simulator's and the real world's normalized GMV is 0.986 and 0.863, and the Pearson correlation is 0.998 and 0.977 with p-value $p < 0.000001$.

When testing, OPE firstly estimates the behavior policy on the testing dataset and then applies WIS for evaluation. For simulation, we bootstrap orders from the testing dataset, and utilize simulator to evaluate the performance of a policy. We also emphasize that this simulation is not suitable for training online reinforcement learning algorithms because it takes dozens of minutes to simulate a day in the city. Such inefficiency struggles to meet the massive training sample demands of online reinforcement learning algorithms.

4.3 Baselines and Metrics

Baselines. We consider the following baselines for comparison.

- GS. GS is currently the online matching algorithm in Didi Chuxing [29], which sorts the orders in descending prices as the preference list for the drivers, and sorts the drivers in ascending pick-up distance as the preference list for passenger orders.
- IPPO [3], MAPPO [28]. Both are the state-of-the-art cooperative MARL algorithms.
- CVnet [24]. CVnet is the previous state-of-the-art and deployed algorithm specific for order dispatching. Note that the more recent [22, 25] are not considered here as their focuses are beyond the scope of this work.

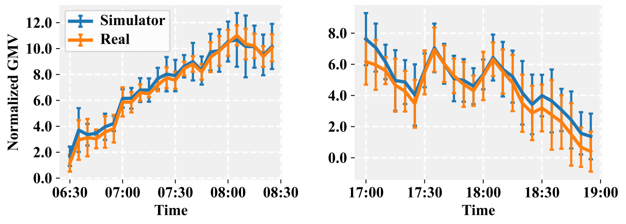

Figure 4: Comparisons on the simulator's and the real world's normalized GMV at morning and evening peak in city A.

We consider the following variants of GRC for the ablation study.

- IL. Independent learning (IL) takes $\lambda_1 = 0$, $\lambda_2 = 1$ and $h = 0$ in Eq. (1). Intuitively, each agent in IL aims to maximize its own long term reward independently.
- LC-lh, with $h \in \{1, 2, 3\}$. LC-lh has $\lambda_1 = 0$ and $\lambda_2 = 1$ in Eq. (1), while it incentivizes local cooperation by optimizing the local aggregated reward of agents in $\mathcal{N}_i^h$ for each agent i.
- GRCS. GRCS only considers the updating signal generated by the squared difference of goal and ordinary states. That is, GRCS assigns $\lambda_1 = 1$ and $\lambda_2 = 0$ in Eq. (1).

Metrics. We consider two metrics for evaluation. Order response rate (ORR) describes the ratio between the number of finished orders and the number of all emerged orders. Gross merchandise volume (GMV) describes the total price of the finished orders.

Table 2: Simulation results in city A and B. Testing is conducted on a 10-day bootstrapped dataset, and each entry represents the mean and standard deviation of these tests. IORR and IGMV are the improved ratios in ORR and GMV respectively, when compared to the GS algorithm.

City	Algorithm	All Day		Morning Peak (06:30-09:30)		Evening Peak (16:30-19:30)	
		IORR (%)	IGMV (%)	IORR (%)	IGMV (%)	IORR (%)	IGMV (%)
A	IPPO	-0.34±1.69	0.20±1.51	-0.14±1.76	-0.13±1.5	0.92±1.53	-0.17±1.44
	MAPPO	-0.16±1.59	0.20±1.47	0.23±1.40	-0.45±1.56	1.63±1.34	-0.19±1.40
	CVnet	0.22±1.60	0.05±1.43	-0.35±1.44	-0.88±1.46	0.27±1.13	0.09±1.55
	IL	0.16±1.57	-0.07±1.48	1.51±1.58	0.64±1.81	0.91±1.37	0.13±1.39
	LC-l1	0.28±1.52	0.16±1.40	0.90±1.29	0.01±1.86	1.92±1.41	0.24±1.25
	LC-l2	0.36±1.53	0.10±1.42	0.61±1.48	-0.54±1.45	1.28±1.28	0.15±1.37
	LC-l3	0.25±1.55	0.03±1.34	1.60±1.53	-0.14±1.62	1.20±1.40	0.31±1.36
	GRCS	0.94±1.92	0.12±1.49	1.87±1.40	0.97±1.41	2.02±1.95	1.16±1.34
	GRC	**1.58**±1.49	**0.42**±1.35	**1.95**±1.45	**1.21**±1.43	**2.40**±1.77	**1.51**±1.63
B	IPPO	0.42±1.10	0.25±1.20	1.06±2.22	0.66±2.04	0.28±1.70	-0.17±2.19
	MAPPO	0.33±1.25	0.07±1.17	0.11±2.19	0.28±2.11	-0.17±1.83	0.27±2.05
	CVnet	0.77±1.19	0.32±1.01	0.33±1.99	0.78±2.21	0.07±1.77	0.51±2.10
	IL	0.58±1.22	0.06±1.10	0.21±2.08	0.79±2.58	-0.03±1.81	0.54±2.44
	LC-l1	0.74±1.04	0.40±0.94	1.16±2.45	0.50±2.38	0.48±1.84	0.34±2.20
	LC-l2	1.03±1.08	0.28±1.03	0.86±2.55	0.41±2.34	0.17±1.56	-0.41±2.18
	LC-l3	1.04±1.13	0.24±0.94	0.35±1.83	0.44±2.27	-0.27±1.42	0.07±2.23
	GRCS	0.99±1.34	0.44±1.24	1.55±2.36	0.84±2.14	0.52±1.49	0.44±2.08
	GRC	**1.74**±1.13	**0.53**±1.18	**2.36**±2.39	**1.76**±2.02	**1.07**±1.69	**1.14**±2.26

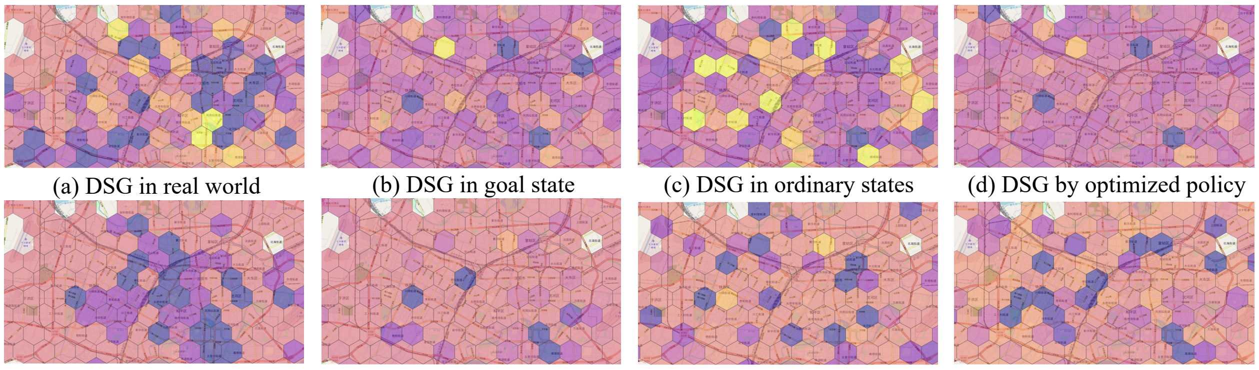
(a) DSG in real world
(b) DSG in goal state
(c) DSG in ordinary states
(d) DSG by optimized policy
(e) CR in real world
(f) CR in goal state
(g) CR in ordinary states
(h) CR by optimized policy

Figure 5: Visualizations of the predicted states in the morning peak at city A. Demand-supply gap (DSG) denotes the normalized difference of unmatched orders and available drivers. Cancellation rate (CR) denotes the normalized ratio of cancellation from the passengers. In (a-d), dark colors represent more drivers than orders, and the darker the color, the larger the gap. Light colors represent more orders than drivers, and the lighter the color, the larger the gap. In (e-h), the darker the color, the higher the CR.

4.4 Experimental Results

4.4.1 Main Results and Ablation Study. Table 1 and 2 present the results of the algorithm under OPE testing and simulation testing. We test the algorithm's performance not only over the entire day but also during the two challenging peak periods within a single day. GS and CVnet did not undergo OPE testing as they are not stochastic algorithms and are not suitable for WIS evaluation.

From the results in the table, it can be observed that IPPO and MAPPO have many negative test outcomes, indicating that their performance is even inferior to that of the GS. This is due to the difficulty of performing accurate credit assignment with a single central reward signal in tasks with a large number of agents, resulting in policy training failure. The CVnet algorithm, by overcoming the shortsightedness of GS, achieves some performance improvement. However, the extent of its performance enhancement is far less than GRC and GRCS. In different cities, different peak periods, and tests on the real-world datasets and the sampled datasets, GRC and GRCS can consistently exceed other baselines.

In the ablation study, we found it is hard to determine the best one among the IL and the LC-lh. Their performance fluctuates with cities and peak periods. This is because it is challenging to ascertain the optimal h for optimization, as it is influenced by the local grid features and similarities of specific cities. The GRCS algorithm that only uses the goal-reaching loss also has consistently high performance, but it falls short in testing on sampled datasets in simulations. We surmise that this is due to the environment model overfitting the real-world dataset. Therefore, the policy guided by the goal state it generates also has certain overfitting results, causing its performance to be slightly inferior under a new sampled dataset. Neither of these two types of algorithms are sufficient to achieve higher performance on their own. For GRC, it utilizes the neighborhood range h that is the most effective in all-day IGMV in OPE test among LC-lh with $h \in \{1, 2, 3\}$ and IL ($h = 0$) in each city. It shows that GRC that combines both updating signals consistently achieves the highest performance in various testing comparisons.

4.4.2 Visualization and Comparison of Predicted States. We visualize and compare the predicted states from the environmental model $\mathcal{E}$. The compared states include the goal states (i.e., (b) and (f) in Fig. 5) and ordinary states (i.e., (c) and (g) in Fig. 5) of $\mathcal{E}$'s predictions for the behavior policy, and the ordinary states (i.e., (d) and (h) in Fig. 5) of $\mathcal{E}$'s predictions for the GRC optimized policy. We choose two easily interpretable metrics, namely demand-supply gap (DSG) and cancellation rate (CR), for the comparison, as shown in Fig. 5. From the figure, it can be observed that the real city state exhibits a clear imbalance in supply and demand, along with a high CR in certain regions. In contrast, the DSG in the goal state is more balanced, and the CR is lower, while the ordinary states are more chaotic and worse in the two metrics than the goal state. (d) and (h) represent the future states of $\mathcal{E}$'s predictions for the optimized policy. Compared to the real states, they have a more balanced DSG and a lower CR, thanks to the training guidance of the goal states.

4.4.3 Influence of the Goal State Horizons. In this section, we investigate the influence of goal state horizons on the policy optimization. We first trained an environmental model capable of predicting the states within the next 6 steps (i.e., an hour). We set the horizon of the goal state as i, where $i \in \{1, 2, 3, 4, 5, 6\}$. For different i, we sample and divide the states of the first i time steps predicted by the environmental model into goal states and ordinary states, and optimize the policy using loss function (1) with $h = 0$. The OPE results shown in Table 3 suggest that as the horizon increases, the performance firstly improves and then gradually declines. We hypothesize that this is because the 2-step prediction considers a longer-term goal state and optimizes the policy to reach it, thereby improving the policy. However, predictions for more distant goal states will have larger uncertainties, which weaken and offset the policy optimization. Therefore, the goal state horizon needs to be conservatively chosen, and large horizons should be avoided.

5 Related works

With the widespread popularity of online ride-hailing services, the order dispatching decisions of online ride-hailing platforms have aroused great interest among researchers. One line of works are specifically interested in theoretically analyzing the algorithms' worst-case performance, known as competitive ratios. To date, various algorithms have been proposed [1, 4, 7–9, 15] since Karp et al. formulated the online bipartite matching (OBM) problem in the 1990s. However, the OBM problems studied in these works are highly simplified, comparing to the OD problem. For instance, they often assume the nodes are one-sided online [4, 8, 15], the edge has no weights [7, 9], only one node appears in each time slot [1, 15], or there is no active departure of nodes in OBM [1, 7, 9, 15]. These simplifications have prevented these algorithms from being applied to real-world OD problems, to the best of our knowledge.

Another line of works exploit MARL to enable the distributed decision-making abilities of drivers [11, 23, 26, 30]. In these works, there is no central server or matching algorithm to gather and match the drivers and orders. Conversely, each driver makes his/her own decision in a distributed way to select orders in the surrounding region. While promising, such approaches are not applicable in the current online-hailing platforms, as the stability and sustainability of large-scale distributed systems still need to be verified.

Furthermore, recent works specified for OD in the online ride-hailing platforms are [22, 24, 25, 27]. The core idea of the series of works lies in learning and adding the values into edge weights to consider the future revenue of destination [27]. Later, [24] proposed to use spatial embeddings and deep models to predict such values, [25] additionally considered the fleet management problem, and [22] deployed online learning module to update the values in real time. However, these works can easily lead to insufficient supply in cold regions, resulting in revenue loss. The core reason found in this study is that the value setting of regions within the same time step is independent of each other. Instead, this work models the impact between the values of different regions from a cooperative game theory, and proposes the GRC algorithm to solve the game. The effectiveness of the algorithm has been confirmed by experimental results over various time periods in different cities.

Table 3: OPE results in the morning peak in city B when the goal state has different horizons.

Horizon of Goal (min)	IORR (%)	IGMV (%)
10	1.14±7.72	1.46±9.87
20	1.43±7.84	1.75±9.78
30	0.71±7.55	0.25±9.97
40	0.24±7.43	0.84±9.98
50	0.84±7.79	0.86±9.94
60	0.51±7.57	1.03±9.86

6 Conclusion

In this work, we rethink the recent works for OD and identifies their sub-optimality due to ignoring the impacts of the region values over other regions. Instead, we model such impacts with game and propose GRC to solve the game based on a novel goal-reaching credit assignment. Experiments validate GRC as it steadily outperforms the baseline algorithms across different cities and peak periods.

Acknowledgments

This work was supported by NSF China (No. U21A20519, U20A20181, 62372288, 62202298), and by DiDi GAIA Research Collaboration Plan.

References

[1] Mohammad Ali Alomrani, Reza Moravej, and Elias B Khalil. 2021. Deep policies for online bipartite matching: A reinforcement learning approach. *arXiv preprint arXiv:2109.10380* (2021).

[2] David Brandfonbrener, Will Whitney, Rajesh Ranganath, and Joan Bruna. 2021. Offline RL without Off-Policy Evaluation. In *NeurIPS*.

[3] 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).

[4] Matthew Fahrbach, Zhiyi Huang, Runzhou Tao, and Morteza Zadimoghaddam. 2022. Edge-weighted online bipartite matching. *J. ACM* (2022).

[5] Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. 2018. Counterfactual multi-agent policy gradients. In *AAAI*.

[6] David Gale and Lloyd S Shapley. 1962. College admissions and the stability of marriage. *The American Mathematical Monthly* (1962).

[7] Zhiyi Huang, Ning Kang, Zhihao Gavin Tang, Xiaowei Wu, Yuhao Zhang, and Xue Zhu. 2018. How to match when all vertices arrive online. In *STOC*.

[8] Zhiyi Huang, Zhihao Gavin Tang, Xiaowei Wu, and Yuhao Zhang. 2019. Online vertex-weighted bipartite matching: Beating 1-1/e with random arrivals. *ACM Transactions on Algorithms* (2019).

[9] Zhiyi Huang, Zhihao Gavin Tang, Xiaowei Wu, and Yuhao Zhang. 2020. Fully Online Matching II: Beating Ranking and Water-filling. In *FOCS*.

[10] Haiming Jin, Yifei Wei, Zhaoxing Yang, Zirui Liu, and Guiyun Fan. 2023. Multi-Intersection Management for Connected Autonomous Vehicles by Reinforcement Learning. In *ICDCS*.

[11] Jiarui Jin, Ming Zhou, Weinan Zhang, Minne Li, Zilong Guo, Zhiwei Qin, Yan Jiao, Xiaocheng Tang, Chenxi Wang, Jun Wang, Guobin Wu, and Jieping Ye. 2019. CoRide: Joint Order Dispatching and Fleet Management for Multi-Scale Ride-Hailing Platforms. In *CIKM*.

[12] R. M. Karp, U. V. Vazirani, and V. V. Vazirani. 1990. An Optimal Algorithm for On-Line Bipartite Matching. In *STOC*.

[13] Ilya Kostrikov, Ashvin Nair, and Sergey Levine. 2022. Offline Reinforcement Learning with Implicit Q-Learning. In *ICLR*.

[14] Sergey Levine, Aviral Kumar, G. Tucker, and Justin Fu. 2020. Offline Reinforcement Learning: Tutorial, Review, and Perspectives on Open Problems. *arXiv preprint arXiv:2005.01643* (2020).

[15] Pengfei Li, Jianyi Yang, and Shaolei Ren. 2023. Learning for Edge-Weighted Online Bipartite Matching with Robustness Guarantees. *arXiv preprint arXiv:2306.00172* (2023).

[16] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. 2017. Multi-agent actor-critic for mixed cooperative-competitive environments. In *NeurIPS*.

[17] James Munkres. 1957. Algorithms for the assignment and transportation problems. *J. Soc. Indust. Appl. Math.* (1957).

[18] OpenAI. 2023. *Introducing ChatGPT*. https://openai.com/blog/chatgpt Sep. 20.

[19] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. In *NeurIPS*.

[20] Doina Precup, Richard S. Sutton, and Satinder P. Singh. 2000. Eligibility Traces for Off-Policy Policy Evaluation. In *ICML*.

[21] 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* (2020).

[22] 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. In *KDD*.

[23] 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 *KDD*.

[24] 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 *KDD*.

[25] 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 *KDD*.

[26] Enshu Wang, Rong Ding, Zhaoxing Yang, Haiming Jin, Chenglin Miao, Lu Su, Fan Zhang, Chunming Qiao, and Xinbing Wang. 2022. Joint Charging and Relocation Recommendation for E-Taxi Drivers via Multi-Agent Mean Field Hierarchical Reinforcement Learning. *IEEE Transactions on Mobile Computing* (2022).

[27] 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 *KDD*.

[28] 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. In *NeurIPS*.

[29] Chen Zhong. 2023. Understanding of Didi's Trading Strategy: Driver Order Matching. https://mp.weixin.qq.com/s/i8IIaMYaFub0a9M9MtFgSw. [Online; accessed 7-Nov-2023].

[30] 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 *CIKM*.

A Extra Experiments

A.1 Comparisons of Different Matching Algorithms

The GRC framework is agnostic to the underlying matching algorithm, such that either KM or GS can be used to match drivers and passenger orders based on the output weights from a GRC policy. In this subsection, we investigate the differences in terms of GMV and pick-up distance when KM or GS is used separately. The results highlight the unique properties of KM and GS and shed light on the trade-offs to be considered for industrial deployment.

In particular, we also add penalties on the pick-up distance into the weights when KM is used, as commonly adopted in previous works [22, 24, 25, 27]: for each $c \in \{0.0, 0.5, 1.0, 2.0\}$, KM-$c$ assigns $\tau_o - c \cdot$ (pick-up distance) to each edge, where τ_o is defined in Sec. 2.2. The IGMV and pick-up distance comparisons are shown in Table 4, where IGMV is the improved GMV versus that of GS. We can observe that, KM-0.0 dominates others in IGMV but incurs the largest pick-up distance. With larger c, both the pick-up distance and IGMV of KM decreases heavily. GS achieves a moderate balance between the two objectives, which is preferable versus KM, as the latter requires an exhaustive search for the c to reach such balance. We further compares the probability for different ranks of drivers (the driver's rank is lower when the pickup distance is smaller) to pick up the passengers in Fig. 6. Notably, GS has the highest probability for matching the nearest driver to each passenger, while KM has lower probabilities and KM-0.0 performs worst.

Table 4: Comparisons on the IGMV and pick-up distance when GS or KM (with penalty) acts as the underlying matching algorithm for GRC.

Matching Alg.	IGMV (%)	Pick-up Distance (km)
GS	1.00±0.00	0.920±0.783
KM-0.0	3.78±0.76	1.903±0.768
KM-0.5	-0.34±0.81	0.923±0.786
KM-1.0	-0.22±0.78	0.919±0.783
KM-2.0	-0.15±0.77	0.912±0.778

B Implementation Details

In experiments, each local state has statistic information of five time steps, including the past three time steps and the same time step over the past two days. For each time step, the used statistic information is 25-d, where 17-d contains information like the number of orders, the number of idle drivers, the cancellations numbers and many others. The left 8-d is the grid id. The action is a scalar. The policy used is a 4-layer linear network, with hidden dimension 32 and relu activation function. The encoder and decoder of the environment

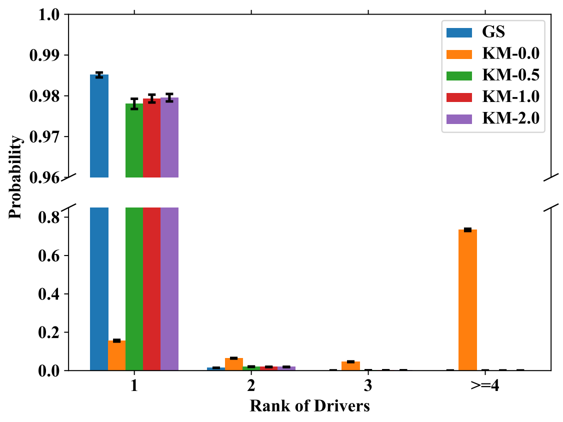

Figure 6: Comparisons on the probability of passengers being picked up by drivers at different distance ranks. The x-axis represents the rank of the drivers in terms of the pick-up distance, i.e., the less pickup distance, the lower the rank. The y-axis represents the probability of driver's rank in the matched driver-order pairs.

model both contain 3 GCN layers, with hidden dimension 256 and relu activation function. The scoring model also uses 3 GCN layers, with hidden dimension 256 and relu activation function. The batch size is 1024. The optimizer used is Adam, with learning rate 0.0003. The K, k used in two cities are 10, 2 and 5, 1, respectively.

C Comparisons on Demand Supply Gap

Following the suggestions of the reviewers, we reproduce a simplified version of the GRC algorithm on the public dataset of Chengdu city from November 2016, demonstrate the reduction effect of GRC on drivers concentration in hot regions, and verify this work's motivation. Code can be found at https://github.com/Zhaoxing1125/GRC.

Fig. 7 illustrates the comparison of the absolute value of the supply-demand difference in the city at different times. The horizontal axis represents ten intervals from low to high, showing the binned order numbers waiting to serve. It is clear from the figure that in areas with more orders (hot regions), the MDP algorithm (KDD-18) accumulates more vehicles than the direct KM algorithm, resulting in excessive drivers concentration. However, the GRC algorithm relatively reduces such drivers concentration. Fig. 8 divides the regions into cold and hot regions and plots the supply-demand difference curve evolving over time. It can be seen that the supply-demand difference of the GRC algorithm is reduced compared to the MDP algorithm in cold regions, and during peak hours in hot regions, the supply-demand difference with the GRC algorithm is also reduced.

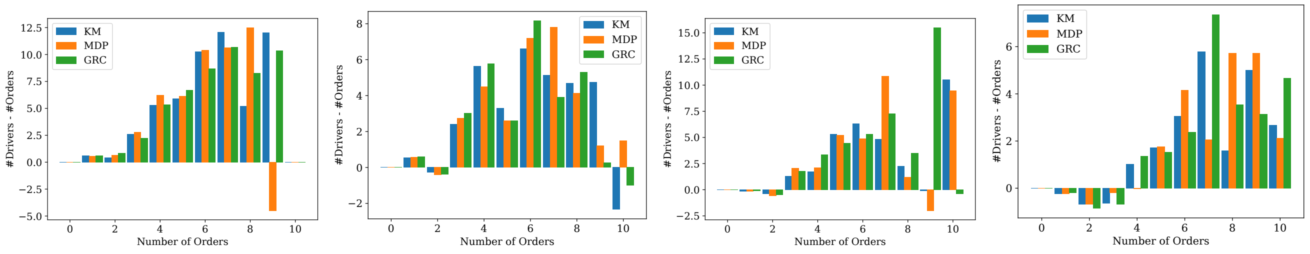

Figure 7: Absolute supply demand gap in 08:00, 08:30, 09:00, and 09:30.

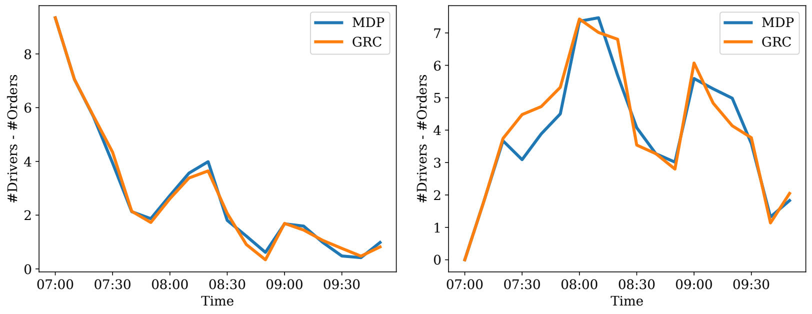

Figure 8: Absolute supply demand gap curve in cold regions and hot regions.