Tackling Sparsity in Designated Driver Dispatch with Multi-Agent Reinforcement Learning

Jiaxuan Jiang
Tsinghua University
Beijing, China
Shanghai Qi Zhi Institute
Shanghai, China
jjx21@mails.tsinghua.edu.cn

Ling Pan
Hong Kong University of Science and Technology
Hong Kong, China
lingpan@ust.hk

Lin Zhou
DiDi Chuxing Technology Co.
Beijing, China
realzhoulin@DiDiglobal.com

Longbo Huang
Tsinghua University
Beijing, China
longbohuang@tsinghua.edu.cn

Zhixuan Fang
Tsinghua University
Beijing, China
Shanghai Qi Zhi Institute
Shanghai, China
zfang@mail.tsinghua.edu.cn

ABSTRACT

Designated driving service is a fast-growing market that provides drivers to transport customers in their own cars. The key technical challenge in this business lies in the design of driver repositioning due to the far distances between drivers and orders, which is caused by complex moving constraints of drivers and the "hub-and-spoke" structure of orders. To address these challenges, this paper proposes Reinforcement Learning for Designated Driver Dispatch (RLD3), a Multi-Agent Reinforcement Learning (MARL) algorithm based on the Partially Observed Markov Decision Process (POMDP) formulation. Our algorithm considers group-sharing structures and frequent potential rewards with heterogeneous costs to achieve a trade-off between heterogeneity and sparsity. Additionally, our algorithm addresses long-term agent cross-effects through window-lasting policy ensembles. We also implement a simulator to train and evaluate our algorithm using real-world data. Extensive experiments demonstrate that our algorithm achieves superior performance compared to existing Deep Reinforcement Learning (DRL) and taxi-reposition methods.

KEYWORDS

Multi-agent reinforcement learning; Fleet management

ACM Reference Format:
Jiaxuan Jiang, Ling Pan, Lin Zhou, Longbo Huang, and Zhixuan Fang. 2025. Tackling Sparsity in Designated Driver Dispatch with Multi-Agent Reinforcement Learning. In *Proc. of the 24th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2025), Detroit, Michigan, USA, May 19 – 23, 2025*, IFAAMAS, 10 pages.

Corresponding authors: Zhixuan Fang (zfang@mail.tsinghua.edu.cn).

Proc. of the 24th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2025), Y. Vorobeychik, S. Das, A. Nowé (eds.), May 19 – 23, 2025, Detroit, Michigan, USA. © 2025 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org).

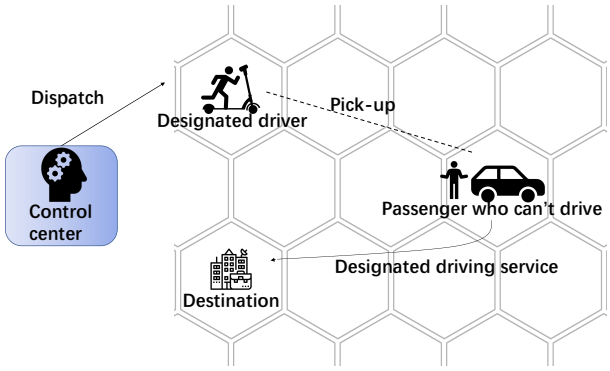

Figure 1: Designated driving.

1 INTRODUCTION

Designated driving, also known as chauffeur service, represents an emerging online platform service wherein professional drivers transport customers who are unable to drive for various reasons, such as intoxication or inexperience. In this service model, the designated driver arrives on an electric scooter and subsequently drives the customer's vehicle to the desired destination, as illustrated in Figure 1. The platform manages the driver dispatching behaviors to improve customers' experience and drivers' income. Designated driving has already become a significant and promising industry, with a market size of over 4 billion in China [4].

As such, one of the critical challenges in this industry is the design of driver dispatch, including matching and repositioning. While typical ride-hailing platforms focus on improving the matching quality between drivers and customers, designated driving platforms still struggle to find a driver for each order. Due to the "hub-and-spoke" pattern of orders, orders' origins are clustered in hot spots such as bars and restaurants, while most destinations are residential areas. After completing orders, the distribution of drivers becomes dispersed and sparse, leading to far distances from potential customers. Thus, repositioning drivers to reduce the future pick-up time is an urgent matter for designated driving.

While optimization methods exist for repositioning problems [32, 41], they require complex modeling of supply and demand dynamics. Recent approaches using Deep Reinforcement Learning (DRL) have made strides in ride-hailing platforms [1, 6, 23, 24, 27, 30, 35, 43, 44]. However, designated driving introduces unique challenges for DRL applications, particularly due to three factors related to sparsity:

- Dataset Sparsity: The number of designated drivers is significantly lower than that of taxi drivers, with a data collection from Hangzhou revealing only around 4,000 designated drivers serving over 8,000 km^2 area.
- Feedback Sparsity: Individual drivers receive limited feedback on order matches, averaging only four per day.
- Agent Interaction Sparsity: The slow movement and long-lasting effects of driver actions necessitate focusing on cumulative interactions.

Additionally, the heterogeneity and scalability among drivers further complicate traditional Multi-Agent Reinforcement Learning (MARL) applications.

This paper proposes a group-sharing window-lasting Reinforcement Learning framework, Reinforcement Learning for Designated Driver Dispatch (RLD3), for designated driver dispatch challenges. By modeling the problem as a Decentralized Partially Observed Markov Decision Process (Dec-POMDP), RLD3 addresses the specific observations of drivers. Key features of RLD3 include:

- Group-sharing Structure: Agents grouped by sharing parameters and experience, balancing sparsity, scalability, and heterogeneity.
- Reward Design: A reward structure reflecting instant order match and long-term order potentials while accommodating diverse movement constraints.
- Window-lasting Interaction: A cumulative action calculation over execution periods to capture agent interactions and decision-making adequately.

We developed a simulator using real-world datasets and conducted extensive experiments, demonstrating that RLD3 outperforms existing Deep Reinforcement Learning (DRL) benchmarks and optimization strategies in completed orders and adherence to movement constraints.

The main contributions of this paper are summarized as follows:

- To the best of our knowledge, we are the first to model the repositioning problem of designated drivers in designated driving markets.
- We propose a novel MARL algorithm, RLD3, to address the sparsity challenges of designated driver repositioning, which builds upon group-sharing structures and window-lasting agent interactions with a potential/cost-aware reward.
- We design a designated driving simulator using real-world datasets and conduct extensive experiments. RLD3 efficiently learns system dynamics and outperforms existing DRL and optimization methods.

2 RELATED WORK

2.1 Fleet Management

Fleet management, also known as the driver reposition problem, has been extensively studied using optimization algorithms [9, 32, 38, 42] and DRL-based techniques [1, 6, 10, 14, 16, 20, 23, 24, 27, 30, 35, 36, 43, 44]. Optimization methods leverage historical data for policy formulation but struggle with real-world demand-supply dynamics. Conversely, DRL algorithms can optimize long-term effects without strong assumptions.

Several DRL studies focus on scalability, heterogeneity, and partial observability. MIX-H [14] uses multi-level controllers for scalability and optimizes joint actions. Deep-dispatching [23] balances local observation and global information, addressing partial observation sparsity. MAC-PPO [20] tailors task scheduling and charging for heterogeneous vehicle types using the MAPPO algorithm.

However, taxi platforms' fast-moving and denser orders reduce sparsity challenges, making direct application to designated driving less effective.

2.2 Multi-Agent Reinforcement Learning

MARL techniques address complex multi-agent problems. MADDPG [25], extending DDPG [21, 22, 28, 29], uses deep networks to approximate action values and manage agent interactions. MAPPO [40], based on the gradient policy improvement of PPO [33], further enhances the sample complexity and performance of multi-agent learning. Despite the benefits within the CTDE paradigm [5], it faces challenges with sparse feedback and agent heterogeneity in designated driving, reducing exploration and learning efficiency.

To enhance exploration, Random Network Distillation (RND) [3] uses an additional value function to estimate intrinsic reward to enhance exploration. In the designated driving platform, due to the unique "hub-and-spoke" structure of orders, the hotspots of orders are more concentrated. Exploring non-semantic information would result in excessive driver movement costs. Curriculum Learning approaches, such as Curriculum Deep Reinforcement Learning [11] and Relevant Curriculum Reinforcement Learning [7], help in learning from sparse feedback by planning the neural network's learning path. However, planning learning paths in multi-agent scenarios is challenging due to the complex dynamics of cooperation and competition among drivers. Mean-Field Reinforcement Learning techniques, such as Mean Field Multi-Agent Reinforcement Learning (MAMFRL) [39] and Multi-Agent Mean Field Q-Learning [8], model agent interactions as the interaction between a single agent and a field effect. Mean-field methods can address the issue of sparse agent distributions but lack consideration for the lasting interaction of different drivers, which should be taken into account since designated drivers have slow movement and complex constraints.

HRL approaches, including Feudal HRL [37], Data-Efficient HRL [26], and Model-Free HRL [31], help tackle scalability and heterogeneity by decomposing problems into sub-agents. However, they must account for complex agent interactions and sparsity issues specific to designated driver reposition applications.

3 RLD3: REINFORCEMENT LEARNING FOR DESIGNATED DRIVER DISPATCH

We consider the designated driving service in one metropolis. There are N drivers on the platform, and orders arrive at the platform to be served. The platform tries to match each order with a driver. Meanwhile, the platform sends movement instructions to relocate idle drivers, aiming to decrease future pick-up time. In the following,

we will first introduce the Decentralized Partially Observed Markov Decision Process (Dec-POMDP) model on driver reposition, and then three unique designs of our algorithm: the grouped structure, the potential reward, and the lasting agent interaction.

3.1 Formulation

We consider discrete time intervals, with each time step representing 30 seconds. Then we divide the metropolis into hexagonal grids, each with a radius of about 180 meters, which is the moving distance of an electric scooter in one time step. At each time step, the platform decides the reposition movement for every idle driver. Due to the limited patience of passengers in the real world, we assume that all orders have thresholds for the waiting time. The order will be canceled if it remains unmatched after the waiting time threshold.

The detailed Dec-POMDP formulation $\langle N, \mathcal{S}, \mathcal{O}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$ is elaborated as follows:

- **Agent** $i \in [N]$: Each driver is considered an agent, resulting in a total of N unique agents. The platform can only reposition idle drivers, as each agent can be in one of three statuses: offline, idle, or working at any given time t.
- **State** $s \in \mathcal{S}$: At each time t, a global state is maintained to encapsulate the statuses of all drivers and orders, including drivers' coordinates, movement distances, work statuses, serving targets, and intended movement trajectories. The state also includes calling time, patience, origin, destination, and serving status for orders.
- **Observation** $s \mapsto_i o_i \in \mathcal{O}$: Drivers have partial observations of the state s. In our implementation, each agent's observation is represented by a 22-dimensional vector:

$$([\#\boldsymbol{order}], [\#\boldsymbol{driver}], [\min \boldsymbol{dist}], t, lat, lng, move), \tag{1}$$

  where the first three terms denote the number of orders to be matched, the number of idle drivers, and the distance to the closest order in six-segment-direction neighborhoods as shown in Figure 2. The last four terms represent current time, latitude, longitude, and the distance already moved.
- **Action** $a_1 \times \cdots \times a_N \in \mathcal{A}$: The platform proposes a joint action instructing the movement policy for all available drivers based on their observations o^t at time t. We assume that idling drivers fully comply with movement instructions. The action space for an individual agent consists of seven discrete actions including moving to six neighboring directions and staying at the current location as shown in Figure 2. Agents located at the boundary and corners of the map have a smaller action space.
- **State Transition** $\mathcal{P} : s \times \boldsymbol{a}_{[N]} \mapsto s'$: The movement of drivers, along with order updates and matches between drivers and orders, induces state transitions in the environment.
- **Reward** $r_i \in \mathcal{R}$: Upon executing an action, each agent receives an immediate distinct reward r_i. The reward at time step t, r_t^i, is calculated by combining three components: the immediate match reward, the move cost, and the neighborhood potential reward, i.e.,

$$r_i^t = mt_i^t + mv_i^t + nb_i^t. \tag{2}$$

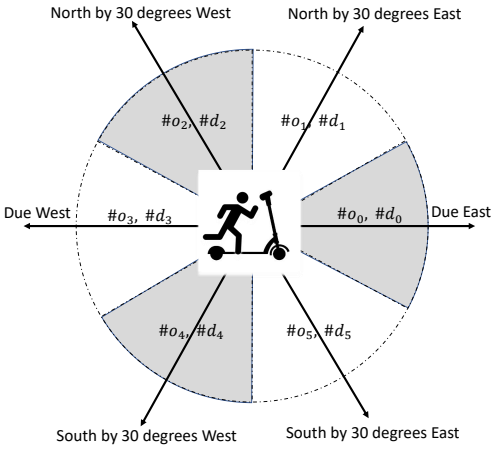

Figure 2: The six action directions and six segments of the neighborhood.

The immediate match reward mt_i^t directly relates to the gross merchandise volume of the platform, which is the objective of our algorithm. To optimize volume without using discriminatory personal information, the immediate match reward is set to a fixed number:

$$mt_i^t = \begin{cases} 50, & \text{if agent } i \text{ is matched with an order at } t; \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

To control drivers' moving distance, we include the move cost mv_i^t as a regularizer that influences the behavior of agents in the reward. Previous empirical study (e.g., [2, 20, 34]) indicates that moving costs of drivers can be divided into several types due to differences in their electric scooters' physical characteristics and individual preferences regarding motion during idle periods. We categorize the move cost into five types, varying from "very low" to "very high", i.e., $c_1 < c_2 < \ldots < c_5$. Every agent has a corresponding type. Thus, the move cost for agent i at time t is set as follows:

$$mv_i^t = \begin{cases} -c_j, & \text{if agent } i \text{ moves;} \\ 0, & \text{if agent } i \text{ stays;} \end{cases} \tag{4}$$

where j is the type index of agent i. In this work, the single-step cost c_j is set as $c_j = 0.5 \times (j+1)$.
Details of the neighborhood potential reward nb_i^t will be introduced later in Section 3.3, as a solution towards feedback sparsity.

3.2 Towards Dataset Sparsity: Group Sharing

To mitigate issues related to sparse data for individual agents and limited feedback regarding matches, we introduce group-sharing structures in our methodology.

As introduced in Section 3.1, the agents are classified into 5 types based on the cost condition, and a natural idea is to place agents of the same type into the same group since they share the same

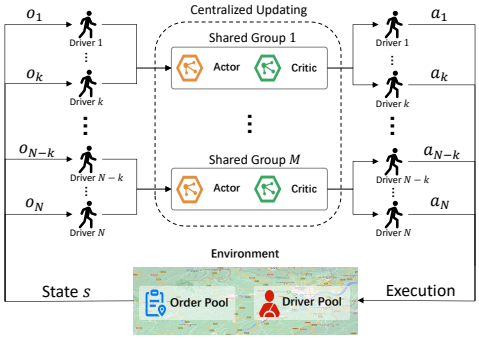

Figure 3: Information flow in the execution stage.

reward function, thereby making their data distributions similar. Based on the concept of grouping, agents within the group share the same network along with their experience data in the training process. Specifically, we divide the N agents into M groups, where M is a fixed number. The number of groups M is a hyper-parameter of the algorithm, and intuitively, the number of groups needs to be no less than the number of types to ensure that the algorithm captures the heterogeneity of different agents in the system.

It is important to emphasize that agent grouping in this algorithm pertains to the design of the neural network structure and is decoupled from the agents' inherent type attributes. Since the number of groups is determined by the heterogeneity of agents, the number of neural networks in the system does not increase linearly with the number of agents. Consequently, RLD3 avoids scalability issues such as value function dimensionality explosion and coordination challenges due to a slow convergence of too many networks during large-scale training.

Under the group sharing structure, RLD3 utilizes double critic-networks and double actor-networks for every group, with the delayed copy used for soft-update. During the training stage, a group network can access the experienced data of all agents belonging to that group, stored in a replay buffer. Therefore, a network can efficiently explore different individuals of the same group in the metropolis and gather more experiences. During the execution stage, each agent calls its corresponding group network to perform policy execution independently. The policy input for each agent is based on its current observation while the output is its deterministic action. To transform the continuous output seven-dimensional vector into a discrete action, the last layer uses Gumble-Softmax [15]. Such mixed strategy ensures that even agents of the same group at the same location may execute different discrete actions, avoiding competition among agents. The information flow during the execution stage is illustrated in Figure 3.

Meanwhile, since agents are grouped, we estimate the influence between these groups using the mean-field effect to deal with agent interactions. Specifically, we calculate the overall impact of all agents in group j on the environment at time step t as g_j^t, which will be introduced in detail in Section 3.4.

3.3 Towards Feedback Sparsity: Space Potential

Since the immediate match reward is highly sparse for the DRL method in designated driving platforms (i.e., it only occurs at the time step with a successful order match, which is rare), we introduce a dense neighborhood potential reward nb_i^t to reflect the potential value of the current area. The intuition is that the distance to an order in the neighborhood reflects how fast an agent can pick up the order. Almost all orders in the neighborhood are attractive to the driver, although the closest ones are especially attractive.

Specifically, we assign potential values to nearby unmatched orders, with higher feedback given to closer orders. We then sum up all potential values to represent the total potential value of the driver's current position. This provides reward feedback to the driver at every time step, compensating for the sparse immediate match reward. The potential reward is defined as follows:

$$nb_i^t = (d^* + 0.1)^{-0.5} + 0.1 \times \sum_{\text{neighbor order } j} (d_{ij} + 0.1)^{-0.5}, \quad (5)$$

where d_{ij} denotes the distance from driver i to order j, and d^* denotes the distance to the closest order. The power index is set to -0.5 to ensure that the potential reward increases as the distance approaches and is a convex function, in order to encourage designated drivers to approach a specific order rather than maintain an equal distance from all orders. It is important to note that the potential reward is merely a complement to the sparse match reward, and its scale is far smaller than the match reward. During training, since the optimization objective of the reinforcement learning algorithm is the cumulative discounted reward, the algorithm will still maximize the order completion rate and, at the same time, make better allocations for the positions of idle drivers.

3.4 Towards Interaction Sparsity: Window Lasting

In designated driver platforms, agents are frequently distanced from one another, leading to sparse, long-term interactions. For example, a driver's income is not directly influenced by the actions of drivers located far away, but rather by the accumulated distribution changes caused by the lasting movements of other drivers. Therefore, we introduce the average action over a time window, instead of a single-step action, when considering other agents' policies.

To achieve this, in addition to recording regular singe-step tuples $(s, \boldsymbol{a}, \boldsymbol{r}, s')$, the buffer calculates and stores the window-lasting actions for all agents. The window-lasting action $\hat{a}_i$ represents the average of sequential actions for the last W time steps within a single idle period:

$$\hat{a}_i^t = \mathbb{E}\left[a_i^s\right], s \sim [t - W, t] \cap T_{\text{last idle}}, \quad (6)$$

where $T_{\text{last idle}}$ refers to the most recent period in which the driver was idling, considering possible different idle periods that may result in diverse moving directions. $\hat{a}_i$ signifies the general movement direction of agent i during this idle period after completing the last order. Thus, the mean-field effect for group j is defined as:

$$g_j^t = \mathbb{E}_{i \in \text{group } j}\left[\hat{a}_i^t\right]. \quad (7)$$

Similar to the group number M, the window length W is also an important hyper-parameter of the algorithm. To best represent the general movement direction of idle drivers between serving

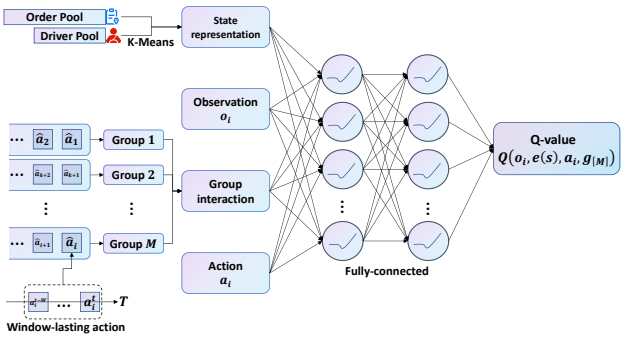

Figure 4: Network structure.

two orders, W should be set to the average time a driver is idling between two consecutive matches, which is 60 (i.e., 30 minutes in real-world simulation) in our work.

Additionally, we use an encoder in the input of the critic to handle complex state representations and their varying dimensions. This encoder is responsible for the distribution of the current unmatched orders and idle drivers respectively. We employ the K-Means algorithm [12] for this encoder. Such an encoder performs clustering on the coordinates of unmatched orders and idle drivers, respectively, and uses the centroid coordinates of the clusters to represent the hot zones of the current distribution of orders and drivers. The encoder converts the variable-dimensional, complex state vectors into a concise and effective fixed-length vector representation. The number of clusters is also a hyper-parameter of the algorithm, which depicts the number of hot spots in the distribution of drivers and orders on the map. As the map size increases, the number of clusters also needs to be set larger.

Therefore, the input of value function network is denoted as $Q_j\left(o_i, encode(s), a_i, \boldsymbol{g_{[M]}}\right)$, where $\boldsymbol{g_{[M]}}$ denotes the concatenated vector of the mean-field effect $g_j (1 \le j \le M)$ for all groups. The detailed value-function structure for every group is shown in Figure 4. All networks utilize two fully connected layers and the GELU activation function [13].

3.5 Network Update

The complete framework is summarized in Algorithm 1. The network update follows the gradient-based actor-critic paradigm. To encourage policy exploration, enabling the value function to incorporate adjustments in learning strategies for better adaptation to changes in the environment, we adopt a policy exploration-based Bellman update rather than a greedy Bellman update. Thus, the loss function for the value network becomes:

$$\begin{aligned} \mathcal{L}(\theta_j) =& \mathbb{E}_{sample_j^t}\left[\left(Q_j^{\pi}\left(o_i, encode(s), a_i, \boldsymbol{g_{[M]}}\right) - y\right)^2\right], \\ y =& r_i + \gamma Q_j^{\pi'}\left(o_i', encode(s'), a_i', \boldsymbol{g'_{[M]}}\right), \end{aligned} \tag{8}$$

where the $sample_j^t$ is a batch of data sampled from the corresponding replay buffer of the updating group network j, and each data comes from all drivers that belong to group j.

Due to the partially observed property and our critic structure, the gradient of the policy network is:

$$\begin{aligned} \nabla_{\theta_j} J(\pi_j) =& \mathbb{E}_{sample_i^t}\Big[\nabla_{\theta_j}\pi_j(a_i \mid o_i) \\ & \nabla_{a_i} Q_j^{\pi}\left(o_i, encode(s), a_i, \boldsymbol{g_{[M]}}\right)\Big|_{a_i=\pi_j(o_i)}\Big]. \end{aligned} \tag{9}$$

Algorithm 1 Reinforcement Learning for Designated Driver Dispatch (RLD3).

Require: order data, driver pool $[N]$, episode number MAX, episode length T, learning rate λ, update rate τ, batch size S, group number M, window length W.

for episode from 1 to MAX **do**
  Initialize environment and receive an initial state s.
  for t from 1 to T and not all drivers are off-line **do**
    Generate action $a_i = \pi_i(o_i)$.
    Execute action $(a_1, a_2, \cdots, a_N)$ and observe reward $\boldsymbol{r}$ and next state s'.
    Push $(s, \boldsymbol{a}, \boldsymbol{r}, s', \hat{\boldsymbol{a}})$ into buffer.
    $s = s'$.
    for group j from 1 to M **do**
      Sample a batch of size S: $(s, o_i, a_i, r_i, s', \hat{\boldsymbol{a}})(i \in \text{group } j)$ from replay buffer.
      Update critic by minimizing $\mathcal{L}(\theta_j)$.
      Update actor using sample policy gradient $\nabla_{\theta_j} J$.
    end for
    Update the target network parameter for each agent i by $\theta_i' = \tau\theta_i + (1-\tau)\theta_i'$.
  end for
end for

4 SIMULATOR

We conduct experiments to evaluate RLD3 using a simulator and real-world data. The simulator models the whole process of how states of drivers and orders evolve on a designated driving platform. Specifically, the simulator includes a driver dispatch module that is available to reposition any idling driver. Therefore, the simulator serves as a training environment for RL algorithms and can evaluate the performance of various dispatch policies. We describe the environment simulator as follows:

4.1 Real-World Dataset

The simulator is built on a real-world dataset. The real-world data is obtained, from Hangzhou, China. The data includes over 4,000 drivers and 16,000 orders per day from Feb 21 to Feb 28, 2022. Each order's information consists of the coordinates and the time of its events including generation, match, completion, and cancellation. Each driver's information consists of log-in time, log-out time, and online coordinates. The data collection process does not include personal information about drivers and orders. To prevent the leakage of driver or passenger behavior patterns, we also utilize virtual IDs. In every experiment episode, we sample a fixed number of orders from the dataset to simulate the order distribution.

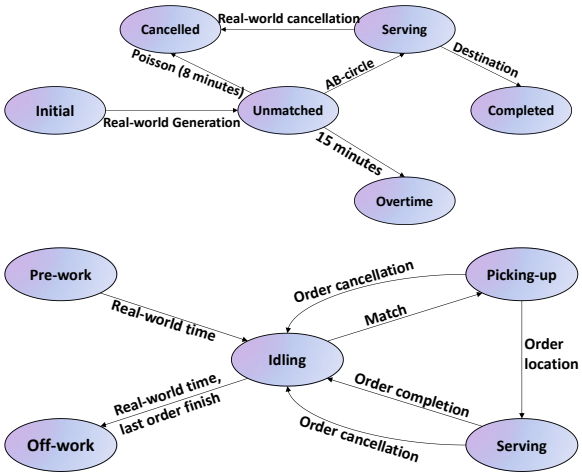

Figure 5: Order(above) and driver(below) state transfer.

4.2 Order State Transfer

For the order process, every appearing order enters the order pool and waits to be matched at a predetermined real-world generation time. During the waiting period, if an unmatched order is not answered within a specified period, it enters the overtime state and fails. The overall timeout is set to 15 minutes, as in real-world applications. In addition to the timeout limit, each order also has its own patience time. Since the cancellation time of canceled orders in the dataset follows an exponential distribution with a mean of 8 minutes, we assume that all orders cancel following a Poisson process with a mean cancellation time of 8 minutes after known real-world patience. However, if an order is canceled by the customer in the real world, the patience of the order will be set to the actual real-world value. The whole state transfer for orders is illustrated in Figure 5.

4.3 Driver State Transfer

Each driver has a scheduled on-work and off-work time. When the current simulation time exceeds the online time, the driver enters the idle state from its actual location in the real world. Idling drivers can either move to a given location according to the dispatch policy or match with an order for service. When an idle driver matches an order, the driver immediately goes to pick up the customer. Once the order is completed, the driver returns to the idle status until the simulation time exceeds their offline time. The whole state transfer for drivers is illustrated in Figure 5.

4.4 Matching Module

The simulator applies a two-step driver-searching algorithm, the AB-circle algorithm, to match drivers and nearby orders. The algorithm is intuitive and is deployed in Hangzhou. At each time step, for orders around which there are idling drivers within the A-circle (with a radius of 3000 meters), the algorithm assigns the order to the closest driver to minimize pick-up time. After matching all such orders, the algorithm calculates a global optimal match between orders and drivers within the B-circle (with a radius of 5000 meters). The optimal match is calculated using the Kuhn-Munkres algorithm [19] in a bipartite graph. The matching process in a single time step is illustrated in Algorithm 2.

Algorithm 2 AB-circle matching algorithm.

Require: Unmatched order pool, idling driver pool.
for order i sorted by generation time **do**
  if there exist drivers within A circle of i **then**
    Match i with the closest driver.
  end if
end for
Construct a bipartite graph G between unmatched orders and idling drivers till then.
for order i in G **do**
  for drivers j in G **do**
    if j is in B circle of i **then**
      Connect i and j with an edge weighted by the negative distance.
    end if
  end for
end for
Run Kuhn-Munkres algorithm to get the maximal match between orders and drivers in G.

5 EXPERIMENT

We conduct experiments based on the proposed simulator and real-world data. Due to the slow execution of most MARL algorithms in large-scale scenarios, we sample an area of approximately 600 km^2, simulating 50 drivers and 500 orders. This area is roughly equivalent to a driver's daily operational range. In real-world applications, we can similarly divide the entire metropolis into hexagonal sections, each approximately 600 km^2, to solve for the locally optimal strategy in each section.

We first present the training process of our model and then compare the performance with other existing methods. Next, we conduct an ablation study to validate the ability of different modules. At last, we study the scalability of our algorithm. Each experiment is repeated with 4 different seeds on a single NVIDIA A100/A40 GPU, and the average result is presented. All algorithms were trained for 1000 episodes, with the first 100 episodes dedicated to exploration.

5.1 Performance Comparison

We compare our algorithm against existing DRL methods, taxi reposition strategies, and optimization policies. Every episode starts at 11:00 and ends at 20:00 when all orders are completed or canceled. Eligible orders are sampled from the dataset per episode.

Benchmark algorithms include independent DDPG [22], MADDPG [25], MAMFRL [39], RND [3], and MAPPO [40]. We also include deep-dispatching [23] and VPS [17] for taxi dispatch comparisons. All DRL algorithms are configured uniformly regarding hidden layers, update intervals, and learning rates. To reveal the performance differences of the algorithm itself, all DRL algorithms use the same hyper-parameters including optimizer, learning rate, update rate, update interval, batch size, hidden dimension, and exploration episode number.

	Algorithm	Testing performance		IID generalization	
		Order ↑	Distance ↓(km)	Order ↑	Distance ↓(km)
Our Algorithm	RLD3	**237.2 ± 3.4**	**7.0 ± 1.3**	**234.0 ± 3.9**	**7.0 ± 1.3**
Taxi-dispatch Algorithm	VPS [17]	232.1 ± 5.7	37.8 ± 7.7	229.1 ± 5.8	38.2 ± 7.6
	Deep-dispatch [23]	230.3 ± 2.3	15.5 ± 2.4	229.0 ± 2.3	15.1 ± 2.3
DRL-based	DDPG [22]	186.9 ± 5.2	27.8 ± 3.0	183.1 ± 5.5	27.9 ± 3.1
	MADDPG [25]	215.7 ± 3.7	29.6 ± 0.6	212.0 ± 4.4	30.2 ± 0.5
	MADDPG-RND [3]	228.6 ± 3.5	65.3 ± 0.7	224.0 ± 3.7	66.3 ± 0.9
	MAMFRL [39]	224.3 ± 3.7	34.3 ± 5.3	221.1 ± 4.8	34.9 ± 5.5
	MAPPO [40]	229.0 ± 3.5	63.1 ± 0.8	225.4 ± 4.5	61.4 ± 10.1
Optimization-based	Myopic-dispatch	229.8	73.2	228.8	73.1
	Myopic-dispatch with PO	225.4	29.1	221.8	29.3

Table 1: Testing performance. The testing performance is evaluated for 10 episodes based on the model that has the best episodic performance, and the results are averaged over the 10 episodes and four seeds. The IID generalization performance is measured using an additional testing dataset of 10 episodes that are separated from the training dataset.

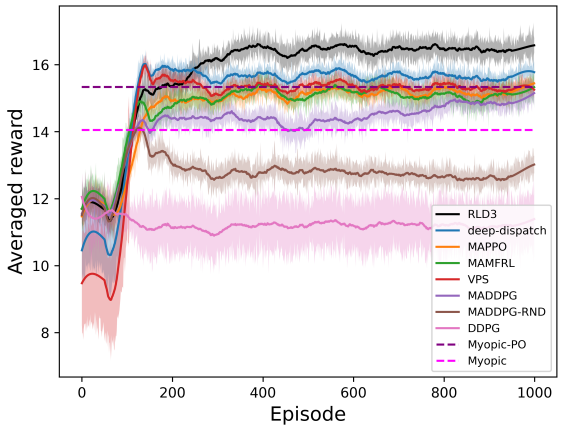

Figure 6: Training performance. The average results across four seeds along with the confidence intervals are presented.

As shown in Figure 6 and Table 1, our model outperforms all other algorithms in terms of the number of completed orders and moving distance. RND and MAPPO fall into no-semantic exploration due to always moving; MADDPG and MAMFRL fail to differentiate the value of different directions when there are no nearby orders, resulting in a significant amount of random walking. Although deep-dispatching addresses the problem of partial observability by combining local and global information, the algorithm is constrained by data sparsity, making it difficult to handle the instability of dynamic agent interactions. While VPS encourages exploration and stability by incorporating an additional value function based on global states, similar to RND, it faces challenges related to excess exploration and instability during later stages of training due to policy changes. For optimization baselines, the myopic driver repositioning policy optimizes the picking-up distances by treating the drivers as servers, which in turn leads to intense competition among drivers for orders. Partial observability helps alleviate this competition, but it may lead to a decrease in optimization effectiveness.

5.2 IID Generalization

We conducted IID Generalization experiments to assess robustness by assuming training and testing data originate from the same distribution [18]. We sampled 500 unseen orders per episode. As shown in Table 1, our algorithm does not decline significantly in IID performance and still outperforms other methods. An interesting phenomenon is that all algorithms demonstrate good IID generalization performance. This is because the designated driving platform itself exhibits sparsity, and the hotspots of orders are concentrated. Since we maintain the same initial state for all drivers and the same order underlying distribution in the IID generalization test, drivers are still able to effectively transfer the learned hotspot information from previous experiences when moving.

5.3 Scalability Evaluation

To test scalability, we proportionally increased the simulated city area and the number of drivers while experimenting with different scales of cluster numbers. We measured the average reward of agents to evaluate the algorithm's performance across different scales. As shown in Figure 7, RLD3 exhibits superlinear performance in large-scale experiments, as larger numbers of agents enable greater diversity and more data sharing when fixing the group number and cluster size. In addition, increasing the number of clusters can enhance the state encoder to summarize the distribution of orders and drivers in large-scale scenarios, which allows for a more accurate description of the current distribution of hot zones, leading to improved performance.

5.4 Ablation Study

Group Number. The group number is a typical hyper-parameter that determines the number of shared networks. A larger group number better represents the heterogeneity, but also increases the

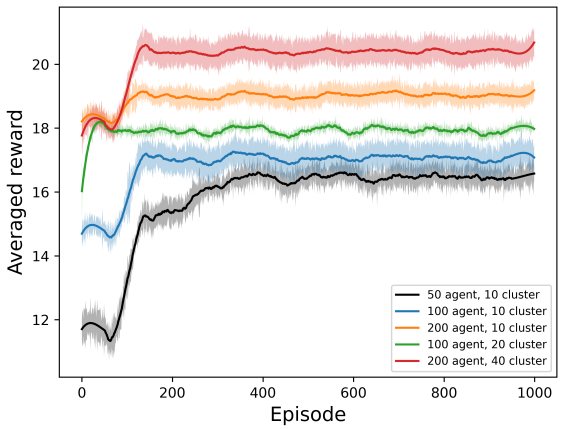

Figure 7: Scalability Evaluation. The figure presents the average reward performance per agent, demonstrating the super-linear improvement of RLD3 in large-scale scenarios.

Algorithm	Order ↑	Distance ↓
RLD3 (5 groups, 60 window length)	**237.2 ± 3.4**	7.0 ± 1.3
RLD3 for 1 group	150.5 ± 9.2	21.5 ± 1.2
RLD3 for 10 groups	236.4 ± 3.6	7.4 ± 0.3
RLD3 for 50 groups	231.7 ± 3.6	9.1 ± 0.4
RLD3 for 10 groups (3:7 uneven)	235.3 ± 3.9	10.1 ± 1.3
MADDPG for 5 groups	223.0 ± 3.6	24.5 ± 1.2
MADDPG-RND for 5 groups	211.5 ± 7.4	56.1 ± 1.8
MAMFRL for 5 groups	227.0 ± 10.2	6.1 ± 1.7
RLD3 without window-lasting	229.5 ± 3.2	27.1 ± 3.4
RLD3 for 30 window length	236.3 ± 4.5	7.8 ± 0.9
RLD3 for 120 window length	231.7 ± 3.6	9.1 ± 0.4
RLD3 without state encoder	232.2 ± 3.7	27.2 ± 1.6
RLD3 without potential reward	223.8 ± 3.8	6.7 ± 1.2

Table 2: Ablation study. The averaged results over 10 testing episodes and four seeds are presented.

storage pressure and training time. Additionally, a large group number may not learn well in sparse feedback situations. The results in Table 2 show that the group-sharing structure helps improve the performance of MADDPG and RLD3. We further investigated the performance of RLD3 under uneven group distributions. Experimental results demonstrated that uneven group allocation leads to marginal performance degradation and increased variance, primarily attributable to training data quantity disparities.

Window-lasting Agent Interaction. Our algorithm uses a window-lasting policy ensemble to better learn the cross-effects of other agents' policies. We evaluated the algorithm without the window average. As shown in Table 2, removing window-averaged interaction impairs learning due to fluctuating agent actions. The optimal window length aids in capturing idle movements, with performance deterioration observed at extreme lengths. In this experiment, the window length is approximately equivalent to the average time between two consecutive matched orders for drivers, and thus the window-averaged action effectively captures the direction of idle movement for the driver during this period. In practical applications, the window length should also be selected based on the average time for a single instance of idle movement for the agent.

State Encoder. To capture the distribution information of orders and drivers during the training stage, we employ an encoder to encode the system's state. The dimensions of the state vector are constantly changing, making it difficult to directly utilize the value function. Therefore, we extract the distribution information of orders and drivers separately using the K-Means method. As shown in Table 2, the K-Means state encoder can assist DRL algorithms in better understanding the state of the designated driving platform, particularly in extracting driver-order distribution information. Additionally, even without state encoders, our algorithm still outperforms others due to the benefits of group-sharing and window-lasting interaction techniques.

Reward Design. We compared different reward components by removing the neighborhood potential reward and move cost, as shown in Table 2. All reward settings were tested with our proposed group-sharing structure and training process. The dense potential reward not only increases performance but also stabilizes the training process, as indicated by the much smaller value function loss. The model without the cost falls into a sub-optimal situation where only order numbers are optimized, ignoring distance constraints.

6 CONCLUSION

In this paper, we tackled the driver dispatch problem in designated driving platforms characterized by sparsity and strict constraints. We proposed a multi-agent deep reinforcement learning (DRL) algorithm formulated as a decentralized partially observed Markov decision process (Dec-POMDP). Our approach leverages a group-sharing structure and a specially designed reward to address sparsity issues and employs a window-lasting agent interaction technique to manage long-lasting agent interactions.

Extensive experiments using a real-world data-based simulator demonstrated that our algorithm outperforms traditional taxi-reposition policies and existing DRL methods in completed order numbers and adherence to moving constraints. Moreover, our algorithm exhibits efficient scalability, underscoring its effectiveness in the designated driver dispatch problem.

Future work will explore self-supervised group forming algorithms like clustering, enhancing agent interaction modeling and algorithm performance. Additionally, we plan to investigate the impact of driver non-compliance on dispatch performance, aiming to improve the real-world applicability of our algorithm.

ACKNOWLEDGMENTS

The work of Jiaxuan Jiang and Zhixuan Fang is supported by Tsinghua University Dushi Program, Shanghai Qi Zhi Institute Innovation Program SQZ202312, and CCF-DiDi GAIA Collaborative Research Funds for Young Scholars. The work of Longbo Huang was support by the National Natural Science Foundation of China Grants 52450016 and 52494974.

REFERENCES

[1] Lina Al-Kanj, Juliana Nascimento, and Warren B Powell. 2020. Approximate dynamic programming for planning a ride-hailing system using autonomous fleets of electric vehicles. *European Journal of Operational Research* 284, 3 (2020), 1088–1106.

[2] María J Alonso-González, Sascha Hoogendoorn-Lanser, Niels van Oort, Oded Cats, and Serge Hoogendoorn. 2020. Drivers and barriers in adopting Mobility as a Service (MaaS)–A latent class cluster analysis of attitudes. *Transportation Research Part A: Policy and Practice* 132 (2020), 378–401.

[3] Yuri Burda, Harrison Edwards, Amos J. Storkey, and Oleg Klimov. 2019. Exploration by random network distillation. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

[4] BusinessGrowthReport. 2022. Global Designated Driving Service Market Research Report 2022. https://www.businessgrowthreports.com/TOC/22043825. Accessed: 2022-9-30.

[5] Caroline Claus and Craig Boutilier. 1998. The Dynamics of Reinforcement Learning in Cooperative Multiagent Systems. In *Proceedings of the Fifteenth National/Tenth Conference on Artificial Intelligence/Innovative Applications of Artificial Intelligence* (Madison, Wisconsin, USA) *(AAAI '98/IAAI '98)*. USA, 746–752.

[6] 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 '22: The 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, August 14 - 18, 2022*. 3838–3848.

[7] Yannis Flet-Berliac and Philippe Preux. 2020. Only Relevant Information Matters: Filtering Out Noisy Samples To Boost RL. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, Christian Bessiere (Ed.). ijcai.org.

[8] Sriram Ganapathi Subramanian, Pascal Poupart, Matthew E. Taylor, and Nidhi Hegde. 2020. Multi Type Mean Field Reinforcement Learning. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems* (Auckland, New Zealand) *(AAMAS '20)*. Richland, SC, 411–419.

[9] Yuhan Guo, Wenhua Li, Linfan Xiao, and Hamid Allaoui. 2024. A prediction-based iterative Kuhn-Munkres approach for service vehicle reallocation in ride-hailing. *International Journal of Production Research* 62, 10 (2024), 3690–3715.

[10] Zhen Guo, Bin Yu, Wenxuan Shan, and Baozhen Yao. 2023. Data-driven robust optimization for contextual vehicle rebalancing in on-demand ride services under demand uncertainty. *Transportation Research Part C: Emerging Technologies* 154 (2023), 104244.

[11] Guy Hacohen and Daphna Weinshall. 2019. On The Power of Curriculum Learning in Training Deep Networks. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA (Proceedings of Machine Learning Research, Vol. 97)*, Kamalika Chaudhuri and Ruslan Salakhutdinov (Eds.). PMLR, 2535–2544.

[12] J. A. Hartigan and M. A. Wong. 1979. A K-Means Clustering Algorithm. *Journal of the Royal Statistical Society: Series C (Applied Statistics)* 28, 1 (1979), 100–108. arXiv:https://rss.onlinelibrary.wiley.com/doi/pdf/10.2307/2346830

[13] Dan Hendrycks and Kevin Gimpel. 2016. Bridging Nonlinearities and Stochastic Regularizers with Gaussian Error Linear Units. *CoRR* abs/1606.08415 (2016). arXiv:1606.08415

[14] Xiaohui Huang, Jiahao Ling, Xiaofei Yang, Xiong Zhang, and Kaiming Yang. 2023. Multi-Agent Mix Hierarchical Deep Reinforcement Learning for Large-Scale Fleet Management. *IEEE Transactions on Intelligent Transportation Systems* 24, 12 (2023), 14294–14305. https://doi.org/10.1109/TITS.2023.3302014

[15] Eric Jang, Shixiang Gu, and Ben Poole. 2017. Categorical Reparameterization with Gumbel-Softmax. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

[16] Shenggong Ji, Yu Zheng, Zhaoyuan Wang, and Tianrui Li. 2019. A deep reinforcement learning-enabled dynamic redeployment system for mobile ambulances. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 3, 1 (2019), 1–20.

[17] Yan Jiao, Xiaocheng Tang, Zhiwei Tony Qin, Shuaiji Li, Fan Zhang, Hongtu Zhu, and Jieping Ye. 2021. Real-world ride-hailing vehicle repositioning using deep reinforcement learning. *Transportation Research Part C: Emerging Technologies* 130 (2021), 103289.

[18] Robert Kirk, Amy Zhang, Edward Grefenstette, and Tim Rocktäschel. 2023. A Survey of Zero-shot Generalisation in Deep Reinforcement Learning. *Journal of Artificial Intelligence Research* 76 (2023), 201–264.

[19] Harold W Kuhn. 1955. The Hungarian method for the assignment problem. *Naval research logistics quarterly* 2, 1-2 (1955), 83–97.

[20] Donghe Li, Chunlin Hu, Qingyu Yang, and Shitao Chen. 2023. Multi Actor-Critic PPO: A Novel Reinforcement Learning Method for Intelligent Task and Charging Scheduling in Electric Freight Vehicles Management. In *2023 IEEE 26th International Conference on Intelligent Transportation Systems (ITSC)*. 1116–1121. https://doi.org/10.1109/ITSC57777.2023.10421834

[21] Zhuoran Li, Ling Pan, and Longbo Huang. 2023. Beyond conservatism: Diffusion policies in offline multi-agent reinforcement learning. *arXiv preprint arXiv:2307.01472* (2023).

[22] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2016. Continuous control with deep reinforcement learning. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, Yoshua Bengio and Yann LeCun (Eds.).

[23] Yang Liu, Fanyou Wu, Cheng Lyu, Shen Li, Jieping Ye, and Xiaobo Qu. 2022. Deep dispatching: A deep reinforcement learning approach for vehicle dispatching on online ride-hailing platform. *Transportation Research Part E: Logistics and Transportation Review* 161 (2022), 102694.

[24] Zhidan Liu, Jiangzhou Li, and Kaishun Wu. 2020. Context-Aware Taxi Dispatching at City-Scale Using Deep Reinforcement Learning. *IEEE Transactions on Intelligent Transportation Systems* (2020).

[25] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. 2017. Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (Eds.). 6379–6390.

[26] Ofir Nachum, Shixiang Gu, Honglak Lee, and Sergey Levine. 2018. Data-Efficient Hierarchical Reinforcement Learning. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett (Eds.). 3307–3317.

[27] Takuma Oda and Carlee Joe-Wong. 2018. MOVI: A model-free approach to dynamic fleet management. In *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*. IEEE, 2708–2716.

[28] Ling Pan, Longbo Huang, Tengyu Ma, and Huazhe Xu. 2022. Plan better amid conservatism: Offline multi-agent reinforcement learning with actor rectification. In *International conference on machine learning*. PMLR, 17221–17237.

[29] Ling Pan, Tabish Rashid, Bei Peng, Longbo Huang, and Shimon Whiteson. 2021. Regularized softmax deep multi-agent q-learning. *Advances in Neural Information Processing Systems* 34 (2021), 1365–1377.

[30] Guoyang Qin, Qi Luo, Yafeng Yin, Jian Sun, and Jieping Ye. 2021. Optimizing matching time intervals for ride-hailing services using reinforcement learning. *Transportation Research Part C: Emerging Technologies* 129 (2021), 103239.

[31] Jacob Rafati and David C. Noelle. 2019. Learning Representations in Model-Free Hierarchical Reinforcement Learning. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*. AAAI Press, 10009–10010.

[32] Jake Robbennolt and Michael W. Levin. 2023. Maximum Throughput Dispatch for Shared Autonomous Vehicles Including Vehicle Rebalancing. *IEEE Transactions on Intelligent Transportation Systems* 24, 9 (2023), 9871–9885.

[33] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal Policy Optimization Algorithms. arXiv:1707.06347 [cs.LG] https://arxiv.org/abs/1707.06347

[34] Mohammad Shahverdy, Mahmood Fathy, Reza Berangi, and Mohammad Sabokrou. 2020. Driver behavior detection and classification using deep convolutional neural networks. *Expert Systems with Applications* 149 (2020), 113240.

[35] Zhenyu Shou and Xuan Di. 2020. Reward design for driver repositioning using multi-agent reinforcement learning. *Transportation research part C: emerging technologies* 119 (2020), 102738.

[36] 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* (Washington DC, USA) *(KDD '22)*. Association for Computing Machinery, New York, NY, USA, 3950–3960. https://doi.org/10.1145/3534678.3539060

[37] Alexander Sasha Vezhnevets, Simon Osindero, Tom Schaul, Nicolas Heess, Max Jaderberg, David Silver, and Koray Kavukcuoglu. 2017. FeUdal Networks for Hierarchical Reinforcement Learning. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017 (Proceedings of Machine Learning Research, Vol. 70)*, Doina Precup and Yee Whye Teh (Eds.). PMLR, 3540–3549.

[38] Xiaoyang Xie, Fan Zhang, and Desheng Zhang. 2018. PrivateHunt: Multi-source data-driven dispatching in for-hire vehicle systems. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 2, 1 (2018), 1–26.

[39] Yaodong Yang, Rui Luo, Minne Li, Ming Zhou, Weinan Zhang, and Jun Wang. 2018. Mean Field Multi-Agent Reinforcement Learning. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018 (Proceedings of Machine Learning Research, Vol. 80)*, Jennifer G. Dy and Andreas Krause (Eds.). PMLR, 5567–5576.

[40] 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 *Advances in Neural Information Processing Systems*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (Eds.), Vol. 35. Curran Associates, Inc., 24611–24624. https://proceedings.neurips.cc/paper_files/paper/2022/file/9c1535a02f0ce079433344e14d910597-Paper-Datasets_and_Benchmarks.pdf

[41] Lingyu Zhang, Tao Hu, Yue Min, Guobin Wu, Junying Zhang, Pengcheng Feng, Pinghua Gong, and Jieping Ye. 2017. A Taxi Order Dispatch Model based On Combinatorial Optimization. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, NS, Canada, August 13 - 17, 2017*. ACM, 2151–2159.

[42] Rick Zhang, Federico Rossi, and Marco Pavone. 2016. Model predictive control of autonomous mobility-on-demand systems. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*. 1382–1389.

[43] Wenqi Zhang, Qiang Wang, Jingjing Li, and Chen Xu. 2020. Dynamic Fleet Management With Rewriting Deep Reinforcement Learning. *IEEE Access* 8 (2020), 143333–143341.

[44] Bolong Zheng, Lingfeng Ming, Qi Hu, Zhipeng Lü, Guanfeng Liu, and Xiaofang Zhou. 2022. Supply-Demand-Aware Deep Reinforcement Learning for Dynamic Fleet Management. *ACM Trans. Intell. Syst. Technol.* 13, 3, Article 37 (2022), 19 pages.