A Dynamic Matching Time Strategy Based on Multi-Agent Reinforcement Learning in Ride-Hailing

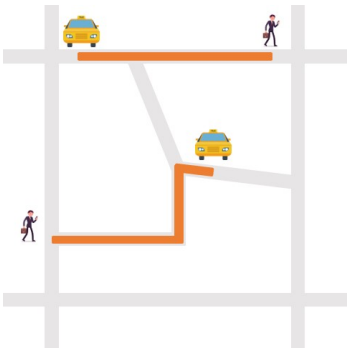
Shuai Li[1], Bing Shi[1,2] ✉, Yaping Deng[1]
[1] School of Computer Science and Artificial Intelligence, Wuhan University of Technology, Wuhan 430070, China
[2] Shenzhen Research Institute of Wuhan University of Technology, Shenzhen 518000, China
bingshi@whut.edu.cn

***Abstract*—For online ride-hailing platforms, choosing the right time to match idle vehicles with passengers is one of the most important factors affecting the platform's profit. On one hand, vehicles and passengers arrive dynamically, and an appropriate delayed matching may generate a highly efficient matching result with more values. On the other hand, different regions may have different states of supply (vehicles) and demand (passengers), and the matching time should be different. At this moment, we need an efficient matching time strategy that takes into account matching time and regional differences to maximize the platform's long-term profit. In this paper, we propose a dynamic matching time algorithm based on multi-agent reinforcement learning, which is called Multi-Region Differentiated Matching Decision. Firstly, we describe the order matching process and then model it as a decentralized partially observable Markov decision process (Dec-POMDP). Secondly, considering that there are regional differences in supply and demand, we divide the overall area based on historical data and propose an algorithm based on multi-agent reinforcement learning to realize multi-region differentiated dynamic matching. Finally, we conduct extensive experiments to evaluate our matching algorithm against benchmark algorithms in a real-world dataset. The experimental results show that our algorithm can outperform benchmark algorithms.**

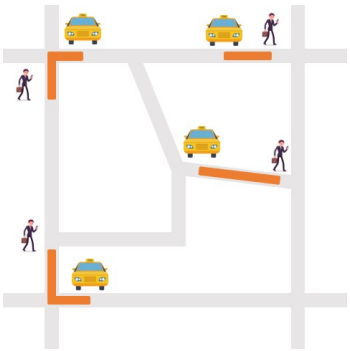
***Index Terms*—Ride-hailing, Delayed matching, Long-term profit, Multi-Agent Reinforcement learning**

I. Introduction

Online ride-hailing has become one of the most important transportation ways in the modern cites. In the ride-hailing system, the platform needs to match idle vehicles with passengers efficiently, since this will significantly affect the platform's profit and passengers' riding experience. Currently, the ride-hailing platform usually matches the passengers with vehicles immediately when the riding orders are raised, such as [1], [2]. In fact, the immediate matching may cause inefficient matching results. For example, as shown in Figure 1(a), if the platform performs matching at time t_1, the cost of the platform and the waiting time of the passengers will increase due to the insufficient number of vehicles and passengers. In contrast, instead of immediate matching, the platform can collect more information about orders and idle vehicles to make more efficient matching decisions in the delayed matching, which is shown in Figure 1(b). However, the passengers may have a tolerance time for waiting for the service. Therefore, the platform needs to find an effective matching time. Furthermore, the riding requests and idle vehicles may change dynamically over the time, thus the platform needs to dynamically determine the matching time based on the arriving orders and idle vehicles.



(a) $T = t_1$, Immediate Matching.



(b) $T = t_2$, Delayed Matching.

Fig. 1. Different matching strategies.

Furthermore, in a city-wide area, different small subareas (called regions) may have different supplies (idle vehicles) and riding demands. In this case, it is inappropriate for the platform to set a uniform matching time for all regions. The platform needs to determine the matching time for each region respectively according to the supply and demand in that region. Moreover, the platform needs to maximize the long-

DOI reference number: 10.18293/SEKE2023-112

term profit, and thus needs to consider the impact of current decisions on the future matching.

Specifically, with the dynamic arrival of passengers and vehicles, the platform needs to choose the right matching time for regions with different supply and demand to maximize long-term profits. Note that it is difficult for the platform to set the matching time at any time since this will result in heavy computing load of keeping monitoring whether the matching information is enough. In this paper, we divide the whole time into several time slots. Therefore, instead of determining the matching at any time, in this paper we determine at what time slot to do the matching, i.e., to determine whether matching at the current time slot or not, to achieve dynamic matching. In summary, the main contributions of this work are as follows. Firstly, the current matching time decision affects future order matching, and each region can only make decisions based on its observed supply and demand information, so this is a sequential decision problem, which we model as a decentralized partially observable Markov decision process (**Dec-POMDP**). Secondly, we divide the whole area into several non-overlapping regions according to the historical order data to realize multi-region differentiated dynamic matching. Thirdly, we design a multi-agent reinforcement learning based algorithm (**MRDMD**) to determine whether matching at the current time slot or not in order to maximize the long-term profit of the platform. Finally, we conduct extensive experiments to evaluate the proposed algorithm based on a real-world dataset. We demonstrate that using differentiated matching time in different regions can significantly increase the platform's long-term profit.

The rest of this paper is organized as follows. In Section II we introduce the related work. In Section III we describe the basic settings and define the problem. We introduce the multi-region differentiated matching decision algorithm in Section IV. Finally, we give experimental analysis in Section V and conclude the paper in Section VI.

II. Related Work

The order matching problem in ride-hailing is widely studied. There exist a lot of works that consider how to maximize the platform's profit. Li et al. [3] propose a non-centralized order matching approach, where vehicles are viewed as agents, and multiple agents work together to maximize the overall profit. Shi et al. [4] propose two order matching mechanisms based on truthful auction, where drivers bid for orders published by the platform and make profits in order to maximize the social welfare of drivers and platform.

There also exist a number of works about maximizing the number of completed orders in order matching problem. Garaix et al. [5] propose direct and iterative algorithms to solve the order matching problem and maximize the number of completed orders. Furthermore, Holler et al. [1] propose a deep reinforcement learning approach by combining deep learning. They treat vehicles as independent agents and perform order matching from the perspective of centralized platform dispatching, thus maximizing the number of completed orders.

Moreover, how to minimize vehicle travel distance is also investigated in the related works. Liao et al. [6] propose a nearest matching algorithm to match the order with the vehicle closest to it. Duan et al. [7] propose an algorithm that can gradually expand the visible range of orders, which can effectively reduce the allocation time of orders and maintain a low travel distance.

The above works usually consider immediate matching between vehicles with passengers. Some works, [3], [8], consider the delayed matching by using the cumulative information. Qin et al. [9] find that dynamic matching can effectively reduce passenger's waiting time compared to immediate matching. However, they do not take into account that in different regions, the platform has different supply and demand information, therefore the matching time can be set differently.

To the best of our knowledge, existing works did not consider that different regions have different supply and demand with dynamic changes over time, and thus they cannot dynamically determine the matching time for each region for each time to maximize the profit of the platform. In this paper, we consider the above factors and design a dynamic matching decision algorithm **MRDMD** for multiple regions to maximize the long-term profit of the platform.

III. Basic Settings

In this section, we first describe how the ride-hailing system works, and then we give the basic settings of the order, vehicle and platform's profit. Furthermore, we describe the problem we intend to solve in this paper.

Figure 2 shows how the ride-hailing system works. When a passenger rises a trip order, first this order will be collected by the platform into the buffer pool. Second, when the size of the buffer pool is suitable, the platform will match those orders with idle vehicles. Then the vehicle that receives the allocated order will transport the passenger to the designated location and charge the appropriate fee.

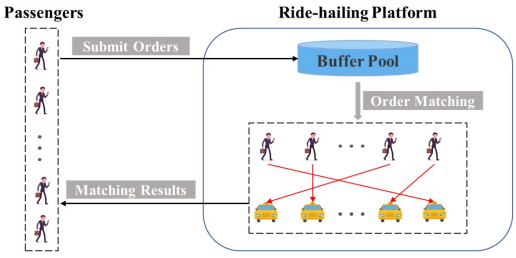


Fig. 2. Online ride-hailing platform.

We divide the whole time into several time slots $\mathcal{T} = \{1, 2, ..., T\}$ and consider a 2-D area $\mathcal{L} = \{1, 2, ..., L\}$. The relevant settings are given below.

*Definition 1 (**Order**):* An order $o_i \in \mathcal{O}$ is a travel request made by a passenger through a smart device, which can appear at any time slot but can only be served by a maximum of one vehicle. An order o_i is expressed as a tuple $o_i = (t_{o_i}, org_{o_i}, des_{o_i}, p_{o_i}, c_{o_i}, w_{o_i}, s_{o_i})$, the specific meaning of these elements is as follows:

- t_{o_i} is the time when the order o_i is submitted.
- org_{o_i} is the passenger's pick-up location, consisting of latitude and longitude.
- des_{o_i} is the destination of the order, consisting of latitude and longitude.
- p_{o_i} is the price of the order.
- c_{o_i} is the cost of the order, which is determined by the distance traveled and the unit cost of the vehicle.
- w_{o_i} is the maximum tolerance time for the passenger.
- s_{o_i} is a status identifier to indicate the status of the order, including pending, completed, and invalid.

*Definition 2 (**Vehicle**):* The vehicle $v_j \in \mathcal{V}$ is represented as a tuple $v_j = (loc_{v_j}, s_{v_j})$, the specific meaning of these two elements is as follows:

- loc_{v_j} represents the current location of the vehicle, consisting of latitude and longitude.
- s_{v_j} indicates the current state of the vehicle, to show whether the vehicle is idle or busy.

We assume that the vehicles are owned by the platform and the vehicles will follow the platform's dispatch to serve passengers, which simplifies the management of the platform and also improves efficiency [8].

*Definition 3 (**Platform's Profit**):* The platform's profit is equal to the value of all completed orders minus the cost of vehicles.

$$EP = \sum_{l=1}^{L} \sum_{t=1}^{T} \sum_{i=1}^{|O_t^l|} (p_{o_i} - c_{o_i}) \tag{1}$$

O_t^l is the completed order set in the region l at time slot t, p_{o_i} and c_{o_i} are the value and cost, respectively, of the ith completed order in region l at time slot t.

Based on the above settings, in this paper, we consider a ride-hailing system where vehicles and orders arrive dynamically and passengers have a maximum tolerance time. We want to dynamically determine the matching time in multiple regions to maximize the long-term profit EP. To achieve this goal, we design a Multi-Region Differentiated Matching Decision (**MRDMD**) algorithm, which dynamically determines the matching time for each region. Note that as we discussed in Section I, it is difficult for the platform to set the matching time at any time. Therefore, in this paper we determine at what time slot the platform will do the matching, i.e., to determine whether matching at the current time slot or not, to achieve dynamic matching.

IV. The Algorithm

In this section, we consider the decentralized partially observable Markov decision process (**Dec-POMDP**), which describes the multi-region differentiated order matching process, and then propose a multi-region differentiated matching decision (**MRDMD**) algorithm to maximize the profit of the platform.

The **Dec-POMDP** in our multi-regional cooperation scenario can be represented as a tuple: $G =< \mathcal{S}, A, \mathcal{P}, r, Z, \mathbb{O}, n, \gamma >$. Where $s \in \mathcal{S}$ represents the area's supply and demand information. At each time slot, each region l obtains its observation $z^l \in Z$ from the environment using the observation function $\mathbb{O}$. Based on the observation, each region chooses an action $a^l \in \mathcal{A}$ and all the actions are combined to form a joint action $a \in \mathcal{A}^L$. After this joint action is performed, the environment state is transferred to the new one according to the state transition function $\mathcal{P}(s_{t+1}|s_t, a_t) : \mathcal{S} \times \mathcal{A}^L \times \mathcal{S} \to [0, 1]$. r represents the reward function, which is shared by all the regions, n represents the number of regions, and $\gamma \in [0, 1)$ is a discount factor that decreases the impact of the past reward. We describe the details of the observation space, action space, and reward function below.

Observation Space: $z_t = ((v_t^1, o_t^1),(v_t^2, o_t^2), ..., (v_t^L, o_t^L)) \in Z^L$, where v_t^l, o_t^l represent the number of vehicles and orders respectively in region l at time slot t.

Action Space: $a_t = (a_t^1, a_t^2, ..., a_t^L) \in A^L$, where $a_t^l = 0$ and $a_t^l = 1$ represent delayed matching and matching in region l at time slot t, respectively. Note that the unmatched orders and vehicles will transfer to next time slot.

Reward Function: At each time slot t, each region l takes the action $a_t^l \in \{0, 1\}$, and then get a reward $r_t^l = \sum_{i=1}^{|O_t^l|}(p_{o_i} - c_{o_i})$ from environment. The global reward $r_t = \sum_{l=1}^{L} r_t^l$ is accounted for all regions in order to obtain the maximum overall profit.

There are multiple regions in our scenario, each requiring individual decisions and ensuring that the long-term profit of the platform is maximized. Therefore, we use QMIX [10] to design **MRDMD**, which is based on decentralized training and distributed execution and is widely used in various multi-agent environments. In our algorithm, there exist three types of networks, the regional network Q_l, the mixed network Q_{tot}, and the hypernetwork. The regional network performs an action based on individual observation z_t^l and previous action a_{t-1}^l and outputs regional action values $Q_l(\tau^l, a_t^l)$, where τ is the observation-action history, maintained by each region to perceive the dynamic changes of region information. The mixed network has weight parameters produced by the hypernetwork consisting of multiple linear layers and receives as input the state $s \in S$. This network receives the action values $Q_l(\tau^l, a_t^l)$ of all regions and outputs the overall action values $Q_{tot}(\tau, a)$. In summary, **MRDMD** needs to ensure that equation 2 is satisfied, which means that each region can greedily choose matching action based on the regional network solely. The **MRDMD** algorithm is shown in Algorithm 1.

$$\text{argmax}_a Q_{tot}(\tau, a) = \begin{cases} \text{argmax}_{a_1} Q_1(\tau_1, a_1) \\ ... \\ \text{argmax}_{a_L} Q_L(\tau_L, a_L) \end{cases} \tag{2}$$

Algorithm 1 takes the number of vehicles and regions as input. The training will take K rounds (line 4), each lasting for T time slots (line 7). At the beginning of the training, each agent acquires an initial observation state (line 6). During training, each agent first gets the action value by using the evaluation network $Q(\theta)$ (line 8). After that, these actions will be collected to get the joint action a (line 9). For each region, if

Algorithm 1 Multi-Region Differentiated Matching Decision (**MRDMD**) Algorithm

Input:

The number of vehicles V and the number of regions L.

Output:

Platform's matching strategy π.

1: Initialize the memory pool $\mathcal{M}$;
2: Initialize the evaluation network $Q(\theta)$ and the mixing network $Q_{tot}(\phi)$
3: Initialize the target network $\hat{Q}(\hat{\theta})$, $\hat{Q}_{tot}(\hat{\phi})$, and set the weight $\hat{\theta} \leftarrow \theta, \hat{\phi} \leftarrow \phi$;
4: **for** $k = 1$ to K **do**
5: $s \leftarrow env.reset()$;
6: Each agent gets the initial observation state z_1^l from s;
7: **for** $t = 1$ to T **do**
8: Each agent inputs its observation z_t^l to the network $Q(\theta)$ to obtain the action value;
9: Forming the joint action a;
10: **for** a_l in a **do**
11: **if** $a_l = 1$ **then**
12: Matching vehicles and orders using **Kuhn-Munkres** algorithm to get the reward $r = p - c$;
13: **end if**
14: **if** $a_l = 0$ **then**
15: No matching is performed, and the reward $r = 0$. Vehicles and orders in this region will be transferred to the next round;
16: **end if**
17: **end for**
18: Get the next state s' and terminated flag $done$;
19: Store the state transition tuple $(s_t, s', z, r, a, done)$ into memory pool $\mathcal{M}$;
20: **if** $len(M) > threshold$ **then**
21: Randomly select b samples from $\mathcal{M}$ for training;
22: Update network based on loss function $\mathcal{L}(\theta)$: $\mathcal{L}(\theta) = \sum_{i=1}^{b}[(r + \gamma \max_{a'} Q_{tot}(\tau', a', s'; \hat{\phi}) - Q_{tot}(\tau, a, s; \phi))^2]$
23: **end if**
24: $s \leftarrow s'$, use the observation function $\mathbb{O}$ to generate the observation state z of each region;
25: **end for**
26: Every round c, $\hat{\theta} \leftarrow \theta, \hat{\phi} \leftarrow \phi$;
27: **end for**

a matching action is selected, it will use Kuhn-Munkres [11] algorithm to find the most profitable matching combination of the current vehicles and passengers (line 12). When no matching is made, the current vehicle and order information is transferred into the next round (line 15). After the matching is completed, we can calculate the real-time reward r of the matching action and get the next state s' (line 18), and then this information will be stored in the memory pool (line 19). When the memory size is greater than a threshold, we will randomly sample some records from the memory pool for learning (lines 20-23). After continuous repeated learning and training, a convergent matching strategy that can maximize the platform's long-term profit is obtained.

V. Experimental Analysis

In this section, we run extensive experiments to evaluate our algorithm. The dataset used in this paper is provided by DiDi, from which we select the data from 13:00 to 15:00 on weekends, which contains the initiation time, the end time, the start location and destination of the order, and also the price obtained by completing the order. Similar to [12]–[14], we divide the area into four regions with different supply and demand based on the number of orders, which is shown in Figure 3.

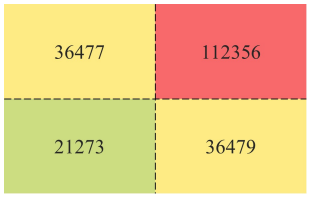


Fig. 3. Area division. The experimental area is about 215.71 square kilometers. Furthermore, we analyze the experimental data and find that the average travel distance to complete an order is 6.98 km. Therefore we divide the area into 4 regions with an area of 53.93 square kilometers. Such a region with the length of 7.34 km can cover the average travel distance of an order. The number in the region represents the number of orders during the experimental period.

Vehicles and passengers arrive dynamically, and their activity range is bounded according to the maximum range of all orders in the dataset. The initial state of each vehicle is idle. The number of vehicles gradually increases from 500 to 2000, with each increase of 500 vehicles. The period is 13:00-15:00, and we use $\Delta t = 10s$ as the length of time slot. There will be 720 time slots during the entire period. We set the vehicle's unit driving cost to 1 CNY/km. Each order has a maximum tolerance time w, and we assume it is independently and identically drawn from a uniform distribution within $[1, 30]$. When a passenger waits longer than the tolerance time, the order will be cancelled. In this experiment, we assume that the vehicle only serves orders within the same region at the current time slot, but after the service is completed, the vehicle can move to other regions. The specific experimental parameters are shown in Table I.

TABLE I
Experimental parameters

parameter	*value*
Time period T	13:00-15:00
Length of time slot Δt	10
Number of vehicles	[500, 1000, 1500, 2000]
Maximum tolerance time w	$U(1, 30)$
Number of regions L	4
Unit driving cost	1

A. Benchmark Approaches and Metrics

We will use the following benchmark approaches and metrics to evaluate our proposed algorithm.

Multi-Region Restricted Q-Learning (**MRRQL**). The **RQL** algorithm [15] sets a maximum matching time interval $[a, b]$, and then uses reinforcement learning to continuously adjust the size of this interval to control the matching time. We modify the **RQL** algorithm to fit our problem, where each region performs the **RQL** algorithm independently, which is called the **MRRQL**.

Multi-Region GREEDY (**MRGREEDY**). Tong et al. [2] find that **GREEDY** algorithm can still achieve very competitive results in most cases. We modify the **GREEDY** algorithm to fit our problem, where each region performs matching independently at each time slot using the **GREEDY** algorithm, and priority is given to match the highest value orders with idle vehicles, which is called the **MRGREEDY**.

Multi-Region UNIFORM (**MRUNIFORM**). The **UNIFORM** algorithm [16] is a commonly used comparison algorithm, which will do the matching for every n time slots. We modify the **UNIFORM** algorithm to increase its dynamic matching property, with a half probability of matching at the current time slot and a half probability of not making a match, which is called the **MRUNIFORM**.

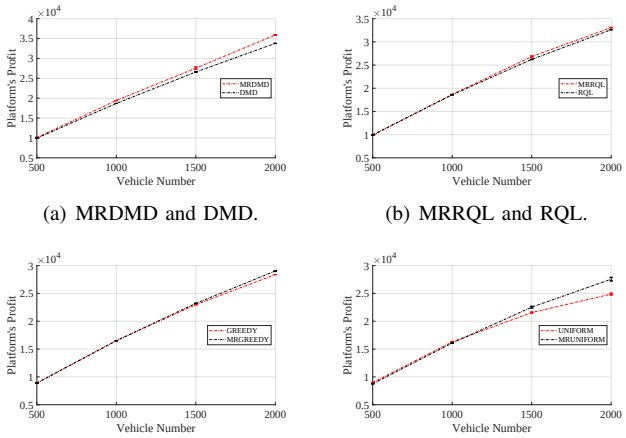
In order to evaluate the performance of **MRDMD**, we consider the following metrics.

- **Total platform's profit.** The total platform's profit refers to the sum of the profit of all vehicles.
- **Order response rate.** Order response rate is the ratio of completed orders to total orders.
- **Pick-up distance.** Pick-up distance is the distance from the vehicle's current location to the order initiation location after the vehicle matches the order.
- **Average extra distance per order.** The average extra distance per order is the average distance an empty car needs to travel to complete an order.

B. Experimental Results

In the experiment, we increase the number of vehicles from 500 to 2000 with step size 500, and the experimental results are shown below.

To prove that our algorithm is effective when combined with the region division, we conduct experiments on the whole area and multiple regions respectively to compare the profit of all algorithms. We use **MR** to represent an algorithm that combined with region division, e.g., **MRDMD** and **DMD** are the same algorithm running on multiple regions and the whole area, respectively. From Figure 4(a) and Figure 4(b) , we can see that **MRDMD** can make higher profits than **DMD**, and similarly **MRRQL** can make higher profits than **RQL**, which means that after the area is divided, the reinforcement learning based algorithm can sense the differentiated information of each region and make independent and effective matching decisions. In contrast, from Figure 4(c) and Figure 4(d), we can find that **MRGREEDY** and **MRUNIFORM** do not perform well as that **GREEDY** and **UNIFORM** do.



(a) MRDMD and DMD. (b) MRRQL and RQL.

(c) MRGREEDY and GREEDY. (d) MRUNIFORM and UNIFORM.

Fig. 4. Platform's profit between multiple regions and overall area.

Next, we will evaluate the proposed algorithm with multiple regions against the benchmark approaches.

Total Platform's Profit: The total platform's profit of the four algorithms is shown in Figure 5. As the number of vehicles increases, we find that the profit of all four algorithms increase. We also find that **MRDMD** performs better than the other three algorithms. **MRRQL** performs better than the other two algorithms. In more detail, we find that the algorithm with dynamic delayed matching time (**MRDMD**, **MRRQL**) brings more profit to the platform compared to the algorithms with immediate matching (**MRGREEDY**) and dynamic random matching (**MRUNIFORM**).

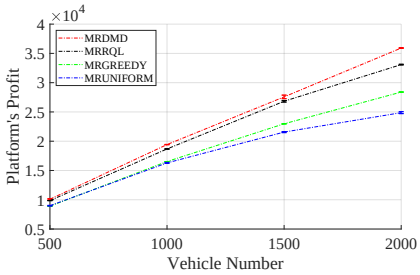


Fig. 5. Total platform's profit.

Order Response Rate: As we can see in Figure 6, when the idle vehicles are insufficient, the order response rate of the four algorithms are very similar, where **MRGREEDY** has a slight advantage. This is because the **MRGREEDY** matches orders with vehicles at each time slot, and therefore has a better order response rate. When the number of vehicles is greater than 1500, **MRDMD** still performs the best, followed by **MRRQL**, **MRGREEDY** and **MRUNIFORM**.

Pick-up Distance: From Figure 7 we can see that the pick-up distance of the four algorithms increases when the number of vehicles is increased to 1500. When the number of vehicles is greater than 1500, the pick-up distance decreases for **MRDMD** and **MRRQL**, while the distance still increases for **MRGREEDY** and **MRUNIFORM**. This may be because **MRDMD** and **MRRQL** can sense the dynamic changes of the regional state about vehicles and orders, and when there

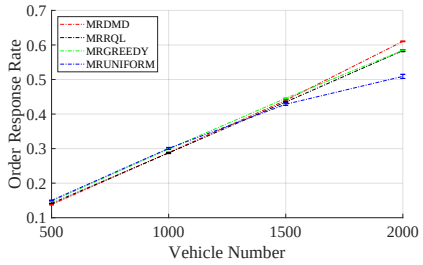


Fig. 6. Order response rate.

exist a large number of vehicles, these two algorithms can accumulate enough information about orders and vehicles to make efficient matching decisions, which may decrease pick-up distance. Finally, we still find that **MRDMD** performs the best.

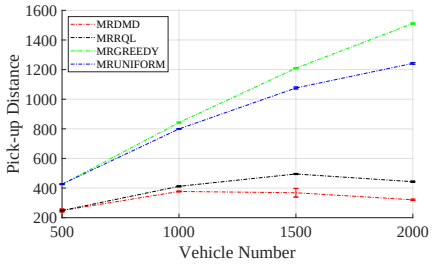


Fig. 7. Pick-up distance.

Average Extra Distance Per Order: From Figure 8, we can see that the average extra distance per order for the four algorithms decreases as the number of vehicles increases and **MRDMD** has better performance. This is because more vehicles bring more matching information, and for an order, a closer vehicle can be selected for matching. Therefore the average distance to complete an order decreases.

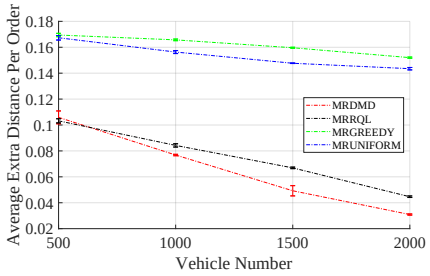


Fig. 8. Average extra distance per order.

VI. Conclusion

In this paper, we propose a multi-region differentiated matching decision algorithm (**MRDMD**) based on multi-agent reinforcement learning by considering the real-time supply and demand status of different regions in order to maximize the ride-hailing platform's profit. In order to evaluate the effectiveness of **MRDMD**, we run experimental analysis based on the real DiDi dataset against three typical benchmark algorithms. The experimental results show that the proposed algorithm can outperform other algorithms. We find that our algorithm with dynamic matching time according to the supply and demand status of each region can bring higher long-term profit and serve more orders. Our analysis can also provide useful insights for designing the realistic matching time strategy for ride-hailing platforms.

Acknowledgement

This paper was funded by the Shenzhen Fundamental Research Program (Grant No.JCYJ20190809175613332), the Humanity and Social Science Youth Research Foundation of Ministry of Education (Grant No.19YJC790111) and the Philosophy and Social Science Post-Foundation of Ministry of Education (Grant No.18JHQ060).

References

[1] John Holler et al. Deep reinforcement learning for multi-driver vehicle dispatching and repositioning problem. In *2019 IEEE International Conference on Data Mining (ICDM)*, pages 1090–1095. IEEE, 2019.

[2] Yongxin Tong, Jieying She, et al. Online minimum matching in real-time spatial data: experiments and analysis. *Proceedings of the VLDB Endowment*, 9(12):1053–1064, 2016.

[3] Minne Li, Zhiwei Qin, et al. Efficient ridesharing order dispatching with mean field multi-agent reinforcement learning. In *The world wide web conference*, pages 983–994, 2019.

[4] Bing Shi, Yikai Luo, and other. Auction-based order-matching mechanisms to maximize social welfare in real-time ride-sharing. In *International Conference on Database Systems for Advanced Applications*, pages 261–269. Springer, 2020.

[5] Thierry Garaix, Christian Artigues, et al. Optimization of occupancy rate in dial-a-ride problems via linear fractional column generation. *Computers & Operations Research*, 38(10):1435–1442, 2011.

[6] Ziqi Liao. Real-time taxi dispatching using global positioning systems. *Communications of the ACM*, 46(5):81–83, 2003.

[7] Yubin Duan, Ning Wang, et al. Optimizing order dispatch for ride-sharing systems. In *2019 28th International Conference on Computer Communication and Networks (ICCCN)*, pages 1–9. IEEE, 2019.

[8] Zhe Xu, Zhixin Li, et al. Large-scale order dispatch in on-demand ride-hailing platforms: A learning and planning approach. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 905–913, 2018.

[9] Guoyang Qin, Qi Luo, et al. Optimizing matching time intervals for ride-hailing services using reinforcement learning. *Transportation Research Part C: Emerging Technologies*, 129:103239, 2021.

[10] Tabish Rashid, Mikayel Samvelyan, et al. Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning. In *International conference on machine learning*, pages 4295–4304. PMLR, 2018.

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

[12] Yongxin Tong, Libin Wang, et al. Dynamic pricing in spatial crowdsourcing: A matching-based approach. In *Proceedings of the 2018 International Conference on Management of Data*, pages 773–788, 2018.

[13] Kaixiang Lin, Renyu Zhao, et al. Efficient large-scale fleet management via multi-agent deep reinforcement learning. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1774–1783, 2018.

[14] Xiaocheng Tang, Fan Zhang, et al. Value function is all you need: A unified learning framework for ride hailing platforms. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 3605–3615, 2021.

[15] Yansheng Wang, Yongxin Tong, et al. Adaptive dynamic bipartite graph matching: A reinforcement learning approach. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*, pages 1478–1489. IEEE, 2019.

[16] Chang Liu, Jiahui Sun, et al. Spatio-temporal hierarchical adaptive dispatching for ridesharing systems. In *Proceedings of the 28th International Conference on Advances in Geographic Information Systems*, pages 227–238, 2020.