META: A City-Wide Taxi Repositioning Framework Based on Multi-Agent Reinforcement Learning

Chenxi Liu, Chao-Xiong Chen, and Chao Chen

***Abstract*— The popularity of online ride-hailing platforms has made people travel smarter than ever before. But people still frequently encounter the dilemma of "*taxi drivers hunt passengers and passengers search for unoccupied taxis*". Many studies try to reposition idle taxis to alleviate such issues by using reinforcement learning based methods, as they are capable of capturing future demand/supply dynamics. However, they either coordinate all city-wide taxis in a centralized manner or treat all taxis in a region homogeneously, resulting in inefficient or inaccurate learning performance. In this paper, we propose a multi-agent reinforcement learning based framework named META (MakE Taxi Act differently in each agent) to mitigate the disequilibrium of supply and demand via repositioning taxis at the city scale. We decompose it into two subproblems, i.e., taxi demand/supply determination and taxi dispatching strategy formulation. Two components are wisely built in META to address the gap collaboratively, in which each region is regarded as an agent and taxis inside the region can make two different actions. Extensive experiments demonstrate that META outperforms existing methods.**

***Index Terms*— Taxi reposition, reinforcement learning, multi-agent learning.**

I. Introduction

The popularity of online ride-hailing platforms has made people travel smarter than ever before. They provides a global eye to "see" the real-time demands and supplies, i.e., taxi-hailing requests from the passengers and taxis available for picking up passengers, generating more cost-effective matches between passengers and taxis. Yet people still encounter the situation frequently, i.e., *passengers have to wait for a long time to be assigned a taxi while empty taxis wander around looking for passengers in other regions*. It is caused by the imbalance between the demands and supplies of taxis in the city [1], [2]. From the perspective of platforms, it is practical to redress the imbalance by proactively repositioning idle taxis to high-demand regions in advance, which will be of great benefit to passengers, drivers, and platforms.

However, repositioning taxis is not a trivial task. Dispatching idle taxis based on the current distribution of passengers' requests is less effective. We aim to seek a globally dispatching strategy at the city scale by repositioning taxis to serve the future demands of passengers. It is a challenging problem due to the following facts: (1) the demands for taxis are unknown because passengers' mobility poses high uncertainty and dynamics [3]; (2) the repositioning of taxis will affect the demands and supplies of taxis in the city, resulting in real-time changes and an unknown impact on the future; (3) the repositioning of taxis should consider the practical issues, e.g., reachability and cost, which are also concerned by platforms.

Reinforcement Learning (RL) can learn decision-making strategies by interacting with a complex environment. It is capable of capturing the long-term effects of actions and handling real-time changes in the environment. Because of such advantages, many researchers propose to use RL-based methods to solve the taxi repositioning problem [4], [5]. The proposed methods fall into two major categories, *centralized* and *distributed* respectively.

- The *centralized* RL solutions take the whole city as an agent. As for the agent, it needs to coordinate all taxis in the city when making repositioning strategies. Since the state space and action space are prohibitively huge due to the large taxi fleet [6], the centralized RL solutions may suffer from the *inefficiency* issue.
- The *distributed* RL solutions introduce the Multi-Agent Reinforcement Learning (MARL) and regard each region in the city as an agent [7]. To ease the training phase, each agent usually learns the optimized policy through a shared deep model with others [6]. In the acting phase, all idle taxis within the same region are suppose to be *homogeneous* and will take the *same* action. Taking the concrete case shown in Fig. 1 (left) as an example, previous MARL solutions would decide all three taxis to move to Region#2 at t_1, leaving one passenger in Region#1 unserved. These two simplifications contribute to the *inaccuracy* issue of MARL.

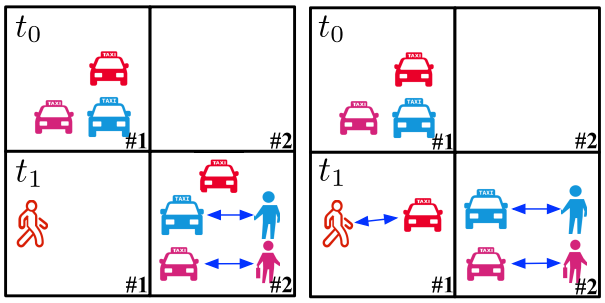

Fig. 1. A comparable example highlighting the effect of **META**.

To achieve an *efficient and accurate* reinforcement learning simultaneously, we present a multi-agent based taxi repositioning framework called **META**, which enables to **M**ak**E** **T**axi **A**ct differently at the same region. Its main distinctive features include: 1) We divide the city into equal-sized grids and take each grid as an agent, which learns the optimized policy *independently*; 2) Taxis in the same grid are *heterogeneous*, i.e., each taxi can take two different actions, i.e., stay still or head to other regions. The first feature ensures the avoidance of side effect caused by unnecessary interactions among different agents during the training of parameter-shared model. Instead of treating each taxi as an agent, the second feature strikes a nice trade-off between the reposition granularity and the overall performance. Since taxis in the same grid can take two different actions, the agents can have more modes of cooperation, leading to a better policy eventually. For instance, with **META**, two idle taxis in Fig. 1 (right) would be relocated to Region#2 and one stay still at Region#1 at t_1.

To sum up, the main contributions are as follows:

- We identify the limitations of existing works, and study the problem of balancing city-wide taxi demands and supplies by decomposing it into two subproblems: *determination of the*

Manuscript received 21 February 2021; revised 7 June 2021; accepted 5 July 2021. Date of publication 16 July 2021; date of current version 9 August 2022. This work was supported in part by the National Natural Science Foundation of China under Grant 61872050 and in part by the Chongqing Basic and Frontier Research Program under Grant cstc2018jcyjAX0551. The Associate Editor for this article was F. Chu. *(Chenxi Liu and Chao-Xiong Chen contributed equally to this work.) (Corresponding author: Chao Chen.)*

The authors are with the College of Computer Science, Chongqing University, Chongqing 400044, China (e-mail: cschaochen@cqu.edu.cn).

Digital Object Identifier 10.1109/TITS.2021.3096226

numbers of reposition taxis and *determination of corresponding targeted moving-in/out region*, respectively.

- We propose **META** to reposition city-wide taxis, which is a two-stage repositioning framework, consisting of the *taxi demand/supply determination* layer based on MARL and the *taxi dispatching strategy formulation* layer based on bipartite graph matching.
- We design a transportation environment simulator to evaluate **META** based on the real taxi dataset from the City of New York (NYC), US. The experiment results demonstrate that **META** outperforms, compared to the homogeneous taxi reposition system, with the balance capability and order response rate increased by 8.71% and 14.4%, respectively.

The rest of this paper is structured as follows. We narrate the preliminaries and problem statement in Sec. II. The design of **META** is presented in Sec. III, and we will evaluate it in Sec. IV. Then we review the related works in Sec. V. Finally, Sec. VI concludes this paper.

II. Problem Formulation

We first introduce some key concepts of the **META** repositioning system before describing the problem of balancing the city-wide taxi demands and supplies.

Agent. A city grid is regarded as an agent i. We divide the city into a series of equal-sized grids, as a result, the number of agents is exactly equal to the number of city grids, denoted as N. Each agent has its own fleet management policy. Specifically, the policy is to determine how many taxis in the grid will be relocated to other grids.

State. The state of an agent i at the time t is denoted as s_t^i. It can be computed by $Num_d^i - Num_s^i$, where Num_d^i and Num_s^i refer to the number of taxi demands and supplies in an agent i, respectively. The global state s^t consists of each grid's state s_t^i.

Action. Agent's action is the number of taxis that needs to dispatch to nearby grids. There are $|a_t^i \times Num_s^i|$ taxis that need to be repositioned, where $a_t^i \in [-1, 1]$. We use a threshold ξ with a small positive value to categorize grids based on a_t^i. If $a_t^i > \xi$, agent i will reposition taxis to other agents, and such agent is called as a **Source** grid. When $a_t^i < -\xi$, agent i expects to be repositioned taxis from other agents, and it is called as a **Sink** grid. As for agents with $|a_t^i| \leq \xi$, the agents will take no action.

Strategy. A repositioning strategy is a set of repositioning pairs. A repositioning pair is a pair of a source grid and a sink grid. The number of repositioning taxis for such a pair is determined by their corresponding actions.

Reward. The reward r_t^i is the revenue that an agent i obtains at the time t, which is defined as follows:

$$r_t^i = 1 - \frac{|Num_d^i - Num_s^i|}{max\{Num_d^i, Num_s^i\}} \tag{1}$$

It reflects the equilibrium degree of the agent i. An agent aims to maximize the accumulated reward with the discount of γ, denoted as $\sum_{k=0}^{\infty} \gamma^k r_{t+k}^i$. We set $\gamma = 0.9$ for all agents.

Heterogeneous Taxis. Heterogeneous taxis indicate that taxis in the same grid can have two different relocation destinations, i.e., some taxis stay still and the rest taxis are dispatched to other grids, which is different from *homogeneous taxis* in previous studies. The idea of heterogeneous taxis can generate more optimized repositioning strategies to better balance city-wide taxi demands and supplies.

Therefore, considering the heterogeneous taxis, the problem of balancing the city-wide taxi demands and supplies can be stated as: given the state s_t^i of each agent at time t, we intend to determine the number and the destinations of the repositioning taxis to develop a repositioning strategy to balance the future distribution of taxi demands and supplies. For each agent, it will try to maximize its accumulated reward.

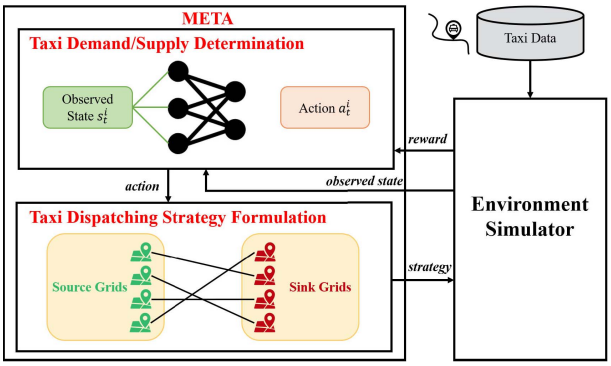

Fig. 2. The system framework of **META**.

III. The META System

In this section, we introduce the **META** system, which consists of two components, i.e., *taxi demand/supply determination* and *taxi dispatching strategy formulation*, respectively.

As shown in Fig. 2, the component of *taxi demand/supply determination* is to generate actions for an agent, based on the state observed from the environment. Specifically, it is to determine the number of taxis for relocating (i.e., sending out or calling in), according to the number of passenger orders and idle taxis in the grid. On such basis, the component of *taxi dispatching strategy formulation* is to develop repositioning strategies for agents, i.e., determining the repositioning pairs. After executing the repositioning strategy, the transport environment will generate the next observed state and bring the reward to each agent, which triggers the next learning round of the **META** system.

A. Taxi Demand/Supply Determination

In this component, we aim to figure out the future demand and supply of taxis in grids, based on historical passenger orders and taxi data. As mentioned above, we model the sub-problem as a Markov game G for N agents, referred as a tuple $G = (N, S, A, R, P, \gamma)$, where N, S, A, R, P, and γ denote the number of agents, set of states, joint action space, reward function, transition possibility, and a future reward discount factor, respectively. In our formulation, we split the global reward and assign each piece to each agent. By maximizing the expected accumulated reward, the agent could learn a policy to generate action at each time step. Compared with the action space consisting of limited discrete values, the agent's actions are continuous in our work. Considering the action values of agents may vary considerably, we normalize the action value to $[-1, 1]$, which is the ratio of the number of taxis that need to be repositioned in the next time step in the grid. In the following, we elaborate details on how to obtain the optimal actions for agents in the dynamical environment.

We extend the deep deterministic policy gradients (DDPG) to a multi-agent setting [8]. DDPG consists of two models: actor (μ-network) and critic (Q-network), as shown in Fig. 3. The actor is a policy network that takes the observed state as input and outputs the exactly continuous action directly, as shown in Eq. 2.

$$a_t^i = \mu_{\theta^i}(s_t^i) \tag{2}$$

where s_t^i is the observed state of the agent i from the environment, and θ^i refers to the μ-network parameters. Note that we clip the value of actions to $[-1, 1]$. The critic is a Q-value network that

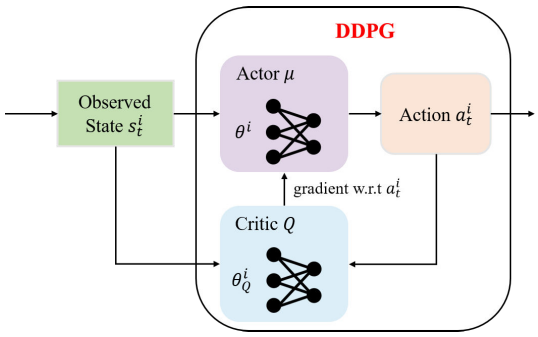

Fig. 3. The architecture of DDPG. The actor and critic are chained together, both having online network and target network.

takes the observed state and action as input and outputs the Q-value. To stabilize learning, target networks are created for both critic and actor. The actor is updated according to Eq. 3.

$$\begin{aligned}\nabla_{\theta^i}\mu &\approx \mathbb{E}_{s\sim\rho^\mu}\left[\nabla_{\theta^i} Q\left(s, a \mid \theta_Q^i\right)\Big|_{s=s_t^i, a=a_t^i}\right] \\ &= \mathbb{E}_{s\sim\rho^\mu}\left[\nabla_a Q\left(s, a \mid \theta_Q^i\right)\Big|_{s=s_t^i, a=a_t^i} \nabla_{\theta^i}\mu\left(s\right)\Big|_{s=s_t}\right]\end{aligned} \tag{3}$$

where $Q(s, a \mid \theta_Q^i)$ is the Q-value derived from the critic, and θ_Q^i refers the Q-network parameters. We adopt a mean-squared Bellman error function as the loss function to guide Q-network update. The loss function can roughly tell us how closely the Q-network comes to the optimal Q-value, as shown in Eq. 4.

$$L\left(\theta_Q^i\right) = \mathbb{E}_{s\sim\rho^\mu}\left[\left(Q\left(s, a \mid \theta_Q^i\right)|_{s=s_t, a=a_t} - y_t\right)^2\right] \tag{4}$$

where y_t is target value of target network in critic, defined as:

$$y_t^i = r_t^i + \gamma\, Q\left(s, a \mid \theta_Q^i\right)\Big|_{s=s_{t+1}, a=\mu(s_{t+1})} \tag{5}$$

In the multi-agent setting, we build a DDPG model for each agent, instead of using the parameter-shared network. Each agent interacts with the environment to learn a policy and a Q-value function concurrently. DDPG uses off-policy data and the Bellman equation to learn an optimal approximator of the Q-value, which in turn is used to learn the optimal action.

B. Taxi Dispatching Strategy Formulation

In this component, we aim to develop optimal taxi dispatching strategies for agents, based on the output actions of the first component. Concretely, it is to determine where the idle taxis should head to in the next time step.

As discussed, agents with different actions represent the grids with different taxi demands. There are two kinds of grids, i.e., source grids and sink grids. The strategy of dispatching idle taxis from a source grid to a sink grid should answer two sub-questions: 1) how many taxis should be dispatched from the source grid, and 2) which sink grid should the taxis go to. To simplify the complexity and focus on the key issue, we assume that all taxis that need to be repositioned in a source grid are dispatched to an identical sink grid to meet its demand. Thus, the problem of formulating taxi dispatching strategies can be converted into the problem of finding the maximum matching of the bipartite graph [9]. Furthermore, we need to consider the cost of dispatching taxis. In this paper, the dispatch cost is defined as the distance a taxi travels from a source grid to a sink grid. We aim to find an optimal matching that source girds could provide as many taxis as possible for sink grids while the total cost is minimized. It is fundamentally a combinatorial optimization problem, called the assignment problem [10].

Formally, we present the mathematical definition of the task. Given a bipartite graph $G = (V, E)$ with bi-partition (V_o, V_i) and a weight function $w(e)$, it is to find a perfect matching M minimizing $w(M)$. V_o and V_i denote the set of source grids and the set of sink grids, respectively. $e = (i, j)$ refers to a dispatch from the source grid i to the sink grid j, and the weight function $w(e) = d_{ij}$ indicates the cost of the dispatch, where d_{ij} is the Manhattan distance between two grids. There are two considerations when designing the weight function: 1) the dispatch cost should be proportional to the distance traveled; 2) in practice, taxi drivers prefer to be dispatched to nearby grids.

As a result, the objective is to minimize the demand-supply gap between source grids and sink grids with the minimal total cost, which is equivalent to:

$$\max \sum_{(i,j)\in M} -d_{ij} \times |x_i - y_j| \tag{6}$$

where x_i and y_j refer to the number of taxis that the source grid i can supply and the demands of the sink grid j, respectively. They are calculated according to Eq 7.

$$\begin{aligned} x_i &= \left|a_t^i \times Num_s^i\right| \\ y_j &= \left|a_t^j \times Num_d^j\right| \end{aligned} \tag{7}$$

where a_t^i and a_t^j refer to the action outputs of the first component; Num_s^i and Num_d^j refer to the number of idle taxis in the corresponding grid at time step t, respectively.

Obviously, it is an unbalanced assignment since both parts of the bipartite graph may have the different number of vertices. To address this issue, we add some *virtual* vertices to the smaller part and connect them to the larger part, using the edge cost of $-\infty$. Finally, we introduce the Kuhm-Munkres (KM) algorithm to find a perfect matching [11]. Note that we apply some trivial techniques in this component. We set a threshold ξ for the input actions. It is negligible when action $|a_t^i| \le \xi$, which makes the component more concerned about effective dispatching. It is also helpful to enhance the system robustness and reduce the influence of noise actions. Moreover, the one-to-one dispatching strategy could potentially alleviate side effects caused by the dynamical environment in MARL, since the changed action only influences the number of repositioning taxis in the current repositioning pair. For agents that do not appear in the repositioning pair, they will overlook such action change unless it changes dramatically.

IV. Experiments

A. Data Description

We use the taxi data provided by The New York City Taxi and Limousine Commission (TLC),[1] from June 1 to June 15 in 2016, in our experiments. The data contains information about the time and location of passengers getting on and off. In the time dimension, we split a day into 144 time steps, i.e., one time step is set to 10 minutes. In the space dimension, we divide the city into equal-sized grids. By removing grids that are unreachable (e.g., mountains, rivers) for passengers and taxis, we obtain 84 valid city grids in total, with the side length of 0.01° in latitude and longitude.

B. Simulator Design

To support the training and evaluation, we design a simulator based on historical taxi data from the Manhattan, NYC. It contains four

[1] https://www1.nyc.gov/site/tlc/about/data-and-research.page

TABLE I

PERFORMANCE RESULTS OF DIFFERENT METHODS WITH DIFFERENT SIZES OF THE TAXI FLEET. FOR *Reward*, WE REGARD THE RESULT OBTAINED BY THE RANDOM METHOD WITH 6000 TAXIS AS THE REFERENCE

Method	6000 taxis		8000 taxis		10000 taxis	
	Reward	ORR	Reward	ORR	Reward	ORR
Random	100.0000	100.00	109.5511	120.96	116.6285	141.33
DQN-5	133.6933	106.83	159.8524	139.10	164.4795	160.04
DQN-7	134.8740	109.96	159.3738	**142.12**	169.0460	160.00
DDPG	**157.0584**	**112.39**	**180.3936**	139.91	**193.8792**	**167.67**

steps at each time slice, including order generation, taxi update, order assignment and strategy making, detailed as follows:

- **Order generation:** The orders in the current time step are generated from the passenger boarding points in the historical data within the same time step.
- **Taxi update:** In the second step, we update the number of idle taxis in each grid. The arriving taxis contain two parts, one is from the passenger-carrying task and the other is from the repositioning task. For taxis that are occupied by their tasks will be considered as "busy taxis"and not be dispatched in the next steps.
- **Order assignment:** The orders in the grid are assigned to the idle taxis in the same grid, leaving either unfilled orders or extra idle taxis.
- **Strategy making:** Last and the most importantly, we take the state as input to the proposed **META** system to generate the taxi repositioning strategy. With the repositioning strategy, taxis will go to corresponding grids. The traveling time depends on the distance between the origin gird and the destination grid.

C. Experiment Setup

1) Baselines: To validate the effectiveness of **META**, we compare the repositioning performance of the adopted DDPG algorithm to other methods, detailed as follows.

- **Random:** It randomly determines the number of taxis that are dispatched in or out in each grid.
- **DQN-5:** The DQN method is an RL-based method. Since the action space of DQN is discrete, we discretize the continuous action space using various intervals. DQN-5 has 5 discrete action values, i.e., $\{0.9, \xi, 0, -\xi, -0.9\}$, where ξ is the threshold defined in Sec. III.
- **DQN-7:** We also implement DQN method to DQN-7, including 7 discrete action values, i.e., $\{0.9, \frac{0.9+\xi}{2}, \xi, 0, -\xi, -\frac{0.9+\xi}{2}, -0.9\}$.

2) Parameter Setting: The actor-network and the critic-network of DDPG are parameterized by a two-layer MLP model, which is same to the DQN method. The optimizer is *AdamOptimizer*, the learning rate and the batch size are respectively set to 0.001 and 64, which are also the default settings of all methods. The random seeds are also set to be same, i.e., 5, 15, 25. We run 3 times to report the average results. Note that the maximal dispatch distance is set to 2 grids and the threshold ξ is set to 0.1. All the methods are implemented with Python 3.7.4, Pytorch 1.2.0, and trained with a single NVIDIA 2080Ti GPU.

D. Results

1) Repositioning Performance: The first set of experiments are designed to evaluate the overall balancing performance for different algorithms. We train each algorithm with 10,000 epochs, and obtain the test results, as shown in Table I. *Reward* is the average accumulated reward of each agent, which can reflect the overall equilibrium performance. The order response rate (ORR) is the ratio of served orders. Both results of *Reward* and *ORR* are normalized using the result of 6,000 taxis with the **Random** method as the standard.

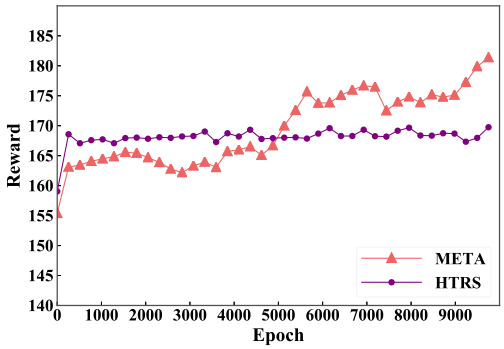

Fig. 4. The training curve in reward of different systems.

As we can see, Reward and ORR increase with the size of the taxi fleet. In each taxi fleet configuration, DDPG almost achieves the best performance in terms of Reward and ORR. The **Random** method is far worse than all the rest methods. **DQN-7** is slightly better than **DQN-5** because the agent in **DQN-7** can output more action values. Compared with the discrete action space of two DQN methods, DDPG has a continuous action space, which means that it can achieve the fine-grained control of agents and generate a better policy.

2) Heterogeneous vs. Homogeneous: To verify the effectiveness of the heterogeneous taxi dispatching design in **META** system, we make a comparison with the Homogeneous Taxi Repositioning System (**HTRS**), which is a typical MARL-based framework [6]. In **HTRS**, it takes the orientations as the action, which is limited discrete action spaces. The **HTRS** determines the optimal orientation for each agent, and dispatches all the available taxis to the neighbor grid (including itself). We compare their performance using a fleet of 8,000 taxis, and the other settings are set default.

Results on the training process of the two systems are shown in Fig. 4 and Fig. 5. We can find that: 1) **META** obtains a higher Reward and ORR than **HTRS** with the increase of the training epoch, demonstrating that **META** is more capable of balancing the demand and supply of taxis in the city; 2) At the early stage of training (i.e., when the training epoch is smaller than 5000), **HTRS** is better than **META** in Reward, which suggests that it is easier to learn the homogeneous repositioning strategy. However, as training progresses, agents in **META** cooperate with each other by repositioning heterogeneous taxis to gain more rewards; 3) The ORR training curves of **META** and **HTRS** fluctuate almost the same as the Reward curves, which shows that a better supply-demand balance brings a higher probability of taking a ride for passengers. 4) Reward

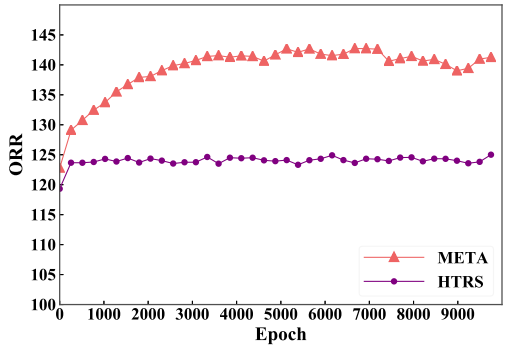

Fig. 5. The training curve in ORR of different systems.

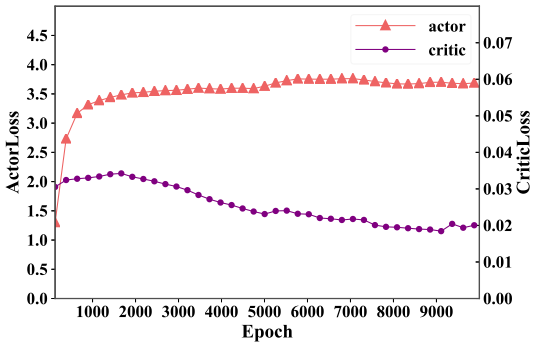

Fig. 6. The training loss of **META**.

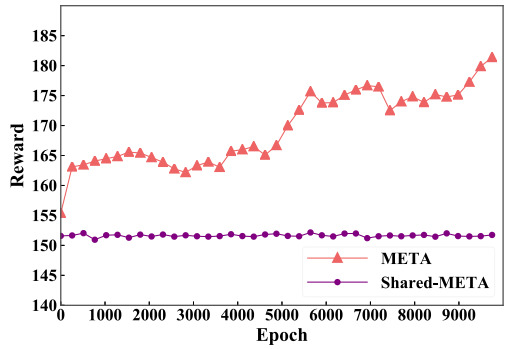

Fig. 7. Comparison results on reward between **META** and **Shared-META** under different training epochs.

in training does not converge for **META** as the training epoch getting bigger, which may be caused by the complex multi-agent environment. In this case, even a small change in policies of agents would play a significant impact at the Reward. As the comparison, we show the training loss results of **META** in Fig. 6. Both networks including the actor network and the critic network converge gradually in terms of the training loss.

After the training, we evaluate the strategies learnt from the two systems with the test data, under the default configuration. **META** obtains 180.39 in Reward and 139.91 in ORR, while **HTRS** gets 165.93 in Reward and 122.30 in ORR. Comparing with **HTRS**, **META** has increased by 8.71% in Reward and 14.4% in ORR, respectively.

3) Impact of Parameter-Shared Networks: To validate the effectiveness of the independent design of **META**, we compare the performance of the proposed **META** system and **META** with shared networks (called **Shared-META**). In the **Shared-META**, all agents use neural network models with the same parameters.

As shown in Fig. 7, we can see that **Shared-META** achieves a stable but always lower Reward than **META** as the training epoch getting bigger, implying that it is difficult for **Shared-META** to learn

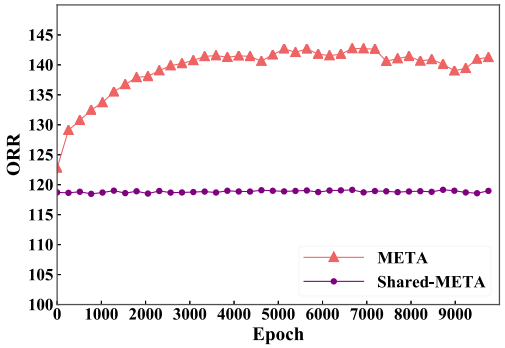

Fig. 8. Comparison results on ORR between **META** and **Shared-META** under different training epochs.

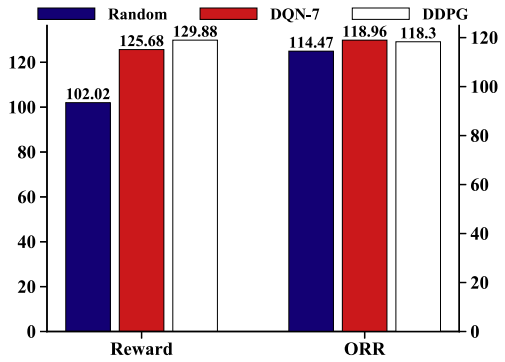

Fig. 9. The performance of the many-to-many dispatching strategy under the default settings of 8000 taxis.

an accurate policy to generate optimal actions. Similarly, as shown in Fig. 8, the ORR of the **Shared-META** remains almost unchanged in the training progress. Since each grid presents its human flow pattern, it is hard to learn such different patterns under a common and shared network setting.

4) One-to-One vs. Many-to-Many: As the second component–*Taxi Dispatching Strategy Formulation*–uses KM algorithm to dispatch taxis, it can only achieve a one-to-one dispatching strategy. To verify the effectiveness of the one-to-one dispatching strategy, we compare it with a many-to-many dispatching strategy based on the maximum network flow.

In the many-to-many dispatching strategy, we built a network flow graph consisting of nodes and links. The node represents a source grid or a sink grid. The supply of the node is the action output of the first component. As for the link, it represents a feasible reposition between a source node and a sink node. We set the capacity of each link to the supply Num_s^i of its source node and set its cost the same as the repositioning cost in Sec. III-B. Finally, the problem is changed to a min cost flow problem [12]. We use the network flow solvers of OR-Tools which is developed by Google [13] to get the optimal solution.

The experiments of the many-to-many dispatching strategy are under the default settings of 8,000 taxis. DQN-5 method has the same algorithm principle as DQN-7 method, so we only evaluate DQN-7 method which has a better performance in the two DQN methods. The results of Random method, DQN-7 method and DDPG method are shown in Fig. 9, its distribution is quite similar to the one-to-one dispatching strategy. For both strategies, the DDPG method gets the best performance in Reward. From the bar chart, we found that the results in Reward of the many-to-many dispatching strategy are worse than the corresponding results of the one-to-one dispatching strategy. Especially for the DDPG method, there is about a 28% drop in the Reward. As mentioned in Sec.III-B, the one-

to-one dispatching strategy could alleviate side effects caused by the dynamic environment in MARL. When we use the many-to-many dispatching strategy in the second component, any changes in the output action of the first component will influence all agents. The environment is more dynamic than the environment of the one-to-one dispatching strategy, which is hard for the RL algorithms to learn the optimal policy.

V. Related Works

A. Taxi Reposition

Repositioning taxis in advance is an imperative way to balance taxi demands and supplies. One trending direction is to associate the problem of repositioning taxi with the *Markov Decision Process (MDP)* model [5], [6], [14], [15]. It has been proved that using MDP could be more effective than the traditional model-based methods [16]. Our work is highly related to those MDP methods in terms of problem setting and methodology. In the MDP methods, the long-term impact of the repositioning strategy can be captured, even if the repositioning strategy is evolving. Among the works using MDP methods, they mainly make use of reinforcement learning [17] to directly learn the optimal action policies by instructing all taxis to interact with the external environment. In [6], a contextual multi-agent reinforcement learning framework is proposed to achieve explicit coordination among a large scale of taxis. In [15], the authors present a novel reinforcement learning framework in a hierarchical way to serve order dispatching tasks and fleet management tasks. These works usually hypothesize the taxis in the same grid are homogeneous, but the real circumstances are often not the case. In this paper, **META** considers the heterogeneous of the taxis in the same grid, so that it can achieve better performance.

B. Reinforcement Learning

Reinforcement Learning [17] is an effective way to solve the MDP problem. Combining with the neural network, deep reinforcement learning [18] uses the neural network to approximate the real function between state space and action space. According to the output of the neural network, the deep reinforcement learning algorithm can be divided into two categories, value-based approaches and policy-based approaches [18]. The value-based approaches output the Q-value which represents the expected reward after taking state-action pair (s, a). The typical algorithm of value-based approaches is deep Q-network (DQN) [19]. The policy-based approaches output the policies directly. It can be the probability distribution of state-action pair [20] or the action value [21]. Although deep reinforcement learning has been successfully applied to many fields, there are still some challenges when applied to practical problems, e.g., repositioning taxis at an urban level. There could be a lot of taxis (agents) in the city, and the city environment is not stationary as it would be influenced by the policy of each taxi. This is fatal to the training of the deep reinforcement learning algorithm.

To mitigate the impact of a dynamical environment on reinforcement learning, multi-agent reinforcement learning (MARL) [7] is presented. The essential idea of MARL algorithms is that if the agents can exchange information with each other, they will make the environment stationary in a sense [22]. MADDPG [22] is the typical MARL algorithm. However, the current MARL algorithms still cannot be applied to the problem with a large-scale agent. Thus, in **META** system, we simplify the MARL algorithm to make it run in the city-wide taxi repositioning system and reduce the dynamics of the environment from other perspectives.

VI. Conclusion and Future Work

In this paper, we proposed a two-stage system framework called **META**, to address the problem of balancing city-wide taxi demands and supplies. **META** applied the MARL to determine the taxi demands/supplies and the KM algorithm to generate repositioning strategies between source grids and sink grids. Finally, we evaluated the **META** system extensively and experiment results demonstrate its effectiveness.

In the future, we will develop a more advanced matching algorithm for the second component in **META** to generate a better repositioning strategy. In more detail, we plan to use RL algorithms to develop many-to-many matching algorithms, which will consider the dynamics of the environment. In this way, the **META** can make strategy more intelligently.

References

[1] B. Li *et al.*, "Hunting or waiting? Discovering passenger-finding strategies from a large-scale real-world taxi dataset," in *Proc. IEEE Int. Conf. Pervas. Comput. Commun. Workshops (PERCOM Workshops)*, Mar. 2011, pp. 63–68.

[2] C. Chen *et al.*, "Crowddeliver: Planning city-wide package delivery paths leveraging the crowd of taxis," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 6, pp. 1478–1496, Jun. 2017.

[3] S. Guo *et al.*, "ROD-revenue: Seeking strategies analysis and revenue prediction in ride-on-demand service using multi-source urban data," *IEEE Trans. Mobile Comput.*, vol. 19, no. 9, pp. 2202–2220, Sep. 2020.

[4] C. Liu, M. Deng, C. Chen, and C. Xiang, "A driver-centric vehicle reposition framework via multi-agent reinforcement learning," in *Proc. Int. Conf. Green, Pervasive, Cloud Comput.*, 2020, pp. 217–230.

[5] Z. Liu, J. Li, and K. Wu, "Context-aware taxi dispatching at city-scale using deep reinforcement learning," *IEEE Trans. Intell. Transp. Syst.*, early access, Nov. 3, 2020, doi: 10.1109/TITS.2020.3030252.

[6] K. Lin, R. Zhao, Z. Xu, and J. Zhou, "Efficient large-scale fleet management via multi-agent deep reinforcement learning," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jul. 2018, pp. 1774–1783.

[7] L. Busoniu, R. Babuska, and B. De Schutter, "A comprehensive survey of multiagent reinforcement learning," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 38, no. 2, pp. 156–172, Mar. 2008.

[8] T. P. Lillicrap *et al.*, "Continuous control with deep reinforcement learning," 2015, *arXiv:1509.02971*. [Online]. Available: http://arxiv.org/abs/1509.02971

[9] A. S. Asratian, T. M. Denley, and R. Häggkvist, *Bipartite Graphs and Their Applications*, vol. 131. Cambridge, U.K.: Cambridge Univ. Press, 1998.

[10] R. Burkard, M. Dell'Amico, and S. Martello, *Assignment Problems: Revised Reprint*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2012.

[11] H. W. Kuhn, "The Hungarian method for the assignment problem," *Naval Res. Logistics Quart.*, vol. 2, nos. 1–2, pp. 83–97, Mar. 1955.

[12] L. R. Ford and D. R. Fulkerson, *Flows in Networks*. Princeton, NJ, USA: Princeton Univ. Press, 2015.

[13] L. Perron and V. Furnon. *Or-Tools*. Google. Accessed: Jul. 12, 2019. [Online]. Available: https://developers.google.com/optimization/

[14] H. Tang, M. Kerber, Q. Huang, and L. Guibas, "Locating lucrative passengers for taxicab drivers," in *Proc. 21st ACM SIGSPATIAL Int. Conf. Adv. Geographic Inf. Syst.*, Nov. 2013, pp. 504–507.

[15] J. Jin *et al.*, "CoRide: Joint order dispatching and fleet management for multi-scale ride-hailing platforms," in *Proc. 28th ACM Int. Conf. Inf. Knowl. Manage.*, Nov. 2019, pp. 1983–1992.

[16] M. Qu, H. Zhu, J. Liu, G. Liu, and H. Xiong, "A cost-effective recommender system for taxi drivers," in *Proc. 20th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2014, pp. 45–54.

[17] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 2018.

[18] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Process. Mag.*, vol. 34, no. 6, pp. 26–38, Nov. 2017.

[19] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529–533, Feb. 2015.

[20] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Proc. Adv. Neural Inf. Process. Syst.*, 2000, pp. 1057–1063.

[21] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *Proc. Int. Conf. Mach. Learn.*, Jan. 2014, pp. 387–395.

[22] R. Lowe, Y. Wu, A. Tamar, J. Harb, O. P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 6379–6390.