Cooperative Multi-Agent Reinforcement Learning in Express System

Yexin Li [1], Yu Zheng [2,3,4,5], Qiang Yang [1]
[1] Hong Kong University of Science and Technology, HKSAR
[2] JD Intelligent Cities Research, Beijing
[3] JD Intelligent Cities Business Unit, Beijing
[4] Institute of Artificial Intelligence, Southwest Jiaotong University, Chengdu
[5] Xidian University, Xi'an
yliby@connect.ust.hk,msyuzheng@outlook.com,qyang@cse.ust.hk

ABSTRACT

Express systems are widely deployed in many major cities. One type of important tasks in the system is to pick up packages from customers in time. As pick-up requests come in real time and there are many couriers picking up packages, how to dispatch couriers to ensure the cooperation among them and to complete more pick-up tasks in a long time, is very important but challenging. In this paper, we propose a reinforcement learning based framework to learn courier dispatching policies. At first, we divide the city into independent regions, inner each of which a constant number of couriers pick up packages at the same time. Besides reducing problem complexity, city division has practical operation benefits. Afterwards, we focus on each region separately. For each region, we propose a Cooperative Multi-Agent Reinforcement Learning model, i.e. CMARL, to learn the optimal courier dispatching policy in it. CMARL tries to maximize the total number of completed pick-up tasks by all couriers in a long time. Our model achieves this target by combining two Markov Decision Processes, one to guarantee the cooperation among couriers, and the other one to ensure the long-term optimization. After obtaining the value functions of these two MDPs, a new value function is designed to trade off them, based on which we can infer the courier dispatching policy. Experiments based on real-world road network data and historical express data from Beijing are conducted, to confirm the superiority of our model compared with nine baselines.

CCS CONCEPTS

• **Information systems** → **Spatial-temporal systems**; • **Theory of computation** → **Reinforcement learning**.

KEYWORDS

Express system, multi-agent reinforcement learning, cooperation

ACM Reference Format:
Yexin Li, Yu Zheng, Qiang Yang. 2020. Cooperative Multi-Agent Reinforcement Learning in Express System. *In Proceedings of the 29th ACM International Conference on Information and Knowledge Management (CIKM'20), October 19–23, 2020, Virtual Event, Ireland.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3340531.3411871

CIKM '20, October 19–23, 2020, Virtual Event, Ireland

ACM ISBN 978-1-4503-6859-9/20/10…$15.00
https://doi.org/10.1145/3340531.3411871

1 INTRODUCTION

Express systems are widely deployed in many major cities, e.g. Beijing, New York City, Paris, etc. providing much convenience to citizens' daily life and largely promoting the development of e-commerce. One important type of tasks in express system is to pick up packages from customers in time. For example, when a customer wants to send a package, he or she logs in the express APP and sends a request, which includes his or her location information, then a courier will go to his or her location to pick up the package.

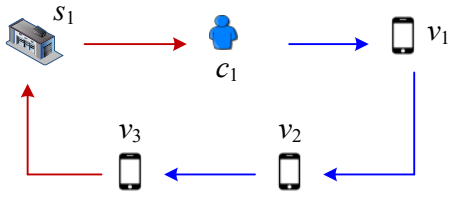


Figure 1: An express system

Figure 1 describes how a courier worked in an express system. Courier c_1 first departed from transit station s_1 and went to pick up package v_1. Later on, another customer sent a pick-up request v_2, then c_1 left his or her current location and went to v_2 to pick it up, so on so forth. After some working time, c_1 returned to transit station s_1 to unload the packages on his or her van. Couriers in the system are required to depart from and return to transit stations by specific time, to fit the schedule of trucks which pick up packages from stations regularly [32][15]. We name this specific time interval as an episode, e.g. 5 hours or one day, etc.

According to practical system operation, each pick-up request should be served in a short waiting time, or its customer may cancel this request. Besides, quick responses from the system can largely improve customers' experiences. Nowadays, many system operators promise and try to serve each request with a waiting time no longer than a threshold ϑ, e.g. one hour. If a customer has waited for a time longer than ϑ, his or her request can be cancelled, and a compensation will be paid to him or her.

Currently, system operators try to complete the massive number of pick-up tasks every day by hiring many couriers. However, this strategy largely increases the operation cost. Besides, immoderate recruitment cannot fundamentally improve operation efficiency. In this paper, we try to operate the system via another strategy, i.e. given a constant number of couriers, we intelligently dispatch them at the beginning of each period, e.g. every ten minutes, thus to complete the most pick-up requests within a waiting time ϑ in each episode.

We partition the city into uniform grids. At the beginning of each period t in the episode, we determine which surrounding grid each courier c_w should go or just stay at the current one. Afterwards, c_w will pick up packages only in the selected grid until the next period $t+1$, when new actions for all the couriers are required again. As we can see, there are nine actions for each courier to select from in each period, i.e. $\{1, 2, 3, ..., 8\}$ for the eight surrounding grids and 0 for the current grid. By continuing choosing working grids for couriers in each period until the end of the episode, we want to maximize the total number of completed pick-up requests, which have been served in a waiting time no longer than ϑ. However, it is a very challenging problem because of the following reasons.

Express systems are large and dynamic. According to historical express data collected in an area of 15×15 square kilometers in Beijing, there are tens of thousands of packages generated in the area every day. In order to complete these tasks, hundreds of couriers are working at the same time in the area. Besides, as pick-up requests can be made by customers whenever they want, practical express systems are often very dynamic.

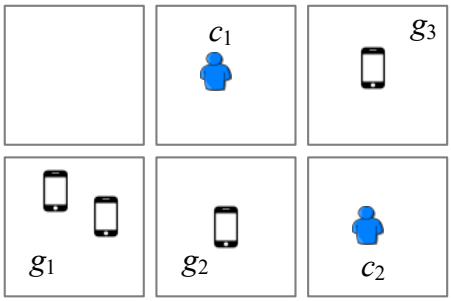


Figure 2: Challenges in system operation

Online dispatching is hard to optimize for a long time. After departing from transit station, each courier chooses where to work in each period. However, each action taken by a courier has long-term effect. Figure 2 gives an example. If courier c_2 goes to grid g_3 instead of g_2, he or she can pick up the same number of packages in the current period because both g_2 and g_3 have only one request. However, as he or she cannot arrive at grid g_1 from g_3 for the two pick-up requests in the next period, g_3 may be a worse choice than g_2 for the current period. Considering this issue, we want to optimize a sequence of actions instead of only the immediate one for each courier. However, even for a single courier, the solution space is too large, i.e. 9^T where T is the number of periods in the episode, not to mention that there are many couriers working at the same time.

Ensuring cooperation among couriers is nontrivial. From Figure 2, we can see that couriers c_1 and c_2 cannot both go to grid g_2 for the only one pick-up request, or there will be conflict between them. However, as there are often hundreds or even thousands of couriers working in the system at the same time, making them work cooperatively is very hard. In each period, if we consider all the n couriers together, the action space is 9^n, which is impractical to handle. If we consider the couriers one by one sequentially in the period, it is very possible that we will get a greedy but not optimal dispatching policy along the courier sequence.

Considering these challenges, we propose a reinforcement learning based framework to learn online courier dispatching policies. Our contributions can be summarized into four-fold.

- We first try to formulate the courier dispatching problem by Central-Agent Reinforcement Learning, i.e. CARL. By theoretical analysis and experiments, we show that CARL cannot ensure long-term optimization in our problem.
- A Cooperative Multi-Agent Reinforcement Learning model, i.e. CMARL, is proposed for our problem. CMARL consists of two Markov Decision Processes, i.e. MDPs, respectively denoted as M_1 and M_2. M_1 is for long-term optimization while M_2 tries to ensure the cooperation among couriers. Based on these two MDPs, we design a new value function to guide where should each courier go and work in real time.
- Although CMARL is proposed for courier dispatching, it is applicable to many systems. Applicable conditions are summarized in this paper.
- Massive experiments based on real-world data from Beijing are conducted to confirm the superiority of our model, compared with nine baselines.

2 PRELIMINARY

Notations and terminologies used throughout this paper are defined in this section. We first partition the city into uniform grids to obtain an $I \times J$ grid map. Each grid is about 500 meters wide and long, which is not large considering that each courier rides a small delivery van when working. Based on the grid map, we then formally define pick-up request and the problem investigated in this paper.

***Definition* 1. Pick-up Request**. A pick-up request is a two-entry tuple, which is denoted as $v_w = (v_w.g, v_w.\tau)$. It means that there is a package pick-up task which was generated at timestamp $v_w.\tau$ with a pick-up location in grid $v_w.g$. Its waiting time cannot be longer than a threshold ϑ, otherwise the customer will cancel v_w and get a compensation.

Problem Statement. Given historical pick-up request data and a constant number of couriers, we try to learn online courier dispatching policies, to guide where should each courier go and work in each period of the episode. In each period t, for each courier, there are nine possible actions for him or her, i.e. he or she can go to one of the eight surrounding grids or just stay at his or her current grid, to pick up packages generated there in t. By continuing choosing working grids for couriers in each period until the

Table 1: Notations

Notations	Descriptions
t	A period in the episode
g_w	A grid in the city
c_w	A courier in the system
n	Number of couriers
v_w	A pick-up request
L_{wt}	Grid where courier c_w is in t
S_{wt}	State of courier c_w in period t
a_{wt}	Action of courier c_w in t
r_{wt}	Immediate reward of courier c_w in t
γ	Discount parameter in MDP

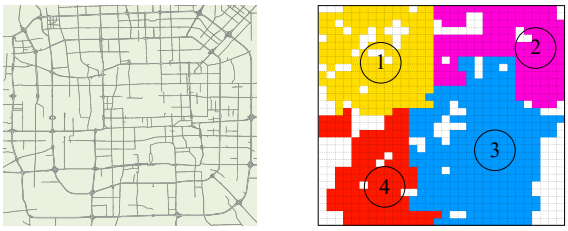
end of the episode, our target is to maximize the total number of completed pick-up tasks, whose waiting time is no longer than ϑ.

3 METHODOLOGIES

Methodologies of our paper are elaborated in this section. Real-world data from Beijing are adopted for examples.

3.1 City Division

As discussed above, the express system is large, making dispatching couriers in it very challenging. Considering this issue, we divide the city into independent regions and focus on each of them separately. Besides to reduce problem complexity, city division also has practical operation benefits, i.e. system operators often prefer managing each region separately instead of the entire city at the same time. In addition, couriers in two areas far away are impossible to interact with each other, thus it is not necessary to consider them together.



A) Beijing urban center B) Four regions

Figure 3: Urban center and regions in it

According to practical settings of the system, there are many choices to divide the city into independent regions, e.g. we can divide the city based on road network, or simply based on geographical distance, etc. In this paper, we adopt an existing method [15] directly. By this method, the city is divided based on historical express data. Main idea of the method consists of two steps, i.e. we first cluster transit stations based on the independence among them, then grids covered by stations in the same cluster make up one region. As city division is not the contribution of this paper, and it is more like a preprocessing step, we do not go into more details about the method but only show some city division results.

Figure 3 A) shows the urban center of Beijing which we investigate in this paper while Figure 3 B) shows the division results of this area, where the grids with same color make up one region. Blank grids mean that there is almost no request there because of some geographical reasons, e.g. it is a lake there. As we can see, we divide the area into four independent regions. In the following context, we focus on each of these regions separately. For simplicity, we still adopt $I \times J$ to denote the grid map of each specific region.

3.2 Courier Dispatching

In this section, we focus on the first region while the other regions can be investigated in a similar way. After departing from transit stations in the first region, couriers choose where to go and work in each period of the episode. As pick-up requests come in real time, and our target is to maximize the total number of completed tasks in a long time, reinforcement learning is adopted to learn the courier dispatching policy in this region.

3.2.1 Markov Decision Process. A reinforcement learning model [26] is often described by a Markov Decision Process, which is made up by six components $\{\mathbf{S}, A, TR, R, \pi, \gamma\}$. State space $\mathbf{S}$ describes the possible states of the agent. Action space A means the actions the agent can take. $\mathbf{S} \times A \times \mathbf{S} \rightarrow TR$ is transition probability, which describes how possible that an agent took action a_t under state S_t will transit to the next state S_{t+1}. Immediate reward function $\mathbf{S} \times A \times \mathbf{S} \rightarrow R$ means the immediate reward received by the agent after taking an action under a state and transiting to the next state. Policy $\mathbf{S} \times A \rightarrow \pi$ describes the probability for the agent to take each action under each state. $\gamma \in [0, 1]$ is a discount parameter. In each period t, the agent under the current state S_t takes an action a_t according to policy π, then transits to the next state S_{t+1}, and receives an immediate reward r_t. In MDP, each action has a long-term return which is defined as Equation 1, where T is the last period of the episode.

$$U_t = r_t + \gamma \times r_{t+1} + \gamma^2 \times r_{t+2} + \ldots + \gamma^{T-t} \times r_T \tag{1}$$

Based on Equation 1, we can define the optimal long-term value function as Equation 2. As we can see, it describes the maximum expected long-term return of each action a_t under each state S_t, by following any policy after period t.

$$Q(S_t, a_t) = \max_\pi E\left[U_t | S_t, a_t, \pi\right] \tag{2}$$

After we obtain the long-term value function, we can easily infer the corresponding optimal policy of this MDP by Equation 3, i.e. we always choose the action which has the maximum optimal long-term value under the current state.

Bellman equation as Equation 4 is often adopted to estimate the optimal long-term value function via an iterative approach. As we can see, it is based on a parametric form of $Q(x, y)$ and a sample pool whose samples are four-entry tuples (S_t, a_t, S_{t+1}, r_t). In real-world

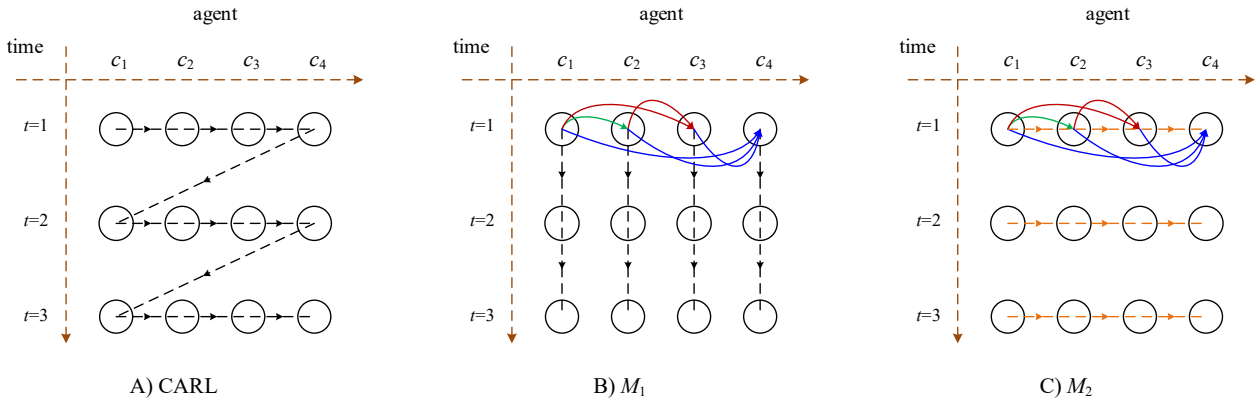


Figure 4: Courier dispatching methodologies

problems, because of the large state and action spaces, deep neural networks are often adopted as the parametric form of $Q(x, y)$.

$$a_t^* = \text{argmax}_a Q(S_t, a) \tag{3}$$

$$Q(S_t, a_t) = r_t + \gamma \times \max_a Q(S_{t+1}, a | S_t, a_t) \tag{4}$$

3.2.2 Sequential Dispatching Setting. Considering that there are many couriers working in the region at the same time, we have two choices to dispatch them in each period. First, we can consider all the n couriers in the region at the same time, i.e. we choose a joint action for them. By this strategy, the action space of each step is 9^n. As there are often tens or even hundreds of couriers in the region, it is impractical to handle such a large action space. On the contrary, the second choice is to consider the couriers one by one sequentially, i.e. in each period t, we assign an action to courier c_1 first, then we assign another action to c_2, so on so forth until all the n couriers have got their actions in t. After that, they will conduct their actions simultaneously at the beginning of t, and work in their new grids until $t + 1$, when new actions will be assigned to them sequentially again. Figure 4 A) gives an example to elaborate this sequential dispatching setting, where there are 4 couriers in the region, i.e. along the horizontal axis, and 3 periods in the episode, i.e. along the vertical axis.

Under sequential dispatching setting, there are only nine choices at each step, which is very feasible. However, it brings another issue, i.e. the cooperation among couriers is hard to guarantee because we consider couriers one by one in each period. It is very possible that we will get a greedy but not optimal result along the courier sequence. In this paper, we focus on the second strategy to make our model trainable and try to ensure cooperation among couriers.

In the following context, to ensure the cooperation among couriers, we first try to formulate the dispatching problem based on Central-Agent Reinforcement Learning, i.e. CARL, which considers the system center as the central and only agent. As there is only one agent, cooperation issue does not exist in CARL. However, by theoretical analysis and experiments, we show that CARL sacrifices long-term optimization for courier cooperation in our problem. Because of this, we try to formulate the problem based on Multi-Agent Reinforcement Learning, i.e. MARL, which considers each courier as an agent. By MARL, long-term optimization can be guaranteed while cooperation among couriers becomes the issue. Considering this, we improve MARL to Cooperative MARL model, i.e. CMARL, which tries to maximize the total completed tasks by all couriers in a long time. After investigating CARL, MARL, and CMARL, we summarize the problems which CMARL is applicable to.

3.2.3 Central-Agent Reinforcement Learning. As claimed above, CARL considers the system center as the central and only agent. Based on the example given in Figure 4 A), we introduce CARL. In each period t, the system center generates action a_{wt} to each courier c_w one by one. As in Figure 4 A), the process of CARL in the episode is $S_{11} \xrightarrow{a_{11}} S_{21} \xrightarrow{a_{21}} S_{31} \xrightarrow{a_{31}} S_{41} \xrightarrow{a_{41}} S_{12} \xrightarrow{a_{12}} S_{22} \xrightarrow{a_{22}} \dots \xrightarrow{a_{33}} S_{43}$ where S_{wt} means the state of c_w in period t. In this example, it means that the system center assigns action a_{11} to courier c_1 in $t = 1$, when the state is S_{11}, and transits to the next state S_{21} of courier c_2 in $t = 1$, then the center assigns action a_{21} to c_2, so on so forth until the last courier c_4 gets his or her action in $t = 1$. After that, all the couriers conduct their actions simultaneously at the beginning of $t = 1$. After $t = 1$, we go to the next period $t = 2$, when the system generates actions to couriers in this period one by one again, so on so forth until the last period.

Adopting Bellman iteration to train the MDP defined above, the sample pool is made up by tuples $\left(S_{wt}, a_{wt}, S_{(w+1)t}, r_{wt}\right)$ when c_w is not the last courier, or $\left(S_{wt}, a_{wt}, S_{1(t+1)}, r_{wt}\right)$ when c_w is the last courier in the region. However, in each period t, couriers get their actions one by one but conduct them at once after all of them have got their actions in t. It means we cannot observe S_{21} directly after assigning a_{11} under S_{11} but can only approximate it.

State. For period t, a courier c_w has a state consisting of four elements as $S_{wt} = (\Upsilon_{wt}, W_{wt}, L_{wt}, t)$. Υ_{wt} means the already came but not served pick-up requests. W_{wt} describes where the other couriers are. Lwt is the current location of c_w in period t. It is

intuitive that W_{wt} and L_{wt} vary over c_w in the same period t. Besides, as actions are generated to couriers one by one, the state of a courier in t should also be impacted by the already determined actions of other couriers before him or her, i.e. Υ_{wt} varies over c_w in t as well. We elaborate this by a running example in Figure 5.

Assuming there are four couriers c_1, c_2, c_3 and c_4 in the region, whose current locations are respectively g_3, g_9, g_{11} and g_6 as shown in Figure 5 A). Figure 5 B) describes the already came but not completed pick-up requests, where each grid with a number means how many requests are located there. For c_1, the request matrix in Figure 5 B), the distribution of the other three couriers in Figure 5 C), i.e. one in g_9, one in g_{11} and one in g_6, his or her current location g_3, and the current period t, make up the state S_{1t}. Based on S_{1t}, an action is assigned to c_1. We assume this action is *go to* g_2 as shown in Figure 5 D). After c_1 got this action, we consider c_2. Firstly, we approximate [15] how many requests are expected to be completed by c_1 inner g_2 in t after he or she will conduct the assigned action. We assume this approximation as $\delta_1 = 3$ in our example. How to obtain this approximation is very simple and will be intuitive after designing the system simulator in section 4.1. Updating the not completed request matrix in Figure 5 B) by reducing δ_1 from the entry corresponding to g_2, we obtain the request matrix for c_2 as Figure 5 E). Generate the distribution of other couriers for c_2 as Figure 5 F), where the location of c_1 has been updated to g_2 considering the action he or she got. Finally, Figure 5 E), F), the current location g_9 and the current period t make up the state of c_2. Repeat these steps until all couriers are considered in period t, then let them conduct their actions, and we then go to the next period.

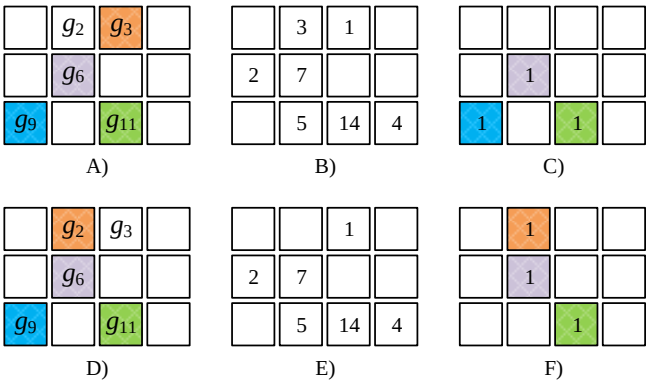


Figure 5: State for each courier in one period

Action and immediate reward are simply defined in CARL. Action $a_{wt} \in \{0, 1, 2, ..., 8\}$ means which grid the system center dispatches courier c_w to at the beginning of period t. Immediate reward r_{wt} is the number of pick-up requests completed by courier c_w inner his or her new grid in period t.

Long-term optimization issue. Although CARL does not have cooperation issue, it can only maximize the total completed tasks in a short but not a long time. Here is the reason. Assume that we adopt Bellman iteration to train the MDP, and the discount parameter γ in Equation 4 is not set large enough, e.g. $\gamma = 0.9$. Under this setting, when there are $n = 40$ couriers in the region for example, the cumulative discount coefficient after the current period t will be smaller than $\gamma^{40} \approx 0.01$. It means that the following periods are almost not considered, not to mention that there may be much more than 40 couriers in the region. But on the contrary, if we set γ larger, the training of CARL in our problem cannot converge. It is also reasonable because how we define the state of the MDP is based on approximating. In a very uncertain system, setting the discount parameter too large is very possible to make Bellman iteration not converge.

Although we only discuss about adopting Bellman iteration to train MDP in this paper, similar issues also exist to other methods, e.g. policy gradient and actor-critic [26], etc. Because by our sequential dispatching setting, we largely extend the length of the MDP, making uncertainty accumulate more. If the uncertainty of each state or state transition is nontrivial, it is too hard to ensure the convergence of these training methods.

3.2.4 Multi-Agent Reinforcement Learning. As CARL cannot guarantee long-term optimization, we try to formulate the dispatching problem based on MARL. Figure 4 B) gives an example of MARL, where each courier is considered as an agent, and they share a common MDP M_1, which is denoted by black dashed lines. At this time, the action generating process is still the same with that shown in Figure 4 A), i.e. actions to couriers are generated one by one in each period. But the MDP changes, i.e. M_1 for each courier c_w in the episode is $S_{w1} \xrightarrow{a_{w1}} S_{w2} \xrightarrow{a_{w2}} S_{w3}$, meaning that the sample pool to train M_1 by Bellman iteration is made up by four-entry tuples $\left(S_{wt}, a_{wt}, S_{w(t+1)}, r_{wt}\right)$. Here the state, action, and immediate reward are defined in the same way as those in 3.2.3.

In fact, the current model is very similar with an existing one [15]. In this study, the authors claimed that their model can guarantee the cooperation among couriers to some extent. Because when they designed the state for courier c_w in period t, actions already got by couriers before c_w in t were also considered. In our model, this is also the case. For example, in Figure 5, when we design the state of c_2 in t, the action courier c_1 got in t is considered. Because we update the uncompleted request matrix Υ_{21} and the courier distribution matrix W_{21} for c_2, after c_1 got his or her action. Similarly, when we design the state of c_3 in t, the actions c_1 and c_2 got are considered, when we design the state of c_4 in t, the actions c_1, c_2, and c_3 got are considered, etc. Colorful arrows in Figure 4 B) describe this correlation among states of couriers in the same period.

According to previous works [15], our analysis, and the experiment results in this paper, we agree that MARL can guarantee the cooperation among couriers to some extent. For example, in Figure 2, after dispatching c_1 to g_2 in t, considering this action, the expected number of uncompleted tasks in g_2 becomes zero, making dispatching c_2 to g_2 in t less possible, thus avoiding the conflict between c_1 and c_2. However, this idea to guarantee the cooperation among couriers is still a greedy strategy instead of an optimal one, i.e. we choose the best action for courier c_1 under his or her current state, then we update the state of c_2 and choose the best action for c_2, so on so forth. In order to guarantee the cooperation among couriers better, we propose CMARL model.

3.2.5 Cooperative Multi-Agent Reinforcement Learning. In this subsection, we first design a third MDP M_2 to guarantee the cooperation among couriers in each period t. M_2 is a central-agent MDP, where the system center is the central and only agent, and generates actions to couriers in each period one by one. An example of M_2 is shown in Figure 4 C), i.e. the yellow dashed lines. It is shared by all the periods in the episode. M_2 tries to optimize the sequence of actions generated to couriers by the system in each period. For any period t, its process is $S_{1t} \xrightarrow{a_{1t}} S_{2t} \xrightarrow{a_{2t}} S_{3t} \xrightarrow{a_{3t}} S_{4t}$, meaning that the sample pool to train M_2 by Bellman iteration is made up by four-entry tuples $\left(S_{wt}, a_{wt}, S_{(w+1)t}, r_{wt}\right)$. Here the state, action, and immediate reward are defined in the same way as those in 3.2.3.

As we can see, because M_2 is based on CARL, it does not have the cooperation issue, i.e. couriers in each period can cooperate well. However, as it only optimizes the sequence of actions in each period separately, it cannot guarantee the long-term optimization target in the episode. But on the contrary, we know that M_1 in 3.2.4 or Figure 4 B) can guarantee the long-term optimization but not the cooperation among couriers. Consequently, we consider M_1 and M_2 as two complementary MDPs. Specifically, M_1 tells each courier how to choose an optimal action considering a long time while M_2 tells each courier how to select an optimal action considering all the couriers in the region. It is reasonable to combine them to guide where each courier should go in each period, to maximize the total completed tasks by all couriers in a long time.

Motivated by the above idea, our CMARL model is made up by M_1 and M_2. Bellman iteration to train M_1 is as Equation 5 while Bellman iteration to train M_2 is as Equation 6, where γ_1 and γ_2 are discount parameters respectively for M_1 and M_2. From Equation 5, we can see that the state transits from S_{wt} to $S_{w(t+1)}$, i.e. vertically, while in Equation 6, it transits from S_{wt} to $S_{(w+1)t}$, i.e. horizontally, matching their definitions shown in Figure 4 B) and C). In other words, M_1 is for long-term optimization while M_2 is for cooperation.

$$Q_1(S_{wt}, a_{wt}) = r_{wt} + \gamma_1 \times \max_a Q_1(S_{w(t+1)}, a|S_{wt}, a_{wt}) \quad (5)$$

$$Q_2(S_{wt}, a_{wt}) = r_{wt} + \gamma_2 \times \max_a Q_2(S_{(w+1)t}, a|S_{wt}, a_{wt}) \quad (6)$$

After obtaining Q_1 and Q_2, a new value function is defined as Equation 7, where α is a parameter to trade off M_1 and M_2. In the online dispatching process, for each courier c_w in period t, we obtain its state S_{wt} first, then choose the action whose new value is the maximum, i.e. $a^*_{wt} = \text{argmax}_{a_{wt}} Q(S_{wt}, a_{wt})$.

$$Q\left(S_{wt}, a_{wt}\right) = Q_1\left(S_{wt}, a_{wt}\right) + \alpha \times Q_2\left(S_{wt}, a_{wt}\right) \quad (7)$$

Deep networks are designed to estimate the value functions Q_1 and Q_2 via Bellman iteration. Many studies [17][28][31][30][11] in deep learning [6] can be referred to. In our work, we simply adopt very basic layers, e.g. convolutional layers, fully connected layers, embedding layers, etc. to construct the networks for Q_1 and Q_2.

3.3 Model Applicability

Although we propose CMARL for courier dispatching problem, the model can be widely applied to many similar problems. As we can see, except when designing each component of the MDPs, i.e. state, action, and immediate reward, no more specific settings of express system are considered or incorporated in CMARL. By reviewing 3.2.3, 3.2.4, and 3.2.5, we summarize that to any problem, which meets the following two conditions, CMARL is applicable.

Condition 1. Many agents work in the system at the same time. Specifically, the number of agents is not strictly constrained. It can be tens or even hundreds.

Condition 2. System dynamics is stochastic, thus how the states of the MDPs transit is very unsure. For example, in express system, when adopting Bellman iteration to train the MDPs, the discount parameters cannot be set too large.

For problems which meet these two conditions, CMARL tries to generate actions to all agents one by one in each period of the episode, thus to optimize the final target. In other words, CMARL tries to guarantee the cooperation among agents and the long-term optimization in the episode at the same time.

Although in this paper, we only adopt Bellman iteration for model training, other existing methods can be adopted as well, e.g. policy gradient and actor-critic [26], etc. Issues discussed in CARL and MARL still exist when adopting other methods. For example, about the long-term optimization issue in 3.2.3, because by our sequential dispatching setting, we largely extend the length of the MDP of CARL, making uncertainty accumulate. If the uncertainty of state or state transition is nontrivial, no matter by which method, it is too hard to ensure the convergence when training the model. About the cooperation issue in MARL, it obviously cannot be addressed by any training method. As a result, CMARL is still a better choice than CARL and MARL to this kind of problems, no matter which training method is chosen.

4 EVALUATION

Based on road network data and historical express data in Beijing, we conduct experiments by CMARL and nine baselines. Express data are provided by one of the largest e-commerce platforms in China. Figure 3 A) shows the urban center where the data have been collected. Data details are summarized in Table 2. In our experiments, we set the episode as 8:00am - 1:00pm, i.e. half day, because it matches how couriers work in the real system. Besides, we also conduct experiments when setting the episode as 2:00pm - 7:00pm while dismissing the analysis and discussions, since the results are similar with those of 8:00am - 1:00pm.

As introduced in 3.1, we first divide the urban center into four regions, then we separately focus on each of the regions. Experiment settings for each region are summarized in Table 3. In the table, # requests means the number of requests each region is expected to have in the episode, which is estimated based on historical express data. According to the workload in each region, we allocate a number of couriers to work in it. In fact, the number of couriers being allocated to each region needs to be decided based on practical system settings, e.g. the workload, the budget, etc. However, in our experiments, we try different settings of the number of couriers

Table 2: Real-world data and simulator parameters

Data	time	1st, Aug. - 15th, Aug. 2018
	Range	$\approx 15 \times 15$ km
	# grids	30×30
	# transit stations	106
	# couriers	1, 786
	# parcels	531, 920
Simulator	Courier speed	5 meters / second
	ϑ	1 hour
	t_ϵ	2 minutes

Table 3: Experiment settings for each region

Region	# requests / episode	# couriers
1	2365	{10, 20, ..., 70}
2	3338	{10, 20, ..., 70}
3	6854	{80, 90, ..., 150}
4	4190	{10, 20, ..., 70}

in each region, because our target is to evaluate the dispatching performance of each model under a random given number of couriers. As we can see from Table 3, because the workload in the third region is obviously larger than the others, we set the number of couriers in it as $n = \{80, 90, ..., 150\}$ while n in other regions is set as $n = \{10, 20, ..., 70\}$. Other values can also be tried, but these values we used are representative enough considering the workload in each region. Figure 6 shows the expected request distribution in each region, the darker, the more requests located in the grid in the episode. Some blank grids do not necessarily mean that there is no request in them, but the requests in them are too few to be shown.

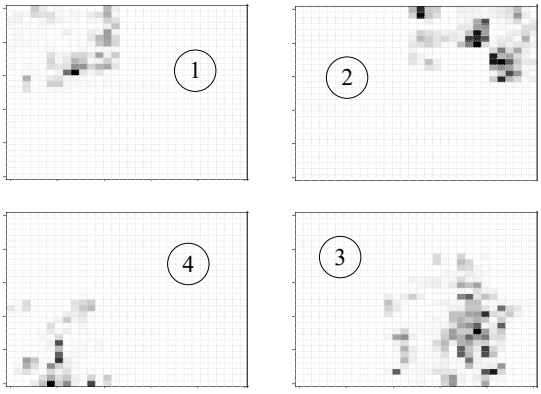


Figure 6: Request distribution in each region

4.1 System Simulator

A system simulator is required to train and evaluate our model and baselines. It simulates how the express system operates in each period of the episode.

At first, we estimate the distance between any two tasks in each grid based on road network data. Assuming that the distance between any two road nodes e_v and e_k is ϕ_{vk}, we estimate the task distance inner grid g_w with a Normal Distribution N_w which fits ϕ_{vk}, where $e_v, e_k \in rn_w$, and rn_w is the road network in g_w. Secondly, based on historical request data, we fit Normal Distributions to generate the number of pick-up requests that will come in each period and located in each grid. After obtaining these distributions, we simulate how the system operates.

Each courier is initially at a random station. In each period t of the episode, we first generate the number of requests that will come to each grid in t, then we update the requests in the system, i.e. we delete the requests which have waited for a time longer than ϑ and add those that will come in t. Afterwards, our simulator simulates the activities of each courier in t one by one. Each courier in the region works as follows.

Step 1. Courier c_w conducts his or her action, i.e. goes to grid L_{wt} to work in period t.
Step 2. If there is any not completed request in L_{wt} and there is remaining time in period t, generate a distance $d_x \sim N_{wt}$ for c_w to go and pick up a package, here N_{wt} is the task distance distribution in grid L_{wt}. Duration for this pick-up task is estimated by Equation 8, where vr is the speed of the courier or the delivery van, and t_ϵ is a constant additional time needed by each task, e.g. the time for package checking or form filling, etc.

$$t_x = \frac{d_x}{vr} + t_\epsilon \tag{8}$$

Check whether the remaining time in t is enough for t_x. If yes, let c_w conduct this pick-up task; update the remaining time in t by deducting t_x; update the remaining requests in L_{wt} by deducting one; then we repeat step 2. Otherwise, c_w terminates his or her working in period t.

Parameters in our simulator, i.e. waiting time threshold ϑ, courier speed vr, and t_ϵ, are summarized in Table 2. Simulating the activities of each courier one by one in each period t matches the sequential action generating process, forcing each courier to consider the already determined actions of others properly, i.e. the meaning of those colorful arrows in Figure 4.

4.2 Metric and Baselines

Metric. Percent of Completed pick-up Requests, i.e. PCR, in the episode is adopted as the metric in our experiments.

Nine baselines are adopted to compare with CMARL, among which the first three are heuristic algorithms while the others are reinforcement learning based models.

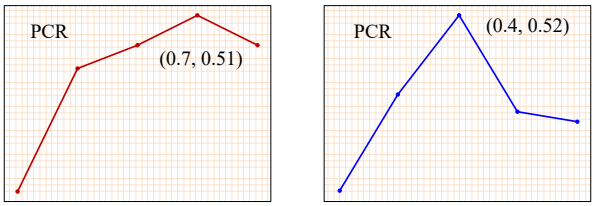
- **Random**. Select a random possible action each time.
- **Greedy**. Select the grid whose number of remaining pick-up requests is maximum among the nine choices.
- **Cooperative Greedy - CG**. Similar with Greedy, but each time a courier gets his or her action, we update the expected remaining pick-up tasks in the corresponding grid. It is the same with what we do when designing the state.

- **Independent DQN - IDQN** [4][27]. Consider each courier as an agent but ignore the cooperation among them. Couriers share a common MDP. IDQN is simple but often adopted to solve multi-agent problems.
- **tMARL** [12]. tMARL is similar with M_1. However, in each period t, for each courier c_w, the state is $S_{wt} = (\Upsilon_{wt}, L_{wt}, t)$, i.e. we do not know the global information of other couriers in the region.
- M_1 **in CMARL** [15]. Only consider the MDP M_1 in 3.2.4. It is almost the same with an existing model [15].
- **Degenerating** M_2 **- DM**. Similar with the MDP M_2 in 3.2.5 but do not update the expected remaining requests according to those already generated actions.
- M_2 **in CMARL**. Only consider the second MDP M_2 in 3.2.5.
- **CARL**. Central-Agent Reinforcement Learning introduced in section 3.2.3.

4.3 Parameters

As we can see, there are three parameters to tune in CMARL, i.e. the discount parameter γ_1 in M_1, the discount parameter γ_2 in M_2, and the parameter α in Equation 7 to trade off M_1 and M_2. In this subsection, we elaborate how to tune each parameter for the first region when the number of couriers is set as $n = 20$. How to tune the parameters when n varies or for other regions is similar.

For parameter γ_1, we respectively set it as $\gamma_1 = \{1, 0.9, ..., 0.6\}$ and conduct M_1 to compare the PCR under each setting. Results are shown in Figure 7 A). As we can see, when $\gamma_1 = 0.7$, M_1 obtains the largest PCR, thus we set $\gamma_1 = 0.7$ when $n = 20$ in the first region. For γ_2, we respectively set it as $\gamma_2 = \{0.6, 0.5, 0.4, 0.3, 0.2\}$ and conduct M_2 to compare the PCR under each setting. Results are shown in Figure 7 B). As we can see, when $\gamma_2 = 0.4$, M_2 obtains the largest PCR, thus we set $\gamma_2 = 0.4$ when $n = 20$ in the first region.



A) Parameter γ_1 B) Parameter γ_2

Figure 7: Parameter tuning

For α, under $\gamma_1 = 0.7$ and $\gamma_2 = 0.4$, we respectively set it as $\alpha = \{0.4, 0.8, 1.2, 1.6, 2\}$ and conduct CMARL to compare the PCR under each setting. We dismiss the figure because of space reason. But from the results, we know that when $\alpha = 1.2$, CMARL obtains the largest PCR, thus we set $\alpha = 1.2$ when $n = 20$ in the first region.

After tuning parameter α under each value of n in the first region, we obtain $\alpha = 0.8, 1.2, 1.2, 1.2, 1.6, 1.6, 1.6$ respectively for $n = 10, 20, 30, 40, 50, 60, 70$. As we can see, when the number of couriers increases, the more important the cooperation among couriers becomes, thus a larger weight α is assigned to M_2, trying to guarantee the cooperation among couriers better.

Table 4: PCR in Region 1

# couriers	10	20	30	40	50	60	70
Random	0.10	0.19	0.27	0.34	0.41	0.47	0.53
Greedy	0.30	0.44	0.49	0.51	0.52	0.52	0.52
CG	0.30	0.46	0.51	0.53	0.54	0.55	0.56
IDQN	0.29	0.49	0.60	0.65	0.74	0.80	0.83
tMARL	0.28	0.50	0.65	0.76	0.84	0.86	0.91
M_1	0.28	0.51	0.66	0.78	0.84	0.88	0.92
DM	0.29	0.49	0.65	0.79	0.85	0.90	0.92
M_2	0.28	0.52	0.72	0.80	0.88	0.94	0.95
CARL	0.27	0.50	0.70	0.79	0.88	0.94	0.96
CMARL	**0.31**	**0.54**	**0.73**	**0.85**	**0.92**	**0.95**	**0.96**

4.4 Performance Comparison

In this section, we focus on the first region. Experiment results and analysis of other regions are similar. Given a constant number of couriers in the first region, i.e. $n = \{10, 20, ..., 70\}$, we respectively conduct experiments by the nine baselines and our model. PCR of each model is summarized in Table 4.

As we can see, among the first three heuristic algorithms, Random performs the worst. It is reasonable as Random does not consider any optimization target. Unexpectedly, when there are not too many couriers, i.e. when $n = 10$, Greedy and CG, which do not consider any cooperation or do not consider cooperation enough, can work as well as our model. However, it is also reasonable because there is rare competition among couriers when $n = 10$, i.e. the region is not crowded at all. Besides, considering that pick-up requests in the region are too many for only $n = 10$ couriers, long-term optimization is not necessary to be explicitly considered under this situation. In other words, because couriers always have enough packages to pick up in each period, there is almost no labor waste. However, when the number of couriers increases, Greedy and CG perform worse than those reinforcement learning based baselines and our model obviously. CG obtains better results than Greedy. Because CG considers the cooperation among couriers to some extent, i.e. when assigning actions to couriers, the actions already generated to some couriers before are also considered, which is similar with the way how M_1 considers the cooperation among couriers. However, CG performs worse than M_1 when $n > 10$, because it does not have a long-term optimization target as M_1 does.

Besides the three heuristic algorithms, we have another six reinforcement learning based baselines, among which the first three are existing models while the other three are degenerated from CMARL proposed in this paper. IDQN is the simplest one which considers each courier as an independent agent without any cooperation with others, thus it is reasonable that it performs worse than M_1 which considers the cooperation among couriers to some extent, when $n > 10$. However, as IDQN has a long-term optimization target, it is still better than the three heuristic algorithms. tMARL does not consider the global information of all couriers in the region as M_1 does, therefore, it also performs worse than M_1, proving that the second component W_{wt} in our state is reasonable and necessary.

DM, M_2, and CARL are degenerated from CMARL proposed in this paper. As introduced in 3.2.5, M_2 tries to maximize the completed pick-up tasks by all couriers in each period. As we can see,

Table 5: PCR in Region 2

# couriers	10	20	30	40	50	60	70
Random	0.09	0.17	0.24	0.31	0.37	0.43	0.48
Greedy	0.24	0.39	0.46	0.50	0.52	0.53	0.54
CG	**0.24**	0.40	0.48	0.52	0.54	0.56	0.57
IDQN	0.22	0.38	0.52	0.61	0.67	0.69	0.74
tMARL	0.22	0.39	0.53	0.62	0.72	0.78	0.81
M_1	0.22	0.40	0.54	0.65	0.74	0.78	0.83
DM	0.23	0.39	0.59	0.69	0.77	0.83	0.85
M_2	0.21	0.40	0.59	0.70	0.80	0.85	0.90
CARL	0.21	0.40	0.61	0.67	0.77	0.85	0.89
CMARL	0.23	**0.43**	**0.61**	**0.73**	**0.81**	**0.87**	**0.91**

Table 6: PCR in Region 3

# couriers	80	90	100	110	120	130	140	150
Random	0.28	0.31	0.34	0.37	0.40	0.42	0.45	0.47
Greedy	0.48	0.50	0.51	0.52	0.52	0.53	0.53	0.54
CG	0.52	0.53	0.54	0.55	0.56	0.57	0.57	0.58
IDQN	0.52	0.50	0.55	0.58	0.60	0.62	0.62	0.66
tMARL	0.52	0.59	0.62	0.66	0.70	0.69	0.72	0.73
M_1	0.53	0.61	0.61	0.67	0.70	0.70	0.75	0.76
DM	0.54	0.60	0.66	0.71	0.74	0.77	0.79	0.82
M_2	0.56	0.61	0.66	0.71	0.76	0.79	0.81	0.83
CARL	0.59	0.62	0.66	0.70	0.75	0.79	0.82	0.85
CMARL	**0.61**	**0.64**	**0.69**	**0.75**	**0.77**	**0.82**	**0.83**	**0.87**

Table 7: PCR in Region 4

# couriers	10	20	30	40	50	60	70
Random	0.05	0.11	0.16	0.20	0.25	0.29	0.33
Greedy	0.18	0.33	0.43	0.50	0.55	0.58	0.61
CG	0.19	0.34	0.45	0.53	0.58	0.63	0.66
IDQN	0.17	0.32	0.42	0.51	0.55	0.61	0.61
tMARL	0.17	0.32	0.46	0.55	0.69	0.72	0.81
M_1	0.18	0.32	0.48	0.58	0.69	0.72	0.82
DM	0.18	0.33	0.45	0.56	0.63	0.72	0.75
M_2	0.17	0.31	0.45	0.58	0.67	0.73	0.79
CARL	0.17	0.31	0.47	0.60	0.67	0.76	0.82
CMARL	**0.19**	**0.36**	**0.51**	**0.64**	**0.71**	**0.80**	**0.85**

when n is small, it performs worse than many other models. Because at this time, the region is not crowded at all, thus the cooperation among couriers is not very necessary to be considered explicitly. Besides, M_2 does not consider the long-term optimization target, and the additional cooperation constraint may make M_2 harder to train, leading to its poorer performance. However, as n increases, M_2 performs better and better because the region becomes more and more crowded, improving the possibility for couriers to have competition. At this time, the cooperation among couriers becomes more and more important. DM performs worse than M_2, proving the reasonableness of our state formulation in 3.2.3, i.e. the necessity of those colorful arrows in Figure 4. CARL can guarantee the cooperation among couriers as M_2 does, but because of the long-term optimization issue, it cannot guarantee long-term optimization. Besides, the MDP of CARL is much longer than others, making it harder to train. As a result, we can see its performance is very unstable, i.e. sometimes it is better than M_2 while sometimes it performs worse than M_2. Our model CMARL, which tries to ensure the long-term optimization in the episode and the cooperation among couriers in the region at the same time, performs obviously better compared with the nine baselines.

Besides the final PCR results, we also investigate the convergence processes of the reinforcement learning based models in this paper, i.e. CMARL and the last six baselines. We see that all these models converge well. However, because of the space reason, we dismiss their figures in this section.

Experiment results in other regions are summarized in Table 5, Table 6 and 7. Specifically, in the third region, the number of couriers is respectively set as $n = 80, 90, \ldots, 150$ instead of $n = 10, 20, \ldots, 70$ as those in the other regions. Because the workload in it is much larger than those in the first, second, and fourth regions. As we can see, experiment results in the other three regions are very similar with those in the first region, confirming that our model is applicable to and has stable performances in different environments.

5 RELATED WORK

Express System Studies. Express systems are widely deployed in many major cities, promoting many studies on large-scale city express [24][21][32][2][15]. In an express system, there are often two types of tasks, i.e. delivering parcels to customers and picking up packages from customers. Zhang et al. [32] proposed some heuristic algorithms to guide couriers to deliver and pick up packages at the same time in the system. Li et al. [15] proposed a multi-agent reinforcement learning based model to manage the couriers to complete these two types of tasks. Problem settings of their study [15] are similar with ours, except that we focus on the pick-up tasks only in this paper. However, our setting is also reasonable, because in many express systems, system operators adopt two different groups of couriers to deliver and pick up packages separately. Besides, our model can be easily extended to solve the problem where both of these two kinds of tasks are considered at the same time. Some studies tried to adopt crowdsourcing to complete the package delivering tasks, e.g. Sadilek et al. [24] asked a group of twitter users to deliver packages, Mclnerney et al. [21] employed mobile users to help to deliver packages, Chen et al. [2] exploited existing taxi business to deliver packages to their customers. As our problem is to cooperatively manage a constant number of couriers to complete more requests in a long time, previous models cannot solve our problem properly.

System Operation. Besides studies on express system, many studies on operation of spatio-temporal systems have also been conducted. Lin et al. [18] gave a solution to manage the large-scale fleet for ride-sharing platform. Wei et al. [29] tried to intelligently control the traffic light to minimize the average waiting time of each vehicle. For bike-sharing system, there are many studies [14][8][19][5][13][16] about how to predict bike usage demand, and then redistribute bikes among stations in the city to reduce the customer loss. As a specific operation problem with specific system settings, we cannot directly adopt these previous models to solve our problem.

Deep and Multi-Agent Reinforcement Learning. In many practical problems, the state or action space is large, therefore, traditional reinforcement learning cannot perform effectively. Deep Reinforcement Learning, i.e. DRL, which adopts the amazing representation extraction ability of deep learning, addresses this issue in many spatio-temporal system operation problems [18][14][29][15].

Besides, many models based on DRL for recommendation problem have been proposed, e.g. Chen et al. [3] designed a robust DQN method to provide better recommendation in an e-commerce platform, Hu et al. [7] used DRL to learn an optimal ranking policy for search engine, etc. In some problems, DRL can even beat human [22][23][25]. Multi-agent reinforcement learning [1][20] has also attracted much attention. Tampuu et al. [27] analyzed the cooperation and competition among two agents in reinforcement learning by carefully designing the reward schemes. Kok et al. [9] tried to learn the coordinated actions of a group of cooperative agents by using sparse representation of joint state-action space. Lin et al. [18] designed a multi-agent model to manage many vehicles in the city. Lee et al. [10] further improved cooperative models by mixing demonstrations from centralized policy. However, these models cannot be directly adopted to solve our problem considering the practical settings of express system.

6 CONCLUSION

In this work, we try to dispatch couriers in express system in real time, thus they can pick up packages from customers cooperatively and optimally in a long time.

At first, to reduce the problem complexity and because of some practical system management benefits, we divide the city into independent regions, in each of which there are a constant number of couriers working at the same time. Afterwards, we respectively focus on each region without considering the others. Before going into our proposed model, we theoretically analyze why CMRL cannot guarantee long-term optimization in the episode, and why MARL cannot guarantee global cooperation among couriers. After that, we propose CMARL to dispatch couriers in each region in real time. CMARL is based on two MDPs, one for long-term optimization and one to ensure the cooperation among couriers. After obtaining the optimal long-term value functions of these two MDPs, we design a new value function to trade off them, based on which we can easily guide where should each courier go and work in each period of the episode. Our model tries to maximize the total completed pick-up tasks by all couriers in a long time, i.e. guarantee both cooperation and long-term optimization at the same time. Massive experiments based on real-world road network data and historical express data from Beijing are conducted, to confirm the superiority of our model compared with nine baselines.

7 ACKNOWLEDGEMENT

We thank the reviewers of this paper for their constructive and kind suggestions. We thank Haoran Ma for helping us to polish this paper. Our this work was supported by the National Key R&D Program of China (2019YFB2101805) and Beijing Academy of Artificial Intelligence (BAAI).

REFERENCES

[1] L. Busoniu, R. Babuska, and D. B. Schutter. 2010. *Multi-Agent Reinforcement Learning: An Overview.* Springer.
[2] C. Chen, D. Zhang, X. Ma, B. Guo, L. Wang, Y. Wang, and E. Sha. 2017. Crowddeliver: Planning City-Wide Package Delivery Paths Leveraging the Crowd of Taxis. *IEEE Trans. Intelligent Transportation Systems* (2017).
[3] S. Chen, Y. Yu, Q. Da, J. Tan, H. Huang, and H. Tang. 2018. Stabilizing Reinforcement Learning in Dynamic Environment with Application to Online Recommendation. In *Proc. SIGKDD.*
[4] N. J. Foerster, M. Y. Assael, D. N. Freitas, and S. Whiteson. 2016. Learning to Communicate to Solve Riddles with Deep Distribute Recurrent Q-Networks. In *arXiv.*
[5] S. Ghosh, M. Trick, and P. Varakantham. 2016. Robust Reposition to Counter Unpredictable Demand in Bike Sharing Systems. In *Proc. IJCAI.*
[6] I. Goodfellow, Y. Bengio, and A. Courville. 2016. *Deep Learning.* MIT.
[7] Y. Hu, Q. Da, A. Zeng, Y. Yu, and Y. Xu. 2018. Reinforcement Learning to Rank in E-commerce Search Engine: Formalization, Analysis, and Application. In *Proc. SIGKDD.*
[8] P. Hulot, D. Aloise, and S. D. Jena. 2018. Towards Station-Level Demand Prediction for Effective Rebalancing in Bike-Sharing Systems. In *Proc. SIGKDD.*
[9] R. J. Kok and N. Vlassis. 2004. Sparse Cooperative Q-Learning. In *Proc. ICML.*
[10] H. Lee and T. Lee. 2019. Improve Cooperative Multi-Agent Reinforcement Learning Algorithm Augmented by Mixing Demonstrations from Centralized Policy. In *Proc. AAMAS.*
[11] T. Li, J. Zhang, K. Bao, Y. Liang, Y. Li, and Y. Zheng. 2020. AutoST: Eicient Neural Architecture Search for Spatio-Temporal Prediction. In *Proc. SIGKDD.*
[12] X. Li, J. Zhang, J. Bian, Y. Tong, and T. Liu. 2019. A Cooperative Multi-Agent Reinforcement Learning Framework for Resource Balancing in Complex Logistics Network. In *Proc. AAMAS.*
[13] Y. Li and Y. Zheng. 2019. Citywide Bike Usage Prediction in a Bike-Sharing System. *Transaction on Knowledge and Data Engineering* (2019).
[14] Y. Li, Y. Zheng, and Q. Yang. 2018. Dynamic Bike Reposition: A Spatio-Temporal Reinforcement Learning Approach. In *Proc. SIGKDD.*
[15] Y. Li, Y. Zheng, and Q. Yang. 2019. Efficient and Effective Express via Contextual Cooperative Reinforcement Learning. In *Proc. SIGKDD.*
[16] Y. Li, Y. Zheng, H. Zhang, and L. Chen. 2015. Traffic Prediction in a Bike-Sharing System. In *Proc. SIGSPATIAL.*
[17] Y. Liang, S. Ke, J. Zhang, X. Yi, and Y. Zheng. 2018. GeoMAN: Multi-level Attention Networks for Geo-sensory Time Series Prediction. In *Proc. IJCAI.*
[18] K. Lin, R. Zhao, Z. Xu, and J. Zhou. 2018. Efficient Large-Scale Fleet Management via Multi-Agent Deep Reinforcement Learning. In *Proc. SIGKDD.*
[19] J. Liu, L. Sun, W. Chen, and H. Xiong. 2016. Rebalancing Bike Sharing Systems: A Multi-Source Data Smart Optimization. In *Proc. SIGKDD.*
[20] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch. 2017. Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments. In *Proc. NIPS.*
[21] J. Mclnerney, A. Rogers, and N. R. Jennings. 2014. Crowdsourcing Physical Package Delivery Using the Existing Routine Mobility of A Local Population. In *Proc. Orange D4D Challenge.*
[22] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. 2013. Playing Atari with Deep Reinforcement Learning. In *arXiv.*
[23] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis. 2015. Human-Level Control through Deep Reinforcement Learning. *Nature* (2015).
[24] A. Sadilek, J. Krumm, and E. Horvitz. 2013. Crowdphysics: Planned and Opportunistic Crowdsourcing for Physical Tasks. In *Proc. ICWSM.*
[25] D. Silver, A. Huang, J. C. Maddison, A. Guez, L. Sifre, G. Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis. 2016. Mastering the Game of Go with Deep Neural Network and Tree Search. *Nature* (2016).
[26] S. R. Sutton and G.A. Barto. 1998. *Reinforcement Learning: An Introduction.* MIT.
[27] A. Tampuu, T. Matiisen, D. Kodelja, I. Kuzovkin, K. Korjus, J. Aru, and R. Vincent. 2015. Multiagent Cooperation and Competition with Deep Reinforcement Learning. In *arXiv.*
[28] D. Wang, J. Zhang, W. Cao, J. Li, and Y. Zheng. 2018. When Will You Arrive Estimating Travel Time Based on Deep Neural Networks. In *Proc. AAAI.*
[29] H. Wei, G. Zheng, H. Yao, and Z. Li. 2018. IntelliLight: A Reinforcement Learning Approach for Intelligent Traffic Light Control. In *Proc. SIGKDD.*
[30] X. Yi, Z. Duan, T. Li, J. Zhang, T. Li, and Y. Zheng. 2019. CityTraffic: Modeling Citywide Traffic via Neural Memorization and Generalization Approach. In *Proc. CIKM.*
[31] J. Zhang, Y. Zheng, and D. Qi. 2017. Deep Spatio-Temporal Residual Networks for Citywide Crowd Flows Prediction. In *Proc. AAAI.*
[32] S. Zhang, L. Qin, and Y. Zheng. 2016. Effective and Efficient: Large-Scale Dynamic City Express. *Transaction on Knowledge and Data Engineering* (2016).