# Can Sophisticated Dispatching Strategy Acquired by Reinforcement Learning?

## A Case Study in Dynamic Courier Dispatching System

Yujie Chen
Zhejiang Cainiao Supply Chain Management Co., Ltd
aisling.cyj@cainiao.com

Yu Qian
Zhejiang Cainiao Supply Chain Management Co., Ltd
qianyu.qy@alibaba-inc.com

Yichen Yao
Zhejiang Cainiao Supply Chain Management Co., Ltd
eason.yyc@alibaba-inc.com

Zili Wu
Zhejiang Cainiao Supply Chain Management Co., Ltd
zili.ziliwu@cainiao.com

Rongqi Li
Zhejiang Cainiao Supply Chain Management Co., Ltd
rongqi.lrq@cainiao.com

Yinzhi Zhou
Zhejiang Cainiao Supply Chain Management Co., Ltd
yinzhi.zyz@cainiao.com

Haoyuan Hu
Zhejiang Cainiao Supply Chain Management Co., Ltd
haoyuan.huhy@cainiao.com

Yinghui Xu
Zhejiang Cainiao Supply Chain Management Co., Ltd
renji.xyh@taobao.com

## ABSTRACT

In this paper, we study a courier dispatching problem (CDP) raised from an online pickup-service platform of Alibaba. The CDP aims to assign a set of couriers to serve pickup requests with stochastic spatial and temporal arrival rate among urban regions. The objective is to maximize the revenue of served requests given a limited number of couriers over a period of time. Many online algorithms such as dynamic matching and vehicle routing strategy from existing literature could be applied to tackle this problem. However, these methods rely on appropriately predefined optimization objectives at each decision point, which is hard in dynamic situations. This paper formulates the CDP as a Markov decision process (MDP) and proposes a data-driven approach to derive the optimal dispatching rule-set under different scenarios. Our method stacks multi-layer images of the spatial-and-temporal map and apply multi-agent reinforcement learning (MARL) techniques to evolve dispatching models. This method solves the learning inefficiency caused by traditional centralized MDP modeling. Through comprehensive experiments on both artificial dataset and real-world dataset, we show: 1) By utilizing historical data and considering long-term revenue gains, MARL achieves better performance than myopic online algorithms; 2) MARL is able to construct the mapping between complex scenarios to sophisticated decisions such as the dispatching rule. 3) MARL has the scalability to adopt in large-scale real-world scenarios.

*Proc. of the 18th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2019), N. Agmon, M. E. Taylor, E. Elkind, M. Veloso (eds.), May 13–17, 2019, Montreal, Canada.* 

## KEYWORDS

Multi-agent reinforcement learning; Courier dispatching problem; Smart cities

**ACM Reference Format:**
Yujie Chen, Yu Qian, Yichen Yao, Zili Wu, Rongqi Li, Yinzhi Zhou, Haoyuan Hu, and Yinghui Xu. 2019. Can Sophisticated Dispatching Strategy Acquired by Reinforcement Learning? . In *Proc. of the 18th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2019), Montreal, Canada, May 13–17, 2019,* IFAAMAS, 9 pages.

## 1 INTRODUCTION

With the rapid development of e-commerce, millions of small online retailers have emerged, resulting in a large growth in demand for pickup service. Only if the couriers pick up the goods within the prescribed time window, the subsequent transportation and delivering service can be carried out timely.

In this study, we consider a courier dispatching problem (CDP) with dynamic customers. In particular, given a number of couriers and unserved requests, decision should be made to assign couriers to pickup requests that maximize the total revenue. Each request has a hard service time window and the information is known when the system receives the customer's request.

A straightforward solution for the CDP is to formulate and solve an optimization problem with the development of the problem. However, these methods rely on appropriately predefined optimization objectives at each decision point, which is hard in dynamic situations. To capture the stochastic spatial-and-temporal pattern of pickup requests and utilize the information in the decision process, we discretize the urban region into cartesian grids. Then, a hierarchical

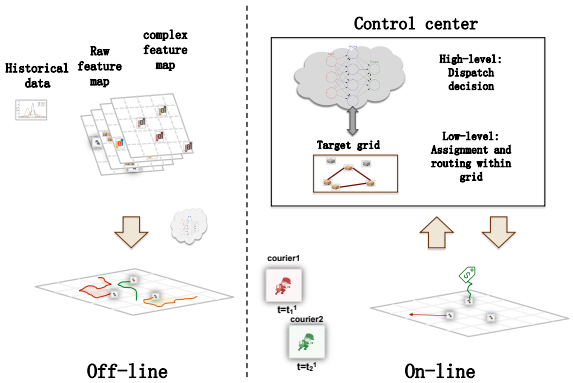


**Figure 1: Overview of the proposed decision mechanism**

solution approach is designed to handle these pickup requests, namely "dispatching-level" and "routing-level". At the "dispatching-level", we adopt a learning model to uncover the spatial-and-temporal pattern of customer demands. The dispatching model aims to find an optimal policy that designates each courier to a target grid and at the same time assigns a service time for each courier in his/her target grid. At the "routing-level", the problem within the target grid is formulated as an orienteering problem with time windows [9]. Briefly, given a set of geographically distributed customers with different prices and time windows, a vehicle aims to maximize its income within limited duration constraint. To solve this problem, we adopt a deterministic insertion algorithm proposed by [9]. Figure 1 depicts our overall decision mechanism.

This paper focuses on the study of learning models at the "dispatching-level". In recent years, multi-agent reinforcement learning (*MARL*) techniques have been successfully applied to a few traditional operational research problems, such as supply chain management [20], inventory optimization [14] and fleet management problem [11]. Although the information quality is improved by many service infrastructures, it still remains significant challenges when adopting *MARL* approach on CDP. Similar to the fleet management problem stated in [11], the CDP also faces the difficulties of 1) problem setting designs including state-action space and appropriate reward functions, 2) large numbers of agents issue and 3) the cooperation problem between agents. In order to provide more realistic decisions, our *MARL* is evaluated in a more complex environment that:

(1) Each pickup request has more complex behavior information such as arrival time, earliest start time, latest start time and service time. Consequently, we can not use a simple linear function to derive the corresponding reward for each action.

(2) Couriers are assigned with tasks whenever it becomes free, and get the actual reward when it fulfills the requests in grids. This setting leads to a delayed reward. Due to the long processing time of each action, mutual influence between agents is magnified, which increases the difficulty in the design of reasonable feedback mechanism.

Our major contributions are as follows:

- It is the first time to solve the CDP using multi-agent reinforcement learning method based on Markov decision process.
- We propose an effective dispatching algorithm with decentralized control and reward shaping to introduce cooperation between agents.
- Our proposed method outperforms a set of well-studied online algorithms for CDP through comprehensive experiments on both of the artificial dataset and the real-world dataset. Moreover, the generalization of our method is confirmed on unseen scenarios beyond the training dataset, mainly on scope of customer time window, dynamic ratio and zone distribution variation.

## 2 RELATED WORK

In this section, we discuss traditional models and solutions to solve the CDP. Additionally, we also present multi-agent reinforcement learning approaches that recently appeared in related problems.

### 2.1 Models for CDP

The courier dispatching problem (CDP) is commonly considered as a dynamic variant of the vehicle routing problem (DVRPs) with time windows [21], or an online matching problem [18]. According to the type of the uncertain information, studies have been carried out in uncertain customer demands [1, 7], stochastic travel time [10, 25], stochastic service time [21] and customer locations [23]. However, the common thing is that all these models tackles the dynamic and stochastic characteristic of the CDP.

### 2.2 Heuristics for CDP

To solve these dynamic problems, we categorize existing approaches based on whether any prediction of future events is utilized in the decision process, namely the myopic strategy and the looking-forward strategy.

In the first group, the solution process is based on known data and does not consider any uncertain information. Ritzinger et al. pointed out that it is hard to compromise between reactiveness and decision quality in dynamic scenarios [16]. The challenge of these myopic solutions is to set up appropriate objective functions. Heuristics from static scenarios can be modified for dynamic situations, such as tabu search [6] and large neighborhood search [8].

Researches from many real-world applications show the underline data pattern of customer demands [12]. The looking-forward strategy predicts future events either implicitly or explicitly. Sungur et al. use stochastic programming with recourse to model the uncertainty in demands [21]. During

the offline phase, a robust optimization is proposed to derive a master plan. During the operation phase, their recoursing rule simply omits non-occurring customers and reschedules new customers. A different approach is to embed sampling techniques into static solvers [17]. The most similar work to our research is to model the CDP using the language of Markov decision processes (MDP). For any practical sized DVRPs, obtaining optimal policies is computationally intractable [5]. Thus, heuristic methods with rollout policy are commonly applied [7, 13, 23]. More practically, a dynamic lookup table technique that simplifies and discretizes the states during the optimization is proposed recently [22, 24].

### 2.3 Multi-agent RL (*MARL*) for planning

There are many obstacles to adopt multi-agent reinforcement learning(*MARL*) algorithms in real-world applications, which often involves complicated interactions between multiple agents. Furthermore, feedback from the environment is also non-linear. Further difficulties include non-stationary environment and prohibitively large and intractable state-action space [11]. With the development of *MARL* and deep learning research, multi-agent systems have been recently studied in a variety of domains including robotic teams and resource management [2, 4]. In a *MARL* setting, it is usually challenging to specify a good objective function, since the returns of agents are correlated [3]. By applying experience sharing technique, homogeneous agents can learn faster and reach better overall performance [19]. Recently, Oroojlooyjadid et al. adopted deep Q-Learning to optimize the replenishment decisions at a given stage in a supply chain management application [14]. More similar to our work, Lin et al. proposed a contextual multi-agent actor-critic(CMAAC) method to find the optimal dispatching rules that balance the supply and demand in a geographical area [11]. However, the size of CMAAC's action space is limited to 6 neighboring girds to move, which is a less optimal decision for real-world online dispatching system.

## 3 PROBLEM STATEMENT AND SIMULATOR DESIGN

*Problem.* Our study is carried out on a $20 \times 20$ grid world, where pickup requests arrive over time. Each pickup request specifies the earliest start time, the latest start time and the service time. We aim to design a dispatching algorithm that maximizes the total revenue of all couriers within limited working hours. In particular, the dispatching algorithm generates two parts of decisions: 1) assigning the courier to a target grid; 2) and assigning a service time for the courier in the target grid at the same decision time.

*Simulator.* To evaluate algorithm performance, we introduce a simulator that generates the pickup requests and executes the decisions of the tested dispatching algorithm. All the later experiments are carried out on this simulator.

Our simulator applies a discrete event driven model. At the start time, all couriers send a request to the dispatching algorithm and receive an instruction. As long as any courier completes his/her current task, the dispatching algorithm assigns a new task to the courier. Whenever a courier arrives his/her target gird, a deterministic insertion algorithm (DIA)[9] is called to give the detailed pick-up instructions. The input parameters of DIA include the target gird arrival time, allowed service time and real-time customers at the arrival time of the target grid. In particular, as shown in Figure 2, the "routing-level" decision is treated as a part of the simulator. When a courier completes all assigned requests, the corresponding revenue and the actual task execution time is recorded for this action and the following activities are conducted sequentially:

- *Order generation:* new pickup requests are dynamically added to the environment as soon as the new request appears, and are removed from the environment when current time goes beyond the customer service time window. Hiring more couriers can reduce the portion of not served requests, and this relates to the tradeoff between the labor cost and platform revenue. In the environment setting, the number of couriers can fulfill above 80% of the requests in a whole day, which is at a reasonable level for performance comparison of different algorithms.
- *Courier status updates:* each courier has three status defined as follows:
  1) free: ready for the next instruction
  2) walking: on the way to target grid
  3) picking: executing the pickup services within the target grid.

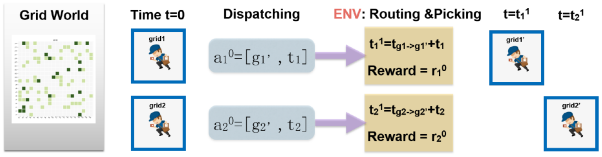


**Figure 2: Environment Settings**

## 4 LEARNING FOR DISPATCHING

In this section, we propose a Multi-Agent Reinforcement Learning (MARL) method to solve the CDP.

### 4.1 MDP Definition

We build the Markov decision process (MDP) from a decentralized point of view that each courier is modeled as an agent. By using the decentralization language, the size of the state-action space is much controllable. The centralized model leads to an exponentially increased size of the state-action space when the number of courier increases. It is worth mentioning that even in a decentralized perspective, global information such as the generation of pickup requests, the status of other agents can be perceived. The other components of the MDP are defined as follows:

- **State** $s_t^i \in \mathcal{S}$: The state of courier $i$ at time step $t$ considers informations in the $9 \times 9$ neighboring grids of the courier $i$. We use image-like tensor input as state, with each channel containing specific information about the environment. Two sets of states are discussed in the experiment, which differs in the information level the dispatch system could observe. At basic level, the dispatch system obtain the current snapshot of customers and couriers. The states include the number of agents, the number of pickup requests and the total price the requests in each grid, plus the distance of each neighbor grid to the current grid. At pre-solved level, dispatch system get extra information by calling the route planning solver. The presumed score of all possible actions at the current time is added as new channels.
- **Action** $a_t^i \in \mathcal{A}$: At the "dispatching-level", the decision includes the target grid and the maximum patrol time within the grid. Therefore, our action space is the cartesian product of candidate grids and time periods. We restrict the candidate grids to the $5 \times 5$ neighborhood grids of the courier $i$. And the maximum patrol time is discretized into $\{0, 10, 20, 30\}$ minutes. To avoid agents moving beyond the gird board, the target grid is clipped at the edge of the board. All the possible actions are represented as one-hot encoding. During the training stage, an action is chosen by roulette wheel selection according to probability distribution given by the model. During the testing stage, the action with the highest probability is selected.
- **Reward Function** $r_t^i \in \mathcal{R} = \mathcal{S} \times \mathcal{A} \to \mathbb{R}$: The reward design determines the optimization goal of the trained model. Intuitively, the reward for courier $i$ at time step $t$ is defined as total price of served requests according to last instruction. Moreover, the cooperation between couriers can be introduced through reward shaping, putting a weight $\alpha$ on how much each courier should care about its individual reward function versus the average of reward of all couriers. For example,
$$r_{t,shaping}^i = r_t^i + \alpha \times R_{team} \tag{1}$$
The effectiveness of considering team work is empirically discussed in later sections.
- **Agent**: A decentralized policy is applied as the homogeneous property of couriers. All the agents establish a common policy through parameters sharing. During the execution phase, each courier is modeled as an agent that selects an action based on its own observations mentioned above.
- **Discount factor**: The discount factor controls the degree of how far the MDP looks into the future. In our application, the state transitions are affected by the joint behaviors of all agents. Meanwhile, the long processing time of each action introduces large uncertainty during the training process. Therefore, a relatively small discount factor is preferable compared with the single-agent scenario. In our setting, the discount factor $\gamma = 0.8$ is adopted.

## 4.2 MARL

This section presents a Multi-Agent Reinforcement Learning based on actor-critic framework. There are two main ideas in the design of *MARL*: 1) Decentralized value function and policy network are used with an expected update; 2) Reward shaping introduces explicit cooperation between agents and considers the whole gains of dispatching system.

**Value Network**. The decentralized state-value function is learned by minimizing the following loss function derived from Bellman equation:

$$L_{\theta_v} = \left(V_{\theta_v}\left(s_t^i\right) - V_{target}\left(s_{t+1}^i; \pi\right)\right)^2 \tag{2}$$

$$V_{target}\left(s_{t+1}^i; \pi\right) = r_t^i + V_{\theta_v'}\left(s_{t+1}^i\right) \tag{3}$$

where we use $\theta_v$ to denote the parameters of the value network, $\pi$ to denote the dispatching policy and $\theta_v'$ to denote the target value network. In order to stabilize the learning process, we fix the target network for a few episodes sampling's time. Moreover, efficient corporation among agents can be established on this decentralized value network.

**Policy Network**. Policy gradient methods work by computing an estimator of the policy gradient and plugging it into a stochastic gradient ascent algorithm. The most commonly used gradient estimator has the form:

$$g = \mathbb{E}_t\left[\nabla_\theta log_{\pi_\theta}\left(a_t|s_t\right) A_t\right] \tag{4}$$

where $\pi_\theta$ is a stochastic policy and $A_t$ is an estimator of the advantage function as time step $t$, which is $A(s_t, a_t)$ for short. In this paper, we use the same objective proposed in the state-of-art actor-critic algorithm from Proximal Policy Optimization(PPO):

$$L_\theta = \mathbb{E}_t\left[\min\left(r_t\left(\theta\right), clip\left(r_t\left(\theta\right), 1-\epsilon, 1+\epsilon\right)\right) A_t\right] \tag{5}$$

$$A_t = -V_{\theta_v}\left(s_t\right) + r_t + \cdots + \gamma^{T-t+1} r_{T-1} + \gamma^{T-t} V\left(s_t\right) \tag{6}$$

where $\theta$ denotes the parameters of policy network and the expectation $\mathbb{E}_t$ indicates the empirical average over samples. The probability ratio $r_t(\theta)$ is computed as follows:

$$r_t\left(\theta\right) = \frac{\pi_\theta\left(a_t|s_t\right)}{\pi_{\theta_{old}}\left(a_t|s_t\right)} \tag{7}$$

Based on the MDP definition, we collect a set of tuples $(s, a, s', r)$ based on our simulator environment mentioned in Section 3. Note that since we do not distinguish individual couriers, the collections of all couriers form a joint-dataset for learning value and actor network. The detailed description of *MARL* is summarized in Algorithm 1.

# 5 DATA DESCRIPTION

In this section, we describe the instances of the simulation environment. We create artificial datasets to empirically test the performance of different algorithms, with the scenario settings varies [15]. In addition, we also apply the model to the real-world pickup-service data to verify the feasibility.

**Algorithm 1:** Multi-agent Training Framework

Initialize replay memory $M$
Initialize actor net and critic net
**for** m = 1 to $N_{episode}$ **do**
  Random choose instance from train set
  Reset environment and get initial state
  **Stage 1: Sampling**
  **while** $t < T$ **do**
    Sample actions $\mathbf{a}_t$ according to policy network, given $\mathbf{s}_t$
    Execute $\mathbf{a}_t$ in the simulator and observe reward $\mathbf{r}_t$, next state $\mathbf{s}_{t+1}$
    Compute value network target as Eq(3) and advantage as Eq(6)
    Store the transitions $(s_t^i, a_t^i, r_t^i, s_{t+1}^i)$ for all couriers $i \in [1, \ldots, N]$ into memory $M$
  **end while**
  **Stage 2: Learning**
  **for** $n_1 = 1$ to $N_1$ **do**
    Sample a batch of experience: $s_t^i$, $V_{target}(s_{t+1}^i; \pi)$
    Update value network by minimizing the value loss Eq(2) over the batch
  **end for**
  **for** $n_2 = 1$ to $N_2$ **do**
    Sample a batch of experience: $s_t^i$, $a_t^i$, $A(s_t, a_t)$
    Update policy network as $\theta \leftarrow \theta + \nabla_\theta L_\theta$ according to Eq(5)
  **end for**
**end for**

## 5.1 Time Horizon

Our pick up services are provided within a fixed time horizon. In our settings, we set the time horizon to 480 minutes, which represents the working period from 8:00 am to 4:00 pm. Within this horizon, discrete pickup requests are generated following a given distribution. Couriers are not required to return to the depot at 4:00 pm, but they would not accept any new requests. The revenue of the day is the total price of all served requests.

## 5.2 Temporal & Spatial Distribution

The arrival rate of new requests in a city varies over time and space. In our data generator, we divide the service area into $G$ grids and partition the time horizon into $M$ intervals. We use $\lambda_{gm}$ to denote the request arrival rate in grid $g$ at time interval $m$. In each time interval $m$, every grid $g$ generates a series of new arrival requests with the arrival rate of $\lambda_{gm}$ in a Poisson process. Given a new generated request $r_i$ generated at time $t_i$, the corresponding time windows is set as follows: 1) the earliest start time $\underline{w_i}$ is at $t_i + \Delta T_1$ , where $t_i$ is the arrival time of request $r_i$ ; 2) the latest that time $\overline{w_i}$ is set to $\underline{w_i} + \Delta T_2$. The parameters $\Delta T_1$ and $\Delta T_2$ are constant value. In addition, customer service time is uniformly chosen from the set {2, 3, 4} minutes, and a customer price is uniformly chosen from the set {1, 2, 3, 4, 5} dollars.

**Table 1: Arrival Rate Matrix**

| Grid Type | Customer Arrival Rate | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| intense | 0.05 | 0.00 | 0.00 | 0.10 | 0.04 | 0.00 | 0.00 | 0.05 |
| peripheral | 0.01 | 0.06 | 0.01 | 0.01 | 0.01 | 0.06 | 0.05 | 0.01 |

**Table 2: Scenario settings**

| Scenario Type | Time Window | Degree of Dynamism | Zone Distribution | Courier Number | Customer Number |
|---|---|---|---|---|---|
| *Base World* | 60 min | 90% | fixed | 10 | 1000 |
| *Median World* | 60 min | 90% | fixed | 30 | 3000 |
| *Large World* | 60 min | 90% | fixed | 100 | 15000 |
| *Small TW* | 20 min | 90% | fixed | 10 | 1000 |
| *Low Dynamism* | 60 min | 50% | fixed | 10 | 1000 |
| *Random Grid* | 60 min | 90% | random | 10 | 1000 |

## 5.3 Testing Scenarios

*5.3.1 Artificial Data.* We design six scenarios as summarized in Table 2 to test our algorithms. Each set of scenarios contains 40 problem instances.

- Grid world: we design a $20 \times 20$ grid world. Each grid is a $1km \times 1km$ square area. The grid world are defined into three types: intense, peripheral and empty, with the percentage of each grid type equals 5%,15%,80% respectively. As shown in Table 2, the type of each grid is fixed initially in all instances except for the *Random Grid*, which has the grid type randomly generated based on the probability.
- Time horizon: time horizon equals 480 minutes and is equally partitioned into eight periods as shown in Table 1.
- Couriers: Three sets of courier number are conducted in our experiments to evaluate the scalability of *MARL* model. Each courier has a moving speed of 0.5 km/minute. All couriers departure from the central gird (10,10).
- Pickup request: according to Table 1, we generate pickup requests for each grid and each time period in a Poisson process and set the current time as $\underline{w_i}$ for request $r_i$. The location of each request is uniformly generated within the corresponding grid.
- Degree of dynamism (DOD): to evaluate the impact of information known in advance to different algorithms, we adjust the number of pickup request known at time 0. For example, if the value of DOD equals 90% as shown in Table 2, we randomly choose 10% of the total generated requests and set their $t_i$ to time 0.

*5.3.2 Real-world Data.* We use the real-world data provided by the courier pickup services of Alibaba in the city of Hangzhou. We select an area of 20km × 20km and divide the area into 20×20 grids. Different to the artificial data, we adopt real distances between grids that are provided by GaoDe map api[1]. We choose 60 continuous days of data and use the first 30 days' data as training set and the second half as the test set for later experiments.

[1] https://lbs.amap.com/

# 6 EXPERIMENT DESIGN

## 6.1 Compared Algorithms

The *MARL* model is compared to the following dispatching algorithms:

- *Random*: This method does not require any information about the current state. It randomly selects a neighborhood grid and a service period from the action space $\mathcal{A}$.
- *GHAV*(Greedy to the highest absolute value): The highest absolute value dispatcher selects a grid with a service period that currently has the highest value per unit time from the action space $\mathcal{A}$.
- *GHEP*(Greedy to the highest expected profit): The highest expected profit dispatcher selects a grid and a service period that has the highest expected profit related to the requested courier. We define the expected profit of each candidate action by considering the traveling time to the target grid and pre-calling the route planning solver to identify possible gains after arriving the target grid.
- *MBM*(Maximum bipartite matching): Maximum bipartite matching dispatcher tries to maximize the total score for all couriers that will be available in 20 minutes. More particularly, in the bipartite graph, there are two disjoint subsets. One sub-graph includes available couriers and the other includes the cartesian product of candidate grids and service periods. We calculate the expected profit of each candidate courier to each grid with each service period. Then a maximum optimal matching problem is solved and the current courier is dispatched.

## 6.2 Experimental Settings

In our experiment, training step lasts for 10000 episodes, and the performance on both the train set and test set are reported. The similar network structure is applied for both value function and policy network, which is parameterized by one fully-connected hidden layer of 200 units with ReLU. After each episode, the sample of transition tuples is written into the memory buffer, which has the size of 20000. In learning step, iteration number $N_1$ and $N_2$ is set to 10, with the batch size of 1024. Adam optimizer is applied, with the learning of $5 \times 10^{-4}$. In considering the collaborative revenue of the whole fleet, performance with the reward reshaping weight $\alpha$ of 0.1, 0.5 and 0.9 has been compared, and $\alpha$ value of 0.5 obtains the highest score with a slight advantage of 0.16% and 0.67% against other two settings, and therefore $\alpha$ value of 0.5 is applied in the following discussion. In the PPO setting, clipping value $\epsilon$ is set to 0.2.

## 6.3 Performance Comparison

As shown in Table 3, the performance of each dispatching strategy is presented. The score is calculated by the percentage of achieved price over the total price of all orders. For the proposed *MARL*, we feed different information to the presentation of states. At basic information level, the dispatch system only obtains the current information of customers and couriers, which is denoted as *MARL-B* in Table 3. Furthermore, we add an additional expected profit channel that is the same information used in *GHEP* and *MBM*, which is denoted as *MARL-EP*

**Table 3: Comparison of performance**

| Strategy | Base World | | Real World | |
|---|---|---|---|---|
| | train set | test set | train set | test set |
| *Random* | 23.35% | 22.99% | 24.56% | 27.78% |
| *GHAV* | 74.17% | 74.07% | 71.24% | 74.86% |
| *GHEP* | 75.12% | 75.51% | 73.47% | 75.30% |
| *MBM* | 81.13% | 80.77% | 76.46% | 75.93% |
| *MARL-B* | 82.80% | 83.00% | 77.86% | 77.96% |
| *MARL-EP* | **84.27%** | **84.45%** | **79.16%** | **79.21%** |

On both the base-world and real-world dataset, *Random* is the worst strategy, which could only get about 23% score of all orders appeared in the whole day. By comparing the performance of *GHAV* and *GHEP*, the expected value function does help couriers to identify the relative best target grid more accurately. This pattern is also shown in the results of *MARL-B* and *MARL-EP*. *MBM* has a significant increase in score compared to *GHAV* and *GHEP*. This is attributed to the consideration of global resource and demand allocation when making a single dispatching decision. *MARL-EP* gives the best achievement in score in all scenarios. These results provide good evidence that *MARL* has the learning ability of sophisticated decision process.

## 6.4 Agent Scale

In discussion of applicability of the *MARL* in urban scale level, *Median World* with 30 couriers, 3000 customers and *Large World* with 100 couriers, 15000 customers are studied, which is significantly larger than the problem instances studied in the MDP modelling group (e.g. [7, 13, 22–24]). The detailed performance on all scenarios is listed in Table 4. The *Median World* has relatively sufficient number of couriers. *MARL-EP* presents about 1% improvement against the *MBM*.

In the *Large World*, we firstly observe the better performance of *GHAV* compared to *GHEP*. This is due to the mutual influence between a large number of agents leads to a rapid change of grid values, thus the expected value function fails to give a correct evaluation of the grid value. Nevertheless, *MARL-EP* still gets the best performance, having at least 3% increase against all human-designed methods.

**Table 4: Performance on extended agent scale**

| Strategy | Median World | | Large World | |
|---|---|---|---|---|
| | train set | test set | train set | test set |
| *Random* | 35.75% | 34.77% | 27.57% | 27.94% |
| *GHAV* | 82.00% | 81.46% | 65.75% | 64.87% |
| *GHEP* | 84.09% | 84.27% | 64.44% | 64.22% |
| *MBM* | 91.37% | 91.45% | 74.71% | 74.62% |
| *MARL-B* | 91.87% | 91.99% | 77.77% | 77.85% |
| *MARL-EP* | **92.45%** | **92.58%** | **78.47%** | **78.51%** |

**Table 5: Performance on different scenario**

| Strategy | Dataset | | |
|---|---|---|---|
| | Small TW | Low Dynamism | Random Grid |
| *Random* | 25.21% | 31.01% | 23.47% |
| *GHAV* | 64.98% | 73.49% | 72.85% |
| *GHEP* | 68.15% | 77.58% | 76.48% |
| *MBM* | 68.76% | 79.00% | 80.68% |
| *MARL-B* | 71.62% | 81.60% | 81.34% |
| *MARL-EP* | **72.50%** | **81.51%** | **82.71%** |

## 6.5 Model Generality

Typically, reinforcement learning is trained on a specific environment setting, the generalization capability of RL model is beyond the scope in many problems. However, due to the uncertainty and unpredictability of online situation, it is worth discussing if there exists certain transferability of the dispatching strategy acquired. On this topic, the generality and extensibility of our model is discussed on types of 3 datasets derived from the base scenario, mainly varying on the aspects of customer service time window, the degree of dynamic orders, and customer zone distribution. In scenario *Small TW*, the time window of available service time is reduced to 20 minutes, which require a more real-time response to new appear customers. In scenario *Low Dynamism*, the ratio of dynamic orders reduced to 50%, and in scenario *Random Grid*, the zone distributions are randomly shuffled.

The detailed performance on all scenarios is listed in Table 5. *MARL-EP* still presents a significant advantage over *GHAV*, *GHEP* and *MBM* on all the scenarios, though the gap is less obvious than on the trained scenario. The resultes proves that *MARL* is capable of handling unseen dynamics in real scenarios.

## 6.6 Imitation Learning

The major shortcoming of reinforcement learning lies in its low sampling efficiency, leading to long warm-up phase during the learning. In many reinforcement learning tasks, if high-quality samples can be generated through historical data or strategies, then the policy might be trained quickly through imitation learning, by using the privilege of past experience and expert strategy. In our experiment, *GHEP* is selected as the imitated objective for *MARL-EP*. Two attempts have been conducted, specifically all *GHEP* strategy before the $300^{th}$ episode, and 20% probability of *GHEP* strategy before $1000^{th}$ episode. The performance is listed in figure 3. It is obvious that imitation learning only accelerate the initial process of the learning curve, while the long-term performance is less favorable than the none-imitated model. The possible explanation is that, reinforcement learning method in this dispatching problem can be easily guided into a local optimum by the imitated strategy. Exploration by self-discovery is quite important in the RL framework, excessive exploitation of expert experience might block the opportunity to learn the best strategy from a long-term perspective.

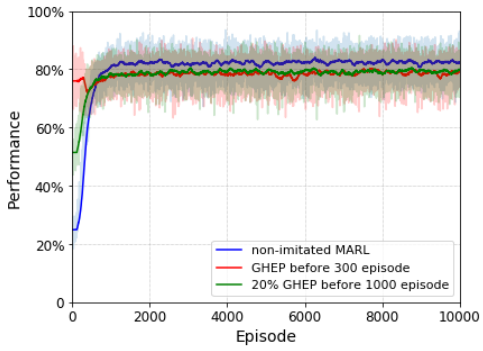


**Figure 3: Performance with imitated strategy from GHEP dispatcher**

## 6.7 Cooperation Discussion

In a multi-agent system, it is likely that not all agents are identically under control, especially in the real business scenario. In the courier dispatch system, couriers might be managed and controlled by different dispatch platforms, and it is worth discussing if *MARL* is capable of learning dispatching strategy in a cooperative environment. In the following discussion, the experiment of four fleets is discussed, each with the courier number of 10. In fleet 1, all 10 couriers share the same policy. In the other three fleets, two sets of dispatching policies are adopted, with each policy controls the implementation of five couriers. In fleet 2, the first 5 couriers use *MARL-EP* policy, and the other 5 couriers use *GHEP* policy. In fleet 3, the first 5 couriers use *MARL-EP* policy, and the other 5 couriers use *Random* policy. In fleet 4, each group of five couriers uses an independent learning environment, in which samples are independently collected and model is separately trained. Due to the homogeneous nature of all couriers, the performance obtained by fleet 4 is approximately equal to fleet 1. However, the learning progress is a bit slower in fleet 4 due to the smaller sample size of each model. In fleet 2, the total performance is a bit lower than the pure *MARL-EP* strategy. At the beginning stage of the learning phase, *GHEP* gets a quite high score, while *MARL-EP* is roughly random. However, as the training process proceeds, the performance of *MARL-EP* gradually grows and surpasses the *GHEP* group at the $1000^{th}$ episode, and the final performance of *MARL-EP* group is 13% higher than the *GHEP* group. In fleet 3, the five couriers in *MARL-EP* group obtains the highest group revenue by making up the loss caused by the inefficient random strategy.

## 6.8 Strategy Analysis

In this section, we aim to gain an insight into the derived dispatching policy by *MARL-EP* due to its outstanding performance. We analyze the trajectory of each courier as shown in Figure 5 from dataset scenario 1. In each subgraph, the color shown in the grid represents the cumulative price of appeared pickup requests in the last two hours, where the light color represents low price and dark color represents

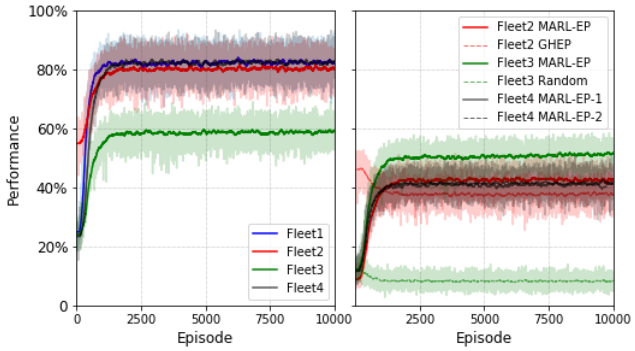


**Figure 4: Performance with two set of strategy**

high price. Further, we set up the same initial state to all tested scenarios. Note that the pickup requests in our problem settings have hard constraints of service time window and disappeared time, which leads to a high requirement of precision spatial and temporal matching. Interesting behavior can be seen from figure 5. a) The random dispatcher generates the longest walk trace and shortest pick-time that actually gain value. b) The *GHEP* shows no cooperation between couriers. This strategy leads to agents' competition in local areas, which results in low overall revenue. c) The *MBM* considers cooperation when there are enough pickup requests showing up. However, it is difficult to make an advisable decision during the period when there are not enough requests. d) In comparison, the *MARL-EP* dispatcher is able to capture the changes of request distribution over time. Take the area highlighted in *MARL-EP* of figure 5 as an evidence, the myopic strategies fail to explore this area due to its isolated and remote location. Only pre-dispatching at the early time could capture the orders of this area.

Figure 6 presents the dispatching paths of different methods given real-world data of Hangzhou city over 8 hours. Similar analysis and conclusion can be drawn. In addition, better supply and demand matching between north and center area over spatial and temporal space is observed in *MARL-EP*.

## 7 CONCLUSION

In this paper, we formulate the courier dispatching problem as a Markov decision process and use a multi-agent reinforcement learning method to solve this problem. We propose an effective dispatching algorithm with decentralized control and reward shaping to encourage cooperation among agents. The results from experiments on both the artificial dataset and the real-world dataset show that the MARL achieves significant improvement over the human-designed dispatching policies. Our method is capable of capturing intrinsic patterns among data and make reasonable decisions from a long-term perspective. Model generality is confirmed on the unseen scenarios beyond the training dataset, mainly on the scope of the customer time window, dynamic ratio, and

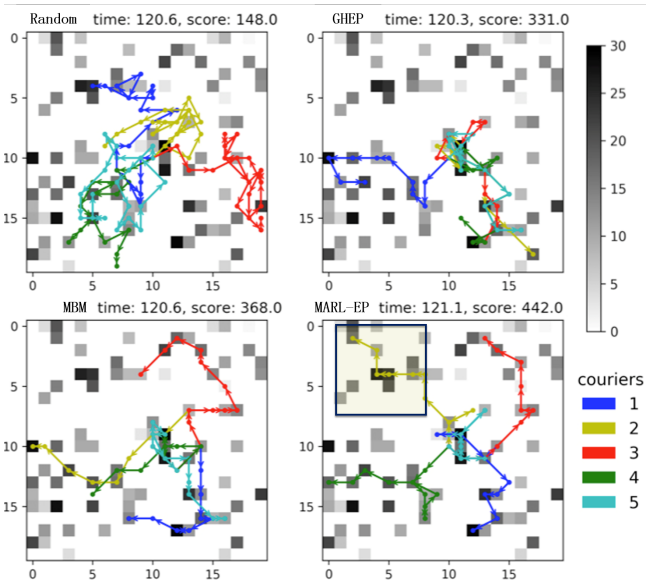


**Figure 5: Trajectory of 5 couriers in the first two hours of Scenario 1**

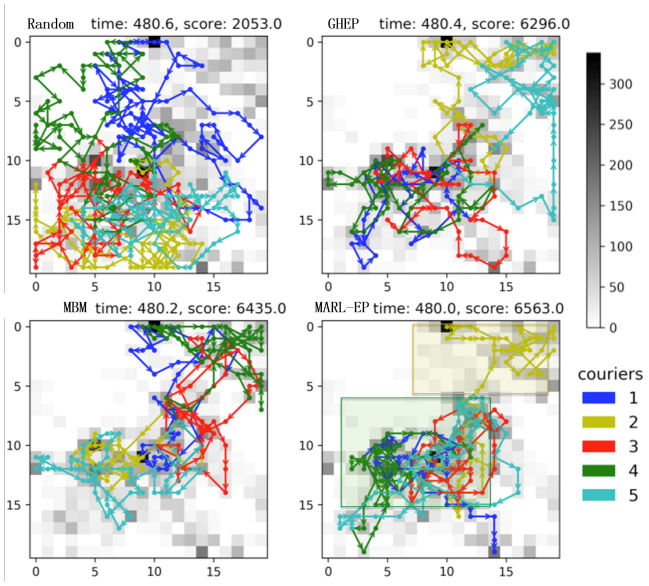


**Figure 6: Trajectory of 5 couriers in the whole day on** $Aug.5^{th}$ 2018

zone distribution variation. Moreover, we show the scalability of the proposed method on significantly larger data sets compared to existing literature.

In future research, we will focus on investigation of more effective network architecture and learning algorithms. Meanwhile, it is beneficial to apply our proposed method to more interesting combinatorial optimization problems in the domain of logistics and online scheduling systems.

## ACKNOWLEDGMENTS

The authors would like to thank the anonymous referees for their valuable comments and helpful suggestions. The authors would also like to show our gratitude to Prof. Yang Yu from School of Artificial Intelligence of Nanjing University for his valuable advices.

## REFERENCES

[1] E Angelelli, R Mansini, and M G. Speranza. 2005. A real-time vehicle routing model for a courier service problem. In *Distribution Logistics*. 87–103.

[2] Bram Bakker, Shimon Whiteson, Leon Kester, and Frans C.A. Groen. 2010. Traffic light control by multiagent reinforcement learning systems. *Studies in Computational Intelligence* 281 (2010), 475–510.

[3] Lucian BusÂÿoniu, Robert Babuška, and Bart De Schutter. 2010. Multi-agent reinforcement learning: An overview. *Innovations in multi-agent systems and applications-1* 310 (2010), 183–221.

[4] Lucian BusÂÿoniu, Robert Babuška, and Bart De Schutter. 208. A Comprehensive Survey of Miltiagent Reinforcement Learning. *IEEE Transactions on Systems, Man, And Cybernetics-Part C: Applications and Reviews* 38, 2 (208). arXiv:arXiv:1011.1669v3

[5] M Dror, G Laporte, and P. Trudeau. 1989. Vehicle routing with stochastic demands: Properties and solution frameworks. *Transportation science* 23, 3 (1989), 166–176.

[6] Michel Gendreau, François Guertin, Jean Yves Potvin, and René Séguin. 2006. Neighborhood search heuristics for a dynamic vehicle dispatching problem with pick-ups and deliveries. *Transportation Research Part C: Emerging Technologies* 14, 3 (2006), 157–174.

[7] JC Goodson. 2013. Rollout policies for dynamic solutions to the multivehicle routing problem with stochastic demand and duration limits. *Operations Research* 61, 1 (2013), pp. 138–154.

[8] Lianxi Hong. 2012. An improved LNS algorithm for real-time vehicle routing problem with time windows. *Computers and Operations Research* 39, 2 (2012), 151–163.

[9] Marisa G Kantor, Moshe B Rosenwein, Marisa G Kantor', and Moshe B Rosenwein2. 1992. The Orienteering Problem with Time Windows. *Source: The Journal of the Operational Research Society* 43, 6 (1992), 629–635.

[10] Gilbert Laporte, François Louveaux, and Hélène Mercure. 1992. The Vehicle Routing Problem with Stochastic Travel Times. *Transportation Science* 26, 3 (1992), 161–170.

[11] Kaixiang Lin, Renyu Zhao, Zhe Xu, and Jiayu Zhou. 2018. Efficient Large-Scale Fleet Management via Multi-Agent Deep Reinforcement Learning. In *arXiv preprint arXiv:1802.06444*. arXiv:1802.06444

[12] C. Malandraki, D. Zaret, J. R. Perez, and C. Holland. 2001. Industrial engineering applications in transportation. In *Handbook of Industrial Engineering: Technology and Operations Management*. 787–824.

[13] Clara Novoa and Robert Storer. 2009. An approximate dynamic programming approach for the vehicle routing problem with stochastic demands. *European Journal of Operational Research* 196, 2 (2009), 509–515.

[14] Afshin Oroojlooyjadid, MohammadReza Nazari, Lawrence Snyder, and Martin Takáč. 2017. A Deep Q-Network for the Beer Game: A Reinforcement Learning algorithm to Solve Inventory Optimization Problems. (2017), 1–38. arXiv:1708.05924

[15] Harilaos N. Psaraftis, Min Wen, and Christos A. Kontovas. 2016. Dynamic vehicle routing problems: Three decades and counting. *Networks* 67, 1 (2016), 3–31.

[16] Ulrike Ritzinger, Jakob Puchinger, and Richard F. Hartl. 2016. A survey on dynamic and stochastic vehicle routing problems. *International Journal of Production Research* 54, 1 (2016), 215–231.

[17] Briseida Sarasola, Karl F. Doerner, Verena Schmid, and Enrique Alba. 2016. Variable neighborhood search for the stochastic and dynamic vehicle routing problem. *Annals of Operations Research* 236, 2 (2016), 425–461.

[18] Michael Z. Spivey and Warren B. Powell. 2004. The Dynamic Assignment Problem. *Transportation Science* 38, 4 (2004), 399–419.

[19] Milos Stankovic. 2016. Multi-Agent Reinforcement Learning. In *13th Symposium on Neural Networks and Applications*.

[20] Tim Stockheim, Michael Schwind, and Wolfgang Koenig. 2002. A Reinforcement Learning Approach for Supply Chain Management. *1st European Workshop on MultiAgent Systems* 40, 6 (2002), 1299–1317.

[21] Ilgaz Sungur, Yingtao Ren, Fernando Ordóñez, Maged Dessouky, and Hongsheng Zhong. 2010. A Model and Algorithm for the Courier Delivery Problem with Uncertainty. *Transportation Science* 44, 2 (2010), 193–205.

[22] Marlin W. Ulmer, Justin C. Goodson, Dirk C. Mattfeld, and Marco Hennig. 2018. OfflineâĂŞOnline Approximate Dynamic Programming for Dynamic Vehicle Routing with Stochastic Requests. *Transportation Science* (2018).

[23] Marlin W. Ulmer, Dirk C. Mattfeld, Marco Hennig, and Justin C. Goodson. 2016. A Rollout Algorithm for Vehicle Routing with Stochastic Customer Requests. In *Logistics management*. Springer,Cham, 217–227.

[24] Marlin W. Ulmer, Dirk C. Mattfeld, and Felix Köster. 2017. Budgeting Time for Dynamic Vehicle Routing with Stochastic Customer Requests. *Transportation Science* 52, 1 (2017), 20–37.

[25] Shangyao Yan, Jenn Rong Lin, and Chun Wei Lai. 2013. The planning and real-time adjustment of courier routing and scheduling under stochastic travel times and demands. *Transportation Research Part E: Logistics and Transportation Review* 53, 1 (2013), 34–47.