TSINGHUA SCIENCE AND TECHNOLOGY
ISSN 1007-0214 05/24 pp356–367
DOI: 10.26599/TST.2023.9010041
Volume 29, Number 2, April 2024

# Reinforcement Learning-Based Dynamic Order Recommendation for On-Demand Food Delivery

Xing Wang, Ling Wang*, Chenxin Dong, Hao Ren, and Ke Xing

**Abstract:** On-demand food delivery (OFD) is gaining more and more popularity in modern society. As a kernel order assignment manner in OFD scenario, order recommendation directly influences the delivery efficiency of the platform and the delivery experience of riders. This paper addresses the dynamism of the order recommendation problem and proposes a reinforcement learning solution method. An actor-critic network based on long short term memory (LSTM) unit is designed to deal with the order-grabbing conflict between different riders. Besides, three rider sequencing rules are accordingly proposed to match different time steps of the LSTM unit with different riders. To test the performance of the proposed method, extensive experiments are conducted based on real data from Meituan delivery platform. The results demonstrate that the proposed reinforcement learning based order recommendation method can significantly increase the number of grabbed orders and reduce the number of order-grabbing conflicts, resulting in better delivery efficiency and experience for the platform and riders.

**Key words:** on-demand food delivery; order recommendation; reinforcement learning; actor-critic network; long short term memory

## 1 Introduction

With the fast development of e-commerce and transportation technologies[1, 2], online-to-offline (O2O) delivery service is gaining increasing popularity in modern life due to its convenience[3]. As a primary delivery scenario in O2O service, on-demand food delivery (OFD) accounts for a majority of transactions in many third-party delivery companies, such as Meituan, GrubHub, Just-Eat, and Deliveroo. According to the statistical financial report of Meituan in 2022, over 50 million orders are maximally generated by customers in a single day[4]. With such a large scale of transactions, it is of great significance to study the OFD business mode and improve the service quality[5].

Figure 1 introduces the main process of OFD service. Three stakeholders (customers, riders, and restaurants) are organized by the food delivery platform. Different orders are collected by the platform and assigned

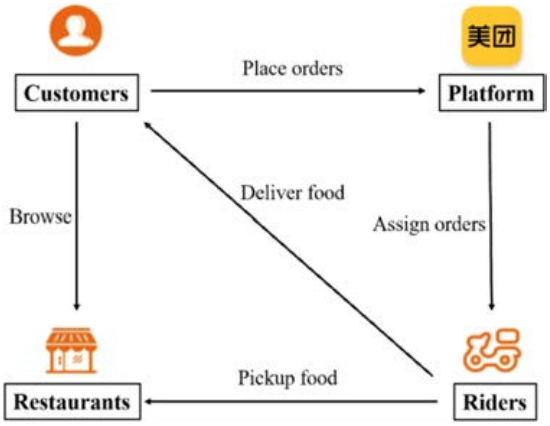


**Fig. 1 Main process of OFD service.**

• Xing Wang and Ling Wang are with Department of Automation, Tsinghua University, Beijing 100080, China. E-mail: wang-x17@mails.tsinghua.edu.cn; wangling@tsinghua.edu.cn.
• Chenxin Dong is with School of Mechanical and Automotive Engineering, Qingdao Hengxing University of Science and Technology, Qingdao 266100, China. E-mail: deyangxuanyi@hotmail.com.
• Hao Ren and Ke Xing are with Meituan, Beijing 100015, China. E-mail: renhao05@meituan.com; xingke@meituan.com.
* To whom correspondence should be addressed.
Manuscript received: 2023-04-19; revised: 2023-05-05; accepted: 2023-05-09

to appropriate riders so that they can be fetched and delivered to customers punctually. As a kernel scheduling process of the OFD service, order assignment determines the matching results between customer orders and riders, which largely influences the delivery efficiency and experience of customers and riders.

In this paper, we deal with the dynamic order recommendation problem in OFD, which is an important and principal order assignment scenario in real-life OFD companies. Different from traditional order dispatching modes, dynamic order recommendation conducts the order assignment procedure by exposing orders to each rider with a sorted order list[6]. Riders can browse the orders in the order list and select the favorite one by hitting the "grab" button on the mobile. For the same order, the rider who hits the button earliest will successfully serve the order. Dynamic order recommendation shares some similarities to item (or text) recommendation in the field of recommender system but differs in the following aspects:

• **The dynamism of food delivery process.** The item sets are relatively static in recommender system field and the user status does not change drastically. But in OFD field, the dynamism of the problem is a main challenge, which mainly results from orders and riders. Old customer orders are grabbed and new customer orders are generated at every single second. Riders are moving with transportation vehicles, which leads to continuously changing positions. Consequently, the dynamism invalidates the solutions at previous step and requires the platform to make instant decisions at every decision-making moment.

• **Multi-objective optimization**[7]**.** Traditional item recommendation usually considers the click through rate (CTR) as the main objective while the OFD platform needs to coordinate customers, riders, and restaurants, which results in different optimization objectives, such as customer satisfaction, rider delivery experience, and delivery efficiency. Improving the OFD service quality requires long-term optimization of these objectives.

• **Order-grabbing conflict between riders.** In dynamic order recommendation, one order can be exposed to different riders at current moment, which means that multiple riders might grab the same order and only one of them can succeed. The occurrence of order-grabbing conflict will harm the rider's delivery experience and cause negative influence to the delivery efficiency of the OFD platform. Therefore, avoiding possible conflict when generating order lists for riders is of great significance.

Due to the above challenges, traditional recommendation technique cannot be applied to solve the dynamic order recommendation problem. To handle the dynamism, we introduce the Markov Decision Process (MDP) and model the dynamic order recommendation as a multi-step sequential decision-making problem. A reinforcement learning based (RL-based) order recommendation method is correspondingly proposed to generate the order lists for different riders at each decision-making moment. Specifically, we design a special actor-critic network based on long short term memory (LSTM) unit to sequentially output the recommendation policy for each rider. The adoption of LSTM unit allows the platform use the information of riders whose order lists are already generated, which will help the actor generate better order lists for the rest riders at current moment so as to avoid order-grabbing conflict. A weighted multi-criteria reward function is proposed to balance different optimization objectives and guide the learning of the designed actor-critic network. The effectiveness of our proposed method is evaluated on the real dataset from Meituan delivery platform in China. The main contributions of the paper include the following aspects.

• We propose an RL-based order recommendation method for the dynamic food delivery problem which considers the competition among different riders. To the best of our knowledge, this is the first work that employs reinforcement learning for order recommendation in dynamic food delivery scenario with multiple riders.

• An actor-critic network based on LSTM unit is constructed to capture the interest of each rider and generate the best order recommendation policy. Besides, three rider sequencing rules are designed to correspond the time step in LSTM unit with different riders so as to sequentially output the recommendation policy for each rider.

• Extensive experiments are conducted on real-world datasets from Meituan. The results demonstrate that the proposed method is significantly superior to the comparative heuristic methods.

The rest of the paper is organized as follows. In Section 2 we review the researches that are related to OFD and recommendation methods. Then Section 3 describes the dynamic order recommendation problem and introduces the MDP formulation. The proposed RL-based order recommendation method is elaborated in Section 4. Computational results are exhibited and

analyzed in Section 5. Finally, Section 6 concludes the paper with some future research directions.

## 2 Related Work

The dynamic order recommendation problem studied in this paper is most related to two fields, i.e., the dynamic food delivery problem and the RL-based recommendation in the scenario of text searching (such as Google and Baidu) and e-commercial platforms (such as Taobao and Amazon). Therefore, we briefly review the problems and methods in these two areas.

### 2.1 Dynamic food delivery problem

Many researches deal with the dynamism by rolling-horizon method and transform the dynamic problem to multiple static sub-problems. Reyes et al.[8] developed optimization-based algorithms to solve the courier assignment and capacity management problems. Rolling horizon method is adopted to alleviate the dynamism and orders are allocated to couriers through a linear assignment model. Yildiz and Savelsbergh[9] introduced a new formulation for dynamic meal delivery routing problem. They assumed that the system could acquire perfect information about order arrivals, based on which they developed a simultaneous column and row generation method to generate solutions effectively. Ulmer et al.[10] considered a stochastic dynamic pickup and delivery problem where customer orders and restaurant preparation time were unknown. They presented a parametrizable cost function approximation to postpone the assignment decisions for selected customers and allow more flexibility in the response to new requests. Liu et al.[11] considered the uncertain travel time in dynamic last-mile delivery problem. A framework that could integrate the travel time predictors with the order assignment optimization was proposed. Besides, they designed two simple heuristics for the multi-period order assignment problem. Chen et al.[12] studied the OFDP and abstracted the problem into a static generalized assignment problem with a rolling horizon policy. They proposed an offline-optimization for online-operation framework based on imitation learning to solve the problem. Steever et al.[13] provided a mixed integer linear programming formulation for the virtual food court delivery problem and developed an auction-based heuristic to implement the dynamic setting. Paul et al.[14] proposed a generic framework for the optimization of first-mile and last-mile of an on-demand meal delivery system. Their objective was to minimize the overall cost per delivery and the delay in order delivery time. Several policies were recommended based on the Just-In-Time concept. Joshi et al.[15] studied the dynamic vehicle assignment problem and developed an algorithm called FOODMATCH, which converted the problem to that of minimum weight perfect matching on a bipartite graph. Moreover, they enhanced the solution quality by reducing batching to a graph clustering problem. Jahanshahi et al.[16] considered a meal delivery service fulfilling dynamic customer requests and developed a deep reinforcement learning solution approach.

Recently, Wang et al.[6] studied the dynamic food delivery problem with rider-centered assignment manner. They focused on the interactions between a single rider and the delivery platform, based on which they proposed a deep reinforcement learning based order recommendation framework to generate satisfactory recommendation policies for the rider.

### 2.2 RL-based recommendation methods

In the field of recommendation systems, recommendation techniques based on reinforcement learning have attracted much attention from researchers[17]. Roughly, RL-based recommendation methods can be divided into two types, i.e., model-based methods and model-free methods. Model-based methods have higher time complexity, which is more difficult to be applied to real recommendation tasks. On the contrary, model-free methods do not require exact formulation of the environment, which makes it more popular than model-based methods[18–20].

Xin et al.[21] proposed a self-supervised reinforcement learning method for sequential recommendation tasks, in which the RL output layer acted as a regularization function to drive the supervision layer to focus on a specific reward. Chen et al.[22] proposed a knowledge-guided deep reinforcement learning method, which integrated the advantages of reinforcement learning and knowledge graph for interactive recommendation. Zhao et al.[23] proposed a new recommendation framework based on deep reinforcement learning that could generate recommendation pages and complementary content at the same time. Deng et al.[24] established a deep recurrent neural network model based on deep learning and reinforcement learning for the problem of financial transaction signal representation. Zou et al.[25] designed an RL-based framework to optimize the long-term engagement goal of users in the recommendation system. It can be seen that current RL-based recommendation

methods are barely applied in the scenario of OFD service. Besides, current researches on recommendation tasks do not consider the inter-correlation between users. However, the influence between riders will incur possible order-grabbing conflict, which is an important factor that affects the efficiency and experience in OFD scenario. Therefore, it is of great value to consider multiple riders and design effective order recommendation methods to enhance the service quality of OFD business.

To the best of our knowledge, this work is the first to consider the interaction between multiple riders and solve the dynamic food delivery problem with RL-based recommendation technique. Compared with current researches, a more realistic problem setting is considered in our study that riders will compete with each other and might fail when trying to grab orders. This setting effectively helps construct the corresponding structure of the actor-critic network and improve the delivery efficiency and experience of customers and riders eventually.

## 3 Problem Formulation

The dynamic order recommendation problem in the OFD scenario can be described as follows. At each decision moment $t$, denote the set of all orders to be assigned as $O_t = \{o_1^t, o_2^t, o_3^t, \ldots\}$ and the set of all riders as $M_t = \{j_1^t, j_2^t, j_3^t, \ldots\}$. For each rider $j$ in $M_t$ at each decision moment $t$, the OFD platform needs to select $n_t$ uitable orders from the order set $O_t$ and generate an order recommendation list $L_j^t = \{o_1^t, o_2^t, o_3^t, \ldots, o_{n_t}^t\}$ for rider $j$ with the recommendation policy $\pi_j^t$. Note that the platform needs to generate $m_t$ different lists for different riders at each moment. After the order lists are generated and exposed to riders, each rider can browse the list and grab the preferred order. If multiple riders select the same order, there will be an order-grabbing conflict, in which the rider who clicks the "grab order" button earlier will grab the order. If the rider is not interested in the order in the recommendation list, they can choose to refresh the order list and get a new one at next decision moment.

Figure 2 gives a graphic illustration of the above problem description and Table 1 presents the meanings of notation symbols. To construct the dynamic order recommendation problem completely, the following constraints are respected.

• Each order can be exposed to multiple riders but should be served by only one rider.

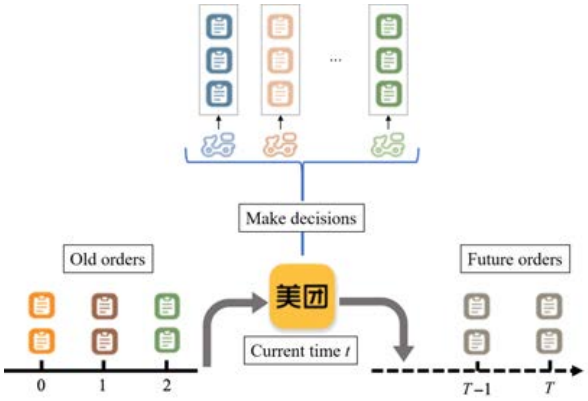


**Fig. 2 Graphic illustration of dynamic order recommendation problem.**

**Table 1 Meanings of notation symbols.**

| Symbol | Meaning |
|---|---|
| $t$ | Decision moment |
| $O_t$ | Set of orders to be assigned at $t$ |
| $M_t$ | Set of available riders at $t$ |
| $m_t$ | Number of riders at $t$ |
| $n_t$ | Length of order list at $t$ |
| $L_j^t$ | Order list of rider $j$ at time $t$ |
| $\pi_j^t$ | Recommendation policy of rider $j$ at $t$ |
| $S, A, P, R, \gamma$ | MDP formulation elements |
| $s, a$ | A certain state or action |
| $p_t$ | Transition probability at $t$ |
| $r_t$ | Instant reward at $t$ |
| $\text{num}_t$ | Number of grabbed orders at $t$ |
| $\alpha, \beta$ | Weighting factors |
| $\mu_t$ | Number of order-grabbing conflicts at $t$ |
| $\Delta\eta_t$ | Increment of route length and time delay at $t$ |
| $g$ | Default penalty when no order is grabbed |

• Each rider can grab at most one order at a decision moment.

• One order can be grabbed by multiple riders but only one of them will succeed.

• The delivery route of each rider obeys the constraints of food delivery route planning problem[26], including precedence constraint, capacity constraint, and time-window constraint.

Besides, the following assumptions are made specially for dynamic food delivery problems.

• The information of orders changes dynamically with the continuous input and output of customer orders.

• The positions of riders change at every second, which means that the status of riders is different at different decision moments.

• Solutions at previous steps will affect the solution space of the subsequent decision moments.

With the above constraints and assumptions, the objective of dynamic order recommendation problem in our study is to generate the best order recommendation list for each rider at each decision step so as to optimize the delivery efficiency of the platform and the delivery experience of customers and riders through a long time duration. For each decision moment, the input of the system is the information of available orders and riders, and the output of the system is different order recommendation lists for all available riders.

Different from the order recommendation problem in Ref. [6], this study considers the order-grabbing conflict between different riders, which means that the platform needs to arrange the order lists more reasonably so as to avoid that multiple riders grab the same order, which will harm both the experience of riders and the efficiency of the delivery system.

To mathematically describe the problem, we abstract the dynamic order problem in the way of reinforcement learning based on MDP. Specifically, we treat the OFD platform as an agent, which is responsible for observing the status of multiple riders and generating different order recommendation policies based on the information of orders and riders. Multiple riders are defined as the environment, which responds to the order lists provided by the platform accordingly. Five essential components $(S, A, P, R, \gamma)$ of MDP formulation are defined as follows:

• **State space *S*:** State $s \in S$ is the original representation of environment (which is the status of multiple riders in our study). A state $s$ is consisted of two parts, where the first is the individual features of each rider which is denoted as $s^j$ for rider $j$ and the second is the interactive features $s^{\text{cross}}$ of all riders. Here we use $s^j$ to characterize the personal status of each rider. $s^{\text{cross}}$ depicts the common status of all riders, which helps measure the order recommendation status of the whole delivery system. The specific meanings of the designed features are shown in Table 2. Note that when calculating $s^j$ for rider $j$, several order attribute features will be involved, which are also listed in Table 2.

• **Action space *A*:** Each action $a_t$ of the agent is defined as a vector with the dimension of $m_t \times \text{dim}$, where $m_t$ is the number of riders and dim is the number of order attribute features. When the OFD platform tries to generate an order recommendation list according to action $a_t$, the vector inner product of action $a_t$ and the attribute features of each order is calculated, resulting in a scalar ranking score for each order. Then all the orders are sorted in a descending order of the ranking score, after which the OFD platform will select the top $n_t$ orders with the highest scores to form the final order recommendation list.

• **Transition function *P*:** According to the feedback of different riders to the displayed order lists at moment $t$ (grab from the list or not and grab which order), the state

**Table 2 Features for state formulation.**

| Type | Name | Meaning |
|---|---|---|
| Individual feature of each rider | onhand_waybill_num | Number of carried orders |
| | grab_last_step | Whether the rider grab an order at last step |
| | grab_success_last_step | Whether the rider succeeds to grab |
| | grab_position | Position of the grabbed order in the list |
| | increment | Increment of route length and time delay |
| | onhand_waybills_features | Order attribute features of carried orders |
| | current_list_features | Order attribute features of order lists |
| | rider_lng, rider_lat | Position of rider |
| Interactive feature of all riders | available_waybill_num | Number of orders left to be grabbed |
| | grab_rider_num | Number of riders occurring grabbing behavior |
| | grab_success_rider_num | Number of riders who succeed grabbing orders |
| | onhand_waybill_num_all | Number of all the carried orders |
| | distance | Route length of all riders |
| Order attribute | is_prebook | Whether the order is a pre-booked order |
| | delivery_distance | Distance between pickup and delivery positions |
| | remain_time | Time left for delivery before promised time of customers |
| | enter_grabpool_dur | Time elapse since order enters the system |
| | prepare_pred_ks | Predicted preparation time of order |
| | waittime_pred | Predicted delivery time of customer |
| | sender_lng, sender_lat, rec_lng, rec_lat | Pickup and delivery positions of orders |

of riders can transfer from $s_t$ to $s_{t+1}$ chronologically. To achieve accurate state transitions, the feedbacks of the riders need to be simulated[27]. Here we adopt two supervised learning classification models based on large amounts of historical data to predict such feedback information. The first model is established to predict whether the order-grabbing behavior will occur in current order list and the second predicts which order the rider will grab if an order-grabbing behavior takes place in current order list. By establishing these two machine learning models, the state transition process of multiple riders can be accurately simulated.

• **Reward function $R$:** Reinforcement learning achieves the optimization of the long-term objective function by designing special reward functions. To optimize the delivery efficiency of the OFD platform and the delivery experience of riders in a period of time, we design the following reward function $r_t$ to guide the agent learning towards a promising direction:

$$r_t = \text{num}_t - \alpha\mu_t - \beta\Delta\eta_t - g \tag{1}$$

where $\text{num}_t$ represents the total number of orders that are grabbed by riders at the moment $t$, indicating that the OFD platform hopes to accomplish the delivery of orders as much as possible since the more orders that are grabbed and served by riders, the more benefits the OFD platform can obtain from the delivery service. $\mu_t$ is the number of order-grabbing conflicts that occur at moment $t$, which imposes a negative reward to the agent to help reduce the number of order-grabbing conflicts so that the delivery experience of riders can be guaranteed. $\Delta\eta_t$ represents the increment of the route length and time delay of all riders after moment $t$, which is used to simultaneously consider whether and how much the delivery efficiency of the platform and the delivery experience of riders are affected. $g$ is the default penalty when there is no grabbing behavior of any rider, indicating that the platform needs to recommend suitable order lists for riders and make the orders be grabbed and served by riders as soon as possible.

• **Discount rate $\gamma$:** Usually, simply focusing on the current feedback of the environment will inevitably cause short-sighted decision. $\gamma$ is designed to balance the short-term reward and long-term reward so that the agent is not trapped in local optimum, which is beneficial for achieving potentially higher cumulative reward. When $\gamma = 0$, the agent only concentrates on the immediate feedbacks while short-term reward and long-term reward are equally taken into consideration when $\gamma = 1$.

## 4 RL-Based Order Recommendation Method

Since there are multiple riders that need to be coordinated at each decision moment, we need to sequentially generate the order list for each rider. At the meantime, the order lists that are already generated need to be considered in subsequent decision steps for the rest riders so as to avoid the order-grabbing conflict.

To achieve such purpose, we propose an RL-based order recommendation method (RLORM) using actor-critic network and long short term memory (LSTM) unit. The overall structure of RLORM is shown in Fig. 3, where the actor-critic network is constructed with LSTM units so that the OFD platform can pay attention to the status of all riders when generating each

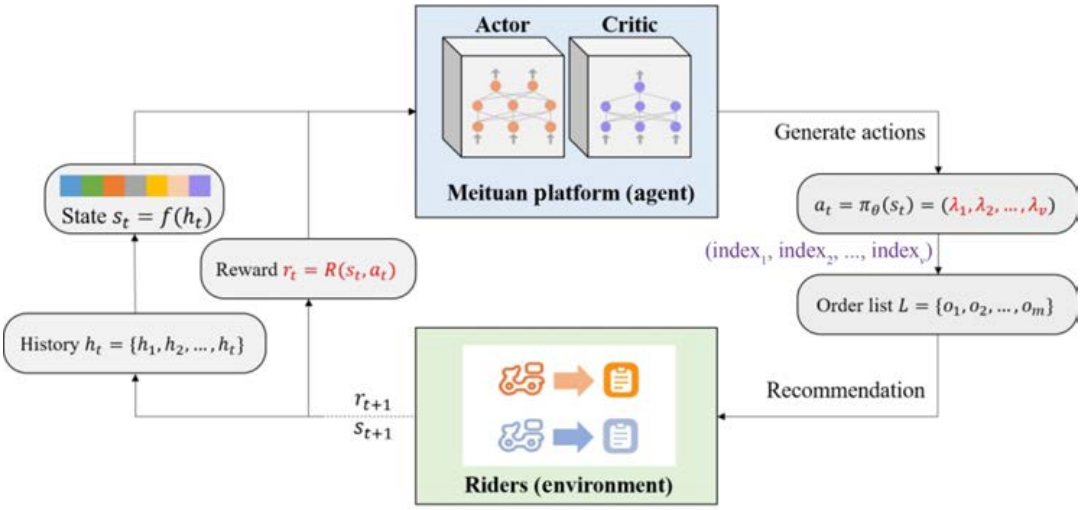


**Fig. 3 Overall structure of the proposed RLORM.**

recommendation policy. The details of the RLORM is introduced in the following content.

### 4.1 Actor-critic network

The actor-critic network consists of two parts: the actor network and the critic network. The actor outputs a recommendation policy $\pi_\theta$ at each interaction moment, trying to obtain the best reward. The critic estimates the value function of the policy output by the actor, which helps evaluate the quality of the policy and assists the actor to adjust the policy generation fashion at the next decision moment.

Since the dynamic order recommendation needs to sequentially generate order lists for $m_t$ riders at each moment, the previous order list is of great value for the subsequent policy generation process. For such sequence-dependent optimization tasks, recurrent neural networks (RNN) are capable of remembering the pre-states of the unit and passing the necessary information to the subsequent state of the unit, which shows a significant advantage in learning sequence-related nonlinear features[28]. Therefore, we adopt the LSTM unit[29] to implement the actor-critic network and model the state of multiple riders.

Specifically, we design an encoder-decoder based actor network to capture the inter-correlation between different riders. The structure of the actor network is shown in Fig. 4, which contains two LSTM units that sequentially work as the encoder and decoder. The LSTM unit of the encoder transforms the input sequence (corresponding to the state characteristics of different riders) into a fixed-length vector $c$ while the LSTM unit of the decoder combines the information of $c$ and $s^{\text{cross}}$, and outputs the order recommendation policy of different riders.

Given the state $s^j$ of each rider at the moment $t$, the input of the encoder can be expressed as $x_t = (s^1, s^2, \ldots, s^j)$. Then the output of the whole actor network at moment $t$ can be calculated as follows:

$$h^{(j)} = f\left(h^{(j-1)}, x_t\right) \tag{2}$$

$$c = h^{(m_t)} \tag{3}$$

$$h'^{(j)} = g(h'^{(j-1)}, y^{(j-1)}, c, s^{\text{cross}}) \tag{4}$$

$$a_t^j = y^{(j)} \tag{5}$$

After action $a_t^j$ is generated, the ranking score of each order can be determined for rider $j$, and the order recommendation list can be formed by selecting the top $n_t$ orders with the largest ranking score.

The critic network uses the same LSTM unit as the actor. The difference is that the output of the LSTM unit is followed with a feed forward neural network so as to predict the value function. Given the state $s^j$, $s^{\text{cross}}$, and action $a_t^j$, the approximate value function $Q^w(s_t, a_t)$ can be calculated with the following equation:

$$Q^w(s_t, a_t) = \text{Linear}(\text{ReLU}(a_t \oplus s^j \oplus s^{\text{cross}})) \tag{6}$$

where $\oplus$ represents the vector concatenation operation. According to the deterministic policy gradient theorem[30] and the temporal differential learning method[31], the loss functions of the actor and critic networks can be calculated as follows:

$$\text{loss}^a = \frac{1}{B}\sum_t -Q^w(s_t, a_t) \tag{7}$$

$$\text{loss}^c = \frac{1}{B}\sum_t (r_t + \Delta Q^w)^2 \tag{8}$$

where $B$ represents the batch size of training data. $\Delta Q^w = \gamma Q^w(s_{t+1}, a_{t+1}) - Q^w(s_t, a_t)$, representing

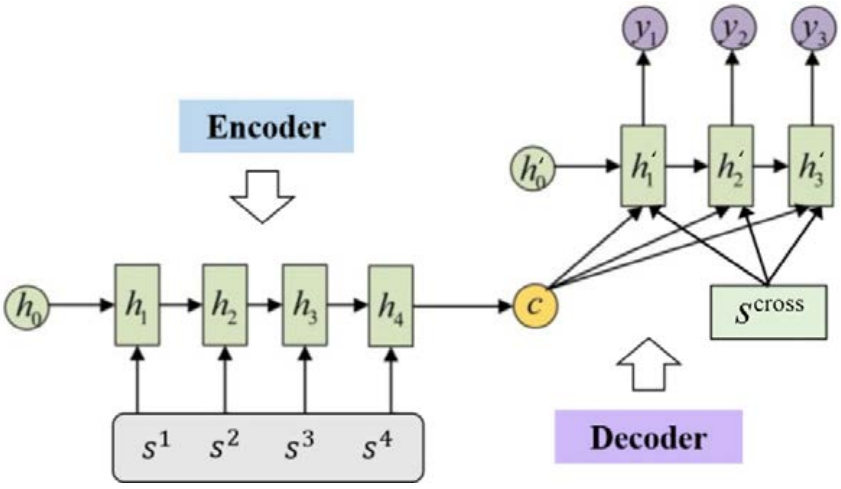


**Fig. 4 Structure of encode-decoder based actor network.**

the difference of value function between next moment and current moment. In order to overcome the dynamic change of network parameters during updating process, we also adopt the target network technology and the sliding average updating method[32].

### 4.2 Rider sequencing rules

From the structure of the actor network, it can be seen that the output of the LSTM unit at each time step corresponds to a single rider's recommendation policy. Therefore, to determine the matching between different time steps of the LSTM unit and different riders, several rider sequencing rules are designed as follows:

• **Sequencing by the number of grabbed orders (Seq-grab):** This rule sorts all riders in a descending order according to the number of orders that are already grabbed by riders. Riders from top of the sorting result to tail corresponds to the time step of the LSTM unit from 1 to $m_t$. The motivation of designing this rule is to generate the order recommendation lists for riders with large numbers of carried orders in advance because these riders are less likely to form new order-grabbing behaviors.

• **Sequencing by the order-grabbing success rate (Seq-succeed):** This rule sorts all riders in an ascending order of the order-grabbing success rate of riders, which gives more priority to riders that often fail to grab orders. This rule aims to improve the average delivery experience of riders and help avoid the occurrence of continuous order-grabbing failures.

• **Random sequencing (Seq-rand):** This rule randomly sorts riders to match the incremental output of LSTM unit.

In the subsequent numerical experiment, the proposed RLORM will be executed using the above rider sequencing rules to compare the performance of different rules.

### 4.3 Training and online test of the actor-critic network

The training steps of actor-critic network are shown in Algorithm 1.

First, we randomly initialize the actor and the critic network. The actor network generates actions $a_t = \left[y^{(1)}, y^{(2)}, \ldots y^{(m_t)}\right]$ based on current state $s_t = \left[s^1, s^2, \ldots, s^{m_t}, s^{\text{cross}}\right]$ and forms the corresponding order recommendation lists of different riders. Then the riders browse the order lists and give feedbacks to the platform. The state transfers from $s_t$ to $s_{t+1}$.

**Algorithm 1 Training steps of the actor-critic network**

**Input:** The size of data batch batch_size, the learning rate of critic network $l^r$, the learning rate of actor network $l^a$, and discount factor $\gamma$.
**Output:** The well-trained parameters of actor-critic network.
Randomly initialize the network parameters $\theta$ and $w$.
Initialize the replay buffer $\mathcal{D}$.
**for** session $= 1$ to $\mathcal{M}$ **do**
  Randomly generate the initial state $s_0$.
  **for** $t = 1$ to $T$ **do**
    Observe current state $s_t = s^1, s^2, \ldots, s^{m_t}, s^{\text{cross}}]$.
    Obtain the action $a_t$ output by actor network.
    Calculate the reward $r_t$ and transfer the state $s_t$ to $s_{t+1}$.
    Save the sample $(s_t, a_t, r_t, s_{t+1})$ to $\mathcal{D}$.
    Randomly take batch_size samples from $\mathcal{D}$ to train the network.
    Optimize the critic network by minimizing $\text{loss}^c$.
    Optimize the actor network by minimizing $\text{loss}^a$.
    Update the parameters $\theta'$ and $w'$ of the target network.
  **end for**
**end for**
Return the trained parameters of actor-critic network.

The online test steps for the actor-critic network are shown in Algorithm 2.

First, the states of riders are randomly initialized. Given current state $s_t$, the action $a_t$ and order recommendation lists of multiple riders can be generated using the well-trained actor network. Then the feedbacks of each rider is obtained by the machine learning models constructed in transition function of MDP formulation. Finally, the reward value at current decision moment is determined by Eq. (1).

**Algorithm 2 Online test steps of the actor-critic network**

**Input:** The size of data batch batch_size, the learning rate of critic network $l^r$, the learning rate of actor network $l^a$, and discount factor $\gamma$.
**Output:** The cumulative reward $r$ of all riders.
Initialize the actor network with the well-trained parameters.
**for** session $= 1$ to $\mathcal{M}$ **do**
  Randomly generate the initial state $s_0, r = 0$.
  **for** $t = 1$ to $T$ **do**
    Observe current state $s_t$.
    Execute action $a_t = \text{Actor}(s_t)$ and generate the order lists.
    Obtain the reward $r_t$ using the trained simulative models.
    Calculate the cumulative reward $r = r + r_t$.
    Transfer the $s_t$ to $s_{t+1}$.
  **end for**
**end for**
Return the cumulative reward $r$.

## 5 Numerical Experiment

### 5.1 Experimental setting

In order to effectively train and test the proposed method, we collect the real delivery data of orders and riders at different times from the Meituan delivery platform to form the training and test dataset. Specifically, 3.66 million interaction samples of different riders and platforms from November 10, 2022 to November 17, 2022 are collected and the ratio of training set to test set is 7:3.

To ensure the fairness and accuracy of the comparative experiments, all algorithms are implemented in Python 3.6 and Tensorflow 2.0, and run on the same Mac OS device equipped with a 2.2 GHz processor and 16 GB RAM. All machine learning models are trained and tested on the same type of servers on Meituan delivery platform.

As for the parameter setting, the network hyper-parameters are set as follows. The maximum number of steps that the rider interacts with the OFD platform is $T = 40$. The maximal number of riders is $m_t = 20$. The output dimension of the LSTM unit is set as 128. The learning rate of the actor and the critic network are set as $10^{-4}$ and $10^{-3}$, respectively. The updating weighting factor $\lambda_1 = \lambda_2 = 0.99$. Batch size $B$ is set as 64. The parameters in the reward function $\alpha$, $\beta$, and $g$ are all set as 0.1 according to real statistical experience. Finally, the number of repetitive training times per sample is set as 500.

We use five comparative metrics to evaluate different methods, which are the total interaction step (TIS), cumulative reward (CR), total number of grabbed orders (TNGO), total number of order-grabbing conflicts (TNOGC), and total increment of route length and time delay (TIRLTD). These metrics are closely related to the delivery efficiency of the platform and delivery experience of riders and can comprehensively evaluate the performance of order recommendation methods.

### 5.2 Comparison of different rider sequencing rules

Figure 5 gives the comparison results of different rider sequencing rules. It can be seen that Seq-succeed achieves the best performance among the three comparative rules, resulting in the shortest interactive steps and the largest cumulative reward.

In addition, from the number of grabbed orders and the number of order-grabbing conflicts, it can be found that compared with the other two rules, Seq-succeed can effectively reduce the number of order-grabbing conflicts and increase the probability of orders being selected by riders, which shows that sorting riders according to the order-grabbing success rate can effectively arrange the orders among different riders and help the OFD platform achieve the overall optimization of the delivery experience of all riders.

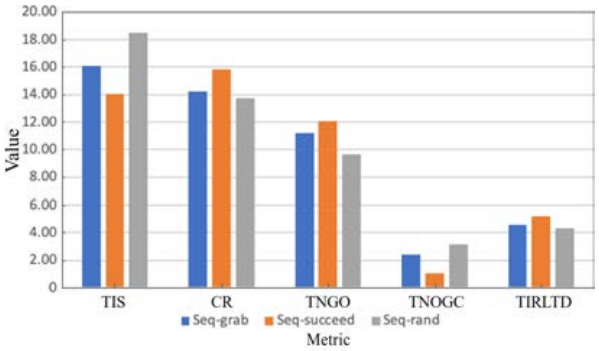


**Fig. 5 Comparison results of different rider sequencing rules.**

Finally, from the increment of route length and time delay, it can be noticed that Seq-succeed has the largest increment among three rules, which is because Seq-succeed increases the number of grabbed orders, resulting in more delivery distance. Despite this, Seq-succeed is still significantly better than the comparative rules in other metrics. Therefore, Seq-succeed is suitable to be used as the best rider sequencing rule in the subsequent comparative experiments.

### 5.3 Comparison with typical order recommendation methods

To validate the effectiveness of our proposed order recommendation method, we compare the RLORM with several representative order recommendation methods in OFD scenario, which are introduced as follows:

• **Fetch_distance:** For each rider at current moment, sort the orders in an ascending order of the pickup distance, and select the first $n_t$ orders to form an order list.

• **Wish_prob:** For each rider at current moment, sort the orders in a descending order according to the probability of grabbing orders (which is estimated by the machine learning model trained by Meituan platform), and select the first $n_t$ orders to form the order list.

• **Distance_score:** For each rider at current moment, calculate the delivery distance increment of the rider's route after each order is (virtually) assigned to the rider, and arrange them in an ascending order of the increment. Select the first $n_t$ orders to form the order list.

• **Random:** Randomly generate an order list for each rider.

Table 3 shows the results of RLORM and the comparative methods. It can be seen that RLORM is significantly better than the comparatives in terms of interactive steps, cumulative rewards, number of grabbed orders, number of order-grabbing conflicts, and increment of route length and time delay. This is because RLORM can pay attention to the recommendation policies of other riders when determining the recommendation policy of current rider. Therefore, the order-grabbing conflict can be effectively avoided, which helps improve the success rate of the order-grabbing, reduce the number of interactive step between riders and the platform, and improve the recommendation and delivery efficiency.

Among the comparative methods, Wish_prob is superior to others because it can use a variety of features to capture the rider's willingness of order-grabbing. Distance_score achieves better results than Fetch_distance, which means that riders are more concerned about the quality of the delivery route rather than the pickup distance after they grab orders. Besides, those orders that have a long delivery distance are less possible to be favored by riders.

## 6 Conclusion and Future Work

In this paper, we study the dynamic order recommendation problem in the OFD scenario where multiple riders and order-grabbing conflict are considered. We model the problem as a multi-step sequential decision-making process based on MDP, and propose an RL-based order recommendation method to generate order recommendation policies for different riders. We design an encoder-decoder based actor-critic network based on LSTM to sequentially form the order recommendation list for each rider. The actor-critic network can take into account the information of order lists of previous riders when deciding the order recommendation policy of current rider, which effectively alleviate the order-grabbing conflict between riders. Besides, several rider sequencing rules are designed and compared. Experimental results show that sequencing with the order-grabbing success rate achieves the best performance. Through extensive comparison, it is proved that the proposed RLORM can effectively improve the delivery efficiency of the OFD platform and the delivery experience of riders.

**Table 3 Comparative results of RLORM with other methods.**

| Method | TIS | CR | TNGO | TNOGC | TIRLTD |
|---|---|---|---|---|---|
| RLORM | **14.070** | **15.844** | **12.036** | **1.049** | **5.187** |
| Fetch_distance | 20.719 | 10.658 | 10.365 | 2.166 | 6.571 |
| Wish_prob | 17.548 | 12.541 | 11.429 | 2.589 | 6.227 |
| Distance_score | 18.424 | 12.202 | 11.781 | 1.436 | 6.373 |
| Random | 28.003 | 2.682 | 6.883 | 4.827 | 8.639 |

For future work, we will try to explore more order assignment patterns in OFD service, such as the one-to-one, many-to-one, and many-to-many matching problems. Besides, it is of great meaning to adapt the reinforcement learning technique to more application scenarios, which might bring much benefits to many realistic businesses.

### Acknowledgment

This work was supported in part by the National Natural Science Foundation of China (No. 62273193), Tsinghua University—Meituan Joint Institute for Digital Life, and the Research and Development Project of CRSC Research & Design Institute Group Co., Ltd.

### References

[1] L. Li, Y. Chai, and Y. Liu, Evolution of e-commerce patterns: Model and economic analysis, (in Chinese), *Journal of Tsinghua University* (*Science and Technology*), vol. 52, no. 11, pp. 1524–1529, 2012.

[2] X. Liu and Y. Li, VRP model and a heuristic algorithm for across-region distribution in the environment of E-commerce, (in Chinese), *Journal of Tsinghua University* (*Science and Technology*), vol. 46, pp. 1014–1018, 2006.

[3] A. Seghezzi, M. Winkenbach, and R. Mangiaracina, On-demand food delivery: A systematic literature review, *Int. J. Logist. Manag.*, doi: 10.1108/IJLM-03-2020-0150.

[4] Meituan, Homepage of Meituan delivery, https://peisong.meituan.com/about, 2023.

[5] C. Li, and L. Miao, Planning methods of regional logistics systems and logistics parks, (in Chinese), *Journal of Tsinghua University* (*Science and Technology*), vol. 44, no. 3, pp. 398–401, 2004.

[6] X. Wang, L. Wang, C. Dong, H. Ren, and K. Xing, An online deep reinforcement learning-based order recommendation framework for rider-centered food delivery system, *IEEE Trans. Intell. Transp. Syst.*, vol. 24, no. 5, pp. 5640–5654, 2023.

[7] E. Jiang, L. Wang, and J. Wang, Decomposition-based multi-objective optimization for energy-aware distributed hybrid flow shop scheduling with multiprocessor tasks, *Tsinghua Science and Technology*, vol. 26, no. 5, pp. 646–663, 2021.

[8] D. Reyes, A. Erera, M. Savelsbergh, S. Sahasrabudhe, and R. J. O'Neil, The meal delivery routing problem, https://optimization-online.org/?p=15139, 2018.

[9] B. Yildiz and M. Savelsbergh, Provably high-quality solutions for the meal delivery routing problem, *Transp. Sci*., vol. 53, no. 5, pp. 1372–1388, 2019.

[10] M. W. Ulmer, B. W. Thomas, A. M. Campbell, and N. Woyak, The restaurant meal delivery problem: Dynamic pickup and delivery with deadlines and random ready times, *Transp. Sci*., vol. 55, no. 1, pp. 75–100, 2021.

[11] S. Liu, L. He, and Z. J. M. Shen, On-time last-mile delivery: Order assignment with travel-time predictors, *Manag. Sci.*, vol. 67, no. 7, pp. 4095–4119, 2021.

[12] J. F. Chen, L. Wang, H. Ren, J. Pan, S. Wang, J. Zheng, and X. Wang, An imitation learning-enhanced iterated matching algorithm for on-demand food delivery, *IEEE Trans. Intell. Transp. Syst*., vol. 23, no. 10, pp. 18603–18619, 2022.

[13] Z. Steever, M. Karwan, and C. Murray, Dynamic courier routing for a food delivery service, *Comput. Oper. Res*., vol. 107, pp. 173–188, 2019.

[14] S. Paul, S. Rathee, J. Matthew, and K. M. Adusumilli, An optimization framework for on-demand meal delivery system, in *Proc. 2020 IEEE Int. Conf. Industrial Engineering and Engineering Management* (*IEEM*), Singapore, 2020, pp. 822–826.

[15] M. Joshi, A. Singh, S. Ranu, A. Bagchi, P. Karia, and P. Kala, Batching and matching for food delivery in dynamic road networks, in *Proc. 2021 IEEE 37th Int. Conf. Data Engineering* (*ICDE*), Chania, Greece, 2021, pp. 2099–2104.

[16] H. Jahanshahi, A. Bozanta, M. Cevik, E. M. Kavuk, A. Tosun, S. B. Sonuc, B. Kosucu, and A. Başar, A deep reinforcement learning approach for the meal delivery problem, *Knowl. Based Syst*., vol. 243, p. 108489, 2022.

[17] L. Wang, Z. Pan, and J. Wang, A review of reinforcement learning based intelligent optimization for manufacturing scheduling, *Complex System Modeling and Simulation*, vol. 1, no. 4, pp. 257–270, 2021.

[18] G. Shani, D. Heckerman, and R. I. Brafman, An MDP-based recommender system, *J. Mach. Lear. Res.*, vol. 6, no. 43, pp. 1265–1295, 2005.

[19] N. Taghipour and A. Kardan, A hybrid web recommender system based on Q-learning, in *Proc. 2008 ACM Symp. on Applied Computing*, Fortaleza, Brazil, 2008, pp. 1164–1168.

[20] X. Bai, J. Guan, and H. Wang, A model-based reinforcement learning with adversarial training for online recommendation, in *Proc. 33rd Int. Conf. Neural Information Processing Systems*, Vancouver, Canada, 2019, pp. 10735–10746.

[21] X. Xin, A. Karatzoglou, I. Arapakis, and J. M. Jose, Self-supervised reinforcement learning for recommender systems, in *Proc. 43rd Int. ACM SIGIR Conf. Research and Development in Information Retrieval*, Virtual Event, China, 2020, pp. 931–940.

[22] X. Chen, C. Huang, L. Yao, X. Wang, W. Liu, and W. Zhang, Knowledge-guided deep reinforcement learning for interactive recommendation, in *Proc. 2020 Int. Joint Conf. Neural Networks* (*IJCNN*), Glasgow, UK, 2020, pp. 1–8.

[23] X. Zhao, L. Zhang, Z. Ding, L. Xia, J. Tang, and D. Yin, Recommendations with negative feedback via pairwise deep reinforcement learning, in *Proc. 24th ACM SIGKDD Int. Conf. Knowledge Discovery & Data Mining*, London, UK, 2018, pp. 1040–1048.

[24] Y. Deng, F. Bao, Y. Kong, Z. Ren, and Q. Dai, Deep direct reinforcement learning for financial signal representation and trading, *IEEE Trans. Neural Netw. Learn. Syst*., vol. 28, no. 3, pp. 653–664, 2017.

[25] L. Zou, L. Xia, Z. Ding, J. Song, W. Liu, and D. Yin, Reinforcement learning to optimize long-term user engagement in recommender systems, in *Proc. 25th ACM SIGKDD Int. Conf. Knowledge Discovery & Data Mining*, Anchorage, AK, USA, 2019, pp. 2810–2818.

[26] X. Wang, L. Wang, S. Wang, J. F. Chen, and C. Wu, An XGBoost-enhanced fast constructive algorithm for food delivery route planning problem, *Comput. Ind. Eng*., vol. 152, p. 107029, 2021.

[27] Y. Tang, L. Li, and X. Liu, State-of-the-art development of complex systems and their simulation methods, *Complex System Modeling and Simulation*, vol. 1, no. 4, pp. 271–290, 2021.

[28] H. Salehinejad, S. Sankar, J. Barfett, E. Colak, and S. Valaee, Recent advances in recurrent neural networks, arXiv preprint arXiv: 1801.01078, 2017.

[29] S. Hochreiter and J. Schmidhuber, Long short-term memory, *Neural Comput*., vol. 9, no. 8, pp. 1735–1780, 1997.

[30] D. Silver, G. Lever, N. Heess, T Degris, D. Wierstra, and M Riedmiller, Deterministic policy gradient algorithms, in *Proc. 31st Int. Conf. Int. Conf. Machine Learning*, Beijing, China, 2014, pp. 387–395.

[31] C. M. Bishop and N. M. Nasrabadi, *Pattern Recognition and Machine Learning*. New York, NY, USA: Springer, 2006.

[32] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, Continuous control with deep reinforcement learning, arXiv preprint arXiv: 1509.02971, 2015.

**Ling Wang** received the BSc degree in automation and the PhD degree in control theory and control engineering from Tsinghua University, Beijing, China in 1995 and 1999, respectively. Since 1999, he has been with Department of Automation, Tsinghua University, Beijing, China, where he became a full professor in 2008. He has authored five academic books and more than 300 refereed papers. His current research interests include computational intelligence based optimization and scheduling. He is a recipient of the National Natural Science Fund for Distinguished Young Scholars of China, the National Natural Science Award (Second Place) in 2014, the Science and Technology Award of Beijing City in 2008, and the Natural Science Award (First Place in 2003, and Second Place in 2007) nominated by the Ministry of Education of China. He is the editor-in-chief for the *International Journal of Automation and Control*, and the associate editor for the *IEEE Transactions on Evolutionary Computation*, *Swarm and Evolutionary Computation*, etc.

**Xing Wang** received the BSc degree from Northwestern Polytechnical University, Xi'an, China, in 2017. He is currently pursuing the PhD degree at Tsinghua University, Beijing, China. His current research interest includes intelligent optimization on complex scheduling problems.

**Chenxin Dong** received the Bachelor degree in vehicle engineering from China Agricultural University in 2010 and the MEM degree from China University of Petroleum (East China) in 2020. He also studied in Kettering University for one year in 2017. Currently, he is a lecturer in Qingdao Hengxing University of Science and Technology, China. His current research interests include optimization and automotive engineering.

**Hao Ren** received the BSc and MS degrees in industrial engineering from Tianjin University, Tianjin, China in 2016 and 2019, respectively. He is an algorithm engineer at Meituan, China. His current research direction is the application of machine learning in online food delivery.

**Ke Xing** received the BSc degree in computer science from Harbin Engineering University, Harbin, China in 2016. He is an algorithm engineer at Meituan, China. His current research interests include the application of machine learning, spatial data mining, and ubiquitous computing.