TSINGHUA SCIENCE AND TECHNOLOGY
ISSN 1007-0214 07/24 pp386–399
DOI: 10.26599/TST.2023.9010069
Volume 29, Number 2, April 2024

A Matching Algorithm with Reinforcement Learning and Decoupling Strategy for Order Dispatching in On-Demand Food Delivery

Jingfang Chen, Ling Wang*, Zixiao Pan, Yuting Wu, Jie Zheng, and Xuetao Ding

Abstract: The on-demand food delivery (OFD) service has gained rapid development in the past decades but meanwhile encounters challenges for further improving operation quality. The order dispatching problem is one of the most concerning issues for the OFD platforms, which refer to dynamically dispatching a large number of orders to riders reasonably in very limited decision time. To solve such a challenging combinatorial optimization problem, an effective matching algorithm is proposed by fusing the reinforcement learning technique and the optimization method. First, to deal with the large-scale complexity, a decoupling method is designed by reducing the matching space between new orders and riders. Second, to overcome the high dynamism and satisfy the stringent requirements on decision time, a reinforcement learning based dispatching heuristic is presented. To be specific, a sequence-to-sequence neural network is constructed based on the problem characteristic to generate an order priority sequence. Besides, a training approach is specially designed to improve learning performance. Furthermore, a greedy heuristic is employed to effectively dispatch new orders according to the order priority sequence. On real-world datasets, numerical experiments are conducted to validate the effectiveness of the proposed algorithm. Statistical results show that the proposed algorithm can effectively solve the problem by improving delivery efficiency and maintaining customer satisfaction.

Key words: order dispatching; on-demand delivery; reinforcement learning; decoupling strategy; sequence-to-sequence neural network

1 Introduction

The wide application of computer and Internet technology has promoted the online-to-offline (O2O) e-commerce which has gained significant growth[1] over the past decades. Meanwhile, with the support of modern logistics, O2O e-commerce spawns the on-demand food delivery (OFD) service. Some of the most well-known OFD providers include Meituan, Ele me, Uber Eats, Grubhub, Deliveroo, and Swiggy. The users only need to place orders on smartphones, and the platforms will inform the restaurants to prepare meals and send riders to deliver meals to the users. Recently, with the rapid development of the lazy economy, the traditional OFD platform is not limited to providing meal delivery and extends to delivering medicine, flowers, and all kinds of goods from supermarkets, which have deeply seeped into most aspects of daily life. Such a purchase mode saves time for users on cooking at home or heading to the stores so that they can spare and enjoy more leisure time. Besides, the OFD service also offers job opportunities with decent wages for couriers and helps restaurants

• Jingfang Chen, Ling Wang, Zixiao Pan, Yuting Wu, and Jie Zheng are with Department of Automation, Tsinghua University, Beijing 100084, China. E-mail: cjf17@tsinghua.org.cn, wangling@mail.tsinghua.edu.cn; pzx19@mails. tsinghua.edu.cn; wyt20@mails.tsinghua.edu.cn; j-zheng18@mails.tsinghua.edu.cn.
• Xuetao Ding is with Department of Delivery Technology, Meituan, Beijing 100102, China. E-mail: dingxuetao@meituan.com.
* To whom correspondence should be addressed.
Manuscript received: 2023-05-31; revised: 2023-06-30; accepted: 2023-07-05

or stores overcome the disadvantages in locations[2] by connecting them with potential online customers. As a result, OFD commerce attracts more and more users, freelancers, and business owners, leading to a continuous increase in market scale. According to the report by Statista, the worldwide market volume of online food delivery is projected to reach 1.65 trillion US dollars by 2027 from 2023 with an expected compound annual growth rate of 12.67%[3].

However, the business expansion also incurs fierce competition, which drives the OFD enterprises to improve service quality for a larger market share. One of the operational issues that many OFD providers are concerned about is how to dispatch orders for higher delivery efficiency and better customer satisfaction. This issue is more prominent during the on-peak dining hours when new orders need to be assigned to riders who have already carried multiple previously dispatched orders to serve. The OFD platforms should not only consider how to deliver new orders on time but also guarantee that the assignment will not influence the unfinished orders too much. In a word, it is crucial to obtain a reasonable dispatching scheme that can meet all demands on time while reducing the detours for riders as well.

Essentially, the order dispatching problem together with the route planning problem can be formulated as a special extension of the vehicle routing problem, which is named as OFD problem in this paper. Besides its NP-hard nature, there are multiple challenges of the order dispatching problem under realistic circumstances, such as high dynamism, large-scale complexity, and stringent time requirements. To be specific, new orders are flooding into the platforms continuously and the locations of riders are changing dynamically, which mean that the platforms need to make real-time decisions. Besides, the number of orders and riders to be scheduled is massive in China. Moreover, to ensure the taste of food and spare enough time for riders to deliver, the platforms are forced to make very quick decisions (usually in tens of seconds). Therefore, the order dispatching problem in OFD has been acknowledged as the ultimate challenge in last-mile logistics[4, 5].

Such a challenging problem has drawn increasing attention from researchers in recent years. Cosmi et al.[6, 7] modeled the OFD problem with one rider and one restaurant as a single machine scheduling problem, which was formulated into mixed integer linear programming (MILP) models, and optimized by a dynamic programming algorithm[6] and optimizer Gurobi[7]; for the similar scenario, Yu et al.[8] derived the lower bounds for various rider capacities and designed two dispatching algorithms to minimize the delay. For a special OFD scenario where a customer could order meals from multiple restaurants, Steever et al.[9] built an MILP model and designed two heuristics, and Wang et al.[10] proposed a three-stage order dispatch scheme that outperformed the heuristics in Ref. [9]. Ulmer et al.[11] presented an anticipatory customer assignment policy with a postponement strategy and time buffer to account for the stochasticity from dynamic requests and food preparation. Liu et al.[12] integrated travel time predictors into the order assignment optimization and proposed a branch-and-price algorithm. Kohar and Jakhar[13] presented an augmented two-index formulation to model the capacitated multi pickup online food delivery problem and proposed a branch-and-cut algorithm. Tao et al.[2] proposed a personalized order assignment and routing model for different types of riders and developed a tabu search algorithm. For a future scenario where food is delivered by unmanned aerial vehicle, Liu[14] built a mixed integer programming model and proposed an optimization-driven progressive algorithm, and Huang et al.[15] proposed an iterated heuristic framework integrated with a simulated annealing based local search to periodically schedule tasks.

Besides, more and more researchers begin to focus on the realistic scenarios originating from the OFD platforms. For the meal routing problem from Grubhub, Reyes et al.[4] developed a rolling horizon algorithm while Yildiz and Savelsbergh[5] proposed a new mathematical model solved by a simultaneous column and row generation method. For the order dispatching problem from Meituan, Chen et al.[16, 17] and Zheng et al.[18] designed a matching algorithm with an adaptive tie-breaking strategy, an imitation-learning enhanced iterated matching algorithm, and a filtration-based iterated greedy algorithm to effectively dispatch orders; besides, multiple efficient route planning techniques were developed in Refs. [19–21]. For the OFD scenario in Swiggy, Kottakki et al.[22] modeled the customer experience as a time-variant piece-wise linear function and built a multi-objective optimization model solved by Gurobi; Paul et al.[23] proposed a generic optimization framework with a batching algorithm and a mathematical model to assign the order batches; Joshi et al.[24] also proposed a FOODMATCH algorithm to group the orders and assign the order batches. For the meal delivery problem from Getir, Jahanshahi et al.[25] and Bozanta et al.[26] modeled the dynamic arrival of orders and the rider behaviors as a Markov decision process where the riders can accept or reject the orders. The authors trained multiple extensions of the deep Q-Network to dispose of the dynamic customer requests.

From the above literature, it is observed that the OFD scenarios considered in different works differ because the operation mode and market size vary across different regions and OFD platforms. In this paper, we typically investigate the OFD problem in China which constitutes the biggest share worldwide according to Statista[3]. Taking Meituan as an example, there are over 2 000 000 orders and over 100 000 riders to be scheduled per day on average in Beijing, which amounts to a large-scale combinatorial optimization problem. Meanwhile, the platform needs to periodically solve the problem every minute. In this case, most existing exact methods or swarm intelligence based optimization algorithms[27] cannot be directly applied because these methods are too time-consuming to solve large-scale instances. The reinforcement learning (RL) approach has shown great potential to solve complex combinatorial optimization problems effectively and efficiently, which can meet the stringent time requirements of OFD. To the best of our knowledge, few research papers have applied RL to the OFD problem except Refs. [25] and [26]. However, the problem scale is smaller (less than 300 daily orders and less than 10 riders) and the new orders can be rejected in Refs. [25] and [26], which is far from the problem scenario in this paper so the proposed RL method cannot be directly employed. Therefore, it is still of significance to develop fast and effective algorithms to meet such extreme OFD requirements on limited computation time and large problem scale.

In this paper, we address the order dispatching problem originating from one of the largest OFD platforms in China, and propose a reinforcement learning and decoupling strategy based matching algorithm (RLDMA). First, to deal with the large-scale complexity, a decoupling method is designed, which reduces the solution space by only considering the riders with the best potential for each new order. Second, to tackle the high dynamism and meet the stringent time requirements, an RL-based dispatching algorithm is developed, which periodically dispatches new orders to appropriate riders. Specifically, a sequence-to-sequence (seq2seq) network is constructed based on the problem characteristic to generate the order priority sequence, which is trained by a specially designed training approach to improve the learning performance; then, a greedy heuristic is employed to effectively dispatch new orders. Numerical experiments are conducted to test the performance of the proposed algorithm by comparing it with other methods. Statistical results show that the proposed algorithm can well solve the problem by improving delivery efficiency and maintaining customer satisfaction.

2 Problem Description

To respond to the dynamically arriving orders in time, the rolling horizon approach is employed, which accumulates the new orders and solves the following static optimization problems in a series of time windows. With the notations in Table 1, the order dispatching problem can be described as follows:

At the current time T, a set of n new orders O received in $[T-\Delta T, T]$ need to be dispatched to a set of m riders Q. Each rider q_j may be delivering a set of old orders O_j^o with a travel speed v_j. Each new order can only be dispatched to a single rider, and each old order cannot be re-dispatched to other riders. Each order o_i needs to be delivered before a committed soft deadline t_i. The

Table 1 Notations.

Notation		Description
Input parameter	T	Current time to make decisions
	ΔT	Length of a time window
	n	Number of new orders received in $[T-\Delta T, T]$
	m	Number of riders
	Q	Set of all riders. $Q=\{q_1,q_2,\ldots,q_m\}$
	O	Set of all new orders. $O=\{o_1,o_2,\ldots,o_n\}$
	O_j^o	Set of undelivered old orders carried by q_j
	O^o	Set of all undelivered old orders. $O^o = O_1^o \cup O_2^o \cup \cdots \cup O_m^o = \{o_{n+1}, o_{n+2}, \ldots, o_{n+u}\}$
	L	Set of all site nodes. $L=\{l_1,l_2,\ldots,l_m\}$
	P	Set of all pickup nodes. $P=\{i+ \mid o_i \in O \cup O^o\}$
	D	Set of all delivery nodes. $D=\{i- \mid o_i \in O \cup O^o\}$
	t_i	Committed delivery time of order o_i
	v_j	Travel speed of rider q_j
	d_{i_1,i_2}	Travel distance from node i_1 to node i_2
Decision variable	O_j^n	Set of new orders dispatched to q_j. $O_j^n \subseteq O$
Intermediate variable	R_j^n	New route of rider q_j
	R_j^o	Old route of rider q_j

old order that has been picked up is associated with a single delivery node $i_- \in D$. Each of the remaining old orders and all new orders is associated with a pickup node $i_+ \in P$ and a delivery node $i_- \in D$. Each rider q_j locates at node $l_j \in L$. Then the problem can be described on Graph $G = (V, A)$, where $V = L \cup P \cup D$ is the node set and A is the arc set. Each arc $(i_1, i_2) \in A$ $(i_1, i_2 \in V, i_1 \neq i_2)$ relates to a travel distance d_{i_1,i_2}. An example is provided in Fig. 1 to better describe the problem. There are several constraints as follows:

- Food cannot be picked up until it has been prepared.
- Food should be picked up first before being delivered.
- The total weight of food that a rider carries must be less than the trunk capacity.

The goal is to optimize customer satisfaction and delivery efficiency by dispatching each new order to riders appropriately. Define the set of new orders dispatched to each rider $O_j^n \subseteq O$ as the decision variables, and then the optimization objective is defined in Eq. (1).

$$\text{Min ADC} = \frac{1}{n}\sum_{q_j \in Q} C\left(O_j^n\right) \tag{1}$$

Equation (1) calculates the average dispatching cost (ADC) after dispatching all orders, where $C(O_j^n)$ is the cost of dispatching a set of new orders O_j^n to rider q_j, and the decision variable sets satisfy $O_1^n \cup O_2^n \cup \cdots \cup O_m^n = O$ and $O_j^n \cap O_{j'}^n = \varnothing$ for any two different riders q_j and $q_{j'}$. Given the intermediate variables new route R_j^n and old route R_j^o, i.e., the route of rider q_j after and before being dispatched with O_j^n, respectively, the dispatching cost $C(O_j^n)$ can be calculated in Eq. (2).

$$C\left(O_j^n\right) = w_t\left|\mathrm{TC}\left(R_j^n\right) - \mathrm{TC}\left(R_j^o\right)\right| + w_d\left|\mathrm{DC}\left(R_j^n\right) - \mathrm{DC}\left(R_j^o\right)\right| + L\left(R_j^n\right) \tag{2}$$

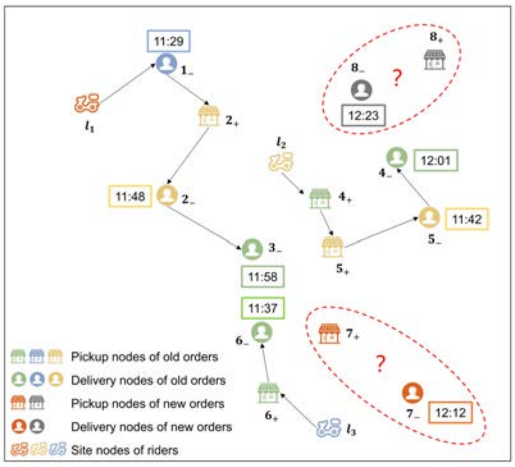
$$L\left(R_j^n\right) = \begin{cases} 0, & \text{if } R_j^n \text{ satisfies all constraints;} \\ M, & \text{if } R_j^n \text{ violates any constraint} \end{cases} \tag{3}$$

where $\mathrm{TC}\left(R_j\right)$ represents the delay of all orders in route R_j, $R_j \in \{R_j^n, R_j^o\}$, which reflects customer satisfaction; $\mathrm{DC}(R_j)$ is the total travel distance of route R_j, which reflects delivery efficiency; w_t and w_d are the weights to balance customer satisfaction and delivery efficiency, respectively; and $L(R_j^n)$ is a penalty term on the new route R_j^n to check if the constraints are satisfied, which is defined in Eq. (3) where M is a large number. The specific calculation of $\mathrm{TC}\left(R_j\right)$ and $\mathrm{DC}\left(R_j\right)$ is consistent with the previous researches[16–18].

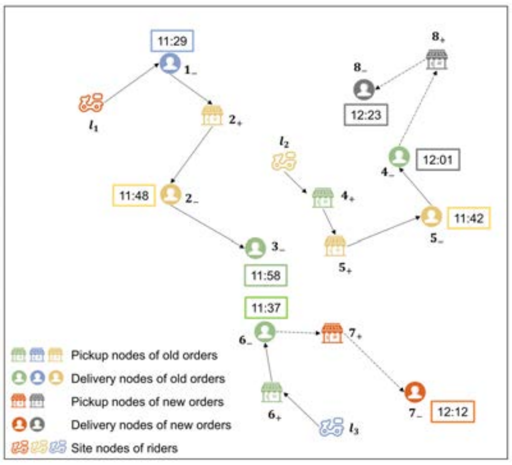
It should be noted that the new route R_j^n starts with the site node l_j, and is constructed with the pickup and delivery nodes of the old orders in O_j^o and the assigned new orders in O_j^n. Scheduling the route of a rider can be modeled as a pickup and delivery problem with a single vehicle and open route. Generally speaking, a route R_j can be obtained by any route planning method. Since this paper mainly studies the order dispatching problem and due to the limited space, the route planning methods used in this paper are not elaborated on but can be referred to in Refs. [17–19].

3 Methodology

As mentioned above, the exact algorithm and meta-



(a) New orders to be dispatched



(b) A feasible solution

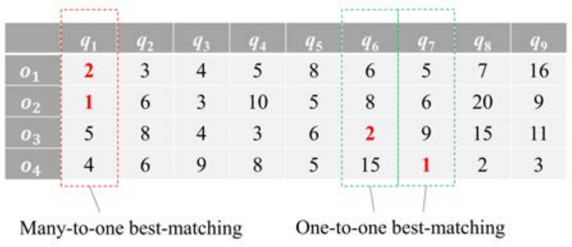
Fig. 1 An example of the order dispatching problem. There are two new orders to be dispatched and three riders with six old orders to serve. The committed delivery time of each order is given alongside the delivery node. A decision needs to be made on how to dispatch new orders o_7 and o_8, the pickup and delivery nodes of which are circled by the red dotted line. A feasible solution is given where o_7 and o_8 are dispatched to q_3 and q_2, respectively.

heuristics cannot well converge in limited decision time due to the large-scale complexity of the problem. Actually, constructive heuristics are usually adopted by the platforms as a trade-off method to meet the stringent time requirements. Therefore, this paper considers the heuristic method to solve the order dispatching problem. However, although it can generate a solution fast, the heuristic sometimes cannot well guarantee the solution quality. To overcome this issue, an effective dispatching heuristic is presented by fusing the advantages of the RL mechanism and optimization strategies. Specifically, to deal with the large-scale complexity, a decoupling method is designed which reduces the matching space based on the analysis of the relationship between new orders and riders. Besides, an RL-based dispatching method is developed, which exploits the dispatching priority of orders to improve the overall solution quality. Finally, according to the priority, the orders are dispatched to the corresponding best riders in order by the greedy heuristic to obtain a high-quality dispatching solution.

3.1 Calculating dispatching costs

The first step of RLDMA is constructing a dispatching cost matrix. Specifically, to evaluate whether it is reasonable to dispatch a new order to a certain rider, the dispatching cost is calculated for each pair of new orders and riders based on Eq. (2). An example of the dispatching cost matrix is illustrated in Fig. 2 where the dispatching cost of the best rider for each new order is marked with red color. According to the dispatching cost matrix, it is natural to consider matching each new order with its best rider. By analyzing each pair of the new order and its best rider, there will be two types of matching relationships.

The first is the one-to-one best matching where a best rider is related to only one new order. In this case, dispatching this new order to any other rider will cause the dispatching cost to increase, and any other new order is related to another rider with a smaller dispatching cost. Thus, it can be indicated that dispatching the new order to its best rider may be the optimal match.



	q_1	q_2	q_3	q_4	q_5	q_6	q_7	q_8	q_9
o_1	2	3	4	5	8	6	5	7	16
o_2	1	6	3	10	5	8	6	20	9
o_3	5	8	4	3	6	2	9	15	11
o_4	4	6	9	8	5	15	1	2	3

Many-to-one best-matching
One-to-one best-matching

Fig. 2 Dispatching cost matrix.

The second type is the many-to-one best matching where a best rider is related to more than one new order, which means that the delivery resources are relatively insufficient since there are more new orders than proper riders. In this case, it is necessary to decide whether to dispatch more than one order to a rider, defined as bundle dispatching for simplicity, or evenly dispatch the new orders to multiple riders. Note that bundle dispatching helps to reduce travel distance but may also be riskier at delivering all the orders on time. Therefore, how to trade off delivery efficiency and customer satisfaction by balancing the workload of riders is crucial for optimizing many-to-one best matching.

3.2 Decoupling

According to the above analysis, it is important to deal with the many-to-one best matching scenario. Since the many-to-one best matching only considers the rider with minimum dispatching cost for each new order, the matching space can be largely reduced. However, this greedy manner also means that it would be easy to be trapped in local minimum because it lacks the exploration of the matching space.

It is discovered from practical experience that the new orders are usually dispatched to one of the riders with smaller dispatching costs. Therefore, more attention can be paid to focusing on optimizing the matching relationship between each new order and these riders rather than all riders. Meanwhile, the exploration ability can also be enhanced since matching space is enlarged compared with the one-to-one and many-to-one best matching. However, a rider may simultaneously be the rider with smaller dispatching costs of multiple new orders, which means that the matching relationship is more complicated than the one-to-one and many-to-one best matching. Thus, it is of significance to reconstruct the matching relationship between new orders and riders.

To this end, this paper proposes a decoupling approach that aims at reducing the matching space while maintaining the exploration ability. For convenience, we define the following terms.

Top riders. Referring to the riders with smaller dispatching costs, which can be determined as follows: For a new order o_i, sort all the riders $q_j \in Q$ in ascending order of dispatching cost $C(O_j^n)$ where $O_j^n = \{o_i\}$, and the first K riders are the top riders of o_i where K is a parameter.

Candidate orders. For each top rider q_j of a new order o_i, o_i is defined as a candidate order of q_j.

Non-critical order-rider set. Which is composed of one new order and its top riders. Each rider in this set is not the top rider of any other new orders.

Critical order-rider set. Which is composed of at least two new orders and their top riders. There is at least one rider in this set that is simultaneously the top rider of more than one new order.

With the above definitions, the steps of the proposed decoupling method are described as follows:

Step 1: Initialize $O^u = O$. Find the top riders for each new order and the candidate orders for each top rider accordingly.

Step 2: Initialize the current order-rider set $(\mathcal{C}, \mathcal{R}) = (\varnothing, \varnothing)$.

Step 3: For the first order in O^u, put the corresponding top riders into set $\mathcal{C}$ and their candidate orders into set $\mathcal{R}$. For each order that is newly put into $\mathcal{C}$, put the corresponding top riders into $\mathcal{C}$ and their candidate orders into $\mathcal{R}$. Repeat the above process until no new orders or riders can be put into $\mathcal{C}$ or $\mathcal{R}$.

Step 4: Remove the orders in $\mathcal{C}$ from O^u. Store the current $(\mathcal{C}, \mathcal{R})$ and classify it as a non-critical order-rider set or critical order-rider set according to the number of new orders in $\mathcal{C}$.

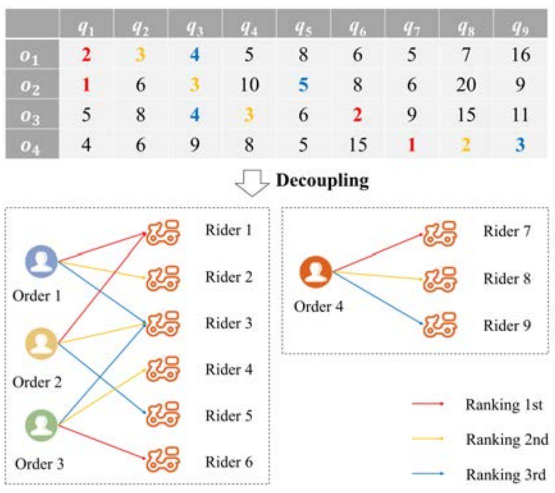
Step 5: If $O^u \neq \varnothing$, then turn to Step 2; otherwise, output all the stored non-critical order-rider sets and critical order-rider sets.

The pseudo-code is given in Algorithm 1. Figure 3 illustrates an example with 4 new orders and 9 riders, where K is set as 3. According to the decoupling method, all the new orders and riders can be decoupled as a critical order-rider set with 3 new orders and 6 riders and a non-critical order-rider set with 1 new order and 3 riders.

Compared with the original matching space with all new orders and riders, the problem scale of each critical or non-critical order-rider set is reduced. Then, for each non-critical order-rider set, the sole new order will be directly dispatched to its best rider. For each critical order-rider set, how to dispatch each new order will be determined by the following RL-based dispatching approach.

Algorithm 1 Decoupling method

```
Input: temporary new order set $O^u$
1: Initialize $\mathcal{T} \leftarrow \varnothing, \mathcal{N} \leftarrow \varnothing, O^u \leftarrow O$
2: while $O^u \neq \varnothing$
3:    Let $o_c \leftarrow$ POP$(O^u)$, $O^c \leftarrow \{o_c\}$
4:    Initialize $\mathcal{R} \leftarrow \varnothing, \mathcal{C} \leftarrow \{o_c\}$
5:    while $O^c \neq \varnothing$
6:       Let $o_c \leftarrow$ POP$(O^c)$
7:       for each top rider $q_j$ of $o_c$
8:          if $q_j \notin \mathcal{R}$:
9:             $\mathcal{R} = \mathcal{R} \cup \{q_j\}$
10:            for each candidate order $o_i$ of $q_j$
11:               if $o_i \notin \mathcal{C}$:
12:                  Let $\mathcal{C} = \mathcal{C} \cup \{o_i\}$
13:                  Let $O^c = O^c \cup \{o_i\}$
14:                  Let $O^u = O^u \setminus \{o_i\}$
15:               end if
16:            end for
17:         end if
18:      end for
19:   end while
20:   if $|\mathcal{C}| > 1$:
21:      $\mathcal{T} = \mathcal{T} \cup \{(\mathcal{C}, \mathcal{R})\}$
22:   else:
23:      $\mathcal{N} = \mathcal{N} \cup \{(\mathcal{C}, \mathcal{R})\}$
24:   end if
25: end while
Output: non-critical order-rider sets $\mathcal{N}$ and critical order-rider
   sets $\mathcal{T}$.
```



	q_1	q_2	q_3	q_4	q_5	q_6	q_7	q_8	q_9
o_1	2	3	4	5	8	6	5	7	16
o_2	1	6	3	10	5	8	6	20	9
o_3	5	8	4	3	6	2	9	15	11
o_4	4	6	9	8	5	15	1	2	3

Fig. 3 An example of the decoupling method.

3.3 RL-based dispatching

For each critical order-rider set, two decisions need to be made. The first is to decide the priority of dispatching each new order. The orders that are dispatched first will change the statuses of the corresponding riders, thus indirectly affecting the dispatching of the remaining new orders and the final dispatching results. This process is sequential decision-making where the RL mechanism can be introduced to explore. The second is to decide which rider each new order should be dispatched to, which directly affects the final dispatching quality.

To balance the exploration and exploitation, the

RL method and greedy heuristic are hybridized for dispatching. Specifically, a seq2seq framework is used to generate the order dispatching priority, which is trained by a specially designed RL method to explore the sequence space. The greedy heuristic is used to determine the rider for each new order, which is beneficial to guaranteeing the solution quality. The details are described as follows.

3.3.1 Encoder

The encoder extracts the information about the new orders and the relationship between them. In this paper, the gate recurrent unit, one type of recurrent neural network, is adopted as the encoder. The mechanism of the encoder is shown in Fig. 4. For each critical order-rider set, the following steps are executed.

Step 1: Use the one-dimensional convolutional neural network (Conv1d) to process the feature matrix constructed by the raw feature vector of each new order. The output of the Conv1d represents the high-dimensional embeddings of the new orders, which is denoted as $\{\boldsymbol{\omega}_1, \boldsymbol{\omega}_2, \ldots, \boldsymbol{\omega}_{n_t}\}$, where n_t is the number of new orders in the current critical order-rider set.

Step 2: Sort the new orders in descending order of critical rate. For a new order o_i, the critical rate is defined as the total number of candidate orders that the top riders of o_i are related to. If the top riders of o_i are simultaneously the top riders of more new orders, the critical rate of o_i is larger. In other words, a larger value of the critical rate indicates that the dispatching results of o_i will affect more new orders.

Step 3: According to the order sequence generated in Step 2, feed each new order into the encoder in turn until the encoder outputs the final hidden state $\boldsymbol{e}_{n_t}$.

The hidden state vector produced by the encoder represents the extracted information and will be fed into the decoder.

3.3.2 Decoder

The decoder is designed for generating the dispatching priority of new orders. The priority depends not only on the information of the new order itself but also on the rider information. This is because dispatching new orders to different riders will influence the subsequent dispatching of the remaining new orders. Therefore, it is crucial to make full use of the information of both new orders and riders.

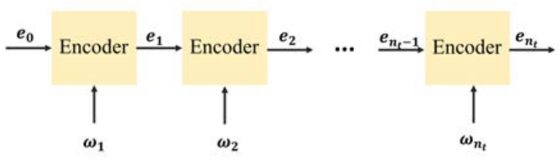


Fig. 4 Illustration of the encoder.

To enable the decoder to learn the relationship between each new order and riders sufficiently, the attention mechanism is employed, which extracts the importance of new orders with different riders and uses it as the input. Besides, the regret mechanism is also introduced, which is combined with the predictions of the network for generating the priority. The exploration is ensured by the network while the bad case can be avoided by the regret value. The mechanism of the decoder is illustrated in Fig. 5. The specific steps of the decoder are elaborated on as follows:

Step 1 (calculating the high-dimensional rider embeddings): Use Conv1d to process the feature matrix constructed by the raw feature vector of each rider. The output of the Conv1d represents the high-dimensional embeddings of the riders, which is denoted as $\{\boldsymbol{r}_1, \boldsymbol{r}_2, \ldots, \boldsymbol{r}_{m_t}\}$, where m_t is the number of riders in the critical order-rider set.

Step 2 (calculating the context vector): Use the final hidden state vector $\boldsymbol{e}_{n_t}$ of the encoder as the initial hidden state vector of the decoder, i.e., $\boldsymbol{d}_0 = \boldsymbol{e}_{n_t}$. Calculate the alignment scores, weights, and context vector with the embeddings of riders $\boldsymbol{r}_j$ and the hidden state vector $\boldsymbol{d}_i$ according to Eqs. (4) to (6):

$$u_j^i = \boldsymbol{v}^{\mathrm{T}} \tanh\left(\boldsymbol{W}_1 \boldsymbol{r}_j + \boldsymbol{W}_2 \boldsymbol{d}_i\right) \tag{4}$$

$$a_j^i = \operatorname{softmax}\left(u_j^i\right) \tag{5}$$

$$\boldsymbol{c}_i = \sum_{j=1}^{m_t} a_j^i \boldsymbol{r}_j \tag{6}$$

where vector $\boldsymbol{v}$, matrix $\boldsymbol{W}_1$, and matrix $\boldsymbol{W}_2$ are trainable parameters, u_j^i is the alignment score of the new order o_i and rider q_j, a_j^i is the normalized attention weight, and $\boldsymbol{c}_i$ is the context vector representing the relationship

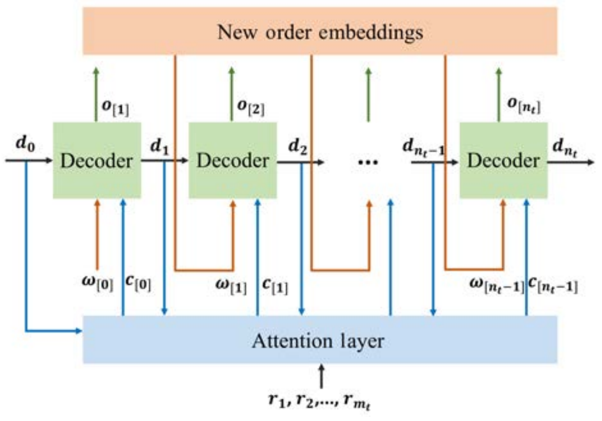


Fig. 5 Illustration of the decoder.

between the new order o_i and all the riders in the current critical order-rider set.

Step 3 (calculating the input of the decoder): Concatenate the high-dimensional embedding $\boldsymbol{\omega}_i$ of the previous output order o_i and the context vector $\boldsymbol{c}_i$ of o_i, which yields the new embedding $\boldsymbol{h}_i$ of o_i as Eq. (7). $\boldsymbol{h}_i$ is fed into the decoder as the next input.

$$\boldsymbol{h}_i = \boldsymbol{\omega}_i || \boldsymbol{c}_i \tag{7}$$

Step 4 (calculating the probability vector of the priority): Use the linear transformation to process the hidden state vector $\boldsymbol{d}_i$ of the decoder, which yields the probability vector of the priority $\boldsymbol{y}_i$ as Eq. (8) shows. Then, use the softmax function to process $\boldsymbol{y}_i$ and obtain the normalized probability vector of the priority $\boldsymbol{p}_i$ as Eq. (9) shows.

$$\boldsymbol{y}_i = \boldsymbol{W}_d \boldsymbol{d}_i + \boldsymbol{b}_d \tag{8}$$

$$\boldsymbol{p}_i = \text{softmax}\,(\boldsymbol{y}_i) \tag{9}$$

where $\boldsymbol{W}_d$ and $\boldsymbol{b}_d$ are the trainable parameter matrix and the trainable bias vector, respectively. It should be noted that the length of the normalized probability vector of priority equals the number of new orders in the critical order-rider set.

Step 5 (generating the order priority sequence): Calculate the regret value rv˙i of each new order o_i according to Eq. (10) and obtain the regret value vector $\mathbf{rv} = [\text{rv}_1, \text{rv}_2, \ldots, \text{rv}_{n_t}]$. Combine the regret value and the normalized probability vector of priority as the final priority vector according to Eq. (11).

$$\text{rv}_i = C_{i,[1]} - C_{i,[2]} \tag{10}$$

$$\boldsymbol{P}_i = \mathbf{rv} + \boldsymbol{p}_i \tag{11}$$

where $C_{i,[1]}$ and $C_{i,[2]}$ are the dispatching cost of the best rider and second-best rider for order o_i, respectively. The new order with the largest priority is selected as the current output new order of the decoder. Meanwhile, use the masking technique to set the priority as zero in case the new order is repeatedly selected. Denote the generated new order priority sequence as $\{o_{[1]}, o_{[2]}, \ldots, o_{[n_t]}\}$.

3.3.3 Training approach

In this paper, the policy-based RL method is employed to train the seq2seq network. According to the above steps, the seq2seq network directly generates the order priority sequence without the need to update the states and calculate the cumulated reward. Thus, the evaluation of the order priority sequence, i.e., the total dispatching costs, can be utilized to replace the cumulated reward for credit assignment. However, the total dispatching costs cannot be directly used due to two reasons: the scales of different critical order-rider sets differ, so the total dispatching costs need to be normalized; besides, simply using the total dispatching costs as the weights in the loss function would lead to the training bias, so a baseline needs to be introduced to make the training more stable. Therefore, the total dispatching cost is processed as follows:

$$C(\pi|s) = \frac{\text{TDC}(\pi|s) - \text{TDC}_{\text{best}}(s)}{\text{TDC}_{\text{best}}(s)} \tag{12}$$

where TDC $(\pi|s)$ is the total dispatching cost that results from policy π on instance s, $\text{TDC}_{\text{best}}(s)$ is the best total dispatching cost on instance s, which is stored during the history training process and used as the baseline, and $C(\pi|s)$ is the weight for credit assignment. The policy gradient method is used for updating the parameters of the networks as Eq. (13) shows and the Monte Carlo sampling is adopted to approximate the expectation.

$$\nabla_\theta J(\theta) = E_{\pi \sim p_\theta(\cdot|s)}\left[C(\pi|s)\,\nabla_\theta \log p_\theta(\pi|s)\right] \tag{13}$$

Note that the total dispatching cost depends on the final dispatching results which are affected by both the order priority sequence and the greedy heuristic. Thus, the total dispatching costs cannot directly reflect the quality of the order priority sequence because the greedy heuristic also influences the total dispatching cost. Besides, different order priority sequences may lead to the same dispatching results. Therefore, it is necessary to investigate which part of the order priority sequence truly affects the final dispatching results. The dispatching cost of the corresponding part will be used to evaluate the quality of the whole order priority sequence and guide the policy network to learn. To this end, a resultant order based selective learning strategy (ROSLS) is designed. In the ROSLS, only the resultant orders will be selected for calculating the dispatching costs and then updating the policy network. For simplicity, the following definitions are given before elaborating on the process of ROSLS.

- **History best order priority sequence seq*:** the order priority sequence that yields the smallest dispatching cost during the training process. Specifically, a buffer will be reserved to store the history best order priority sequence and its total dispatching cost, i.e., $\text{TDC}_{\text{best}}(s)$. The buffer will be updated once a better order priority sequence with a smaller total dispatching cost is found.

- **Resultant order:** the order that is in different positions in the current order priority sequence seq$^\pi$ generated by policy π and the history best order priority sequence seq*. For an example with 6 new orders, assume that seq$^\pi$ and seq* are 1-3-4-2-6-5 and 1-3-6-2-4-5, respectively, and the resultant orders are 4 and 6.

The process of the ROSLS is as follows. First, calculate the resultant dispatching cost $C^r(O_j^n)$ of each order in seq$^\pi$ according to Eq. (2). Specifically, for the rider that is dispatched with only one new order, the resultant dispatching cost is calculated as $C^r(O_j^n) = C(O_j^n)$; for the rider that is dispatched with more than one new order, the dispatching cost is averaged as the resultant dispatching cost, i.e., $C^r(O_j^n) = C(O_j^n)/|O_j^n|$. Then, find the resultant orders and use the corresponding resultant dispatching cost to calculate the weights in Eq. (12). This process is realized by masking the resultant dispatching cost of the other new orders so that only the resultant dispatching cost of the resultant orders will be included in the training loss.

3.3.4 Greedy dispatching heuristic

To ensure the solution quality, the following greedy heuristic is adopted to dispatch each new order. Specifically, for each critical order-rider set, the order priority sequence is generated based on the seq2seq network. Then, dispatch each new order in order priority sequence to its best rider in turn. Each time a new order o_i is dispatched, the dispatching costs among the remaining new orders and the rider that are dispatched with o_i will be recalculated, and the best rider for each remaining new order will be updated according to current dispatching costs. Repeat the above steps until all the new orders in all the critical order-rider sets are dispatched.

4 Experiment

In this section, the performance of the proposed method is evaluated and analyzed by comparing it with other methods in terms of different metrics.

4.1 Experiment setting

We collect the history delivery data generated from 10:00 am to 1:00 pm in Longyan, Fujian province, China, as datasets. The datasets are separated into the training set and test set according to the dates. The data generated in the first couple of dates are used as the training set and the remaining are used as the test set. The length of the time windows is set as 1 min. Each instance is related to a specific time window where a number of riders and new orders need to be scheduled.

The parameters are set as follows. The weight coefficients w_t and w_d for calculating the dispatching cost are both set as 1. The sizes of the rider embeddings and order embeddings are both 100 and the size of the hidden layer of the recurrent neural networks is 50. There is only one crucial parameter K in the proposed algorithm, which is the number of the top riders for each order for the decoupling method. A larger K means that the critical order-rider set will include more new orders and riders so that the matching space is larger but the algorithm will consume more computation time. Therefore, it is not suggested to set K as an overlarge value. To properly set K, we carried out a preliminary experiment to investigate the influence of K on the scale of the critical order-rider set as shown in Figs. 6 and 7.

Specifically, the numbers of riders or new orders are recorded on each instance as K changes. The average, maximal, and minimal values among all instances are plotted in Figs. 6 and 7.

From Fig. 6, it is natural that the number of riders in a critical order-rider set becomes larger when each new order is related to more top riders. However, the number of new orders in a critical order-rider set tends to be saturated as K increases according to Fig. 7. This is because the new orders are mostly connected by the top riders with higher ranks. The riders that rank lower are usually not appropriate to be dispatched with a new order so including such riders is hard to absorb other

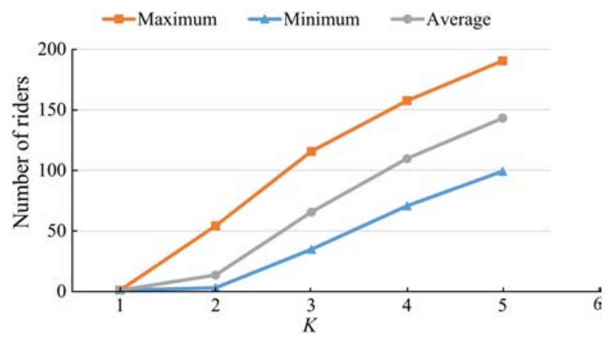


Fig. 6 Trend of the rider number in a critical order-rider set.

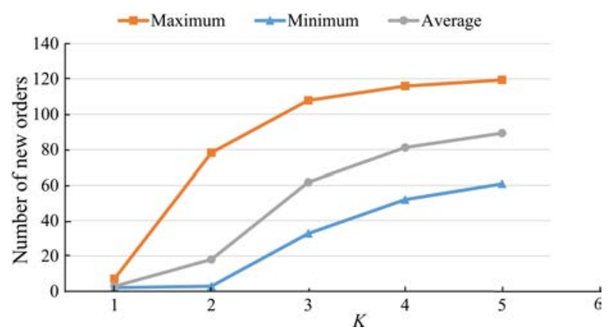


Fig. 7 Trend of new order number in a critical order-rider set.

new orders. In other words, the new orders that share the same top riders are usually clustered spatially or temporally, so there must be a threshold of K that can well decouple all the riders and new orders into different sets/clusters. According to the above results and analysis, we set a moderate value of K as 5.

To make a fair comparison, several effective dispatching heuristics are adopted to compare with the proposed method. The first heuristic is the best matching with the regret operator (BM-REG)[16], which first dispatches orders to the best riders and dispatches the remaining orders to the riders with the largest regret values. To test the effectiveness of the proposed RL-based training strategy, we set two variants of the proposed method, i.e., rdGD and gGD, which are the same as the proposed method except that they use different rules to generate the order priority sequence. Specifically, the rdGD randomly generate the order priority sequence and gGD sort the orders in ascending order of dispatching cost corresponding to their best riders.

The relative percentage deviation (RPD) in Eq. (14) is adopted as the main metric to evaluate the optimization performance of the algorithms.

$$\text{RPD} = \frac{\text{ADC}_{\text{alg}} - \text{ADC}_{\text{best}}}{\text{ADC}_{\text{best}}} \times 100 \tag{14}$$

where ADC_{alg} is the average dispatching cost, i.e., the optimization objective in Eq. (1), obtained by a certain algorithm alg, and ADC_{best} is the best average dispatching cost obtained among all algorithms as the baseline. The smaller the RPD, the better the optimization performance.

Besides, the following observation metrics are also employed to evaluate the performance of algorithms in terms of effectiveness and efficiency.

- Average increased distance (AID):

$$\text{AID} = \frac{1}{n}\sum_{j=1}^{m}(\text{DC}(R_j^n) - \text{DC}(R_j^o)) \tag{15}$$

- Delay rate (DR):

$$\text{DR} = \frac{|O_d|}{n} \times 100\% \tag{16}$$

where $\text{DC}(R_j^n)$ and $\text{DC}(R_j^o)$ are the total travel distances of the old and new routes for a rider q_j, respectively. The AID reflects the average increased travel distance for delivering a new order. A smaller value of AID indicates a higher delivery efficiency. $O_d \subseteq O$ is the set of orders that are delivered with a delay. The delay rate represents the proportion of the delayed orders in all orders. The smaller the delay rate, the better the customer satisfaction.

4.2 Experiment result and discussion

The results of RPDs, AIDs, and DRs are listed in Tables 2–4. The instances are classified into 9 groups by the number of new orders n in the second column. Accordingly, the number of riders m is recorded in the third column for each group. The interval represents the range of the number of new orders or riders for each instance in this group. For example, the instance that

Table 2 Results of RPD.

Group	n	m	RPD			
			RLDMA	BM-REG	rdGD	gGD
1	[10, 20)	[173, 296]	**0.527**	1.745	1.335	0.751
2	[20, 30)	[213, 433]	**1.548**	2.893	1.681	2.074
3	[30, 40)	[297, 503]	**0.746**	4.005	1.517	2.179
4	[40, 50)	[403, 580]	**0.917**	2.192	1.761	3.115
5	[50, 60)	[245, 572]	2.601	4.693	**1.312**	4.642
6	[60, 70)	[544, 641]	**0.497**	4.009	3.474	4.971
7	[70, 80)	[568, 661]	**0.826**	4.713	5.471	5.558
8	[80, 90)	[558, 660]	**2.089**	4.656	2.319	3.205
9	$\geqslant 90$	[604, 695]	**0.558**	5.358	3.734	5.670
Average			**1.145**	3.807	2.512	3.574

Table 3 Results of average increased distance.

Group	n	m	AID			
			RLDMA	BM-REG	rdGD	gGD
1	[10, 20)	[173, 296]	**575.485**	583.888	582.771	577.006
2	[20, 30)	[213, 433]	**498.987**	508.952	501.670	503.257
3	[30, 40)	[297, 503]	**397.209**	409.136	398.939	401.995
4	[40, 50)	[403, 580]	**320.214**	323.426	322.565	326.041
5	[50, 60)	[245, 572]	317.134	326.552	**315.029**	322.820
6	[60, 70)	[544, 641]	**232.845**	240.029	238.968	243.020
7	[70, 80)	[568, 661]	**198.721**	207.144	208.937	209.349
8	[80, 90)	[558, 660]	185.893	190.443	**185.891**	186.850
9	$\geqslant 90$	[604, 695]	**172.272**	179.489	176.069	179.930
Average			**322.084**	329.895	325.649	327.808

Table 4 Results of delay rate.

Group	n	m	DR (%)			
			RLDMA	BM-REG	rdGD	gGD
1	[10, 20)	[173, 296]	1.724	**1.695**	1.714	1.726
2	[20, 30)	[213, 433]	3.708	3.791	3.729	**3.596**
3	[30, 40)	[297, 503]	4.059	**4.001**	4.237	4.141
4	[40, 50)	[403, 580]	3.088	3.103	**2.966**	3.103
5	[50, 60)	[245, 572]	3.337	3.408	**3.070**	3.296
6	[60, 70)	[544, 641]	**2.527**	2.636	2.571	2.548
7	[70, 80)	[568, 661]	3.061	2.998	**2.962**	2.993
8	[80, 90)	[558, 660]	**1.950**	2.024	2.085	2.006
9	$\geqslant 90$	[604, 695]	2.631	**2.536**	2.583	2.595
Average			2.898	2.910	**2.880**	2.889

contains 10–19 new orders will be classified in Group 1. Then, the statistics of the number of riders among all instances in Group 1 are calculated, which corresponds to a minimum of 173 riders and a maximum of 296 riders. The value in each cell is the average value of all instances in this group.

From Table 2, it can be seen that the proposed RLDMA yields the best RPDs on nearly all groups. This phenomenon indicates that the RLDMA is able to deal with scenarios with different numbers of new orders, which validated the adaptability of the RLDMA. Note that on Group 5 the RLDMA is inferior to the rdGD. This may be because the rdGD has good exploration capability since it randomly generates the order priority sequence. However, this also indicates that rdGD lacks the exploitation ability relatively so that it is inferior to the RLDMA on most groups. Nevertheless, the RLDMA still surpasses the other two algorithms on Group 5. Besides, the average RPD of RLDMA is the smallest among all the algorithms as the last row shows. Therefore, it can be concluded that the proposed RLDMA performs the best in terms of the optimization metric.

Furthermore, as the RLDMA outperforms the rdGD and gGD, the effectiveness of the proposed seq2seq network and the training approach is validated, which means that the proposed method can well extract the relationship between orders and riders so that it can generate an order priority sequence with high quality.

To investigate the performances of the algorithms more specifically, we illustrate the boxplots on each group in Fig. 8. It can be seen that on Groups 1 to 5, the median of RLDMA is close to that of rdGD or gGD. Especially, on the first group, the median RPDs of all algorithms approximate to zero. This is because the problem scale of the instances in Group 1 is small and

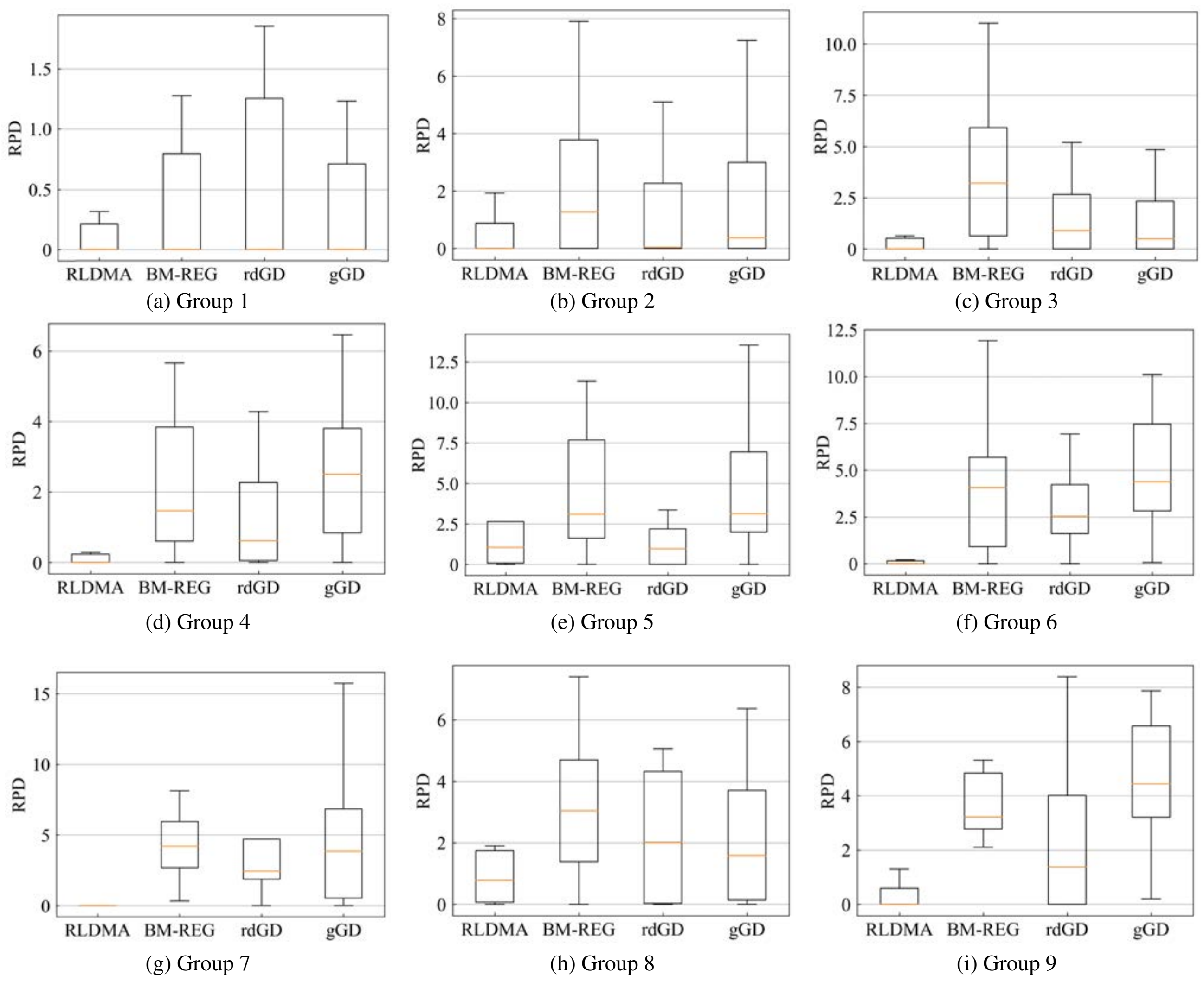


Fig. 8 Boxplot of the RPDs on each group.

it is easier for algorithms to achieve good dispatching results. However, the boxplots on Groups 6 to 9 show that as the number of new orders increases, the gap of median between the RLDMA and other algorithms becomes larger. This phenomenon indicates that the advantage of RLDMA is more obvious on larger scales of instances, which demonstrates the effectiveness of RLDMA to deal with the large-scale complexity.

Besides, it is obvious that the range of RPDs for RLDMA is smaller than those of the other algorithms on all groups. Especially, the maximal RPDs of the RLDMA are also remarkably smaller on all groups., Although the rdGD is capable of exploration and achieves good results on Group 5 with small media and deviation as mentioned before, the performance of rdGD is not stable enough since it also fluctuates intensely on Groups 1, 8, and 9. Thus, the RLDMA outperforms significantly the rdGD in terms of stability. In a word, the performance of RLDMA is more stable than those of the other algorithms, which demonstrates the robustness of the RLDMA.

Tables 3 and 4 show the results of delivery efficiency and customer satisfaction. From Table 3, it is discovered that the AIDs of RLDMA are smaller than those of the other algorithms on 7 out of 9 groups. Although on Groups 5 and 8, the RLDMA performs worse than the rdGD, it still outperforms the other two algorithms. Overall, the average AID of RLDMA is the best. Therefore, it can be concluded that the RLDMA is able to increase the delivery efficiency compared with other algorithms. Table 4 shows that the rdGD performs best on the delay rate. Despite the inferiority of RLDMA to rdGD and gGD on some groups, the delay rates of all algorithms are close either on each group or from the perspective of average results. Thus, it indicates that the RLDMA can guarantee customer satisfaction not to change overly. All in all, it can be concluded that the RLDMA can improve delivery efficiency while maintaining customer satisfaction to some degree.

5 Conclusion

In this paper, the order dispatching problem in the on-demand food delivery service is addressed. To overcome the challenges of high dynamism, large-scale complexity, and stringent time requirements, an effective matching algorithm with reinforcement learning and decoupling strategy is proposed. The problem is formulated into a series of static problems in continuous time windows to deal with the high dynamism. The large matching space can be reduced by the proposed decoupling method which only considers the riders with the best potential for each new order. The stringent requirements on computation time and solution quality can be satisfied by fusing the greedy dispatching heuristic and the seq2seq network trained by a specifically designed reinforcement learning method. Numerical experiments demonstrate the superiority of the proposed method by comparing it with other existing methods, which can improve delivery efficiency and well maintain customer satisfaction.

From this study, several conclusions can be drawn as follows. It is recommended to collaboratively combine the reinforcement learning technique and the optimization methods to solve real-world combinatorial optimization problems, especially for scenarios that limit the decision time. The optimization methods can well ensure the solution quality, while the network trained by reinforcement learning is good at fast generating a solution or enhancing the solution quality by extracting implicit useful information from the data. Besides, it is also a good attempt to tailor the reinforcement learning methods by introducing the problem characteristics or the valuable information during the optimization or training process, which benefits in improving the performance.

In future work, other effective divide-and-conquer mechanisms[28] can be introduced to deal with the large-scale complexity. Besides, other reinforcement learning approaches can be employed for better modeling of the order dispatching process, such as multi-agent reinforcement learning[29]. We will also continue to study the order dispatching problem by considering the long-term objectives to typically deal with the high dynamism. Besides, it is also worth investigating the scheduling problems with uncertainty for on-demand food delivery. Furthermore, it is interesting to develop effective hyper-heuristic or reinforcement learning methods to solve other complex combinatorial problems[30, 31].

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] Y. Huang, Y. Chai, Y. Liu, and J. Shen, Architecture of next-generation e-commerce platform, *Tsinghua Science and Technology*, vol. 24, no. 1, pp. 18–29, 2019.

[2] J. Tao, H. Dai, W. Chen, and H. Jiang, The value of personalized dispatch in O2O on-demand delivery services, *Eur. J. Oper. Res.*, vol. 304, no. 3, pp. 1022–1035, 2023.

[3] Statista, Online food delivery — worldwide, https://www.statista.com/outlook/dmo/eservices/online-food-delivery/worldwide# revenue, 2023.

[4] D. Reyes, A. Erera, M. Savelsbergh, S. Sahasrabudhe, and R. O'Neil, The meal delivery routing problem, http://www.optimization-online.org/DB_FILE/2018/04/6571. pdf, 2018.

[5] 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.

[6] M. Cosmi, G. Oriolo, V. Piccialli, and P. Ventura, Single courier single restaurant meal delivery (without routing), *Oper. Res. Lett.*, vol. 47, no. 6, pp. 537–541, 2019.

[7] M. Cosmi, G. Nicosia, and A. Pacifici, Scheduling for last-mile meal-delivery processes, *IFAC-PapersOnLine*, vol. 52, no. 13, pp. 511–516, 2019.

[8] H. Yu, X. Luo, and T. Wu, Online pickup and delivery problem with constrained capacity to minimize latency, *J. Comb. Optim.*, vol. 43, pp. 974–993, 2022.

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

[10] K. Wang, Y. Zhou, and L. Zhang, A workload-balancing order dispatch scheme for O2O food delivery with order splitting choice, *J. Theor. Appl. Electron. Commer. Res.*, vol. 17, no. 1, pp. 295–312, 2022.

[11] 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.

[12] S. Liu, L. He, and Z. J. Max Shen, On-time last-mile delivery: Order assignment with travel-time predictors, *Manag. Sci.*, vol. 67, no. 7, pp. 4095–4119, 2020.

[13] A. Kohar and S. K. Jakhar, A capacitated multi pickup online food delivery problem with time windows: A branch-and-cut algorithm, *Ann. Oper. Res.*, doi: https://doi.org/10.1007/s10479-021-04145-6.

[14] Y. Liu, An optimization-driven dynamic vehicle routing algorithm for on-demand meal delivery using drones, *Comput. Oper. Res.*, vol. 111, pp. 1–20, 2019.

[15] H. Huang, C. Hu, J. Zhu, M. Wu, and R. Malekian, Stochastic task scheduling in UAV-based intelligent on-demand meal delivery system, *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 8, pp. 13040–13054, 2022.

[16] J. F. Chen, L. Wang, S. Wang, X. Wang, and H. Ren, An effective matching algorithm with adaptive tie-breaking strategy for online food delivery problem, *Complex Intell. Syst.*, vol. 8, pp. 107–128, 2022.

[17] 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.

[18] J. Zheng, L. Wang, L. Wang, S. Wang, J. F. Chen, and X. Wang, Solving stochastic online food delivery problem via iterated greedy algorithm with decomposition-based strategy, *IEEE Trans. Syst. Man Cybern.*: *Syst.*, vol. 53, no. 2, pp. 957–969, 2023.

[19] J. F. Chen, S. Wang, L. Wang, J. Zheng, Y. Cha, J. Hao, R. He, and Z. Sun, A hybrid differential evolution algorithm for the online meal delivery problem, in *Proc. 2020 IEEE Congress on Evolutionary Computation* (*CEC*), Glasgow, UK, 2020, pp. 1–8.

[20] X. Wang, S. Wang, L. Wang, H. Zheng, J. Hao, R. He, and Z. Sun, An effective iterated greedy algorithm for online route planning problem, in *Proc. 2020 IEEE Congr. Evol. Comput.* (*CEC*), Glasgow, UK, 2020, pp. 1–8.

[21] 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.

[22] K. K. Kottakki, S. Rathee, K. Mitra Adusumilli, J. Mathew, B. Nayak, and S. Ahuja, Customer experience driven assignment logic for online food delivery, in *Proc. 2020 IEEE Int. Conf. Ind. Eng. Eng. Manag.* (*IEEM*), Singapore, 2020, pp. 827–831.

[23] 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. Ind. Eng. Eng. Manag.* (*IEEM*), Singapore, 2020, pp. 822–826.

[24] M. Joshi, A. Singh, S. Ranu, A. Bagchi, P. Karia, and P. Kala, Batching and matching for food delivery in dynamic road networks, http://arxiv.org/abs/2008.12905, 2020.

[25] 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.

[26] A. Bozanta, M. Cevik, C. Kavaklioglu, E. M. Kavuk, A. Tosun, S. B. Sonuc, A. Duranel, and A. Basar, Courier routing and assignment for food delivery service using reinforcement learning, *Comput. Ind. Eng.*, vol. 164, p. 107871, 2022.

[27] J. Hu, H. Wu, B. Zhong, and R. Xiao, Swarm intelligence-based optimisation algorithms: An overview and future research issues, *Int. J. Autom. Control*, vol. 14, nos. 5&6, pp. 656–693, 2020.

[28] W. Fang, R. Min, and Q. Wang, Large-scale global optimisation using cooperative co-evolution with self-adaptive differential grouping, *Int. J. Autom. Control*, vol. 15, no. 1, pp. 58–77, 2021.

[29] W. Fan, P. Chen, D. Shi, X. Guo, and L. Kou, Multi-agent modeling and simulation in the AI age, *Tsinghua Science and Technology*, vol. 26, no. 5, pp. 608–624, 2021.

[30] F. Zhao, S. Di, J. Cao, J. Tang, and Jonrinaldi, A novel cooperative multi-stage hyper-heuristic for combination optimization problems, *Complex System Modeling and Simulation*, vol. 1, no. 2, pp. 91–108, 2021.

[31] 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.

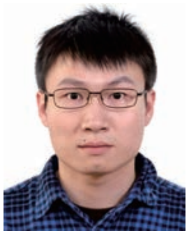
Jingfang Chen received the PhD degree in control theory and control engineering from Tsinghua University, Beijing, China in 2023. He is currently a postdoctoral researcher at Department of Automation, Tsinghua University, China. His main research interest includes intelligent optimization on complex scheduling problems.

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, 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, 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.

Zixiao Pan received the BSc degree in automation from Wuhan University of Technology, Wuhan, China in 2019. He is currently pursuing the PhD degree in control theory and control engineering at Tsinghua University, Beijing, China. His main research interests include the distributed and green scheduling with intelligent optimization and reinforcement learning.

Yuting Wu received the MSc degree from Dongbei University, Shenyang, China in 2020. She is currently pursuing the PhD degree in control theory and control engineering at Tsinghua University, Beijing, China. Her main research interests include intelligent optimization and seru production system scheduling.

Jie Zheng received the BSc degree in automation from Tsinghua University, Beijing, China in 2018. She is currently pursuing the PhD degree in control theory and control engineering at Tsinghua University, Beijing, China. Her main research interest includes the scheduling problem under uncertainty with intelligent optimization.



Xuetao Ding received the MS degree in artificial intelligence from Tsinghua University, Beijing, China in 2013. He is currently at Department of Delivery Technology, Meituan, Beijing, China. His main research direction is machine learning and decision intelligence, including causal learning for pricing, order dispatching algorithm, supply demand optimization, and recommendation.