Complex & Intelligent Systems (2022) 8:107–128
https://doi.org/10.1007/s40747-021-00340-x

ORIGINAL ARTICLE

An effective matching algorithm with adaptive tie-breaking strategy for online food delivery problem

Jing-fang Chen[1] · Ling Wang[1] · Shengyao Wang[2] · Xing Wang[1] · Hao Ren[2]

Received: 15 December 2020 / Accepted: 15 March 2021 / Published online: 8 April 2021

Abstract
With the prosperity of e-commerce, ordering food online has become increasingly prevalent nowadays. Derived from the dispatching problem in Meituan, a real online food delivery (OFD) platform in China, this paper addresses an OFD problem (OFDP). To solve the OFDP efficiently, an effective matching algorithm with adaptive tie-breaking strategy (MAATS) is proposed by collaboratively fusing the optimization methods with machine learning (ML) techniques. First, to efficiently generate a partial solution with a certain quality, a best-matching heuristic is proposed. Second, to break the ties occurring in the best-matching heuristic and obtain a complete solution with high quality, multiple tie-breaking operators are designed. Third, to adapt to different scenarios, the tie-breaking operators are utilized in a dynamic way which is achieved by using ML methods including decision trees and a specially-designed deep neural network. Fourth, problem-specific features are extracted as decision information to assist the ML models to predict the best tie-breaking operator for use in the current scenario. Preliminary offline simulations are carried out on real historical data sets to validate the effectiveness of the proposed algorithm. Moreover, rigorous online A/B tests are conducted to evaluate the performance of MAATS in practical applications. The results of offline and online tests demonstrate both the effectiveness of MAATS to solve the OFDP and the application value to improve customer satisfaction and delivery efficiency on Meituan platform.

Keywords Online food delivery · Dispatching problem · Adaptive tie-breaking · Machine learning · Problem-specific

Introduction

Thanks to the extensive applications of Internet technologies, the past decade has witnessed the explosive spreading of e-commerce. As a new business form spawned by e-commerce, the O2O (online to offline) business has emerged in various fields due to the maturity of mobile payment [16, 19]. One popular type of the O2O business is the online food delivery (OFD) service, which enables the customers to order food with only a few taps on the smartphone. Such convenience has attracted increasing number of customers and brings about a vast commercial opportunity. In the United States, the gross food sales online are expected to grow with an annual compound rate of 16% from 2017 according to Morgan Stanley Research [24]. Besides, China leads the way in market share and achieves great success [16, 17]. The number of online eaters in China is more than 450 million in 2019, increased by 12.7% since 2018 [22]. Moreover, the transaction volume of OFD market reached 642.3 billion yuan in 2019, which is nearly five times the transaction volume in 2015 [21]. In the future, it is of prospect that the OFD market will continue to grow.

There are two categories of OFD operations [37]. The first is restaurant-to-consumer delivery operation, such as McDonalds, Kentucky Fried Chicken, Domino's Pizza, Xibei, which can provide OFD services by the restaurants themselves or third-party logistics. The second is

✉ Ling Wang
wangling@tsinghua.edu.cn

Jing-fang Chen
cjf17@mails.tsinghua.edu.cn

Shengyao Wang
wangshengyao@meituan.com

Xing Wang
wang-x17@mails.tsinghua.edu.cn

Hao Ren
renhao05@meituan.com

1 Department of Automation, Tsinghua University, Beijing 100084, China

2 Meituan, Beijing 100102, China

platform-to-consumer delivery operation, such as Uber Eats, Meituan, Ele.me, Just Eat, Swiggy. The OFD platforms integrate a variety of partner restaurants and offer them delivery services. Taking Meituan, a major OFD platform in China, as an example, an entire OFD process is illustrated in Fig. 1. Customers can search for favorite restaurants, choose from full menus, make an order and pay for the food on the OFD platforms. Once receiving the food ordering requests, the OFD platform will inform the restaurants to prepare food and appoint delivery persons, aka riders, to pick up food from the restaurants and deliver it to the customers. The benefits of the OFD platforms are twofold. First, The OFD platforms decrease the burden of management and marketing [17] so that the partner restaurants can invest more time and money on the food quality. In other words, restaurants can share the professional delivery resources [18] from the OFD platforms to save the cost of employing and managing riders; furthermore, the OFD platforms attract more customers for the restaurants even if they are not situated in popular dining regions. Second, on the other side of the market, the OFD platforms improve the quality of modern life. The customers can enjoy delicious food at their own choices from a wide array of restaurants and menus. Besides, the OFD platforms bring convenience to the people who are unwilling or have no time to cook. In consequence, more and more people are becoming accustomed to ordering food online for instant or pre-booked delivery [17], which promotes the prosperity of the OFD market.

However, the explosive growth of OFD market also brings about huge challenges to OFD platforms. Firstly, the OFD platforms are faced with high dynamism from continuous order requests received over time, especially in lunch or dinner time. Secondly, the number of orders to be disposed is massive. According to the statistics from Meituan in Nov. 2019, the average number of orders received in lunch time

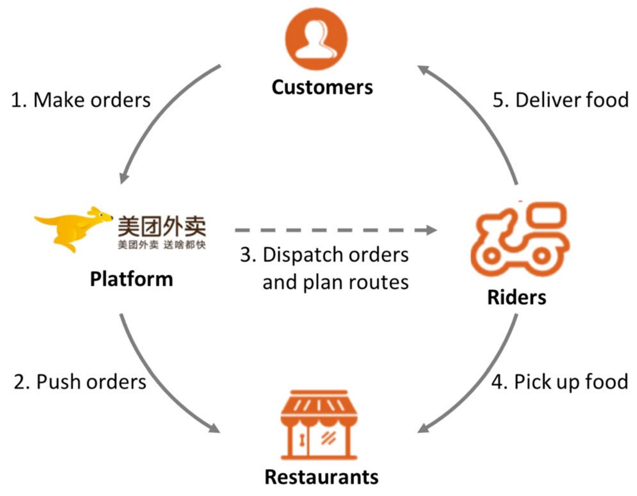


Fig. 1 Online food delivery service of Meituan platform

per day exceeds 6 million. Thirdly, the orders need to be delivered at or ahead of the estimated time of arrival (ETA) committed by the platform. Fourthly, the number of riders is limited and there will be shortages of rider resources in the on-peak meal time or bad weather. Due to the above difficulties, the optimization problem in OFD is regarded as the ultimate challenge in last-mile logistics [28]. How to increase the punctuality of delivering orders with limited resources becomes a prime issue for OFD companies. Efficient and effective optimization technologies are in demand to solve the large-scale, highly dynamic, and multi-constraint optimization problem for OFD platforms to provide high-quality services.

In this paper, we investigate the dispatching problem arising in the real food delivery scenario of Meituan platform and address an OFD problem (OFDP). To deal with high dynamism and urgency, the OFDP is transformed into a static optimization problem within a time window. A matching algorithm with an adaptive tie-breaking strategy (MAATS) is proposed to effectively solve the OFDP by determining the matching relationship between newly-arrived orders and riders. The MAATS is mainly composed of a best-matching heuristic, multiple tie-breaking operators, and a machine learning (ML) model. The best-matching heuristic dispatches the orders to the best riders to ensure the solution quality. However, the best-matching heuristic may result in ties where more than one order is matched with a single rider. To effectively break the ties, multiple tie-breaking operators are proposed based on different optimization objectives to adapt to different scenarios. An adaptive tie-breaking strategy is designed to dynamically determine the best tie-breaking operators to utilize in the current scenario, which is implemented by ML techniques of an eXtreme Gradient Boosting (XGBoost) algorithm and a modified Deep Factorization Machine (mDeepFM). Problem-specific features are extracted to further improve the prediction precision of the ML models. Offline simulation experiments on real historical data from Meituan platform are carried out to validate the effectiveness of MAATS for solving the OFDP. Furthermore, we conduct A/B tests on the real online scheduling system to evaluate the practical performance of MAATS. The results of the offline simulations and online A/B tests show that the proposed MAATS is capable of increasing the delivery efficiency and customer satisfaction.

The remainder of our paper is organized as follows. “Literature review” gives the literature review of OFDP and the problems similar to OFDP. In “Problem description”, the OFDP is described in detail. “Methodology” introduces the proposed MAATS. The experiment results of offline simulations and online AB tests are shown and analyzed in “Experiment”. Finally, “Conclusions” ends this paper with conclusions and future work.

Literature review

The OFDP is an emerging field of research. Although there are publications [4, 39] studying the optimization problems existing in food or meal delivery services, these problems do not originate from the background of OFD. In this section, we focus on the works which consider the food delivery problem arising from the background of OFD. Besides, in the literature, the OFDP is most similar to the extensively-studied dynamic pickup and delivery problem (DPDP). Therefore, the researches of DPDPs are reviewed in this section as well.

Online food delivery

Lu et al. [19] considered a simplified case where orders from the same restaurant could only be assigned to one rider and proposed two genetic algorithms to solve the order assignment and the routing problem respectively. Steever et al. [31] studied a special scenario of OFDP where a single order might include the food from multiple restaurants. The authors formulated a mixed-integer linear programming (MILP) model and designed an auction-based heuristic which incorporated future-looking measurements to determine the order assignment. Liu et al. [18] addressed two versions of the on-demand food delivery problem, i.e. the opportunistic online takeout ordering and delivery (OTOD) where the taxi driver could deliver food when carrying passengers and the dedicated OTOD where the taxi driver could only deliver food. To solve the OTOD problems, they proposed a two-stage approach combined by a constructive algorithm and an adaptive large neighborhood search algorithm with a simulated annealing mechanism. Liu [17] presented an online optimization-driven algorithm for the on-demand meal delivery problem where the food was carried by drones. Luo et al. [20] proposed a genetic annealing algorithm to schedule the routes for delivery persons and studied the relationship between the delivery lateness and the tip that customers offered. Reyes et al. [28] proposed a rolling horizon algorithm to solve the meal delivery routing problem considering dynamic vehicle routing and courier shift scheduling. Yildiz and Savelsbergh [38] presented a novel formulation for the problem addressed in [28] and designed a simultaneous column and row generation method, the efficacy of which was demonstrated by experiments on the instances derived from real-life data. Cosmi et al. [7, 8] considered a special scenario with single-courier and single-restaurant, which was modeled as a single machine scheduling problem. The authors formulated this problem into several MILP models solved by Gurobi [7] and proposed a dynamic programming algorithm [8]. Ulmer et al. [33] presented an anticipatory customer assignment policy to deal with the stochasticity from the arrival time of new orders and the ready time of restaurants. Yu et al. [40] derived the lower bounds of online pickup and delivery problem and proposed two online dispatching algorithms. Chen et al. [5] proposed a hybrid differential evolution algorithm with two phases to determine the route planning and order dispatching respectively. Zheng et al. [43] developed a two-stage algorithm to solve the fuzzy online order dispatching problem considering the uncertainty of food preparation.

Dynamic pickup and delivery

For a comprehensive review of the DPDP before 2010, interested readers may refer to [3]. In this part, we focus on the recent publications studying the DPDP. Muñoz-Carpintero et al. [25] considered the DPDP under a dial-a-ride service scenario. The dial-a-ride system was formulated with a hybrid predictive control approach and the problem was solved by a genetic algorithm and a particle swarm optimization algorithm. Zhu et al. [44] proposed a multi-objective memetic algorithm embedded with a locality-sensitive hashing-based local search to simultaneously minimize route length, response time, and workload. Arslan et al. [2] formulated the DPDP into the offline problem by using the rolling horizon method and designed an exact recursive algorithm to solve the matching problem between tasks and drivers in crowdsourced delivery. Aleksandrov et al. [1] proposed several dispatching heuristics and designed a system based on neural network model that was able to determine the best heuristic to be adopted in current decision process for an online PDP. Fkaier et al. [14] proposed a K-means algorithm to assign clustered requests and used a dynamic programming algorithm to schedule the routes. Ferrucci and Bock [13] presented a real-time control approach to simultaneously handle tour plan execution and tour plan adaptation where the latter was realized by a tabu search algorithm. Sheridan et al. [30] proposed a dynamic nearest neighbor policy to assign vehicles to customers to optimize mean system time of the customers. Vonolfen and Affenzeller [34] proposed an intensity-based waiting strategy and a parametrization approach to adapt the strategy to different problem environments.

The literatures reviewed above address different OFDPs and DPDPs and provide various solution methodologies. However, these studies do not completely accord with the problem in this paper, which originates from the Meituan platform. Also, the existing algorithms cannot be adopted directly to solve our OFDP due to different problem characteristics and extremely-limited computation time. Therefore, we address the OFDP and propose specially-designed methods in this paper.

Problem description

With notations in Table 1, the OFDP can be transformed into a static optimization problem within a time window $[T - \Delta T, T]$ by using the rolling horizon strategy as follows.

At current time T, A set of n new online orders received in $[T - \Delta T, T]$ need to be dispatched to m riders. Each rider q_j carries n_j^o old orders which have already been dispatched to q_j before T but have not been picked up or delivered yet. The assignment of each old order to riders is determined in the previous decision-making process and known as inputs at T. Each rider q_j has a site node l_j where q_j locates at T. Each order $o(i)$ is associated with a pair of nodes $\left(i_+, i_-\right)$ or a single node i_-, where i_+ is the pickup node and i_- is the delivery node. To be specific, since some of the old orders have been picked up, it is not necessary to include the pickup nodes of these orders into problem formulation. Each order has a soft deadline DDL_{i_-} that is either appointed by customers or committed by the platform. Then the OFDP can be defined on a graph $G = (V, A)$, where $V = L \cup P \cup D \cup D^p$ is the set of nodes and A is the set of arcs. L is the set consisting of the site nodes of all riders, P is comprised by the pickup nodes of all new orders and the old orders that have not been picked up, D is the delivery node set of unpicked orders and D^p is the delivery node set of picked orders. Each arc $\left(i_1, i_2\right) \in A\left(i_1, i_2 \in V, i_1 \neq i_2\right)$ is related to a travel time $t\left(i_1, i_2\right)$ and a travel distance $d\left(i_1, i_2\right)$. The constraints are listed as follows.

- The food must be picked up before being delivered.
- The food cannot be picked up before it has been prepared by the restaurant.
- A new order can only be dispatched to one rider.
- Old orders cannot be reassigned to other riders.
- The total weight of food that a rider carries cannot exceed the trunk capacity.

The objective is to minimize the average dispatching cost (ADC) by determining the matching relationship between new orders and riders:

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

where $C\left(O_j^n\right)$ is the dispatching cost after rider q_j is dispatched with a set of new orders O_j^n, which can be calculated as follows.

Consider a given route $R_j = \left[r_j(0), r_j(1), \ldots, r_j\left(N_j\right)\right]$ $(r_j(0) = l_j)$ of q_j, where $r_j(i) \in V$ is the i-th node of route R_j and the length of R_j is $N_j + 1$. The time $\mathrm{AT}_{j,i}$ when q_j arrives at node $r_j(i)$ can be calculated as (2).

Table 1 Notations

Input parameters	
T	Current time to make decisions
n	Number of new orders
m	Number of riders
Q	Set of riders. $Q = \{q_1, q_2, \ldots, q_m\}$
O^n	Set of new orders. $O^n = \{o^n(1), o^n(2), \ldots, o^n(n)\}$
O_j^o	Set of unserved old orders dispatched to q_j before T. $O_j^o = \{o_j^o(1), o_j^o(2), \ldots, o_j^o\left(n_j^o\right)\}$
O^o	Set of old orders. $O^o = \{o^o(1), o^o(2), \ldots, o^o(n^o)\} = O_1^o \cup O_2^o \cup \cdots \cup O_m^o$
O	Set of all orders. $O = \{o(1), o(2), \ldots, o(N)\} = O^n \cup O^o$. $N = n + n^o$
L	Set of site nodes. $L = \{l_1, l_2, \ldots, l_m\}$
P	Set of pickup nodes. $P = \{i_+ \mid o(i) \in O\}$
D	Set of delivery nodes of new orders. $D = \{i_- \mid o(i) \in O^n\}$
D^p	Set of delivery nodes of old orders which have been picked up.
DDL_{i_-}	Deadline of delivering order $o(i)$.
PT_{i_+}	Time when food is prepared at node i_+
$t(i_1, i_2)$	Travel time from node i_1 to node i_2
$d(i_1, i_2)$	Travel distance between node i_1 and node i_2
Decision variables	
O_j^n	Set of new orders dispatched to q_j. $O_j^n = \left\{o_j^n(1), o_j^n(2), \ldots, o_j^n\left(n_j^n\right)\right\} \in O^n$

$$\mathrm{AT}_{j,i} = T + \sum_{k=1}^{i} t\left(r_j(k-1), r_j(k)\right) + \sum_{r_j(k)\in R_j(i)\cap P} \mathrm{WT}_{j,k}, \tag{2}$$

where $R_j(i) = \left[r_j(0), r_j(1), \ldots, r_j(i)\right]$ is a partial route of R_j and $\mathrm{WT}_{j,k} = \max\left\{0, \mathrm{PT}_{r_j(k)} - \mathrm{AT}_{j,k}\right\}$ is the duration that q_j needs to wait for at pickup node $r_j(k)$ for the preparation of food. $\mathrm{PT}_{r_j(k)}$ is the time when the food is prepared at pickup node $r_j(k)$. The tardiness of delivering food to node $r_j(i)$ is defined as $\mathrm{TA}_{j,i} = \max\left\{0, \mathrm{AT}_{j,i} - \mathrm{DDL}_{r_j(i)}\right\}$ and a nonlinear penalty of the tardiness is set as (3):

$$\mathrm{PE}_{j,i} = \begin{cases} 0, & \mathrm{TA}_{j,i} \le 0 \\ \theta \mathrm{TA}_{j,i}^2, & 0 < \mathrm{TA}_{j,i} < \Theta \\ \kappa \mathrm{TA}_{j,i} + \sigma, & \mathrm{TA}_{j,i} \ge \Theta \end{cases}, \tag{3}$$

where θ, Θ, κ and σ are parameters. The time cost TC_j of route R_j is defined as the summation of the penalty in (4):

$$\mathrm{TC}_j = \sum_{r_j(k)\in R_j\cap(D\cup D^p)} \mathrm{PE}_{j,k}. \tag{4}$$

The distance cost DC_j of route R_j is defined as the total travel distance in (5):

$$\mathrm{DC}_j = \sum_{k=1}^{N_j} d\left(r_j(k-1), r_j(k)\right). \tag{5}$$

For q_j, the old route is constructed with the pickup and delivery nodes of orders in O_j^o and the new route is constructed with the pickup and delivery nodes of orders in $O_j^o \cup O_j^n$. Denote the time cost of old route and new route as TC_j^o and TC_j^n respectively, and the corresponding distance cost as DC_j^o and DC_j^n, respectively. Given that the riders are unwilling to change their routes too much after being dispatched with new orders, the difference between the new route and old route is considered for constructing the objective function, which derives the dispatching cost in (6):

$$C\left(O_j^n\right) = \left|\mathrm{TC}_j^n - \mathrm{TC}_j^o\right| + \left|\mathrm{DC}_j^n - \mathrm{DC}_j^o\right|. \tag{6}$$

Before calculating $C\left(O_j^n\right)$, the new route and old route will be scheduled by route planning methods. It should be noted that the route planning methods are basic components for solving the OFDP, which determine the dispatching costs and thus affect the final dispatching results. However, as space is limited and we mainly concentrate on the dispatching techniques in this paper, the adopted route planning methods are not elaborated and interested readers can refer to our previous work [5, 35, 42] for details.

Methodology

Due to the high dynamism, urgency and large-scale characteristics of OFDP, it is infeasible to solve this problem using exact methods. Indeed, the computation time is so limited in the online scheduling system that it is not applicable to implement a search process for finding the optimal solution online. In this situation, some efficient and effective heuristics become the first choices for OFDP. To obtain a high-quality solution in a very short time, we propose the following MAATS framework illustrated in Fig. 2, which mainly consists of a best-matching component and an adaptive tie-breaking component. In the best-matching component, a cost matrix is calculated by tentatively dispatching each new order to each rider. Then a partial solution with certain quality will be produced by efficiently matching new orders with best riders, i.e., the riders with minimum dispatching cost. As for adaptive tie-breaking, when ties occur where an order is matched with multiple best riders, the ML model will predict the best operator to break the ties based on the decision information extracted from input data and cost matrix. By using the best operator, a specific order is selected and then dispatched to the best rider. Repeat the above steps, and all orders will be dispatched efficiently and effectively.

Best-matching

In best-matching, a cost matrix $\left(C_{i,j}\right)_{n\times m}$ is first generated for decision. Each element $C_{i,j}$ represents the cost of dispatching new order $o(i)$ to rider q_j according to (6), i.e., $C_{i,j} = C\left(O_j^n\right)$ and $O_j^n = \{o^n(i)\}$. For each new order, the rider with minimum dispatching cost in the cost matrix is defined as its best rider. After finding the best rider for each order, there may

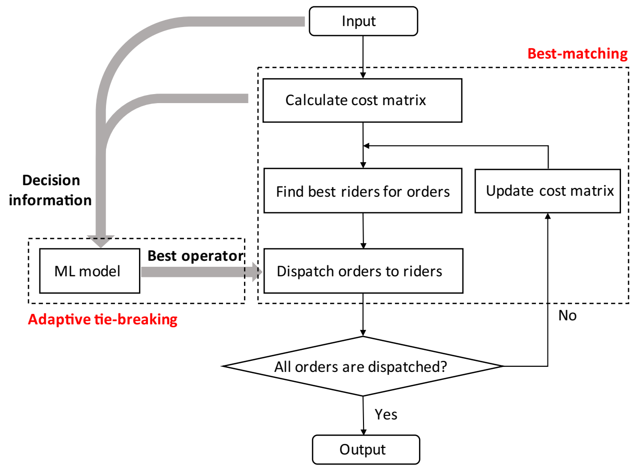
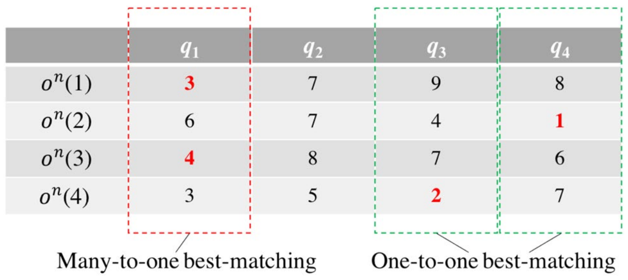


Fig. 2 Framework of MAATS



	q_1	q_2	q_3	q_4
$o^n(1)$	**3**	7	9	8
$o^n(2)$	6	7	4	**1**
$o^n(3)$	**4**	8	7	6
$o^n(4)$	3	5	**2**	7

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

Fig. 3 An example of cost matrix and best-matching

appear to be two types of results as the example in Fig. 3 shows. The first type is one-to-one best-matching where a rider is regarded as the best rider by only one order. The second type is many-to-one best-matching where a rider is regarded as the best rider by more than one order. For the sake of clarity, we define these orders as candidate orders and their best rider as critical rider in many-to-one best-matching, while the orders and riders in one-to-one best-matching are defined as non-candidate orders and non-critical riders respectively.

For one-to-one best-matching, each non-candidate order is dispatched to its corresponding best rider (non-critical rider). However, for many-to-one best-matching, only one candidate order is selected and then dispatched to the critical rider instead of dispatching all candidate orders to the critical rider owing to the following reasons:

1. It lacks of information to make decisions because the overall cost of dispatching all candidate orders to the critical rider is unknown. In the cost matrix, each element only represents the cost of dispatching one order to one rider. The overall cost cannot be obtained by simply calculating the sum of each cost due to the nonlinearity of the cost function.
2. Dispatching all candidate orders to the critical rider may not be the best choice. Obviously, it is riskier for a rider to deliver punctually when serving multiple customers simultaneously. Besides, it is possible to result in unbalance of workloads among riders when critical riders are associated with too many candidate orders. In other words, some riders may be dispatched with an excessive number of orders while others may be dispatched with none.

Apparently, in many-to-one best-matching, the unselected candidate orders are not dispatched to any rider in current loop. These orders will be dispatched to its best riders by executing the best-matching in the next loop. Before finding the best riders in the next loop, the cost matrix needs to be updated. Specifically speaking, for a rider q_j who is dispatched with a new order $o(i)$, the old route will be tentatively replaced by the current new route which is constructed with the nodes of newly-dispatched order $o(i)$ and the old order set O_j^o. Then, the costs of dispatching the remaining orders to q_j will be recalculated. Therefore, for the unselected candidate orders, their best riders may not be the same best riders in the previous loop due to the update of cost matrix. Then, the best-matching will be re-executed based on the updated cost matrix.

Tie-breaking operators

For many-to-one best-matching, how to select an appropriate candidate order has a great influence on the final dispatching results. This is because dispatching different candidate orders to the critical rider will affect the dispatching process of the remaining candidate orders. This selecting procedure is similar to the tie-breaking mechanism for solving permutation-based combinatorial optimization problems where some insertion-based algorithms need to select one from multiple partial solutions whose objective values are same. Selecting different partial solutions will influence the subsequent procedure to construct an entire solution, thus resulting in different final solutions. Therefore, how to break ties has a great influence on the performance of insertion-based algorithms [12]. In MAATS, the tie occurs whenever there is many-to-one best-matching. In addition, there may be plenty of ties occurring in on-peak hour due to numerous orders and shortage of delivery resources. Therefore, it is significant to break ties for best-matching. Based on different optimization goals, multiple tie-breaking operators are designed as follows.

- MIN operator: dispatch the candidate order with minimum dispatching cost to the critical rider.
- MINT operator: dispatch the candidate order with minimum time cost to the critical rider.
- MIND operator: dispatch the candidate order with minimum distance cost to the critical rider.
- MAX operator: dispatch the candidate order with maximum dispatching cost to the critical rider.
- REG operator: dispatch the candidate order with maximum regret value to the critical rider.

The regret value $\mathrm{rv}(o(i))$ of dispatching order $o(i)$ is defined as the cost difference between the second-best rider and the best rider, which can be calculated according to (7):

$$\mathrm{rv}(o(i)) = C_{i,[2]} - C_{i,[1]}, \tag{7}$$

where $C_{i,[1]}$ and $C_{i,[2]}$ is the minimum and second minimum dispatching cost of the i-th row in the cost matrix, respectively.

The former three are greedy operators which consider minimizing the cost from different perspectives. However, the greedy methods are easy to be trapped into local minima. To alleviate such a situation, the latter two operators are designed. The MAX operator considers dispatching the order that seems hardest to deliver (with largest dispatching cost) in case there may not be appropriate riders to deliver this order in the subsequent loops of best-matching. Note that the critical rider and the order selected by MAX operator are still best-matching even if the selected order has the largest cost among candidate orders. Therefore, the MAX operator will not deteriorate the quality of the final solution too much. The REG operator is inspired by the insertion heuristics in [29] which incorporate a kind of future-looking information. As mentioned before, the critical rider (ranking first) in current loop may not be the best rider for the remaining candidate orders in the next loop. In this situation, the second-best rider (ranking second) is most probable to be the best rider for the remaining candidate orders after the cost matrix is updated. The regret value is to measure the difference between the best rider and second-best rider, and a larger regret value indicates that it is more inappropriate to dispatch the order to the second-best rider. Therefore, the decision-maker will be regretful for dispatching the order to an inappropriate second-best rider instead of the best rider. In short, the REG operator can avoid bad cases to some degree.

Adaptive tie-breaking strategy

According to the no free lunch theorem for optimization [36], no algorithm can outperform any other algorithm on every instance. Similarly, the above tie-breaking operators exhibit different performances under different circumstances. However, if we can select the best operator at each time T, the all-day performance is possible to be improved compared with using any sole dispatching operator at each time T. To adaptively select the best operator, we propose an adaptive tie-breaking strategy based on ML techniques.

Multi-label classification

The selection of best tie-breaking operator can be transformed into a classification problem where each operator is regarded as a class label. The goal is to predict the label that the current instance (the static optimization problem at T) is associated with. However, there may be multiple tie-breaking operators that result in the same best dispatching solution. Therefore, it is a multi-label classification problem (MLCP) where each instance may be associated with a set of labels.

To deal with the multi-label classification problem [32], there are two categories of methods, i.e., problem transformation method (PTM) and algorithm adaption method (AAM). The former transforms a multi-label classification problem into one or more single-label classification problems or regression problems. The latter adapts specific learning algorithms to solve the multi-label classification problem directly. Briefly speaking, the PTMs try to fit data to the algorithm, while the AAMs try to fit the algorithm to data [41].

Feature design

Feature engineering is defined as the process to extract features from raw data by using domain knowledge, which plays an important role in data preparation for machine learning. Suitable features can improve the predictive performances of machine learning algorithms. However, useful features usually rely on the domain expertise of data scientists, iterative trial and error, and model evaluation [26], so it is not easy to extract effective features. Different from predicting some real-life parameters which may have highly-relevant factors, it is more difficult to extract highly-relevant features for predicting the best operators because the features need to serve as decision information for optimization. Since the best operator is determined and labeled by the final optimization results, the prediction model should have the ability to predict the optimization results to distinguish the best operators. Therefore, sufficient information that is possible to affect the optimization results should be provided for the model besides the basic information of the problem characteristics. To achieve this, we integrate the information from multiple dimensions into the feature space where the features are mainly categorized into problem-oriented features (information of the problem characteristics) and operator-oriented features (information affecting the optimization results).

Problem-oriented features

1) Spatial–temporal features

- City id: each city is numbered with a unique integer value.
- Day of week day: Day $\in \{1, 2, \dots, 7\}$ represents the day of week, which ranges from 1 to 7. 1 denotes Sunday, 2 denotes Monday, etc.
- Hour of day hour: Hour $\in \{0, 1, \dots, 23\}$ represents the hour of day at the current time.
- Minute of day minute: Minute $\in \{0, 1, \dots, 1439\}$ correlates with the minute of day at current time.

2) Rider number features

- The number of critical riders ncr, which equals to the number of many-to-one best-matchings in the first loop.
- The number of non-critical riders nncr, which equals to the number of one-to-one best-matchings in the first loop.
- The number of best riders: ncr + nncr.
- The proportion of critical riders accounting for best riders: ncr/(ncr + nncr).

3) Order number features

- The number of candidate orders, which equals to the sum of the candidate order number of each critical rider.
- The number of new orders, which equals to the sum of the candidate order number and non-candidate order number. (The number of non-candidate orders is not included because it equals to the number of non-critical riders)
- The proportion of candidate orders accounting for new orders.

The rider number and order number features only contain the integral information of riders and orders, which do not consider the detailed information about each critical rider, each candidate order and the correlations between them. To describe the relationships between critical riders and candidate orders, we design the following aggregate features. However, it is not reasonable to consider each characteristic of each rider or order as one feature because: (1) this will lead to feature explosion since there are thousands of riders and orders; (2) the number of features will be uncertain since the number of riders or orders is unfixed. Therefore, the characteristics of riders or orders are aggregated by calculating the statistics of the same type of features. In this paper, the mean, sum, medium, maximum, minimum values, and standard deviation are considered as specific statistics.

4) Aggregate order number features

The statistics for each critical rider are computed and the statistics of the same type are averaged by the number of critical riders, which finally yields $3 \times 6 = 18$ features in total.

- Aggregate candidate order number: statistics of $\left\{n_1^c, n_2^c, \dots, n_m^c\right\}$. n_j^c denotes the candidate orders that rider q_j matches.
- Aggregate old order number: statistics of $\left\{n_1^o, n_2^o, \dots, n_m^o\right\}$. n_j^o denotes the number of old orders that rider q_j carries.
- Aggregate candidate-old order ratio: statistics of $\left\{\text{rco}_1, \text{rco}_2, \dots, \text{rco}_m\right\}$. $\text{rco}_j = n_j^c / \left(n_j^o + S\right)$ denotes the ratio of the candidate order number n_j^c divided by old order number of q_j, where S is a small decimal.

5) Aggregate geography-ETA features

The geographic and ETA relationships between orders are also extracted as features. Indeed, one of the reasons why multiple orders match a same best rider is that there exist similarities between candidate orders either in geographical distribution or ETA. Apparently, it is reasonable to dispatch a rider with two or more orders simultaneously with close restaurant locations, customer locations and ETAs. We denote this situation as bundle-dispatching. The operator that can lead to more bundle-dispatching results is more possible to outperform other operators. Therefore, the geographic and temporal relationships between orders are valuable information for predicting the best operators. The spatial–temporal order features can be mainly classified into three types as follows.

- ETA features, which consider the similarities on ETAs of different candidate orders from a critical rider.
- Geography features, which consider the similarities on the geographical distribution of different candidate orders from a critical rider.
- Geography-ETA cross features, which consider the similarities on the aggregate relationship between geographical distribution and ETAs of different candidate orders from a critical rider.

6) Aggregate cost features

The cost matrix reveals the relationship between critical riders and candidate orders, which also serves as an indispensable information source for classification. The cost features mainly contain the statistics of route cost, time cost and distance cost in cost matrix.

Due to limited space, the specific design of aggregate geography-ETA features and aggregate cost features is attached in the Appendix.

Operator-oriented features The tie-breaking operators differ in selecting different candidate orders. Hence, incorporating the selection information is helpful for classification. For each operator $X \in$ {MIN, MAX, MINT, MIND, REG}, the following features are calculated, which yield $5 \times 7 = 35$ features in total.

- The average regret value of unselected orders: $\frac{1}{|R^c|} \sum_{j \in R^c} \sum_{i \in O_j^c, i \neq o_j^X} \frac{\text{rv}(i)}{\left|O_j^c\right|}$, where R^c is the set of critical riders,

O_j^c is the candidate order set of q_j, o_j^X is the candidate order dispatched to rider q_j by operator X and $|\cdot|$ represents the cardinality of a set.

- The average regret value of selected orders: $\frac{1}{|R^c|}\sum_{j\in R^c} \mathrm{rv}\left(o_j^X\right)$.
- The average total regret cost of selection: $\frac{1}{|R^c|}\sum_{j\in R^c}\left(C_{o_j^X,[1]} + \sum_{i\in O_j^c, i\neq o_j^X} C_{i,[2]}\right)$.
- The average best cost of the selected orders: $\frac{1}{|R^c|}\sum_{j\in R^c} C_{o_j^X,[1]}$.
- The average second-best cost of the unselected orders: $\frac{1}{|R^c|}\sum_{j\in R^c}\sum_{i\in O_j^c, i\neq o_j^X} C_{i,[2]}$.
- The average candidate rider number of unselected orders: $\frac{1}{|R^c|}\sum_{j\in R^c}\sum_{i\in O_j^c, i\neq o_j^X} m_i^c$, where m_i^c is the number of riders that order $o(i)$ can be dispatched to.
- The average candidate rider number of selected orders: $\frac{1}{|R^c|}\sum_{j\in R^c} m_{o_j^X}^c$.

Classification model

Since there is no prior knowledge about which method solves our problem better, both PTM and AAM are attempted in this study. For PTM, we transform the MLCP into five independent binary classification problems using binary relevance [41]. Each classifier corresponds to a specific dispatching operator and predicts whether this operator yields the smallest dispatching cost among all operators, which is realized by XGBoost algorithm. For AAM, we extend the DeepFM [15] as a new model named as mDeepFM to directly predict the best operators. Both methods belong to supervised learning which learns from the training data and produces a model to predict the label of new instances.

XGBoost XGBoost, first proposed by Chen [6] in 2016, is a powerful and popular tool in data mining fields. As a member of tree algorithms, XGBoost aims at improving the traditional gradient boosting decision tree in both model performance and computational speed. XGBoost reduces the look-up times of creating individual trees and supports parallel computing [11, 27], which makes it much faster than many other existing algorithms. Besides, due to its portability, scalability and flexibility, XGBoost can be operated on various platforms and provide multiple interfaces for users to define necessary parameters. Therefore, XGBoost achieves great success in many areas, especially the machine learning competitions, such as Kaggle where 17 out of 29 winning solutions uses XGBoost in 2015 and KDDCup where top 10 winning solutions all used XGBoost in 2015 [10]. Owing to the above merits, we choose XGBoost as a classification model to solve the MLCP.

XGBoost is a boosting algorithm which is a kind of ensemble learning techniques. The boosting algorithm generates multiple base models sequentially that compensate the deficiencies of each other to achieve better prediction performance [9, 23]. For XGBoost, the base model is decision tree. A decision tree starts from a single root node. New nodes are generated from the root node by selecting appropriate features and splitting the training data into two sets. Each new node, aka internal node, is associated with a set of training data and then can be used to generate the next new nodes. Repeat generating new nodes to split the training data precisely, and a decision tree will be produced. The end node of each branch is denoted as leaf node and related to a label. The split training data of each leaf node is classified as the class that belongs to the corresponding label. An example is given in Fig. 4 to illustrate the classification process of a tree ensemble model.

The specific details of XGBoost is introduced as follows. Given a data set $S = \left\{\left(\boldsymbol{x}_l, Y_l\right) | \boldsymbol{x}_l \in R^m, Y_l \in R\right\} (|S| = n)$ where $\boldsymbol{x}_l \in R^m$ is the feature vector and Y_l is the corresponding label ($Y_l = 1$ if the corresponding operator is the best; otherwise, $Y_l = 0$). Considering an ensemble model with K decision trees, the prediction $\hat{Y}_l$ for an instance ($\boldsymbol{x}_l$,Y_l) can be calculated by summing up the leaf score $f_k\left(\boldsymbol{x}_l\right)$ of each tree as (8):

$$\hat{Y}_l = \sum_{k=1}^{K} f_k\left(\boldsymbol{x}_l\right), \tag{8}$$

where f_k is a function that maps $\boldsymbol{x}_l$ to a specific leaf score $w_{q(\boldsymbol{x}_l)}$, i.e., $f_k\left(\boldsymbol{x}_l\right) = w_{q(\boldsymbol{x}_l)}$, and $q\left(\boldsymbol{x}_l\right)$ maps an instance $\boldsymbol{x}_l$ to a leaf index, i.e., $q\left(\boldsymbol{x}_l\right) : R^m \to \{1, 2, \ldots, T\}$. The mapping relationship is determined by the tree structure.

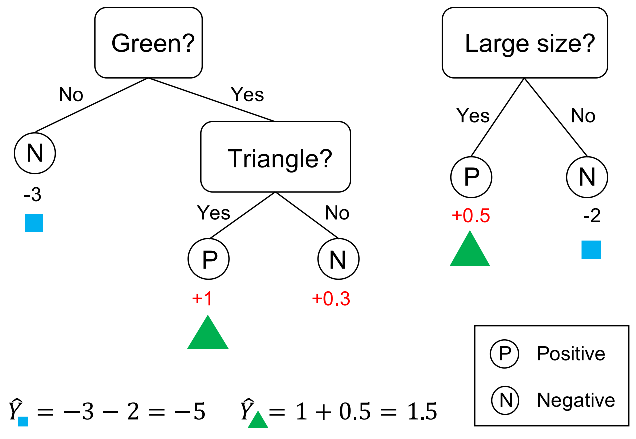


Fig. 4 Example of tree ensemble model

Based on the predictions calculated above, the objective function of XGBoost is defined in (9) which includes a loss function term and a regularization term. The former measures the deviation between predictions and true labels for the data set while the latter serves as a penalty on the model complexity:

$$O = \sum_{l=1}^{n} L\left(Y_l, \hat{Y}_l\right) + \sum_{k=1}^{K} \Omega\left(f_k\right). \tag{9}$$

As a typical boosting algorithm, XGBoost adds a tree in each step. Without loss of generality, when generating the t-th tree, the objective function can be rewritten as (10) by using the Taylor expansion of the above function to the second order:

$$\begin{aligned} O_t &= \sum_{l=1}^{n} L\left(Y_l, \hat{Y}_l^{(t-1)} + f_t(\boldsymbol{x}_l)\right) + \sum_{k=1}^{t} \Omega\left(f_k\right) \\ &\cong \sum_{l=1}^{n} \left[L\left(Y_l, \hat{Y}_l^{(t-1)}\right) + g_l f_t(\boldsymbol{x}_l) + \frac{1}{2} h_l f_t^2(\boldsymbol{x}_l)\right] + \Omega\left(f_t\right) + \sum_{k=1}^{t-1} \Omega\left(f_k\right), \\ &\Leftrightarrow \tilde{O}_t = \sum_{l=1}^{n} \left[g_l f_t(\boldsymbol{x}_l) + \frac{1}{2} h_l f_t^2(\boldsymbol{x}_l)\right] + \Omega\left(f_t\right) \end{aligned} \tag{10}$$

where $g_l = \frac{\partial L\left(Y_l, \hat{Y}_l^{(t-1)}\right)}{\partial \hat{Y}_l^{(t-1)}}$ and $h_l = \frac{\partial^2 L\left(Y_l, \hat{Y}_l^{(t-1)}\right)}{\partial \left(\hat{Y}_l^{(t-1)}\right)^2}$ are first and second-order gradient statistics on the loss function. It should be noted that the 1st to $(t-1)$-th terms are removed from the objective function since these terms are determined in the previous steps and can be regarded as constants. By introducing the specific calculation of penalty term $\Omega\left(f_k\right) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} \omega_j^2$ and splitting the data sets into subsets $I_j = \{l \mid q(\boldsymbol{x}_l) = j\}$ that are associated with leaf j, the final form of the objective function can be obtained as follows:

$$\begin{aligned} \tilde{O}_t &= \sum_{l=1}^{n} \left[g_l f_t(\boldsymbol{x}_l) + \frac{1}{2} h_l f_t^2(\boldsymbol{x}_l)\right] + \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} \omega_j^2 \\ &= \sum_{j=1}^{T} \left[\omega_j \sum_{l \in I_j} g_l + \frac{1}{2}\omega_j^2 \left(\sum_{l \in I_j} h_l + \lambda\right)\right] + \gamma T. \end{aligned} \tag{11}$$

Given that minimizing the objective function is a quadratic problem, the optimal leaf score of t-th tree and optimal objective can be easily obtained as follows:

$$\omega_j^* = -\frac{\sum_{l \in I_j} g_l}{\sum_{l \in I_j} h_l + \lambda}, \tag{12}$$

$$\tilde{O}_t^* = -\frac{1}{2} \sum_{j=1}^{T} \frac{\left(\sum_{l \in I_j} g_l\right)^2}{\sum_{l \in I_j} h_l + \lambda} + \lambda T. \tag{13}$$

The optimal values are calculated based on a specific tree structure. However, it is impracticable to traverse all possible structures. Usually, we start from the root node and add branches iteratively by using greedy methods which evaluate the quality of split according to the gain of objective function after split as (14):

$$\Delta O_{\text{split}} = \frac{1}{2}\left[\frac{\left(\sum_{l \in I_{\text{L}}} g_l\right)^2}{\sum_{l \in I_{\text{L}}} h_l + \lambda} + \frac{\left(\sum_{l \in I_{\text{R}}} g_l\right)^2}{\sum_{l \in I_{\text{R}}} h_l + \lambda} - \frac{\left(\sum_{l \in I} g_l\right)^2}{\sum_{l \in I} h_l + \lambda}\right] - \lambda, \tag{14}$$

where I_{L} and I_{R} denote the data sets of left and right nodes after split. The larger the gain, the better the split. However, it is time-consuming to enumerate all possible split. Therefore, XGBoost uses an approximate algorithm [6] for split finding. With the approximate algorithm, the trees can be fast generated and utilized for predictn.

mDeepFM The DeepFM is first proposed to predict the click-through rate (CTR), which exhibits good performance in the recommendation system [15]. The typical characteristic of the DeepFM is to model low-order and high-order interactions of features by integrating factorization machine (FM) and deep neural network (DNN). Compared with its counterpart Wide and Deep model which uses a linear model to learn the low-order interactions of features by considering both raw features and cross-product features, the DeepFM employs the FM to learn from raw features directly and thus relies less on feature engineering. Besides, DeepFM makes full use of latent vectors to realize end-to-end learning and thus does not need to pre-training the FM layer. Therefore, we choose the DeepFM model for solving the MLCP.

Given that the prediction of CTR is a binary classification problem, the basic DeepFM cannot deal with multi-label classification directly. Therefore, we extend the DeepFM and propose a new model mDeepFM. By adding a transformation layer, the mDeepFM has two merits: (1) solving the MLCP directly by outputting a vector where each element denotes the probability of being the best operator for a certain operator; (2) making more precise predictions by further learning the correlations between low-order and high-order features. The structure of mDeepFM is illustrated in Fig. 5.

Different from the XGBoost, each instance for the mDeepFM is preprocessed by converting the category features into one-hot coding sequences and integrating the label of each operator into a label vector. Consider an instance $(\boldsymbol{x}_l, Y_l) \in S$ where $\boldsymbol{x}_l = \left(x_{\text{field}_1}, \ldots, x_{\text{field}_c}, x_{\text{field}_{c+1}}, \ldots, x_{\text{field}_a}\right)$ is an a-dimension vector. $x_{\text{field}_1}, \ldots, x_{\text{field}_c}$ represent the category features and $x_{\text{field}_{c+1}}, \ldots, x_{\text{field}_a}$ represent the continuous features. Then each category feature is extended by one-hot coding, i.e.,

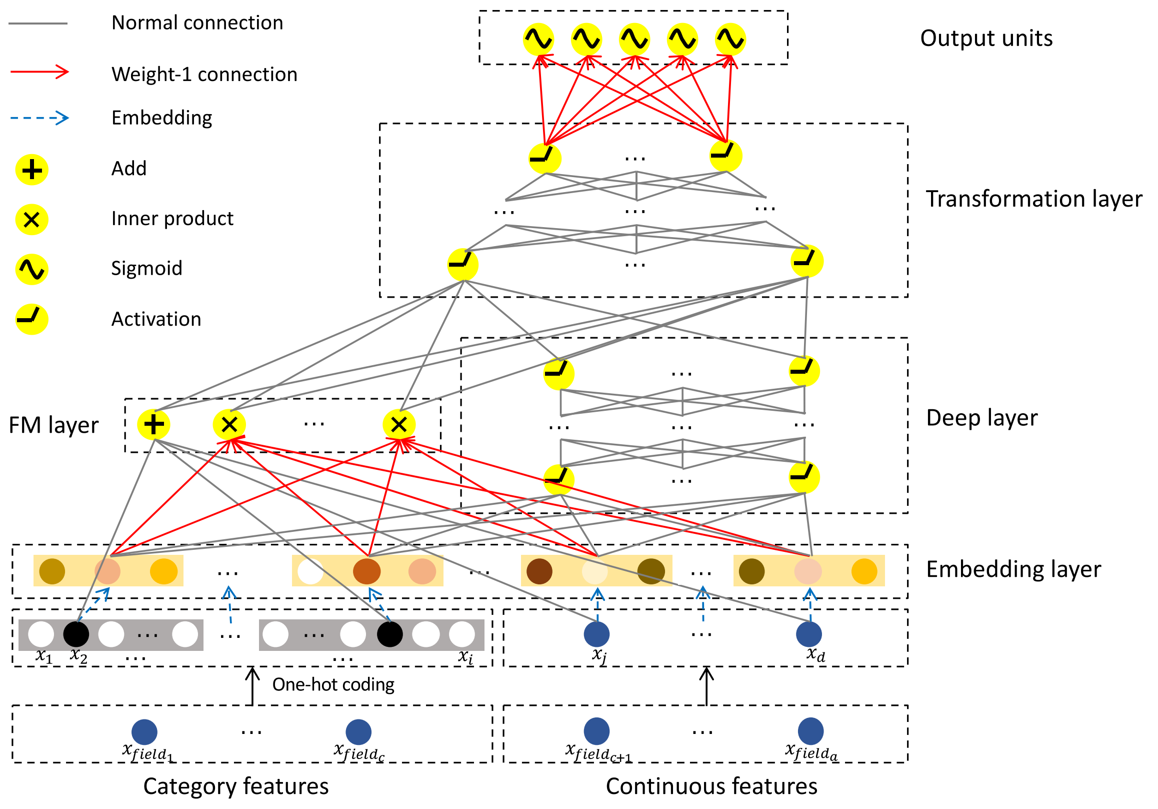


Fig. 5 Structure of mDeepFM

$x_{\text{field}_i} \Rightarrow \boldsymbol{x}'_{\text{field}_i} = (0, \ldots, 0, 1, 0, \ldots, 0), \quad i \in \{1, 2, \ldots, c\}$. By concatenating the one-hot coding category features and continuous features, the feature vector $\boldsymbol{x}_l$ can be reconstructed as a d-dimension vector $\boldsymbol{x}'_l = (x_1, \ldots, x_d) \in R^d$. In what follows, the details of each component are introduced.

1) FM component

The FM is used for modeling the interactions of single and pairwise features. Given an instance $(\boldsymbol{x}'_l, Y_l)$, the output of FM component in DeepFM can be calculated as follows:

$$\hat{Y}_{\text{FM}} = \sum_{i=1}^{d} w_i x_i + \sum_{i=1}^{d} \sum_{j=i+1}^{d} \langle \boldsymbol{v}_i, \boldsymbol{v}_j \rangle x_i x_j, \tag{15}$$

where the first-order weights $w_i \in R$ and latent vectors $\boldsymbol{v}_i = [v_{i,1}, v_{i,2}, \ldots, v_{i,e}] \in R^e$ $(\forall i \in \{1, 2, \ldots, d\})$ are parameters, e is a hyperparameter denoting the dimensionality of the factorization and $\langle \boldsymbol{v}_i, \boldsymbol{v}_j \rangle = \sum_{f=1}^{e} v_{i,f} \cdot v_{j,f}$ is the inner product of vector $\boldsymbol{v}_i$ and $\boldsymbol{v}_j$. Instead of a single weight $w_{i,j} \in R$, the FM employs the inner product of two latent vectors as the weight of a second-order term. This special interactive way of pairwise features enables FM to train latent vector $\boldsymbol{v}_i$ and $\boldsymbol{v}_j$ whenever x_i or x_j equals to zero, while using a single weight $w_{i,j}$ will result in no interaction $(w_{i,j} = 0)$. By introducing the specific calculation of $\langle \boldsymbol{v}_i, \boldsymbol{v}_j \rangle$, The prediction $\hat{Y}_{FM}$ can be rewritten as follows:

$$\hat{Y}_{\text{FM}} = \sum_{i=1}^{d} w_i x_i + \frac{1}{2} \sum_{f=1}^{e} \left(\left(\sum_{i=1}^{d} v_{i,f} x_i \right)^2 - \sum_{i=1}^{d} v_{i,f}^2 x_i^2 \right). \tag{16}$$

To solve the MLCP directly, the outputs of FM are fed into a transformation layer in mDeepFM. However, feeding all weighted first-order and second-order terms will introduce many parameters between the FM layer and transformation layer. Besides, feeding the summation $\hat{Y}_{\text{FM}}$ will lose useful information to some degree. Therefore, the weighted first-order and second-order terms are transformed into a vector $\boldsymbol{o}_{\text{FM}}$ with a length of $a + e$ before being fed into the next layer:

$$\boldsymbol{o}_{\text{FM}} = [\boldsymbol{o}_1, \boldsymbol{o}_2],$$

where $\boldsymbol{o}_1 = \left[o_1^{(1)}, o_1^{(2)}, \ldots, o_1^{(a)}\right]$ is the first-order vector and $\boldsymbol{o}_2 = \left[o_2^{(1)}, o_2^{(2)}, \ldots, o_2^{(e)}\right]$ is the second-order vector. The elements of $\boldsymbol{o}_1$ and $\boldsymbol{o}_2$ can be calculated as (17) and (18) respectively:

$$o_1^{(i)} = \begin{cases} w_j, & x_j \in \boldsymbol{x}'_{\text{field}_i}, x_j \neq 0 \forall i = 1, 2, \ldots, c \\ w_j x_j, & x_j = x_{\text{field}_i} \forall i = c+1, \ldots, a \end{cases}, \tag{17}$$

$$o_2^{(f)} = \frac{1}{2}\left(\left(\sum_{i=1}^{d} v_{i,f} x_i \right)^2 - \sum_{i=1}^{d} v_{i,f}^2 x_i^2 \right), \quad \forall f \in \{1, 2, \ldots, e\}. \tag{18}$$

To be specific, $\boldsymbol{o}_1$ is an a-dimension vector and each element is related to each field of features. For category features, $o_1^{(i)}$ equals to the weight which is related to the dimension with none-zero value under one-hot encoding; for continuous features, $o_1^{(i)}$ equals to the corresponding first-order term. $\boldsymbol{o}_2$ is an e-dimension vector inspired by (16), each element of which is contributed by all features and one dimension of each latent vector.

2) Embedding

To reduce the dimension of sparse data, the embedding layer compresses the one-hot vectors into dense vectors; to balance the contribution of category and continuous features, the embedding layer maps each continuous feature to a vector. In mDeepFM, the weight vectors of FM are employed for producing the embeddings as the DeepFM does. The computation of embedding for each field of features is as follows:

$$\boldsymbol{e}_i = \begin{cases} \boldsymbol{v}_j, & x_j \in \boldsymbol{x}'_{\text{field}_i}, x_j \neq 0 \forall i = 1, 2, \ldots, c \\ x_j \boldsymbol{v}_j, & x_j = x_{\text{field}_i} \forall i = c+1, \ldots, a \end{cases}. \tag{19}$$

3) Deep component

The deep component is constructed by a fully-connected feed-forward network to model high-order feature interactions. The input of deep layer is the output of the embedding layer $\boldsymbol{a}_{\text{D}}(0) = [\boldsymbol{e}_1, \boldsymbol{e}_2, \ldots, \boldsymbol{e}_a]$. The output of each hidden layer can be calculated as $\boldsymbol{a}_{\text{D}}(l+1) = \sigma(\boldsymbol{W}_{\text{D}}(l)\boldsymbol{a}_{\text{D}}(l) + \boldsymbol{b}_{\text{D}}(l))$, $0 \leq l < H_{\text{D}}$, where $\boldsymbol{W}_{\text{D}}(l)$ and $\boldsymbol{b}_{\text{D}}(l)$ are the weight matrix and bias vector of the l-th layer respectively and H_{D} is a parameter representing the number of hidden layers in deep component. The output of the deep component is as follows:

$$\boldsymbol{o}_{\text{DNN}} = \boldsymbol{a}_{\text{D}}(H_{\text{D}}). \tag{20}$$

4) Transformation component

The transformation component is also a fully-connected feed-forward network to further learn the correlations between low-order and high-order features, which concatenates the outputs of FM $\boldsymbol{o}_{\text{FM}}$ and deep layer $\boldsymbol{o}_{\text{DNN}}$ as input $\boldsymbol{a}_{\text{T}}(0) = [\boldsymbol{o}_{\text{FM}}, \boldsymbol{o}_{\text{DNN}}]$. The transformation layer is composed of multiple hidden layers and the output of each layer is $\boldsymbol{a}_{\text{T}}(l+1) = \sigma(\boldsymbol{W}_{\text{T}}(l)\boldsymbol{a}_{\text{T}}(l) + \boldsymbol{b}_{\text{T}}(l))$, $0 \leq l < H_{\text{T}}$, where H_{T} is the number of hidden layers in the transformation component. The output of a hidden layer is fed into next layer with less neurons to compress the outputs gradually into a vector with the length of label number. The output of the transformation layer $\boldsymbol{a}_{\text{T}}(H_{\text{T}})$ is finally fed into a sigmoid function to obtain the predictions as (21):

$$\hat{\boldsymbol{Y}}_{\text{mDeepFM}} = \text{sigmoid}(\boldsymbol{a}_{\text{T}}(H_{\text{T}})). \tag{21}$$

In accord with the XGBoost, the operator with the largest probability is adopted as the final operator to break ties at current time T.

Experiment

In this section, offline and online experiments are conducted to evaluate the performance of the proposed algorithm. As mentioned before, the best operator is selected and executed at each time T so that the all-day performance of the dispatching system will be better than using a single operator. However, since the predictions of the classifiers cannot be accurate all the time, the all-day performance will be deteriorated when making wrong predictions compared with ideal classifiers. Therefore, it is of significance to investigate whether the MAATS outperforms the best-matching using a single tie-breaking operator, which is the principal intention of the offline experiments. However, the offline experiments only serve as preliminary simulations to demonstrate the effectiveness of MAATS for solving the OFDP, which cannot reflect the performance of the proposed algorithm when applied to practice due to the changing and uncertain real situation. Therefore, rigorous online A/B tests are conducted on the Meituan dispatching system to investigate whether the proposed algorithm is of application value.

Offline preliminary simulation

Settings of offline experiment

1) Datasets and parameters

The historical data on Meituan platform are used to generate training set and test set. The data are selected from the whole month of July 2020 in 21 cities based on the daily number of orders (denoted as *DNO*), including 4 large cities (DNO ≥ 250,000), 12 medium cities (100,000 ≤ DNO < 250,000) and 5 small cities (DNO < 100,000), which yield 2,580,805 instances in total. Each instance is related to a static optimization problem at a specific time T. The many-to-one best-matching instances are adopted as the training and test sets (Table 2). Each many-to-one best-matching instance is

Table 2 Information of data sets

	One-to-one instances	Many-to-one instances		Total
		Training set	Test set	
Number	870,765	1,318,222	391,818	2,580,805
Proportion	33.74%	51.08%	15.18%	100.00%

run 5 times for each operator independently and the operators that yield minimum *ADC* are defined as labels.

The main parameters are set as follows. For calculating the penalty of tardiness, θ, Θ, κ and σ are empirically set as 0.06, 20, 8 and 136, respectively. For XGBoost, max_depth (to control tree depth), min_child_weight (to control splitting) and learning_rate (to control the step shrinkage) are set as 5, 4 and 0.1, respectively. For mDeepFM, the relu function is taken as activation function, the embedding length is set as $e=10$, the structure of deep layer is 512-512-512, the structure of the transformation layer is 256-128-64-32-5 and the four spatial–temporal features are defined as the category features while the remaining are continuous features.

2) Evaluation metrics
- Metric of classification

The area under ROC (receiver operating characteristic) curve (AUC) is taken as the metric to evaluate the prediction accuracy of models, which is a widely used metric in the field of binary classification. The value of AUC is between 0 and 1. The larger the AUC is, the more accurately the model predicts.

- Metric of optimization and observation

To test the effectiveness of the adaptive tie-breaking strategy, the best-matching heuristic combined with each single tie-breaking operator and random selection of tie-breaking operators is considered as the comparative algorithm, which yields BM-MIN, BM-MAX, BM-MINT, BM-MIND, BM-REG and BM-RAND. Two versions of MAATS with different ML models are denoted as MAATS-XGB and MAATS-mDeepFM. To evaluate the optimization performance for solving OFDP, the objective *ADC* in (1) is taken as the optimization metric. Since it is hard to infer the delivery efficiency from *ADC*, we define the following two observation metrics to evaluate the delivery efficiency from the perspective of distance and time:

$$\mathrm{AID} = \frac{1}{n}\sum_{j=1}^{m}\left(\mathrm{DC}_j^n - \mathrm{DC}_j^o\right), \tag{22}$$

$$\mathrm{ACT} = \frac{1}{N}\sum_{j=1}^{m}\left(\mathrm{LLT}_j - \mathrm{FAT}_j\right), \tag{23}$$

where DC_j^n and DC_j^o is the distance cost of new route and old route, respectively for rider q_j, LLT_j is the time when q_j leaves the last node, FAT_j is the time when q_j arrives at the first node, n is the number of new orders and N is the number of all orders. AID represents the average increased distance on the route when dispatching a rider with a new order. ACT represents the average consumed time for a rider to serve an order.

The relative percentage deviation RPD is taken to evaluate the performance of an algorithm from different views of metrics:

$$\mathrm{RPD}_{\mathrm{Metric}}(\mathrm{alg}) = \frac{\mathrm{Metric}_{\mathrm{alg}} - \mathrm{Metric}_{\mathrm{bst}}}{\mathrm{Metric}_{\mathrm{bst}}} \times 100, \tag{24}$$

where $\mathrm{Metric} \in \{\mathrm{ADC}, \mathrm{AID}, \mathrm{ACT}\}$, $\mathrm{Metric}_{\mathrm{alg}}$ is the Metric of a certain algorithm and $\mathrm{Metric}_{\mathrm{bst}}$ is the best Metric obtained among all algorithms. A smaller *RPD* indicates a better performance.

Results of the offline experiment

1) Distribution of the data set

Before testing the performance of the proposed algorithm, we investigate the distribution of the data sets to help the analysis of the following experimental results. The data sets are grouped according to the number of critical riders and candidate orders. The number of instances for each group is counted and the proportion of instances that each group accounts for is illustrated in Figs. 6 and 7. From Fig. 6, it is clear that most many-to-one best-matchings occur with less than 100 critical riders. Besides, the instances with 1–10 critical riders account for nearly 60% of all instances. From Fig. 7, it can be seen that most many-to-one best-matchings occur with less than 100 candidate orders. Besides, the instances with less than 10 candidate orders account for nearly 40% of all instances and the instances with 11–100 candidate orders account for nearly 50% of all instances.

Figure 8 illustrates the distribution of different tie-breaking operators under different number of critical riders and candidate orders. For each operator, the instances are classified into positive ones and negative ones. If the operator performs best on an instance, then this instance is marked as a positive instance for this operator, otherwise a negative instance. A larger proportion of positive instances means that the corresponding operator achieves the best operator more frequently. Therefore, the performances of different operators can be ranked as: REG > MAX > MINT > MIN ≈ MIND, which reveals that the REG operator outperforms the other operators in general.

2) Importance of features

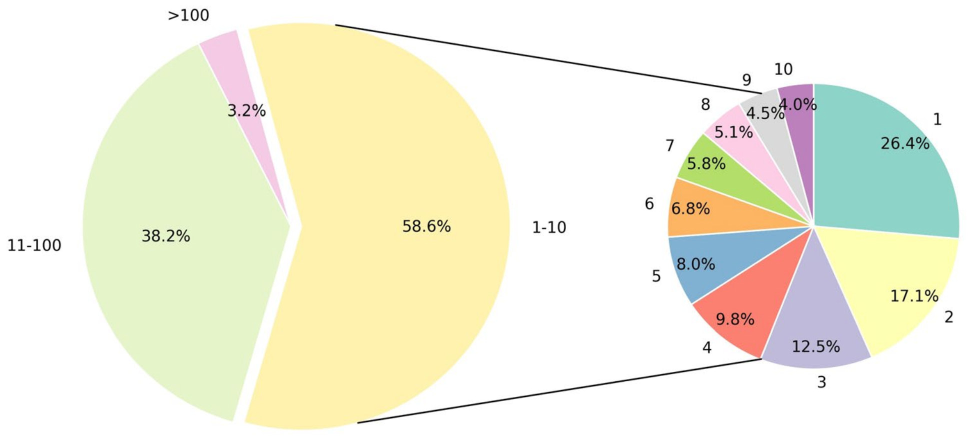


Fig. 6 Data distribution by the number of critical riders

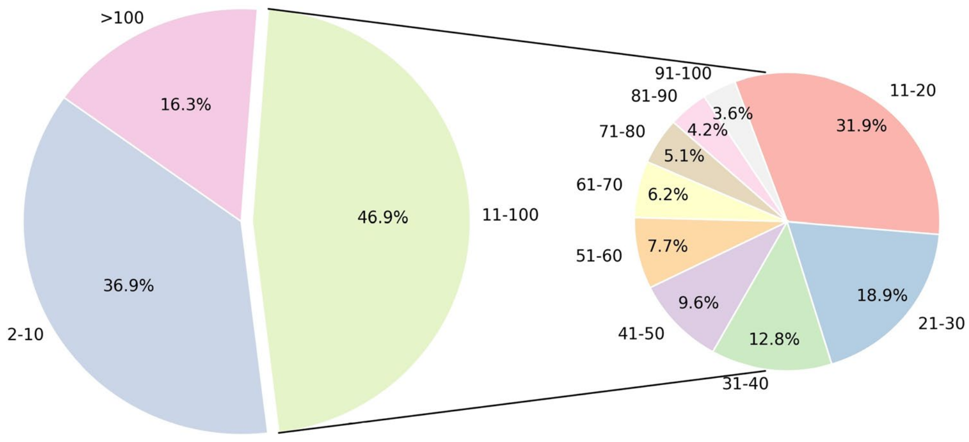


Fig. 7 Data distribution by the number of candidate orders

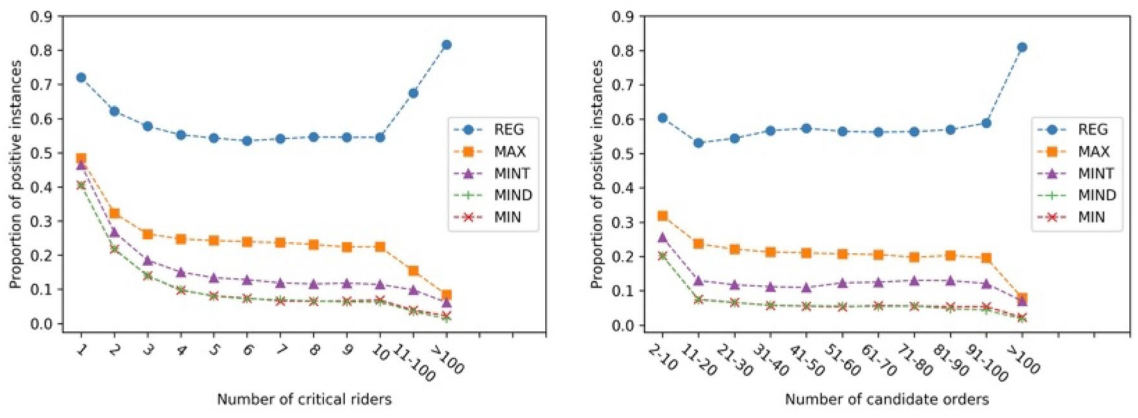


Fig. 8 Distribution of different operators

We analyze the contribution of each type of features based on the gain in (14), which is denoted as "Score". The larger the score, the more important the feature. The scores of the top 50 features are illustrated in Fig. 9 in descending order. It can be seen that the spatial–temporal feature (City id) ranks first, which indicates that different regions require different operators. All rider number features and order number features rank in the top 50, which reveals the necessity of incorporating the basic problem characteristic information. Besides, the operator-oriented features exhibit remarkable importance because these features account for nearly half of the top 50 features. Moreover, quite a number of aggregate features enter in the top 50.

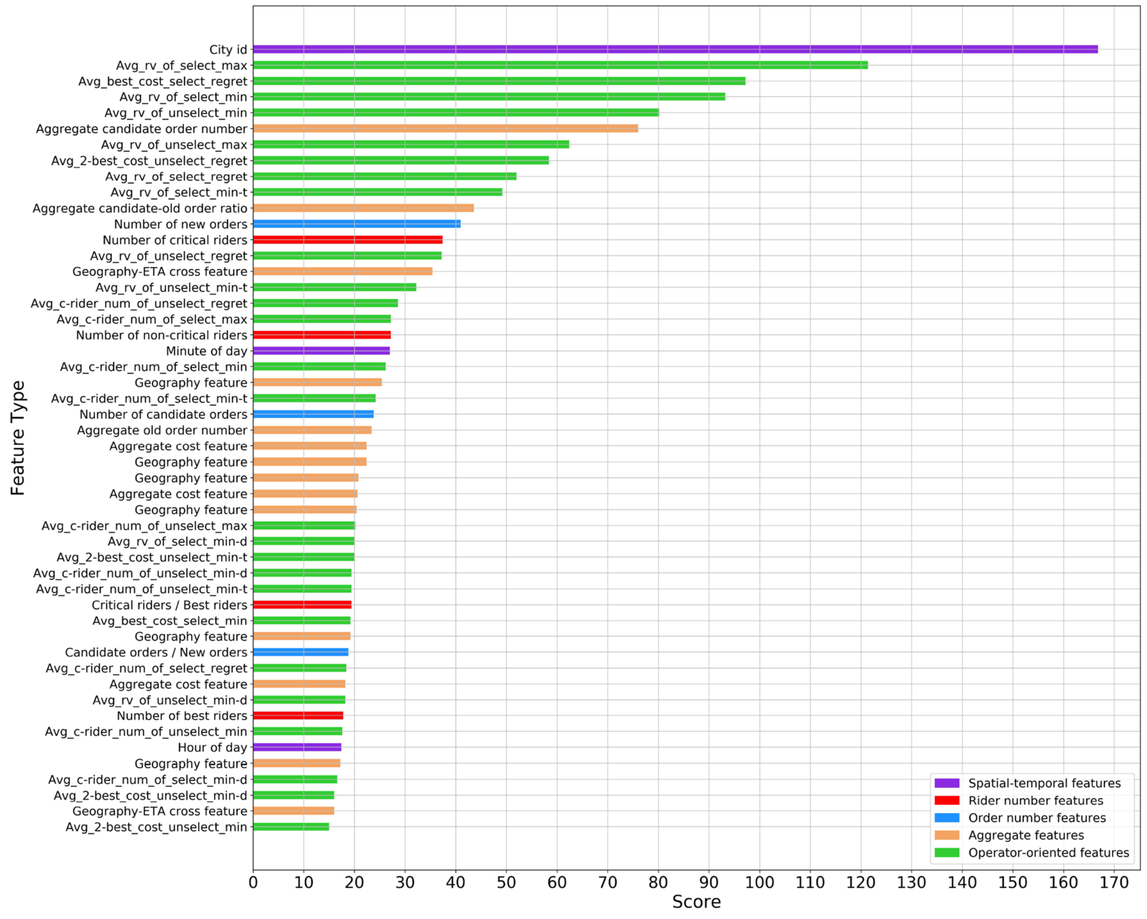


Fig. 9 Importance of features

Table 3 AUC on predicting each tie-breaking operator

Model	Operator				
	MIN	MAX	MINT	MIND	REG
XGBoost	0.86522	0.77785	0.78102	0.86220	0.67454
mDeepFM	**0.86579**	**0.79336**	**0.78719**	**0.86956**	**0.69645**

To sum up, the results indicate that the design of problem-specific features is effective.

3) Prediction accuracy of the models

The AUCs of XGBoost and mDeepFM on predicting each operator are listed in Table 3. It can be seen that the prediction performance of mDeepFM is slightly better than XGBoost. Besides, the AUCs on different operators vary for both models and the AUCs of both models on a specific operator are close. This special phenomenon can be explained according to Fig. 8. To be specific, it is natural that the models tend to have a bias to the class with more instances to gain more accuracy. In other words, the more unbalanced the number of positive instances and negative instances, the more easily the models distinguish. Therefore, the models predict the performances of MIN and MINT operators more accurately since their ratios between positive and negative instances are most unbalanced, while it is more difficult to predict the performance of REG operator since the ratio is close to 0.5.

4) Performance of the algorithms

The results of *RPD*s under different metrics are listed in Tables 4, 5, 6, 7, 8 and 9. To investigate the performances of algorithms under different critical rider number (denoted as m_{cr}) and candidate order number (denoted as n_{co}), the *PRD*s are grouped by different m_{cr} and n_{co} based on the data distribution discussed above. Each cell within the tables is the average *RPD* on the same group of instances. Tables 4 and 5 shows that the BM-REG performs best among all the comparative algorithms, which is in accord with the analysis of Fig. 8. Besides, the MAATS-mDeepFM outperforms the MAATS-XGB on most instances, especially when m_{cr} and

Table 4 RPDs on ADC grouped by the number of critical riders

m_{cr}	BM-MIN	BM-MAX	BM-MINT	BM-MIND	BM-REG	BM-RAND	MAATS-XGB	MAATS-mDeepFM
1	2.376	2.076	2.076	2.402	1.221	1.997	1.097	**1.057**
2	2.173	1.843	1.865	2.17	0.968	1.787	0.937	**0.918**
3	2.059	1.698	1.785	2.019	0.821	1.682	0.817	**0.798**
4	2.018	1.587	1.691	1.979	0.718	1.623	0.713	**0.699**
5	1.959	1.517	1.615	1.918	0.650	1.520	0.651	**0.633**
6	1.961	1.412	1.564	1.888	0.630	1.507	0.623	**0.618**
7	1.875	1.341	1.519	1.814	**0.547**	1.414	**0.547**	**0.547**
8	1.784	1.362	1.455	1.769	0.528	1.356	0.528	**0.521**
9	1.704	1.327	1.405	1.695	0.478	1.328	0.478	**0.476**
10	1.677	1.293	1.356	1.663	**0.428**	1.271	**0.428**	0.433
11–100	1.575	1.169	1.169	1.571	**0.230**	1.138	**0.230**	0.231
> 100	2.171	1.976	1.571	2.237	0.241	1.664	0.241	**0.240**
Average	1.944	1.550	1.589	1.927	0.622	1.524	0.608	**0.598**

Bold value represents the best result in a row

Table 5 RPDs on ADC grouped by the number of candidate orders

n_{co}	BM-MIN	BM-MAX	BM-MINT	BM-MIND	BM-REG	BM-RAND	MAATS-XGB	MAATS-mDeepFM
2–10	2.110	1.806	1.836	2.113	1.005	1.769	0.944	**0.916**
11–20	1.965	1.473	1.619	1.914	0.628	1.514	0.627	**0.618**
21–30	1.793	1.349	1.441	1.755	0.470	1.368	0.470	**0.466**
31–40	1.703	1.284	1.358	1.707	0.394	1.303	0.394	**0.398**
41–50	1.599	1.241	1.266	1.606	**0.366**	1.238	**0.366**	**0.366**
51–60	1.613	1.256	1.251	1.653	**0.290**	1.218	**0.290**	0.297
61–70	1.548	1.166	1.179	1.547	**0.211**	1.143	**0.211**	0.212
71–80	1.529	1.153	1.152	1.546	**0.181**	1.083	**0.181**	**0.181**
81–90	1.517	1.122	1.125	1.466	**0.219**	1.100	**0.219**	**0.219**
91–100	1.488	1.110	1.062	1.466	**0.200**	1.084	**0.200**	**0.200**
> 100	1.819	1.388	1.268	1.820	**0.157**	1.282	**0.157**	**0.157**
Average	1.699	1.304	1.323	1.690	0.375	1.282	0.369	**0.366**

Bold value represents the best result in a row

n_{co} is small, which mainly owes to the reason that the prediction accuracy of the mDeepFM is better than the XGBoost according to Table 3. The average *RPD*s of both MAATS-mDeepFM and MAATS-XGB are smaller than the comparative algorithms on the instances with small m_{cr} and n_{co}. However, when m_{cr} and n_{co} increase, the performances of both the above methods are getting close to the BM-REG. This is because the unbalanced positive and negative instances cause the models to overfit the REG operator, especially for the XGBoost. Nevertheless, the overall average *RPD*s in last line show that both adaptive tie-breaking algorithms surpass all other algorithms, which demonstrates the effectiveness of the adaptive tie-breaking strategy. Besides, it can be seen that the random selection of operators also performs better than using a single operator on some instances with small m_{cr} and n_{co}. Therefore, it can be concluded that hybrid and adaptive utilization of multiple tie-breaking operators is superior to the utilization of a single operator.

The RPDs of the observation metrics are listed in Tables 6, 7, 8 and 9. Tables 6 and 7 show that the MAATS-mDeepFM obtains all best average *RPD*s considering the average increased distance, which means that the MAATS-mDeepFM saves more travel distances than the others when dispatching new orders. However, in Tables 8 and 9, an interesting result is that the BM-MAX achieves the best performance on the average *RPD*s of average consumed time. One inference can be that the MAX operator is capable of balancing the burdens of riders, thus shortening the total travel time for each rider on average. To be specific, dispatching the candidate order with the largest cost to a

Table 6 RPDs on AIDs grouped by the number of critical riders

m_{cr}	BM-MIN	BM-MAX	BM-MINT	BM-MIND	BM-REG	BM-RAND	MAATS-XGB	MAATS-mDeepFM
1	1.575	1.220	1.356	1.562	0.505	1.180	0.435	**0.335**
2	1.741	1.000	1.391	1.633	0.494	1.243	0.469	**0.419**
3	1.859	0.905	1.416	1.754	0.463	1.280	0.458	**0.386**
4	1.925	0.793	1.420	1.810	0.397	1.267	0.397	**0.339**
5	1.878	0.722	1.392	1.765	0.384	1.218	0.383	**0.320**
6	1.770	0.617	1.241	1.653	0.300	1.144	0.300	**0.242**
7	1.771	0.577	1.215	1.687	0.249	1.077	0.249	**0.225**
8	1.619	0.512	1.126	1.527	0.239	1.006	0.239	**0.202**
9	1.536	0.465	1.049	1.449	0.167	0.939	0.167	**0.132**
10	1.519	0.432	1.017	1.426	0.141	0.887	0.141	**0.135**
11–100	1.356	0.414	0.914	1.277	0.018	0.792	0.018	**0.012**
> 100	2.343	1.041	1.724	2.171	– 0.053	1.452	– 0.053	**– 0.055**
Average	1.741	0.725	1.272	1.643	0.275	1.124	0.267	**0.224**

Bold value represents the best result in a row

Table 7 RPDs on AIDs grouped by the number of candidate orders

n_{co}	BM-MIN	BM-MAX	BM-MINT	BM-MIND	BM-REG	BM-RAND	MAATS-XGB	MAATS-mDeepFM
2–10	1.537	0.842	1.211	1.467	0.455	1.114	0.426	**0.354**
11–20	1.792	0.683	1.294	1.698	0.333	1.164	0.334	**0.282**
21–30	1.674	0.514	1.160	1.569	0.195	1.021	0.195	**0.158**
31–40	1.428	0.427	0.987	1.348	0.117	0.864	0.117	**0.100**
41–50	1.328	0.366	0.869	1.246	0.072	0.797	0.072	**0.061**
51–60	1.384	0.355	0.906	1.301	0.054	0.814	0.054	**0.047**
61–70	1.303	0.371	0.855	1.230	0.019	0.747	0.019	**0.015**
71–80	1.24	0.336	0.807	1.183	0.017	0.712	0.017	**0.016**
81–90	1.248	0.369	0.820	1.179	**0.009**	0.738	**0.009**	**0.009**
91–100	1.289	0.345	0.845	1.209	**– 0.005**	0.729	**– 0.005**	**– 0.005**
> 100	1.805	0.675	1.273	1.688	**– 0.032**	1.074	**– 0.032**	**– 0.032**
Average	1.457	0.480	1.002	1.374	0.112	0.889	0.110	**0.091**

Bold value represents the best result in a row

critical rider may lead to a high burden for this rider. Thus, in the next loop, this rider is not appropriate to be dispatched with any new orders so that the remaining candidate orders are dispatched to other riders. Nevertheless, the gap on ACTs between BM- MAX and each adaptive tie-breaking algorithm is much closer than those between BM-MAX and the other algorithms. However, on ADCs and AIDs, the BM-MAX is significantly inferior to the two adaptive tie-breaking algorithms. Therefore, the two adaptive tie-breaking algorithms still outperform the other algorithms in general. From another viewpoint, the OFDP is actually a multi-objective optimization problem so it is natural that the metrics conflict with each other.

Online A/B test

According to the analysis of Fig. 8 and the results in Tables 4, 5, 6, 7, 8 and 9, the BM-REG performs best among all comparative algorithms in general and the MAATS-mDeepFM is better between the two adaptive tie-breaking methods on a majority of instances. Therefore, the BM-REG and MAATS-mDeepFM are chosen as the representative methods for online A/B test.

Settings of online A/B test

A/B test is a commonly-used method to compare two versions with a single variable. In this paper, the way of using tie-breaking operators is taken as the variable, while the

Table 8 RPDs on ACTs grouped by the number of critical riders

m_{cr}	BM-MIN	BM-MAX	BM-MINT	BM-MIND	BM-REG	BM-RAND	MAATS-XGB	MAATS-mDeepFM
1	0.140	**0.028**	0.075	0.137	0.053	0.089	0.038	**0.028**
2	0.131	**0.020**	0.076	0.125	0.038	0.079	0.035	0.026
3	0.131	**0.010**	0.070	0.128	0.035	0.077	0.035	0.022
4	0.136	**0.004**	0.071	0.132	0.030	0.076	0.030	0.021
5	0.136	**0.000**	0.072	0.132	0.032	0.075	0.032	0.020
6	0.126	**−0.005**	0.061	0.123	0.024	0.067	0.024	0.015
7	0.124	**−0.007**	0.060	0.121	0.020	0.063	0.020	0.014
8	0.113	**−0.009**	0.055	0.110	0.020	0.059	0.020	0.014
9	0.110	**−0.010**	0.052	0.107	0.014	0.053	0.014	0.009
10	0.105	**−0.013**	0.047	0.103	0.013	0.050	0.013	0.009
11–100	0.090	**−0.013**	0.039	0.089	0.004	0.042	0.004	0.003
> 100	0.127	**−0.002**	0.058	0.126	−0.001	0.061	−0.001	−0.001
Average	0.122	**0.000**	0.061	0.119	0.024	0.066	0.022	0.015

Bold value represents the best result in a row

Table 9 RPDs on ACTs grouped by the number of candidate orders

n_{co}	BM-MIN	BM-MAX	BM-MINT	BM-MIND	BM-REG	BM-RAND	MAATS-XGB	MAATS-mDeepFM
2–10	0.126	**0.019**	0.069	0.123	0.042	0.077	0.034	0.025
11–20	0.128	**−0.002**	0.065	0.126	0.026	0.069	0.026	0.017
21–30	0.120	**−0.009**	0.057	0.117	0.017	0.060	0.017	0.012
31–40	0.099	**−0.010**	0.048	0.098	0.011	0.049	0.010	0.008
41–50	0.091	**−0.014**	0.040	0.089	0.008	0.044	0.008	0.006
51–60	0.093	**−0.014**	0.042	0.093	0.007	0.045	0.008	0.006
61–70	0.088	**−0.014**	0.038	0.087	0.004	0.040	0.004	0.003
71–80	0.086	**−0.014**	0.035	0.085	0.004	0.038	0.005	0.004
81–90	0.086	**−0.013**	0.036	0.085	0.004	0.040	0.003	0.003
91–100	0.089	**−0.016**	0.036	0.087	0.003	0.041	0.003	0.003
> 100	0.108	**−0.010**	0.046	0.107	0.001	0.051	0.001	0.001
Average	0.101	**−0.009**	0.047	0.100	0.012	0.050	0.011	0.008

Bold value represents the best result in a row

Table 10 Results of online A/B test in City 1

Metric	Comparison dates			Experimental dates			Δ_{real}
	C-region	E-region	Δ_{inh}	C-region	E-region	Δ_{exp}	
antd	1.1732	1.1563	−0.0169	1.1858	1.1602	−0.0256	**−0.0087**
adt	29.8743	30.0947	0.0024	30.4021	29.9114	−0.4907	**−0.4931**
pr	96.76%	96.82%	0.06%	96.75%	96.94%	0.19%	**0.13%**
pp_{55}	5.43%	6.22%	0.79%	5.25%	5.06%	−0.19%	**−0.98%**
or_{15^+}	1.31%	1.33%	0.02%	1.18%	1.16%	−0.02%	**−0.04%**

Bold value represents the best result in a row

BM-REG and MAATS-mDeepFM share the same other algorithmic settings. The online A/B test is rigorously set as follows. Two cities are selected, each of which is split into a comparison region (C-region) and an experimental region (E-region). To make a fair comparison, the two regions are generated as similar as possible with respect to the following metrics during a period of time (denoted as comparison dates). We implement the BM-REG in the C-region and the MAATS-mDeepFM in the E-region for a period of experimental time (denoted as experimental dates). The test starts immediately after the comparison dates and the experimental dates last the same as the comparison dates. The online observation metrics are categorized into two types as follows.

1) Rider efficiency-based metrics
- average normalized travel distance:

$$\text{antd} = \frac{1}{|O|} \sum_{o(i) \in O} \frac{\text{ad}_i}{\text{nd}_i}, \tag{25}$$

- average delivery time (unit: minute):

$$\text{adt} = \frac{1}{|O|} \sum_{o(i) \in O} \left(\text{dt}_i - \text{rt}_i\right), \tag{26}$$

2) Customer satisfaction-based metrics
- punctual rate:

$$\text{pr} = \frac{\left|O_{\text{punc}}\right|}{|O|} \times 100\%, \tag{27}$$

- Proportion of 55-min orders:

$$\text{pp}_{55} = \frac{\left|O_{55}\right|}{|O|} \times 100\%, \tag{28}$$

- 15-min overtime rate:

$$\text{or}_{15} = \frac{\left|O_{15+}\right|}{|O|} \times 100\%. \tag{29}$$

O is the set of all orders during a day, ad_i is the actual travel distance of the rider to deliver order $o(i)$, nd_i is the navigation travel distance to deliver $o(i)$, dt_i is the time when $o(i)$ is delivered to the customer, rt_i is the time when $o(i)$ is received by the platform, O_{punc} is the set of orders that are delivered punctually, O_{55} is the set of orders that satisfy $\text{dt}_i - \text{rt}_i > 55$ min and O_{15+} is the set of orders that are delayed for more than 15 min.

Results of online A/B tests

The results of the A/B tests are shown in Tables 10 and 11. Although the metrics of C-region and E-region are as close as possible, there still exist differences between the two regions. We denote the difference of metrics between E-region and C-region during comparison dates as their inherent difference (Δ_{inh}) and the difference of metrics during experimental dates as an experimental difference (Δ_{exp}). To exclude the influence of inherent differences, each experimental difference is revised by subtracting the inherent difference, which yields the final result (Δ_{real}). From the final results, it can be seen that the MAATS-mDeepFM outperforms the BM-REG on all online observation metrics in both cities. Therefore, it can be concluded that the proposed algorithm is of application significance to improve the customer satisfaction and delivery efficiency for the platform.

Conclusions

This paper addresses an online food delivery problem derived from a real online food delivery platform, Meituan. To dispose the dynamism and urgency, the problem is transformed into a static optimization problem. An effective matching algorithm with an adaptive tie-breaking strategy is proposed for solving the problem. A best-matching heuristic is embedded to fast generate a partial solution possibly with ties while guaranteeing the quality. To construct a complete solution with high quality, multiple tie-breaking operators are designed based on different intentions. To fit in different scenarios, the best operator is adaptively utilized to break ties each time when solving the static optimization problem. The selection of the best operator is converted into a multi-label classification problem which is solved by specially-designed machine learning methods. Offline simulations are conducted on the real historical data from Meituan, which demonstrates the effectiveness of the proposed algorithm to solve the OFDP. Online A/B tests are carried out in real-world applications, which validate the practicable value of the proposed algorithm to improve customer satisfaction and delivery efficiency. Moreover, the experiment results also reveal the success of adaptive mechanism to utilize hybrid

Table 11 Results of online A/B test in City 2

Metric	Comparison dates			Experimental dates			Δ_{real}
	C-region	E-region	Δ_{inh}	C-region	E-region	Δ_{exp}	
antd	1.0455	1.0111	−0.0344	1.0473	0.9857	−0.0616	**−0.0272**
adt	28.4564	27.7113	−0.7451	30.4046	29.2081	−1.1965	**−0.4514**
pr	98.07%	98.07%	0.01%	97.84%	97.89%	0.05%	**0.04%**
pp_{55}	2.05%	1.22%	−0.83%	3.89%	2.58%	−1.39%	**−0.56%**
or_{15+}	0.49%	0.49%	−0.00%	0.55%	0.52%	−0.02%	**−0.02%**

Bold value represents the best result in a row operators, problem-specific strategy to extract useful knowledge and the incorporation of ML techniques to assist decision making in complex optimization system.

In future work, the study on improving the performances of tie-breaking operators will be furthered. Also, it is interesting to develop more efficient and effective algorithms for solving the online food delivery problem by introducing other optimization methods such as genetic programming or other ML technologies such as deep reinforcement learning. Besides, it is valuable to consider the uncertainty arising in the online food delivery process such as uncertain food preparation time, delivery time, and rider behaviors.

Appendix

The design of the aggregate geography-ETA features and aggregate cost features is elaborated in this part. The statistics for each critical rider are computed and the statistics of the same type are averaged by the number of critical riders, which yield $16 \times 6 = 96$ features in total.

Aggregate geography-ETA features

1. ETA features

- Statistics of $\{\Delta\mathrm{ETA}_{i,i'} | i_-, i'_- \in D\}$. $\Delta\mathrm{ETA}_{i,i'} = \mathrm{ETA}_i - \mathrm{ETA}_{i'}$ denotes the difference of ETAs between two candidate orders of a critical rider.

2. Geographical features

The measurements of geographical distribution include the distance of two nodes and the angle of two vectors formed by two pairs of nodes.

2.1. Basic geographic relationship of two orders

- Statistics of $\{d_{i_+,i_-} | i_+ \in P_j^c, i_- \in D_j^c\}$, where $d_{i_+i_-}$ denotes the distance between the pickup node and delivery node of an order, P_j^c and D_j^c denotes the candidate order set of a critical rider q_j.
- Statistics of $\{d_{i_+,i'_+} | i_+, i'_+ \in P_j^c\}$, where d_{i_+,i'_+} denotes the distance between the pickup nodes of two candidate orders of a critical rider q_j.
- Statistics of $\{d_{i_+,i'_-} | i_+ \in P_j^c, i'_- \in D_j^c, i_+ \neq i'_+\}$,here d_{i_+,i'_-} denotes the distance between the pickup node of one candidate order and the delivery node of another candidate order.
- Statistics of $\{d_{i_-,i'_-} | i_-, i'_- \in D_j^{\mathrm{c}}\}$, where d_{i_-,i'_-} denotes the distance between the delivery nodes of two candidate orders of a critical rider q_j.
- Statistics of $\{\langle i_+i_-, i'_+i'_- \rangle | i_+, i'_+ \in P_j^{\mathrm{c}}, i_-, i'_- \in D_j^{\mathrm{c}}\}$., where $\langle i_+i_-, i'_+i'_- \rangle$ denotes the angle between vector i_+i_- and vector $i'_+i'_-$, and the notation of vector i_+i_- means it starts at i_+ and ends at i_-, the same below.
- atistics of $\{\langle i_+i_-, i_+i'_+ \rangle | i_+, i'_+ \in P, i_-, i'_- \in D\}$, where $\langle i_+i_-, i'_+i'_- \rangle$ denotes the angle between vector i_+i_- and vector $i_+i'_+$.
- Statistics of $\{\langle i_+i_-, i_+i'_- \rangle | i_+, i'_+ \in P, i_-, i'_- \in D\}$, where $\langle i_+i_-, i'_+i'_- \rangle$ denotes the angle between vector i_+i_- and vector $i_+i'_-$.

2.2. Geographic proximity of two orders

- Statistics of $\left\{ \max\left(d_{i_+,i'_+}, d_{i_-,i'_-} \right) \right\}$.
- Statistics of $\left\{ d_{i_+,i'_+} + d_{i_-,i'_-} \right\}$.
- Statistics of $\left\{ \max\left\{ d_{i_+,i_-} / d_{i'_+,i'_-} / (\langle i_+i_-, i_+i'_+ \rangle + \langle i_+i_-, i_+i'_- \rangle + S), d_{i'_+,i'_-} / d_{i_+,i_-} / (\langle i'_+i'_-, i'_+i_+ \rangle + \langle i'_+i'_-, i'_+i_- \rangle + S) \right\} \right\}$,where S is a small decimal, the same below.

3. Geography-ETA cross features

- Statistics of $\{\max\{\Delta\mathrm{ETA}_{i,i'} / (d_{i_+,i'_-} + S), \Delta\mathrm{ETA}_{i',i} / \left(d_{i'_+,i_-} + S \right)\}\}$.
- Statistics of $\{\left|\Delta\mathrm{ETA}_{i,i'}\right| / (d_{i_-,i'_-} + S)\}$, where S is a small decimal.

Aggregate cost features

- Statistics of $\left\{ \mathrm{TC}_j^n\left(o_j^{\mathrm{c}}(1)\right), \mathrm{TC}_j^n\left(o_j^{\mathrm{c}}(2)\right), \ldots, \mathrm{TC}_j^n\left(o_j^{\mathrm{c}}\left(n_j^{\mathrm{c}}\right)\right) \right\}$, where $o_j^{\mathrm{c}}(i) \in O_j^{\mathrm{c}}\left(i = 1, 2, \ldots, n_j^{\mathrm{c}}\right)$ is the candidate order of critical rider q_j and $\mathrm{TC}_j^n\left(o_j^{\mathrm{c}}(i)\right)$ is the time cost of dispatching $o_j^{\mathrm{c}}(i)$ to q_j.
- Statistics of $\left\{ \mathrm{DC}_j^n\left(o_j^{\mathrm{c}}(1)\right), \mathrm{DC}_j^n\left(o_j^{\mathrm{c}}(2)\right), \ldots, \mathrm{DC}_j^n\left(o_j^{\mathrm{c}}\left(n_j^{\mathrm{c}}\right)\right) \right\}$, where $\mathrm{DC}_j^n\left(o_j^{\mathrm{c}}(i)\right)$ is the distance cost of dispatching $o_j^{\mathrm{c}}(i)$ to q_j.
- Statistics of $\left\{ C_j^n\left(o_j^{\mathrm{c}}(1)\right), C_j^n\left(o_j^{\mathrm{c}}(2)\right), \ldots, C_j^n\left(o_j^{\mathrm{c}}\left(n_j^{\mathrm{c}}\right)\right) \right\}$, where $C_j^n\left(o_j^{\mathrm{c}}(i)\right)$ is the cost of dispatching $o_j^{\mathrm{c}}(i)$ to q_j in the cost matrix.

Acknowledgements This study is supported by the National Science Fund for Distinguished Young Scholars of China (No. 61525304) and the National Natural Science Foundation of China (No. 61873328), and Meituan.

Declarations

Conflict of interest The authors declare that we have no conflict of interest.

References

1. Aleksandrov M, Barahona P, Kilby P, Walsh T (2013) Heuristics and policies for online pickup and delivery problems. In: 2013 AAAI conference on artificial intelligence (AAAI), Bellevue, Washington, USA
2. Arslan AM, Agatz N, Kroon L, Zuidwijk R (2019) Crowdsourced delivery—a dynamic pickup and delivery problem with ad hoc drivers. Transp Sci 53:222–235
3. Berbeglia G, Cordeau J-F, Laporte G (2010) Dynamic pickup and delivery problems. Eur J Oper Res 202:8–15
4. Bräysy O, Nakari P, Dullaert W, Neittaanmäki P (2009) An optimization approach for communal home meal delivery service: a case study. J Comput Appl Math 232:46–53
5. Chen J, Wang S, Wang L et al (2020) A hybrid differential evolution algorithm for the online meal delivery problem. In: 2020 IEEE congress on evolutionary computation (CEC), Glasgow, United Kingdom
6. Chen T, Guestrin C (2016) XGBoost: a scalable tree boosting system. In: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining (KDD), San Francisco, USA, pp 785–794
7. Cosmi M, Nicosia G, Pacifici A (2019) Scheduling for last-mile meal-delivery processes. IFAC-Pap 52:511–516
8. Cosmi M, Oriolo G, Piccialli V, Ventura P (2019) Single courier single restaurant meal delivery (without routing). Oper Res Lett 47:537–541
9. Dev VA, Eden MR (2019) Gradient boosted decision trees for lithology classification. In: Proceedings of the 9th international conference on foundations of computer-aided process design (FOCAPD), Raleigh, North Carolina, USA, vol 47, pp 113–118
10. Dhaliwal S, Nahid A-A, Abbas R (2018) Effective intrusion detection system using XGBoost. Information 9:149
11. Feng Y, Wang D, Yin Y et al (2020) An XGBoost-based casualty prediction method for terrorist attacks. Complex Intell Syst 6:721–740
12. Fernandez-Viagas V, Framinan JM (2014) On insertion tie-breaking rules in heuristics for the permutation flowshop scheduling problem. Comput Oper Res 45:60–67
13. Ferrucci F, Bock S (2014) Real-time control of express pickup and delivery processes in a dynamic environment. Transp Res Part B Methodol 63:1–14
14. Fkaier ZK, Chaar BF (2013) Online K-means based heuristic for the dynamic pickup and delivery problem solving. In: 2013 world congress on computer and information technology (WCCIT), Sousse, Tunisia, pp 1–6
15. Guo H, Tang R, Ye Y et al (2017) DeepFM: a factorization-machine based neural network for CTR prediction. In: Proceedings of the 26th international joint conference on artificial intelligence (IJCAI), Melbourne, Australia, pp 1725–1731
16. Li C, Mirosa M, Bremer P (2020) Review of online food delivery platforms and their impacts on sustainability. Sustainability 12:5528
17. Liu Y (2019) An optimization-driven dynamic vehicle routing algorithm for on-demand meal delivery using drones. Comput Oper Res 111:1–20
18. Liu Y, Guo B, Chen C et al (2019) FooDNet: toward an optimized food delivery network based on spatial crowdsourcing. IEEE Trans Mob Comput 18:1288–1301
19. Lu Y, Wu Y, Zhou Y (2017) Order assignment and routing for online food delivery: two meta-heuristic methods. In: Proceedings of the 2017 international conference on intelligent systems, metaheuristics and swarm intelligence (ISMSI), Hong Kong, China, pp 125–129
20. Luo H, Liufu M, Li D (2020) Intelligent online food delivery system: a dynamic model to generate delivery strategy and tip advice. arXiv preprint arXiv: 2002.01713.
21. Meituan, CFLP (2019) Report on the development of Chinese immediate delivery business in 2019. http://pdf.dfcfw.com/pdf/H3_AP202006011381522218_1.pdf. Accessed 15 Dec 2020
22. Meituan, China Hospitality Association (2020) Report on the development of Chinese take-out industry in 2019 and the first half of 2020. https://ncstatic.clewm.net/rsrc/2020/0628/09/84f7f3e18c6e27cb32227534f640bd45.pdf. Accessed 15 Dec 2020
23. Mitchell R, Frank E (2017) Accelerating the XGBoost algorithm using GPU computing. PeerJ Comput Sci 3:e127
24. Morgan Stanley Research (2017) Is online food delivery about to get 'amazoned'? https://www.morganstanley.com/ideas/online-food-delivery-market-expands. Accessed 18 May 2020
25. Muñoz-Carpintero D, Sáez D, Cortés CE, Núñez A (2015) A methodology based on evolutionary algorithms to solve a dynamic pickup and delivery problem under a hybrid predictive control approach. Transp Sci 49:239–253
26. Nargesian F, Samulowitz H, Khurana U et al (2017) Learning feature engineering for classification. In: Proceedings of the 26th international joint conference on artificial intelligence (IJCAI), Melbourne, Australia, pp 2529–2535
27. Ogunleye A, Wang QG (2019) XGBoost model for chronic kidney disease diagnosis. IEEE/ACM Trans Comput Biol Bioinform. https://doi.org/10.1109/TCBB.2019.2911071
28. Reyes D, Erera A, Savelsbergh M et al (2018) The meal delivery routing problem. Optim Online
29. Ropke S, Pisinger D (2006) An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. Transp Sci 40:455–472
30. Sheridan PK, Gluck E, Guan Q et al (2013) The dynamic nearest neighbor policy for the multi-vehicle pick-up and delivery problem. Transp Res Part Policy Pract 49:178–194
31. Steever Z, Karwan M, Murray C (2019) Dynamic courier routing for a food delivery service. Comput Oper Res 107:173–188
32. Tsoumakas G, Katakis I (2007) Multi-label classification: an overview. Int J Data Warehous Min 3:1–13
33. Ulmer M, Thomas BW, Campbell AM, Woyak N (2017) The restaurant meal delivery problem: dynamic pick-up and delivery with deadlines and random ready times. Transp Sci. https://doi.org/10.1287/trsc.2020.1000
34. Vonolfen S, Affenzeller M (2016) Distribution of waiting time for dynamic pickup and delivery problems. Ann Oper Res 236:359–382

35. Wang X, Wang S, Wang L et al (2020) An effective iterated greedy algorithm for online route planning problem. In: 2020 IEEE congress on evolutionary computation (CEC), Glasgow, United Kingdom
36. Wolpert DH, Macready WG (1997) No free lunch theorems for optimization. IEEE Trans Evol Comput 1:67–82
37. Yeo VCS, Goh S-K, Rezaei S (2017) Consumer experiences, attitude and behavioral intention toward online food delivery (OFD) services. J Retail Consum Serv 35:150–162
38. Yildiz B, Savelsbergh M (2019) Provably high-quality solutions for the meal delivery routing problem. Transp Sci 53:1372–1388
39. Yildiz H, Johnson MP, Roehrig S (2005) A genetic algorithm for the home-delivered meals location-routing problem. Heinz Coll Res
40. Yu H, Luo X, Wu T (2020) Online pickup and delivery problem with constrained capacity to minimize latency. J Comb Optim. https://doi.org/10.1007/s10878-020-00615-y
41. Zhang M-L, Zhou Z-H (2014) A review on multi-label learning algorithms. IEEE Trans Knowl Data Eng 26:1819–1837
42. Zheng H, Wang S, Cha Y et al (2019) A two-stage fast heuristic for food delivery route planning problem. In: Informs annual meeting, Seattle, Washington, USA
43. Zheng J, Wang S, Wang L et al (2020) A two-stage algorithm for fuzzy online order dispatching problem. In: 2020 IEEE congress on evolutionary computation (CEC), Glasgow, United Kingdom
44. Zhu Z, Xiao J, He S et al (2016) A multi-objective memetic algorithm based on locality-sensitive hashing for one-to-many-to-one dynamic pickup-and-delivery problem. Inf Sci 329:73–89