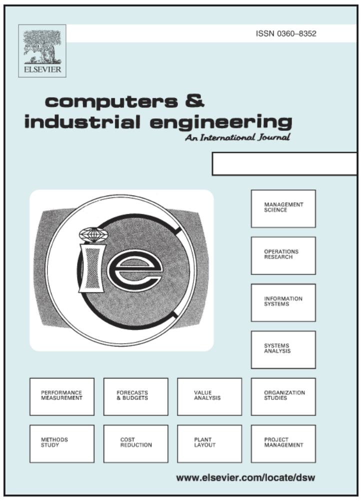


An XGBoost-enhanced fast constructive algorithm for food delivery route planning problem

Xing Wang, Ling Wang, Shengyao Wang, Jing-fang Chen, Chuge Wu

PII:	S0360-8352(20)30699-9
DOI:	https://doi.org/10.1016/j.cie.2020.107029
Reference:	CAIE 107029
To appear in:	*Computers & Industrial Engineering*
Received Date:	26 October 2020
Revised Date:	25 November 2020
Accepted Date:	6 December 2020

An XGBoost-enhanced fast constructive algorithm for food delivery route planning problem

ABSTRACT

As e-commerce booms, online food ordering and delivery has attracted much attention. For food delivery platforms, planning high-quality routes for drivers so as to accomplish the delivery tasks efficiently is of great importance. This paper addresses a food delivery route planning problem (FDRPP), which considers one driver delivering multiple orders from restaurants to customers. Due to the immediacy of the delivery tasks, very limited computational time is provided for generating satisfactory solutions. We mathematically formulate the FDRPP and propose an Extreme Gradient Boosting-enhanced (XGBoost-enhanced) fast constructive algorithm to solve the problem. To construct a complete route, an insertion-based heuristic with different sequencing rules is adopted, together with an acceleration strategy based on geographic information to speed up the insertion procedure. In order to avoid the waste of computational time, we design an adaptive selection mechanism to select sequencing rules for route construction. A classification model using XGBoost is established to predict the performance of different sequencing rules. Through analysis of the route construction procedure, three types of problem-specific features are designed to improve the performance of XGBoost. The effectiveness of the proposed algorithm is demonstrated by conducting experiments on datasets from Meituan food delivery platform, which shows that large amounts of computational time can be saved by our proposed algorithm, while guaranteeing the quality of solutions.

Keywords: food delivery route planning problem; fast constructive algorithm; adaptive selection mechanism; classification model; XGBoost

1. Introduction

With the development of e-commerce, modern life has become more and more convenient recently. People can enjoy a large diversity of services without walking outdoors, among which online food ordering and delivery service occupies a large portion. According to the report from Meituan and CFLP (2019), 70% of the online orders are generated by food delivery platforms. An earlier research by Morgan Stanley Research (2017) predicted that online food delivery was expected to grow by 16% annual compound rate in the next 5 years in the US. Taking Meituan, the largest food delivery platform in China, as an example, the maximal number of transaction orders in a single day has risen up to 30 million and the total number of transaction orders through 2019 has amounted to 18 billion (Meituan & CFLP, 2019). Worldwide food delivery companies such as Grubhub, Uber Eats, and Just Eat, are also developing fast with transaction amounting up to 94 billion dollars (Hirschberg, Rajko, Schumacher, & Wrulich, 2016). Stimulated by such a great market opportunity, online food ordering and delivery is developing and prospering very quickly.

While providing convenience and making considerable profits, food delivery platforms have to face with tremendous pressure. Not only the management of company staff, but also the performance of the dispatching system needs to accommodate to the increasing number of requests and service quality required by customers. Fig. 1 illustrates the general operation of the dispatching system in Meituan, where firstly customers use food ordering application on cellphone to select food they desire. After that, the dispatching system collects orders and pushes them to the restaurants and drivers. Restaurants prepare the food and drivers shuttle between restaurants and customers to finish the delivery tasks. To guarantee the well functioning of the dispatching system, order assignment and food delivery route planning need to be considered in decision-making. The former dispatches orders to appropriate drivers and the latter organizes a feasible route for every single driver. During peak hours of a day, explosive number of orders are generated, which brings difficulty in finding satisfactory solutions from enormous solution space. Meanwhile, due to the immediacy of delivery tasks, dispatching schemes need to be determined in a very short period of time. Therefore, it is necessary to design highly efficient algorithms for the dispatching system, considering both academic significance and practical needs.

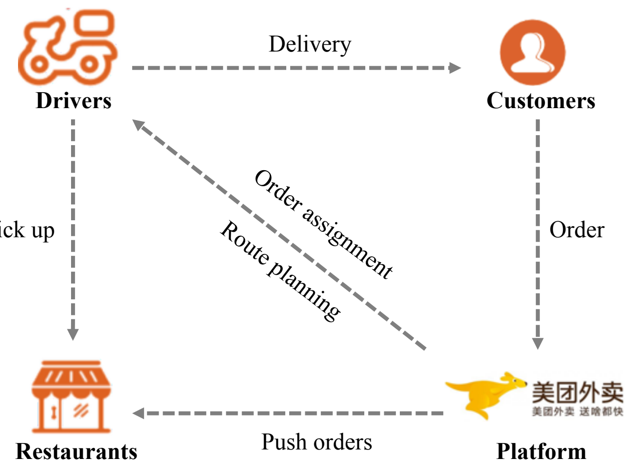


Fig. 1. Operation of the dispatching system

In this paper, we study the food delivery route planning problem (FDRPP), which is more fundamental and essential than order assignment since it serves as an indispensable part to evaluate an order assignment scheme. According to statistics from Meituan, the food delivery route planning module is called most frequently in the whole dispatching system, nearly 5 billion times a day. Therefore, saving computational time of the food delivery route planning algorithm is of great significance. Besides, the solution quality of FDRPP also needs to be promised since it directly influences the satisfaction of both drivers and customers. A well-planned route can not only assign the priority of orders properly to optimize the customer satisfaction, but also shorten the travel distance of the route as much as possible to reduce the burden of the driver. Hence, FDRPP needs to be solved effectively and efficiently to improve the performance of the dispatching system.

However, most traditional algorithms are not suitable for solving FDRPP since they usually consume too much computational time to reach satisfactory results. Under this situation, we develop an Extreme Gradient Boosting-enhanced (XGBoost-enhanced) fast constructive algorithm, denoted as XGB-FCA, to solve the problem efficiently. XGB-FCA functions as a hybrid algorithm which combines the machine learning technology and traditional constructive heuristics. It can save large amounts of computational time while guaranteeing the quality of solutions. To be specific, we adopt an insertion-based heuristic with different sequencing rules to construct high-quality routes. An acceleration strategy based on geographic information is used to speed up the insertion procedure. To further save computational time and keep the solution quality at the same time, we design an adaptive selection mechanism to select sequencing rules for route construction. A classification model based on XGBoost is established to predict the performance of different sequencing rules. Moreover, problem-specific features are designed to improve the performance of XGBoost.

The remainder of the paper is organized as follows. In Section 2, we briefly review the related literature related to FDRPP. Section 3 introduces the notation and formulation of FDRPP. Section 4 describes the details of the proposed XGB-FCA. Experimental results on datasets from Meituan food delivery platform are shown in Section 5. Finally, we close the paper with some conclusions and future research ideas in Section 6.

2. Literature review

The FDRPP is most similar to the single-vehicle pickup and delivery problem (SVPDP), which is a basic version of pickup and delivery problem (PDP), and an extension of traveling salesman problem (TSP). Therefore, we spend the majority of our review on SVPDP and its variants.

The SVPDP occurs when a single vehicle transports goods between pickup and delivery locations to fulfill customer demands. Algorithms for solving SVPDP can be classified into three categories, which are exact algorithms, constructive heuristics and meta-heuristics. Exact algorithms and constructive heuristics are well studied in the early stage of research. Lokin (1979) was the first to study the SVPDP. He extended the TSP studied by Little, Murty, Sweeney, and Karel (1963), formulated the mathematical model of SVPDP by adding additional precedence constraints, and proposed a branch and bound algorithm to solve instances of small scale. Kalantari, Hill, and Arora (1985) adapted the branch and bound algorithm by Little et al. (1963) and designed an exact algorithm, considering cases of finite and infinite vehicle capacity. Experiment showed that the algorithm could solve instances with scale up to 31 customers. Van der Bruggen, Lenstra, and Schuur (1993) proposed a two-stage heuristic based on variable depth search, where a feasible solution was constructed at the first stage and then improved by means of exchange operators at the second stage. Renaud, Boctor, and Ouenniche (2000) modified the classical heuristics initially designed for TSP to solve SVPDP, and further designed three different perturbation heuristics to generate nearly-optimal solutions (Renaud, Boctor, & Laporte, 2002). Cordeau, Iori, Laporte, and Salazar González (2010) studied the SVPDP with last-in-first-out (LIFO), where the goods that were picked last needed to be delivered first. They proposed three different mathematical models and proved several valid inequalities, based on which an effective branch and bound algorithm was developed.

Since 21st century, many meta-heuristic algorithms have been proposed for solving SVPDP. Landrieu, Mati, and Binder (2001) first proposed a tabu search algorithm for SVPDP with time windows. A basic tabu search algorithm and a probabilistic tabu search algorithm were designed with multiple operators. Carrabs, Cordeau, and Laporte (2007) considered the LIFO constraint and proposed a variable neighborhood search algorithm. They designed eight different construction rules to generate initial solutions, followed by three new local search operators whose computational complexities were reduced. Jih, Kao, and Hsu (2002) extended the genetic algorithm (GA) and developed a family competition genetic algorithm (FCGA) to solve the SVPDP. Compared with traditional GA, FCGA was more efficient and possible to generate best solutions due to effective selection of operators. Also attracted by GA, Hosny and Mumford (2007, 2010) focused on the design of encoding scheme and genetic operators. Huang and Ting (2010) adapted the ant colony optimization (ACO) algorithm to solve SVPDP. They dealt with the precedence constraint by using a tabu list. A repair operator was also designed to fix the infeasible solutions. Edelkamp and Gath (2014) designed a nested Monte-Carlo search (NMCS) algorithm for SVPDP, which used the Monte-Carlo random search to characterize the relationships between locations and extracted effective knowledge to determine the best location to visit next. Besides the standard SVPDP, researchers were also inspired by many real-life scenarios and therefore many variants of SVPDP were studied, such as multi-commodity capacitated PDP (Psaraftis, 2011), single-vehicle preemptive PDP (Kerivin, Lacroix, & Mahjoub, 2012) and single-commodity SVPDP (Martinovic, Aleksi, & Baumgartner, 2008). For more variants and models of SVPDP that consider dynamism, uncertainty and additional constraints, we refer to Parragh, Doerner, and Hartl (2008) and Cordeau, Laporte, and Ropke (2008) for interested readers to find more information.

Although there are a large number of algorithms for SVPDP, they are not directly applicable for FDRPP due to unacceptable computational time. As we mentioned earlier, food delivery route planning serves as an extremely fundamental part in the whole dispatching system and the route planning module is called very frequently. Therefore, very limited computational time is provided for the route planning algorithm to run. Any waste of time is supposed to be prevented. To cope with this situation, many researchers turn to the machine learning technology for more efficient methods to solve routing problems. Bello, Pham, Le, Norouzi, and Bengio (2017) presented a framework to tackle combinatorial optimization problems using neural networks and reinforcement learning. They focused on TSP, and trained a recurrent neural network that predicted a distribution over different city permutations. Cooray and Rupasinghe (2017) combined the machine learning with GA to solve the vehicle routing problem. The GA was enhanced by machine learning techniques through tuning its parameters. Their study revealed that, by identifying the underlying characteristics of data, a particular GA could significantly outperform any generic GA with competitive computational times after tuning. Nazari, Oroojlooy, Snyder, and Takac (2018) examined the application of reinforcement learning as a non-parametric model-free method to solve the vehicle routing problem. With a small case study on a mid-sized network, they demonstrated the significant advantages of using reinforcement learning. Mao and Shen (2018) discussed the application of reinforcement learning as a non-parametric model-free method in solving the adaptive routing problem in stochastic and time-dependent network. Yu, Yu, and Gu (2019) proposed a novel deep reinforcement learning-based neural combinatorial optimization strategy. They transformed the online routing problem to a vehicle tour generation problem, and proposed a structural graph embedded pointer network to develop these tours iteratively. Bengio, Lodi, and Prouvost (2020) surveyed and highlighted how machine learning could be used to build combinatorial optimization algorithms and concluded that machine learning was able to overcome the disadvantages that existed in traditional optimization algorithms, such as slow convergence and large computational time.

From the literature review, it can be seen that machine learning technology is popular and of great potential for solving combinatorial optimization problems, especially those with tight restriction on the computational time. Hence, we embed the machine learning technology into traditional optimization methods to solve the FDRPP effectively and efficiently in this paper.

3. Problem description and mathematical formulation

3.1. Problem description

In FDRPP, there is one driver serving several customers. A route must be constructed to satisfy the transportation requests. Each customer is associated with an order which consists of either only a single delivery location or both pickup and delivery locations. The former case indicates that the driver has already fetched the order at the restaurant while the latter indicates the order is not fetched. For each location, there is a positive service time for the driver to execute tasks. After all the locations have been visited, the driver will park at the last location of the route. The FDRPP is to minimize the total cost of the route, consisting of two parts that are the time delay of the route and the distance traveled by the driver, respectively.

The basic characteristics of the problem are as follows. (1) The driver must start from current location and end at one of the delivery locations. (2) Each location has to be visited exactly once. (3) The order cannot be picked up before its earliest pickup time. (4) Positive time delay will occur if actual delivery time of the order is later than its estimated time of arrival. (5) The total demand carried by the driver cannot exceed the maximum trunk capacity. (6) The pickup location has to be visited before the corresponding delivery location.

A typical illustration for FDRPP is shown in Fig. 2. In this case, order 1, 2 and 3 need to be served by the driver, which are colored in blue, yellow and green. Order 2 has already been picked up while order 1 and 3 have both pickup and delivery locations to be visited. Therefore, the pickup locations can be denoted as 1^+ and 3^+ and the delivery locations are 1^-, 2^- and 3^-. To form a feasible route, the driver can start from current location and sequentially visit 1^+, 2^-, 1^-, 3^+ and 3^-, as shown in Fig. 2. In addition, Fig. 3 shows an infeasible way to form a route where the driver visits the delivery location before the corresponding pickup location.

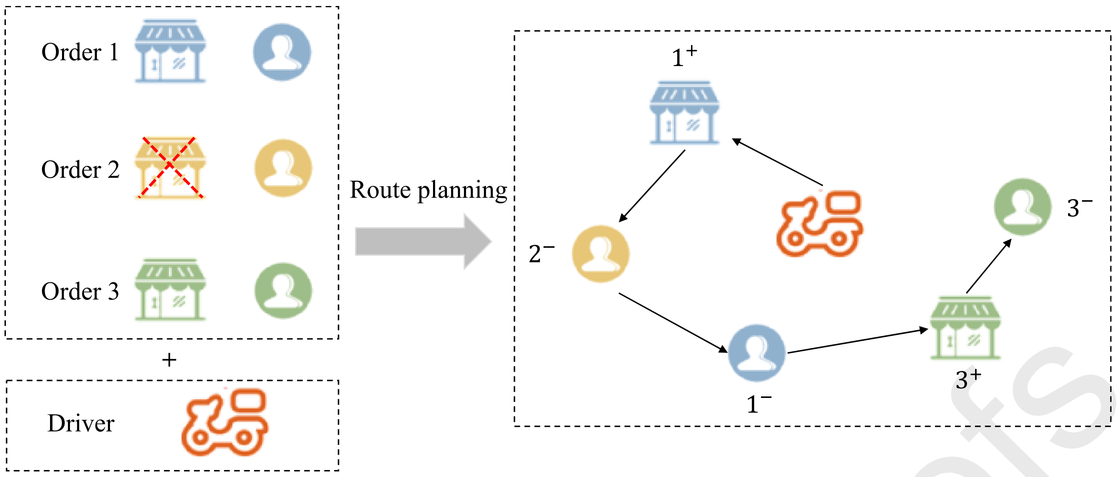


Fig. 2. A typical illustration for FDRPP

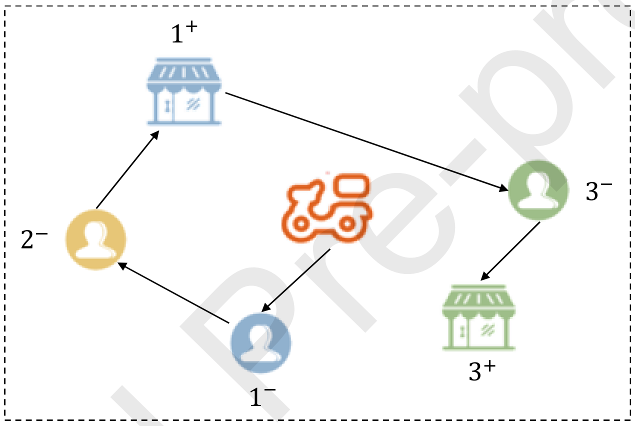


Fig. 3. An infeasible route for FDRPP

3.2. Mathematical formulation

According to the description in Section 3.1, the mathematical model of FDRPP is formulated as follows with notations listed in Table 1.

- Decision variables

  $x_{k,l}$: if the driver travels directly from location k to l, then $x_{k,l} = 1$; otherwise, $x_{k,l} = 0$.

  A_k: the time when the driver arrives at location k.

  T_k: the time when the driver departs from location k.

  u_k: the current demand that the driver is carrying when leaving location k.

- Objective function

$$\min TC = \alpha \sum_{i \in O} \max\{0, T_{i^-} - ETA_i\} + \beta \sum_{k \in S} \sum_{l \in S} d_{k,l} x_{k,l} \tag{1}$$

- Constraints

$$\sum_{\substack{k \in S \\ k \neq l}} x_{k,l} = 1, \forall l \in P \cup D \tag{2}$$

$$\sum_{\substack{l \in S \\ l \neq k}} x_{k,l} \leq 1, \forall k \in P \cup D \tag{3}$$

$$\sum_{k \in P \cup D} x_{k,cl} = 0 \tag{4}$$

$$\sum_{k \in P \cup D} x_{cl,k} = 1 \tag{5}$$

$$\sum_{(k,l) \in B \times B} x_{k,l} \leq |B| - 1, \forall B \subseteq P \cup D : 2 \leq |B| \leq n_1 + 2n_2 \tag{6}$$

$$x_{k,l} = 1 \Rightarrow u_l = u_k + q_l, \forall k,l \in S \tag{7}$$

$$u_{cl} = q_{cl} \tag{8}$$

$$0 \leq u_k \leq Q, \forall k \in S \tag{9}$$

$$A_l \geq T_k + t_{k,l} - (1 - x_{k,l})M, \forall k,l \in S \tag{10}$$

$$A_l \leq T_k + t_{k,l} + (1 - x_{k,l})M, \forall k,l \in S \tag{11}$$

$$A_k \leq T_k - s_k, \forall k \in P \cup D \tag{12}$$

$$T_{cl} = 0 \tag{13}$$

$$T_{i^+} \leq T_{i^-}, \forall i \in O_2 \tag{14}$$

$$T_{i^+} \geq EPT_i, \forall i \in O_2 \tag{15}$$

$$x_{k,l} \in \{0,1\}, \forall k \in S \tag{16}$$

The first term of (1) calculates TD and the second term is DIS. The weight factors α and β (both set as 1 in this paper) are used to balance TD and DIS. Constraints (2) ensures that each location associated with orders has only one predecessor location. Constraint (3) limits the number of successor locations to be not larger than 1, which prevents the driver from entering more than one location after leaving current location. On the other hand, it also allows the route to be non-closed. Constraint (4) and (5) guarantee that the driver starts from current location and enters another one, and indicate that the driver doesn't need to go back to the starting location when finishing all delivery tasks. Constraint (6) eliminates sub-tours and ensures that each location is visited exactly once by exactly one driver, together with constraints (2) and (3). Constraints (7), (8) and (9) are the capacity constraints. Since there may be orders that are already fetched, the initial demand of the driver q_{cl} is not always zero. Constraints (10) and (11) are related to the visiting time of each location and guarantee the feasibility of the schedule. To be specific, if location k and l are consecutive in the route of the driver, then the arrival time at location l is equal to the departure time from location k, plus the travel time between location k and l. Otherwise, constraints (10) and (11) are inactive. Constraint (12) introduces the relationship between the departure time and the arrival time at a certain location and constraint (13) sets the departure time of the driver from current location equal to zero. Constraint (14) ensures that for each order, the pickup location is always visited before visiting the corresponding delivery location. Constraint (15) limits the pickup location to be visited after the earliest pickup time. Lastly, constraint (16) defines the binary variable x for each possible pair of locations.

Table 1. Notations for the mathematical model.

Symbol	Explanation
O_1	Set of orders with only delivery locations
O_2	Set of orders with both pickup and delivery locations
$O = O_1 \cup O_2$	Set of all orders
n_1	Cardinality of O_1
n_2	Cardinality of O_2
P	Set of all pickup locations
D	Set of all delivery locations
cl	Current location of the driver
$S = P \cup D \cup \{cl\}$	Set of all locations
q_k	Demand to be transported at location k (positive for pickup locations and negative for delivery locations)
q_{cl}	Initial demand carried by the driver
EPT_i	Earliest pickup time of order i
ETA_i	Estimated time of arrival of order i
Q	Maximum trunk capacity
$d_{k,l}$	Travel distance between location k and l
$t_{k,l}$	Travel time between location k and l
s_k	Service time at location k
i	Index of orders
i^+	Pickup location of order i
i^-	Delivery location of order i
k,l	Index of locations
B	Subset of $P \cup D$
M	A large number
α,β	Weight factors
TC	Total cost of the route
TD	Time delay of the route
DIS	Distance traveled by the driver

4. XGB-FCA for FDRPP

In this section, we present the proposed XGB-FCA for solving FDRPP in a relatively short computational time, while obtaining satisfactory results. The flowchart of XGB-FCA is shown in Fig. 4, which is composed of two parts, the route construction and the adaptive selection mechanism. The route construction part adopts an insertion-based heuristic to construct a complete route. Firstly, order classification divides orders into two categories according to whether they have pickup locations. After the division, a priority list is generated by sorting orders according to a certain sequencing rule. Then greedy insertion places the candidate order into the best position of the partial route. During this procedure, an acceleration strategy is used to speed up the insertion. The adaptive selection mechanism selects valid sequencing rules for route construction to avoid the waste of computational time. It is based on a classification model using XGBoost, which predicts the performance of different sequencing rules using an already trained mapping function. To improve the performance of XGBoost, we conduct deep analysis on the route construction procedure and design some problem-specific features. We will elaborate the main components of XGB-FCA in the following subsections.

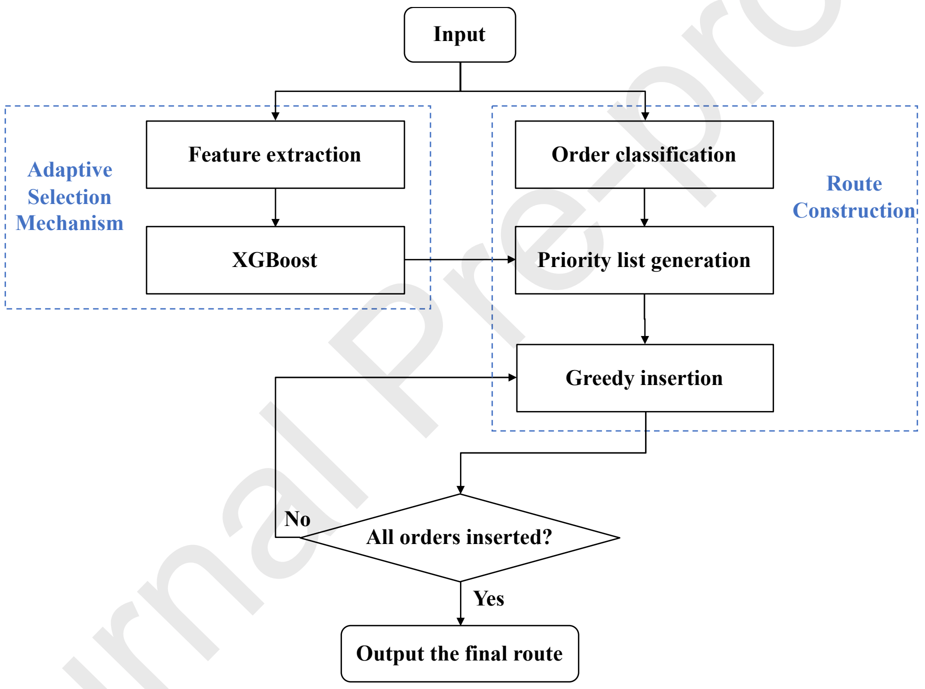


Fig. 4. The outline of XGB-FCA for FDRPP

4.1. Route construction

4.1.1. Insertion-based heuristic

To construct a complete solution for FDRPP, we adopt an insertion-based heuristic that was firstly proposed in our previous work (Wang et al., 2020). The main steps of the heuristic are described as follows.

Step 1 (Order classification). Classify the orders into two categories. The first set contains orders from O_1 and the second contains orders from O_2;

Step 2 (Priority list generation). Sort the orders inside each set according to a certain sequencing rule and put the first set ahead of the second set to construct a priority list;

Step 3 (Greedy insertion). Sequentially take out the orders from the priority list and insert them into the partial route. For orders in the first set, directly place them in the best position of the route; For orders in the second set, try all the possible pairs of pickup and delivery locations, then choose the best pair that does not violate the capacity constraint and precedence constraint.

Step 3 is executed iteratively until a complete solution is constructed. The sequencing rule mentioned in step 2 is essential since it determines the sequence of orders to insert into the partial route. Four sequencing rules were proposed and compared in Wang et al. (2020). Here, we adopt the best two of them, which are the RETA and RU.

- **Rule with estimated-time-of-arrival (RETA).** It sorts orders according to increasing ETA. Orders with a closer deadline are put ahead so as to avoid large amounts of time delay.
- **Rule with urgency (RU).** It sorts orders in a descending order of urgency, which is computed by the following equation,

$$\delta_i = \frac{\mu_i}{ETA_i - ct} \tag{17}$$

where δ_i is the urgency of order i and ct is the current time. For orders from first set, μ_i is the distance between the delivery location of order i and the current location of the driver. For orders from second set, μ_i is the distance between current location of the driver and the pickup location, plus the distance between the pickup location and the delivery location of order i.

According to the results from Wang et al. (2020), RETA performs slightly better than RU in terms of solution quality. To validate, we conduct experiments on two real delivery areas from Meituan food delivery platform. From the result in Appendix A, it can be found that although the solution quality of RU is slightly worse than RETA, the number of instances where RU performs better than RETA is larger. Moreover, for a large portion of instances, the performance of RETA is the same as RU. This result shows that RETA and RU are both effective for route construction and each of them is favorable for some special kinds of instances. Therefore, no single sequencing rule can cover all kinds of situations and they need to be used together to promise the quality of the route. A simple way is to execute the insertion-based heuristic twice with two different sequencing rules and select the best solution. However, although the quality of solution is kept in this way, the computational time is nearly doubled, which is not what we desire because for online food ordering and delivery, every second is valuable. This motivates us to introduce other methods to accelerate the insertion-based heuristic and to prevent the waste of computational time, while keeping good solution quality.

4.1.2. Acceleration strategy

During the greedy insertion procedure of route construction, an acceleration strategy based on geographic information can be used to avoid the “bad” insertion attempts, which was first proposed by Zheng et al. (2019). They claimed that in China, restaurants and customers were geographically close to each other. Therefore, pickup and delivery locations could be clustered by hierarchical clustering and with these clusters, the construction of a feasible route could be sped up. The clustering algorithm is briefly described as follows.

Given a clustering range R, for each location k, if it is not classified, it generates a new group and makes itself a center point. For each location l, if it not classified, it is classified into group k if $d_{k,l} < R$; if it is classified to group k' and is not a center point, then it is reclassified into group k if $d_{k,l} < d_{k',l}$.

According to the cluster tag of each location, two insertion strategy could be derived. To be concise, we simply list the lemmas and the corresponding conclusions here.

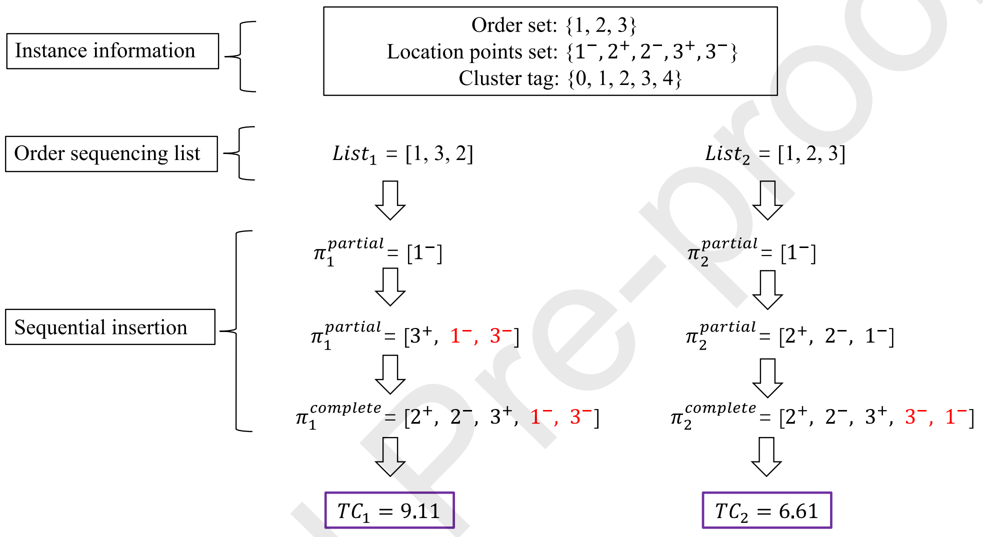
- **Lemma 1 (Insertion before own group).** If location l is classified to group k, then inserting location l to groups before group k is worse than inserting location l to group k.
- **Lemma 2 (Insertion after own group).** If location l is classified to group k, then there is always an insertion better than the insertion of location l to group k' after group k.

From Lemma 1 and Lemma 2, it is concluded that location l is only worth inserting when it is inserted into its own group, say group k, or between groups after group k. This strategy can reduce the searching space when inserting the candidate location points. Nevertheless, this acceleration strategy does not always promise the correctness of reduction. We will show in the following content that there are cases where the acceleration strategy excludes good positions and hence results in the inferiority of solutions.

4.1.3. Analysis of route construction

As we mentioned in Section 4.1.1, two sequencing rules, which are the RETA and RU, are adopted in the insertion-based heuristic to construct complete routes. From the comparison in Appendix A, it is shown that their performances are comparable. For most of the instances, RETA and RU reach the same result while for others, they show different performance regarding the total cost of route. Therefore, analyzing the route construction procedure so as to distinguish the performance of RETA and RU is of great importance. Through analyses of examples, we summarize two possible factors that cause the inferiority of a certain sequencing rule as follows.

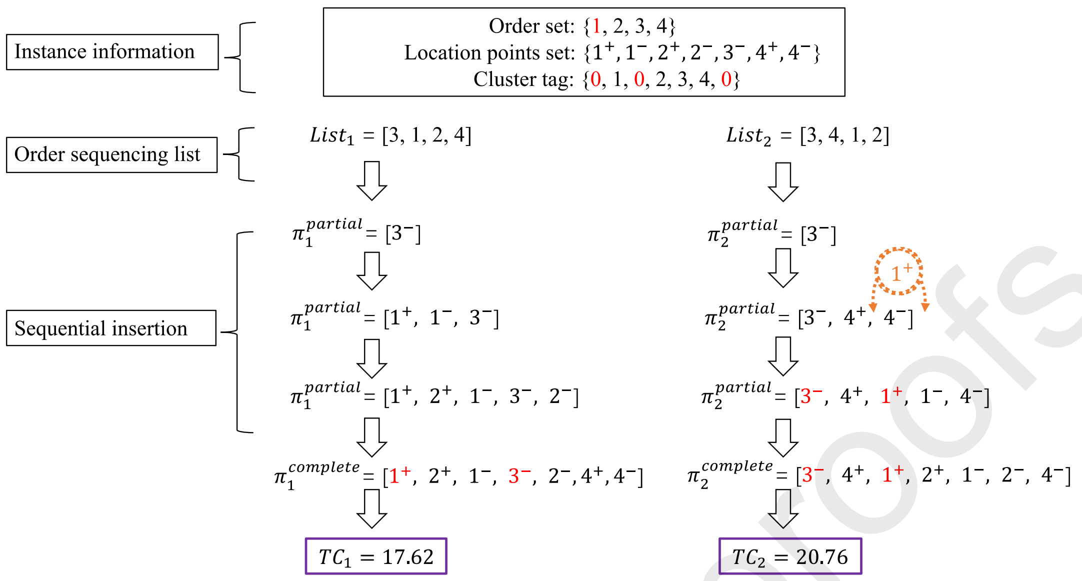
- **Influence caused by local optima.** During route construction, each candidate order will be inserted into the position that minimizes the total cost of current partial route. However, inserting the candidate order in this position is short-sighted, and cannot capture possibly useful information in the future. Consequently, the currently selected best position may not be the best position after the route is completely constructed due to the continuous insertion of subsequent orders. Fig. 5 shows an example of the route construction procedure that is influenced by local optima. Three orders are considered in the example. List_1 represents the order sequencing list generated by RETA and List_2 represents the order sequencing list generated by RU. π_1 and π_2 represent the permutation of route. In this example, RU performs better than RETA, with a total cost of 6.61. The major difference between two complete routes lies in the relative position of location 1^- and 3^-. From the result of RU, we can tell that putting location 3^- ahead of 1^- is the best way to achieve lower total cost. However, during the construction of π_1^{partial}, location 3^- is placed behind the location 1^- because it results in a lower total cost in terms of current partial solution, which causes the inferiority of the final route.



Instance information
Order set: {1, 2, 3}
Location points set: $\{1^-, 2^+, 2^-, 3^+, 3^-\}$
Cluster tag: {0, 1, 2, 3, 4}
Order sequencing list
$List_1$ = [1, 3, 2]
$List_2$ = [1, 2, 3]
Sequential insertion
$\pi_1^{partial} = [1^-]$
$\pi_2^{partial} = [1^-]$
$\pi_1^{partial} = [3^+, 1^-, 3^-]$
$\pi_2^{partial} = [2^+, 2^-, 1^-]$
$\pi_1^{complete} = [2^+, 2^-, 3^+, 1^-, 3^-]$
$\pi_2^{complete} = [2^+, 2^-, 3^+, 3^-, 1^-]$
$TC_1 = 9.11$
$TC_2 = 6.61$

Fig. 5. An example of route construction influenced by local optima

- **Influence caused by acceleration strategy.** The acceleration strategy considers the geographical information of locations but ignores the temporal information of orders. Therefore, some extremely urgent orders may not be inserted into the head of the route, which will probably result in the inferiority of solutions. An illustrative example can be found in Fig. 6. Four orders are involved, among which order 3 contains only a delivery location and order 1 is a very urgent one with small ETA value. Note that the cluster tags of location $1^+, 2^+$ and 4^- are all equal to zero. Therefore, these locations will possibly be influenced by acceleration strategy during insertion. List_1 is the order sequencing list generated by RETA and List_2 is the list generated by RU. For this example, RETA performs better than RU with total cost of 17.62. From the construction procedure of $\pi_1^{\mathrm{complete}}$ we can see that it is better to arrange location 1^+ ahead of location 3^-. However, when inserting location 1^+ into π_2^{partial}, two candidate positions are considered instead of four because we only insert locations into their own group or between groups after that group according to the acceleration strategy. Therefore location 1^+ is destined to be placed behind location 3^- in the final route π_2^{complete}, which is not a satisfactory arrangement.


Instance information
Order set: {1, 2, 3, 4}
Location points set: {1⁺, 1⁻, 2⁺, 2⁻, 3⁻, 4⁺, 4⁻}
Cluster tag: {0, 1, 0, 2, 3, 4, 0}
Order sequencing list
List₁ = [3, 1, 2, 4]
List₂ = [3, 4, 1, 2]
Sequential insertion
π₁^partial = [3⁻]
π₁^partial = [1⁺, 1⁻, 3⁻]
π₁^partial = [1⁺, 2⁺, 1⁻, 3⁻, 2⁻]
π₁^complete = [1⁺, 2⁺, 1⁻, 3⁻, 2⁻, 4⁺, 4⁻]
TC₁ = 17.62
π₂^partial = [3⁻]
π₂^partial = [3⁻, 4⁺, 4⁻]
1⁺
π₂^partial = [3⁻, 4⁺, 1⁺, 1⁻, 4⁻]
π₂^complete = [3⁻, 4⁺, 1⁺, 2⁺, 1⁻, 2⁻, 4⁻]
TC₂ = 20.76
Fig. 6. An example of route construction influenced by acceleration strategy

Since the performance of the sequencing rules is seriously impacted by these factors, it is difficult to predict the performance of different rules precisely. Nevertheless, there are still some useful information that can be extracted from the analysis to assist in estimating the performance of RETA and RU. For instance, when the cluster tag of every location is different from each other, each cluster will be different and every position can be inserted, as shown in Fig. 5. Another similar example is that all the locations have the same cluster tag or the number of locations with the same cluster tag is large. Under this situation, the portion of same clusters will be large and the effect of restriction from acceleration strategy will weaken. Besides, spatial and temporal information of orders can be utilized to characterize the instances and the sequencing rules. Inspired by these analyses, we design three types of problem-specific features in Section 4.2.3, which are used in XGBoost to help predict the classification result of sequencing rules.

4.2. Adaptive selection mechanism

In this section, we introduce the adaptive selection mechanism, which is proposed to determine the sequencing rules to use in priority list generation. As mentioned in Section 4.1.1, using both RETA and RU promises the solution quality but doubles the computational time of route construction. Considering that RETA and RU perform same for a large portion of instances (see Appendix A), there is a lot of useless search, which results in much waste of computational time. To avoid such waste and keep the solution quality, we use the following mechanism to select sequencing rules adaptively. If RETA performs equally as RU, which means that one of them is unnecessary, then we randomly select one from them; otherwise, we take both of them as valid rules for route construction and select the one with a lower objective function value.

To achieve the adaptive selection mechanism, we need to predict the performance of RETA and RU and decide whether they are the same. Therefore, a classification model is established in the way of supervised learning. Specifically, it is implemented with XGBoost algorithm, together with three types of problem-specific features. Details of them are elaborated in the following contents.

4.2.1. **Classification model**

The classification model used in this paper falls into the area of supervised learning, which is a machine learning technique often requiring the existence of ground-truth (also known as label) when training the model. It learns from the training data and extract latent knowledge, which is normally hidden in the combination of different parameters. To be more straightforward, the training process of the classification model forms an accurate mapping function to correlate the input variable and output variable. When new data (without label) enter the classification model, the already trained mapping function will help to classify the data and predict the corresponding category. In this paper, we utilize this technique to predict the performance of different sequencing rules.

Specifically, we take XGBoost as an efficient implementation of gradient boosted decision trees, which is an ensemble learning technique that uses a sequence of decision trees. XGBoost was first proposed by Chen and Guestrin (2016) and further utilized by many researchers (Dhaliwal, Nahid, & Abbas, 2018; Pan, 2018; Parsa, Movahedi, Taghipour, Derrible, & Mohammadian, 2020) due to the effectiveness of reducing the computing time and providing optimal use of memory resources.

XGBoost functions as a tree algorithm. The structure of a decision tree is similar to a real tree, containing root node (topmost node), internal nodes (conditional nodes), and leaf nodes (end nodes). Tree algorithms generally start from the root node, branch out through the internal nodes and finally end up at the leaf nodes. During the splitting up into branches and edges, the features of data (also called attributes) act as the conditions to determine whether the branching continues or not. At the end of the branch, the splitting is done, reaching a final classification result. Fig. 7 presents an explanation of how a decision tree works with a simple example of predicting whether a snake is venomous or not ("Positive" for venomous and "Negative" for non-venomous).

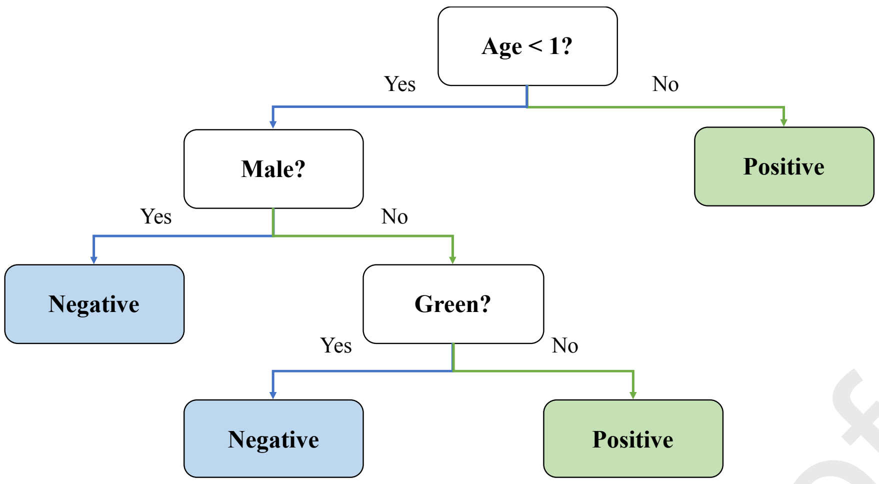


Fig. 7. Explanation of how a decision tree works

4.2.2. XGBoost algorithm

This section introduces the derivation of loss function for evaluating split candidates in practice. Given a dataset $\mathcal{D} = \{(z_j, y_j) | j = 1,2,\ldots,m, z_j \in \mathbb{R}^p, y_j \in \mathbb{R}\}$, there are m observations z_j and each of the observation is associated with p features. Moreover, a corresponding dependent variable y_j denotes the ground-truth of z_j. Let $\hat{y}_j$ be the result predicted by the tree ensemble model, which is calculated as follows,

$$\hat{y}_j = \phi(z_j) = \sum_{r=1}^{R} f_r(z_j),\ f_r \in F \tag{18}$$

f_r represents an independent tree structure with leaf scores. $f_r(z_j)$ represents the score generated by the r-th tree to the j-th observation. F is the space of trees. The objective function we aim to minimize consists of two parts: the training loss and the regularization.

$$\mathcal{L}(\phi) = \sum_j \ell(y_j, \hat{y}_j) + \sum_r \Omega(f_r) \tag{19}$$

In the above equation, ℓ is a loss function to measure the difference between the predictive value and the ground truth, which is implemented using logarithmic loss function in this study. Ω is a term for penalizing the complexity of the model in case of generating extreme parameters, which is calculated as follows,

$$\Omega(f) = \gamma L + \frac{1}{2}\lambda \|w\|^2 \tag{20}$$

where L is the number of leaves and w is the magnitude of leaf weight. γ and λ are parameters for controlling the degree of penalty. The regularization term contributes to generating smooth the final learnt weights and therefore avoids the over-fitting of data.

Since it is not a straightforward task to train all the trees in parallel (Dhaliwal et al., 2018), XGBoost tries to optimize the objective function "tree by tree". It adds a best tree f_v at step v, which is selected by minimizing the following objective function based on the current $v - 1$ trees.

$$\mathcal{L}^{v} = \sum_{j=1}^{m} \ell\left(y_j, \hat{y}_j^{(v-1)} + f_v(z_j)\right) + \Omega(f_v) \tag{21}$$

This formula could be simplified by the second-order approximation. Specifically, for a tree structure σ and the available observations of leaf a is denoted as $I_a = \{j|\sigma(z_j) = a\}$, the corresponding optimal objective is calculated.

$$\tilde{\mathcal{L}}^{(v)}(\sigma) = -\frac{1}{2}\sum_{a=1}^{L} \frac{\left(\sum_{j \in I_a} g_j\right)^2}{\sum_{j \in I_a} h_j + \lambda} + \gamma L \tag{22}$$

Considering that in practice, it is impossible to enumerate all the possible tree structures, the following loss reduction after the tree split is given,

$$\mathcal{L}_{split} = \frac{1}{2}\left[\frac{\left(\sum_{j \in I_{left}} g_j\right)^2}{\sum_{j \in I_{left}} h_j + \lambda} + \frac{\left(\sum_{j \in I_{right}} g_j\right)^2}{\sum_{j \in I_{right}} h_j + \lambda} - \frac{\left(\sum_{j \in I} g_j\right)^2}{\sum_{j \in I} h_j + \lambda}\right] - \gamma \tag{23}$$

where I is a subset of available observations in the current node and I_{left}, I_{right} are the subsets of available observations in the left and right nodes after the split, respectively (Pan, 2018). g_j and h_j represent the first and second order gradient values on the loss function.

$$g_j = \partial_{\hat{y}_j^{(v-1)}} \ell\left(y_j, \hat{y}_j^{(v-1)}\right) \tag{24}$$

$$h_j = \partial^2_{\hat{y}_j^{(v-1)}} \ell\left(y_j, \hat{y}_j^{(v-1)}\right) \tag{25}$$

Note that the procedure of splitting in XGBoost is different from that of an ordinary decision tree. The latter relies on the subsequent pruning and does not consider the complexity of the tree when splitting, while the former takes the complexity into account by parameter γ. Therefore, no extra pruning is used in XGBoost.

4.2.3. Feature extraction

The features of data affect the performance of the model to a great extent. Good features can help the model learn from the data more quickly and accurately. However, there is no universal pattern of feature extraction for every problem. The way to design features depends largely on the analysis of the problem and the knowledge and experience of developers. During the extraction of features, one must evaluate whether the features are closely related to the objective function and consider the interpretability of different features. In this section, we design three types of features based on the analysis of route construction procedure in Section 4.1.3. Features and their meanings are listed in Appendix B. According to aspects the features are related to, we divide the features into three categories, which are the order, route and cluster. To indicate the types of features, the names of features start with "order", "route" or "cluster". By using the suffix of "sta", we refer to the statistics of the corresponding feature, including summation, minimum, maximum, average and standard deviation.

The order features mainly focus on the attributes of orders and try to measure them in terms of time, space and other indices. Since the sequencing rules works directly with the orders, this kind of features have a close relationship to the priority list generated by different sequencing rules. Note that the calculation of most features involves ETA of orders and current time, which are also utilized by RETA and RU, such as the "order_remaintime_sta", "order_level_sta" and "order_urgency_sta". Note that "order_level_sta" is a set of features that reflect the difficulty of serving all the orders. The level of an order i is defined as the sum of pickup level $\mathrm{PL_i}$ and delivery level $\mathrm{DL_i}$, which is calculated as follows,

$$\mathrm{Level_i} = \mathrm{PL_i} + \mathrm{DL_i} \tag{26}$$

$$\mathrm{PL_i} = \frac{\mathrm{EPT_i} - \mathrm{EPT_{min}}}{\mathrm{EPT_{max}} - \mathrm{EPT_{min}}} \tag{27}$$

$$\mathrm{DL_i} = \frac{\mathrm{ETA_i} - \mathrm{ETA_{min}}}{\mathrm{ETA_{max}} - \mathrm{ETA_{min}}} \tag{28}$$

where $\mathrm{EPT_{min}}$ is the minimal earliest pickup time among all orders and $\mathrm{EPT_{max}}$ is the maximal earliest pickup time among all orders. For orders that are already fetched, the pickup level equals zero. $\mathrm{ETA_{min}}$ is the minimal estimated time of arrival among all orders and $\mathrm{ETA_{max}}$ is the maximal estimated time of arrival among all orders. Besides, information from different kinds of time that are related to orders are used to form temporal features. To be specific, the creation time of order refers to the moment when the order is created by the dispatching platform. The order time refers to the moment when the restaurant checks the order and confirms the activeness of the order.

The route features use the information of location points in the route and characterize instances from the perspective of graph. The center refers to the average center of locations. Through the Euclidian distances between pickup locations, delivery locations and driver's location, the degree of aggregation of these location points is quantified. Therefore, the structural information of possible routes is extracted to assist in predicting the performance of route construction.

The initial motivation of designing cluster features is similar to that of route features, which is to extract some information from the physical structure of location points. The difference is that the cluster features are related to the acceleration strategy. According to the analysis of acceleration strategy in Section 4.1.3, the cluster information of location points will affect the insertion procedure of orders, and further influence the quality of route. Accordingly, we design some relevant features to reflect the influence of cluster results. For example, "cluster_num" refers to the sum of all different cluster tags and "cluster_same_portion" refers to the portion of location points that have the same cluster tag. By designing these cluster features, we provide more information about the instances for XGBoost to help make the final decision.

5. Computational experiments

5.1. Experimental settings

To test the performance of the proposed XGB-FCA, we generate two kinds of datasets from Meituan food delivery platform, each of which consists of real route planning tasks. The first contains citywide route planning tasks, where we select Beijing as the representative one. The second collects nationwide route planning tasks, and thus orders and drivers from different cities will occur in this dataset, which makes it more complicated than the first one. The number of location points in two datasets ranges from 6 to 25. Before entering XGBoost, the training and test datasets are checked for abnormal values. If any, they will be eliminated to guarantee the justifiability of data. The details of two datasets are shown in Table 2.

The route construction heuristic is coded in Java and run on a MacBook Pro with 2.2 GHz processors and 16 GB of RAM under Mac OS. The training and test of XGBoost are conducted on servers in Meituan.

Table 2. Details of datasets

Dataset	Type	Number of entries	Ratio of positive and negative entries
Citywide	Train	17306287	1:1
	Test	1260000	1:1
Nationwide	Train	51718808	1:1
	Test	5841573	1:1

5.2. Model evaluation metrics

To evaluate the performance of the classification model, we introduce the confusion matrix and several indices, such as Precision, Recall (also known as True Positive Rate, TPR), False Positive Rate (FPR), ROC curve and PR curve, which are commonly used in the field of machine learning. Fig. 8 shows the structure of the confusion matrix, which is mainly composed of four parts, i.e. True Positive (TP), True Negative (TN), False Positive (FP) and False Negative (FN). Here, Positive refers to that the performance of RETA and RU is different, labelled as 1, and Negative means RETA and RU result in the same total cost of the route, labelled as 0. Under this setting, TP means that a known positive value is predicted as positive, i.e., the performance of RETA and RU is different and it is also predicted as different. Correspondingly, TN means a known negative value is predicted as negative. Similarly, FP means a known negative value is predicted as positive and FN means a known positive value is predicted as negative. In other words, TP and TN are both correct predictions while FP and FN are incorrect predictions.

	Predicted Positive (1)	**Predicted Negative (0)**
Actual Positive (1)	True Positive (TP)	False Negative (FN)
Actual Negative (0)	False Positive (FP)	True Negative (TN)

Fig. 8. Structure of confusion matrix

Given a confusion matrix defined as before, we can derive the calculation of Precision, Recall (TPR) and FPR as follows.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{29}$$

$$\text{Recall (TPR)} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{30}$$

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}} \tag{31}$$

Here, Precision reflects the portion of True Positives among all entries that are predicted as Positive. Recall (TPR) reflects the portion of True Positives among all actual Positive entries, describing the probability of correct prediction of actual Positive values. FPR describes the probability of wrong prediction of actual Negative values.

With TPR and FPR, we can obtain ROC curve by plotting the TPR against the FPR. It can be concluded from the definition of TPR and FPR that the larger the TPR and the smaller the FPR, the more accurate and efficient the model and algorithm. In other words, we expect the curve to be close to the left-hand and top-hand borders as much as possible, and hence, the area under the ROC curve, denoted as AUC, is expected to be close to 1. Similarly, we can get the PR curve by plotting the Precision against the Recall. The PR curve is supposed to be close to the right-hand and top-hand borders and the area under the PR curve, denoted as AUPR, is also expected to be as large as possible. AUC and AUPR are two important metrics when evaluating the performance of the classification model.

5.3. Parameter settings

To use XGBoost, there are many parameters and one is supposed to determine the specific parameter settings in order to perform machine learning tasks effectively. In our experiments, we concentrate on the complexity of the model and adjust the following three parameters.

The "max_depth" parameter is used to control the depth of a tree. The bigger the value is, the more complex the algorithm becomes.

The “min_child_weight” parameter takes positive integer values and sets a threshold on the sum of observation weights in a leaf node after splitting. If the sum of weights is larger than the threshold, the leaf node is worth further splitting. The algorithm becomes more conservative as this parameter increases.

The “learning_rate” parameter performs the step shrinkage so as to provide more space for further improvement. Appropriate setting of this parameter helps prevent the overfitting problems.

For other parameters of XGBoost, we use the default setting as follows. The “objective” is “binary: logistic”. “sub_sample” is set as 1. “colsample_bytree” is 1. “lambda” and “alpha” are set as 1 and 0, respectively.

Table 3 presents the parameters to be adjusted and their corresponding ranges. The “max_depth” ranges from 3 to 9 for citywide dataset while for the nationwide dataset, it ranges from 5 to 11. The range of “min_child_weight” and “learning_rate” are the same for both datasets. To find the best parameters, a commonly used method is to start from the parameter that is least related to others and tune the parameters one by one. Here we choose the “max_depth” to be firstly adjusted, and then the “min_child_weight”, finally the “learning_rate”. The model is run to calculate the best value of AUC and AUPR and determine the best combination of parameters. Since citywide and nationwide datasets are different in complexity, we conduct experiments to adjust the parameters of XGBoost on citywide and nationwide datasets separately. Fig. 9 and 10 show the results achieved. From the results we can see that for citywide dataset, the best AUC and AUPR are achieved when “max_depth”, “min_child_weight” and “learning_rate” are set as 7, 1, 0.1, respectively. Similarly, for nationwide dataset, the corresponding parameters are set as 9, 1, 0.1, respectively. The higher the AUC and AUPR, the more powerful the model is in predicting the correct classification result.

Table 3. Parameter values

Parameter	Value			
max_depth	3 (5)	5 (7)	7 (9)	9 (11)
min_child_weight	1	2	3	4
learning_rate	0.05	0.1	0.2	0.3

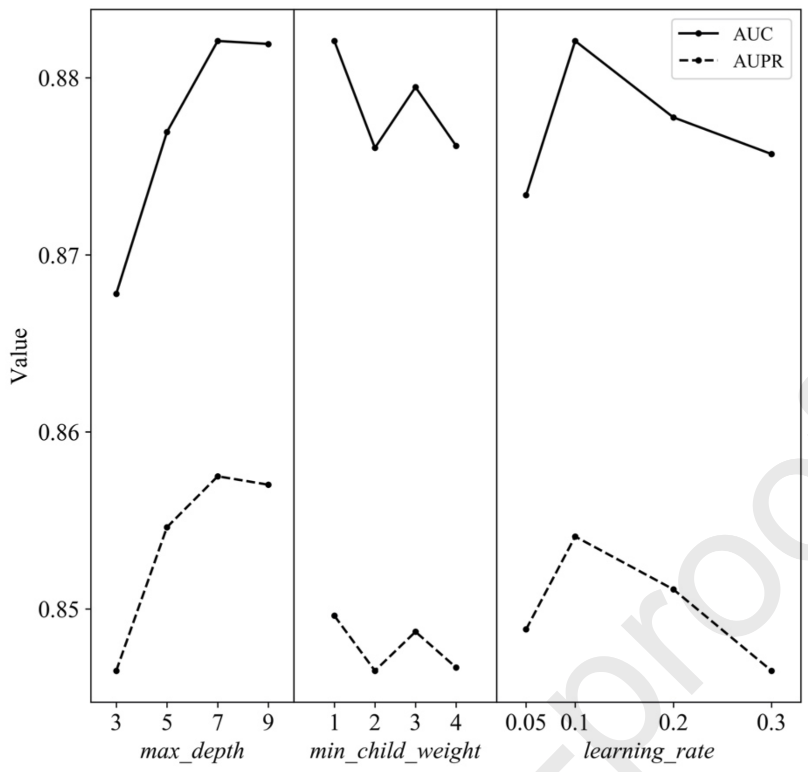


Fig. 9. Parameter adjustment on citywide dataset

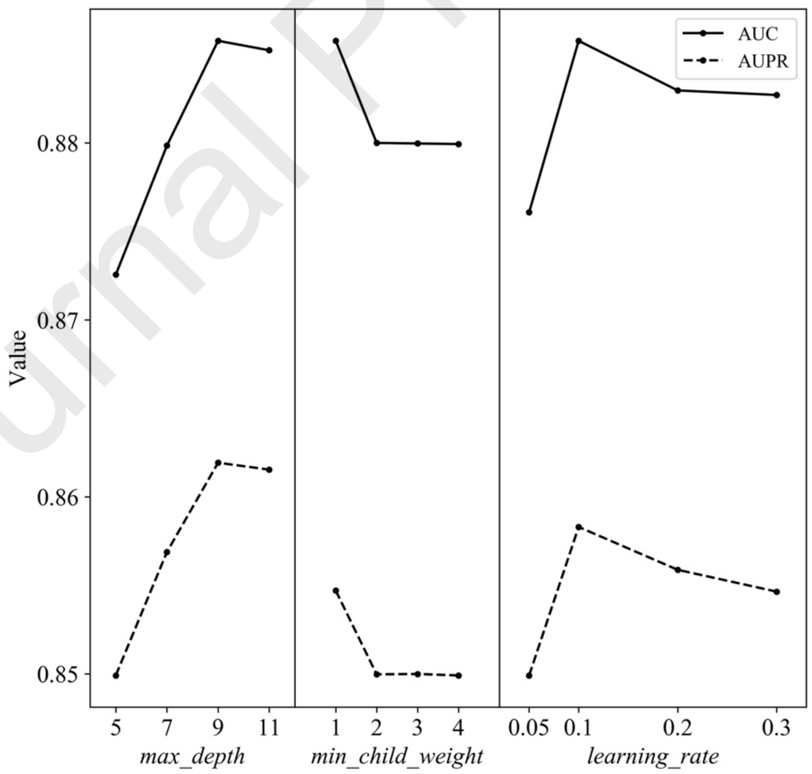


Fig. 10. Parameter adjustment on nationwide dataset

5.4. Comparative results

The performance of XGBoost under different datasets is summarized in Table 4 (using tuned parameters mentioned in Section 5.3). It can be found that the AUC and AUPR of the classification model reach a satisfying level on both datasets, which means that the classification model performs well, and is also robust enough.

Table 4. Performance of XGBoost

Dataset	Precision	Recall	AUC	AUPR
Citywide	82.57%	76.02%	0.882	0.857
Nationwide	82.74%	76.18%	0.886	0.862

To validate the effectiveness of using XGBoost in solving FDRPP, we compare XGB-FCA with several heuristics and analyze the results in terms of the total cost of route and the consumed computational time. The first heuristic is to execute the insertion-based heuristic twice with RETA and RU, as mentioned in Section 4.1.1, and the solution with smaller total cost is selected as the final solution. The second is using the RETA as the only sequencing rule for route construction. Similarly, the third one is using the RU as the only sequencing rule. Moreover, it is our concern that whether XGBoost performs better than a random strategy, therefore the fourth heuristic is to randomly select a sequencing rule from RETA and RU. Finally, we provide the worst performance of two sequencing rules to show the improvement of each heuristic explicitly. To be of convenience, the compared heuristics mentioned above are denoted as Both_heu, RETA_heu, RU_heu, Rand_heu and Worst_heu, from the first to the last. It is obvious that Both_heu achieves the smallest total cost of route but consumes the largest computational time. Therefore, we take the performance of Both_heu as the benchmark and calculate the relative percentage deviation (RPD) of a certain heuristic heu over Both_heu as follows.

$$\mathrm{RPD} = \frac{\mathrm{value_{heu}} - \mathrm{value_{Both\_heu}}}{\mathrm{value_{Both\_heu}}} \times 100 \tag{32}$$

Here $\mathrm{value_{heu}}$, $\mathrm{value_{Both\_heu}}$ are the objective function value of heu and Both_heu, respectively. The smaller the RPD, the better the solution quality. The RPDs of TC, TD and DIS are all listed in results. The computational time (CT) is compared by dividing the computational time of heu by that of Both_heu.

Table 5 presents the comparative result under citywide dataset. It can be seen that RETA_heu, RU_heu and Rand_heu all consume approximately half the time of Both_heu. However, the total costs of them reach 4.84%, 4.45% and 4.56%, respectively. Moreover, the time delays of them are especially large. It is unacceptable to apply methods with such a large deviation of total cost in real dispatching platform since the efficiency of dispatching systems and the satisfaction of customers will be affected. On the contrary, XGB-FCA promises the quality of route with only 0.48% deviation of total cost, while the computation time is reduced to 76.98% of the benchmark. Similar results and conclusions can be summarized from Table 6 for nationwide dataset. Besides, comparing Table 5 and 6, we can conclude that the deviation of total cost is nearly doubled on nationwide dataset for all heuristics except XGB-

FCA, which demonstrates the effectiveness and robustness of our proposed algorithm.

Table 5. Comparison of different algorithms on citywide dataset

Algorithm	RPD			CT
	TC	TD	DIS	
Both_heu (Benchmark)	(0.00)	(0.00)	(0.00)	(100%)
RETA_heu	4.84	6.61	1.69	51.36%
RU_heu	4.45	6.62	0.59	52.49%
Rand_heu	4.56	6.49	1.14	51.95%
Worst_heu	9.29	13.22	2.27	100%
XGB-FCA	**0.48**	**0.67**	**0.16**	76.98%

Table 6. Comparison of different algorithms on nationwide dataset

Algorithm	RPD			CT
	TC	TD	DIS	
Both_heu (Benchmark)	(0.00)	(0.00)	(0.00)	(100%)
RETA_heu	9.40	10.81	1.86	52.39%
RU_heu	8.34	9.80	0.52	53.67%
Rand_heu	8.87	10.30	1.19	52.84%
Worst_heu	17.75	20.61	2.38	100%
XGB-FCA	**0.60**	**0.68**	**0.16**	76.95%

5.5. Feature importance

Another concern of the classification model lies in the effectiveness of features we design. Using an in-built function of XGBoost, we plot the importance of features (F-score) in Fig. 11 and 12, which represents the citywide result and nationwide result, respectively. From Fig. 11 we can tell that the top five important features on citywide dataset are the “route_pickpoint_portion”, “cluster_eta_minus_ct_same_sta”, “order_remaintime_sta”, “cluster_same_portion” and “order_level_sta”, most of which are the order features and cluster features. As for nationwide result in Fig. 12, route features are rising in the feature importance list, such as “route_delivery_center_dis_sta” and “route_pick_current_dis_sta”. It can be concluded from Fig. 11 and 12 that our designed features are of great importance in the branching process of XGBoost, which shows the necessity of feature extraction.

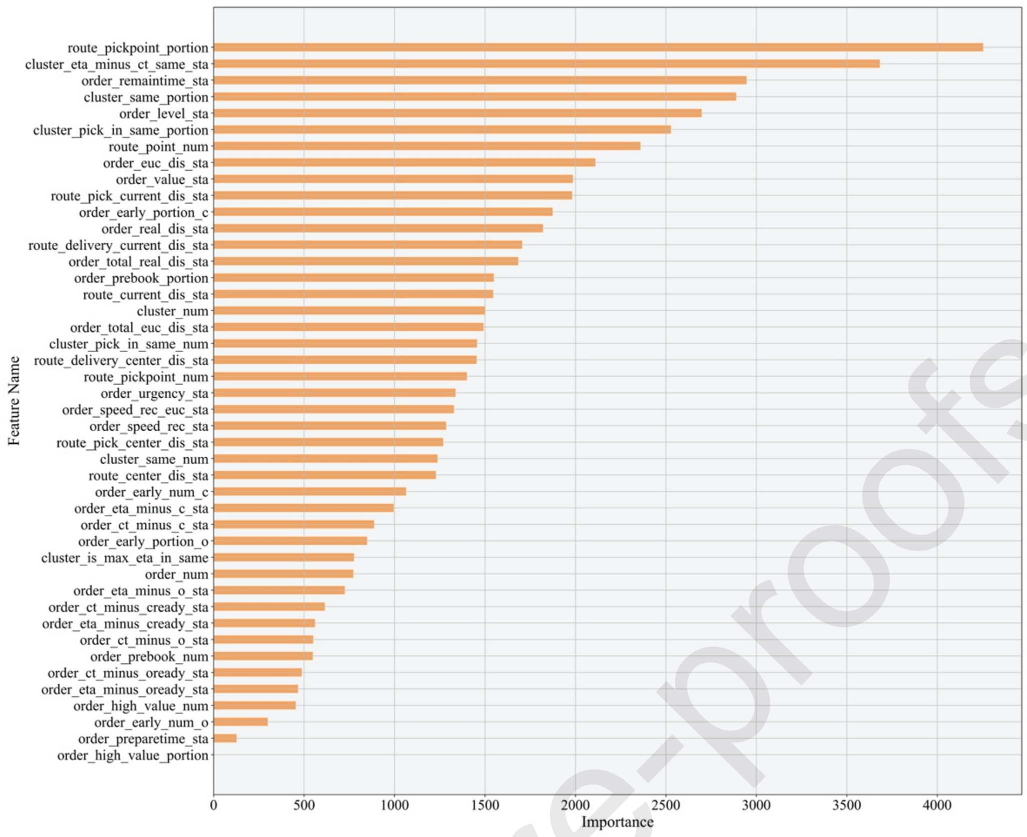


Fig. 11. Feature importance on citywide dataset

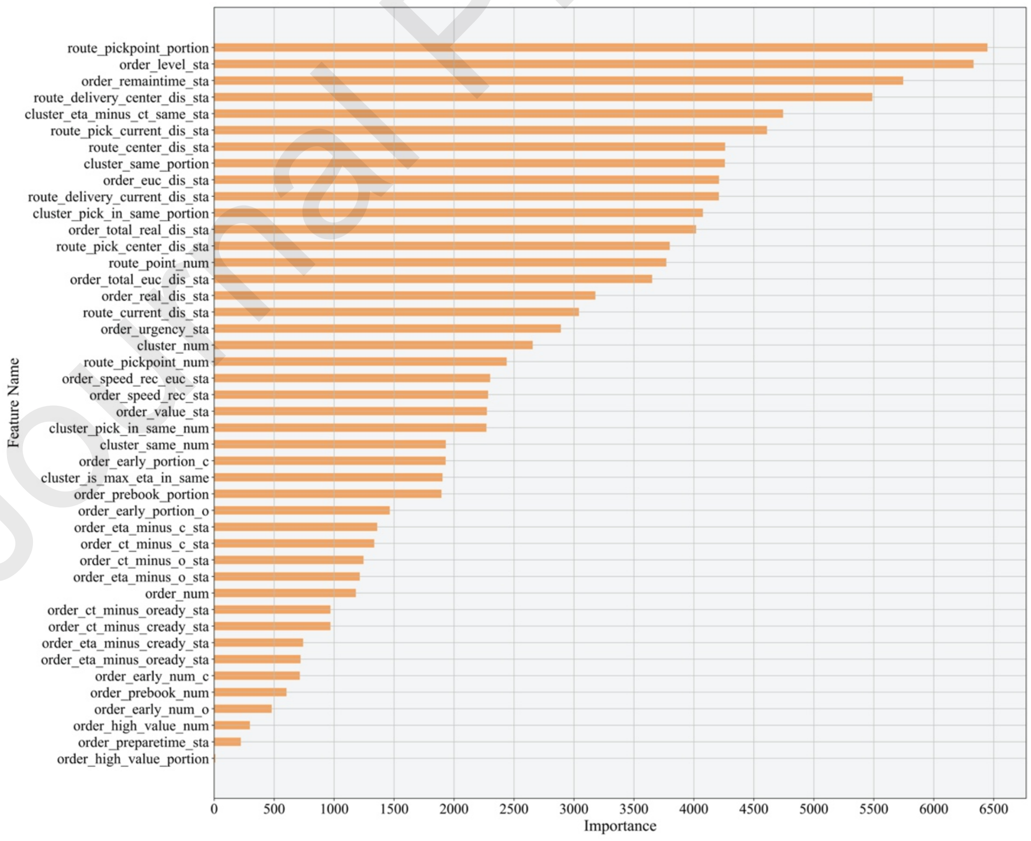


Fig. 12. Feature importance on nationwide dataset

5.6. Further investigation on the performance of XGB-FCA

To further investigate performance of XGB-FCA and examine whether the ratio of positive and negative entries of the dataset affects the performance of XGBoost, we variate the ratio of positive and negative entries and depict the changes of AUC, AUPR, RPD and CT in Fig. 13. From Fig. 13 (a) it can be seen that AUC fluctuates very little under different ratios. This indicates that our model can be utilized under extreme scenarios in real dispatching systems, which further demonstrates the robustness of XGBoost. Fig. 13 (b) shows that AUPR decreases as the number of negative entries increases, which is a normal phenomenon since TP is naturally less than FP under dataset with large portion of negative entries. From Fig. 13 (c) (d) it can be told that the solution quality and the consumed time become better as the number of negative entries increases. The consumed time can be at most reduced to close to 64% of the Both_heu heuristic. In a word, the classification model we developed in this paper is insensitive to the fluctuation of positive and negative entry ratio. The XGB-FCA is able to provide satisfactory solutions under multiple scenarios, considering the total cost of the route and the computational time simultaneously.

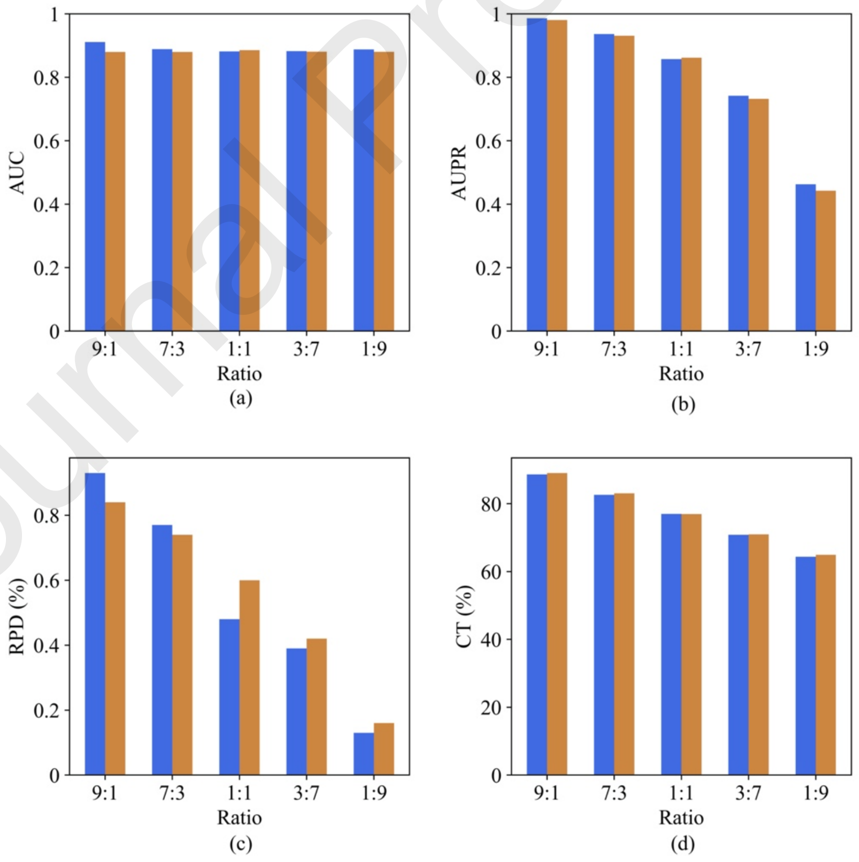


Fig. 13. Performance of XGB-FCA on different ratios of positive and negative entries (Blue bar for citywide result and brown bar for nationwide result)

6. Conclusions

In this paper, we propose an XGBoost-enhanced fast constructive algorithm to solve the food delivery route planning problem. Experimental results on datasets from Meituan food delivery platform demonstrate that large amounts of computational time can be saved by the XGB-FCA, while keeping the solution quality. The effectiveness of the XGB-FCA mainly owes to the following aspects. Firstly, the adopted insertion-based heuristic can generate satisfactory solutions and the acceleration strategy can speed up the insertion procedure. Secondly, the adaptive selection mechanism is able to avoid the waste of computational time by predicting performance of sequencing rules using XGBoost. Finally, the specially designed features are effective in improving the performance of XGBoost.

Since problem-specific features are designed to improve the performance of XGBoost in the proposed XGB-FCA for solving FDRPP, there may be difficulties to directly apply it for solving other optimization problems. Nevertheless, the idea of combining machine learning technology and traditional optimization methods is of great significance for designing algorithms to solve other complex optimization problems effectively and efficiently.

Our future work is to develop more efficient methods and advanced technologies, such as reinforcement learning to solve the food delivery route planning problem with other realistic constraints. Moreover, it is also important to study order assignment problem and the routing problem simultaneously. Finally, considering the uncertainty of food delivery platforms, such as the travel time uncertainty (Wu, Hifi, & Bederina, 2017), delay uncertainty (Wu & Hifi, 2020) and the inconsistency between recommended routes and actual routes of drivers, is of our study interest as well.

Appendix A. Performance of RU and RETA

RU and RETA are compared on real instances from Meituan food delivery platform. The results are shown in Table A.1, where two real different delivery areas are involved. "Total_cnt" represents the total number of routes in the area. "RETA_cnt" represents the number of routes where RETA performs better than RU. "RU_cnt" represents the number of routes where RU performs better than RETA. "Same_cnt" represents the number of routes where RETA and RU have the same performance. "RETA_gap" is the gap of RETA to the best sequencing rule (between RU and RETA) in terms of the objective function and "RU_gap" is the performance gap of RU. The smaller the gap, the better the solution quality.

Table A.1 Performance of RU and RETA

Area	Total_cnt	RETA_cnt	RU_cnt	Same_cnt	RETA_gap	RU_gap
Area_1	45236	5451	7924	31861	3.50%	3.50%
Area_2	19231	1821	3337	14073	3.51%	3.75%

Appendix B. Features of data

Table B.1 Features of data

Category	Name of Feature	Meaning of Feature
Order	order_num	Total number of orders
	order_prebook_num	Number of prebooked orders
	order_prebook_portion	Portion of prebooked orders
	order_early_num_c	Number of orders whose earliest pickup time (based on the creation time) is earlier than current time
	order_early_portion_c	Portion of orders whose earliest pickup time (based on the creation time) is earlier than current time
	order_early_num_o	Number of orders whose earliest pickup time (based on the order time) is earlier than current time
	order_early_portion_o	Portion of orders whose earliest pickup time (based on the order time) is earlier than current time
	order_preparetime_sta	Statistics of the predicted preparation duration of each order
	order_ct_minus_c_sta	Statistics of difference between current time and the creation time of each order
	order_eta_minus_c_sta	Statistics of difference between ETA and the creation time of each order
	order_ct_minus_cready_sta	Statistics of difference between current time and the earliest pickup time (based on the creation time) of each order
	order_eta_minus_cready_sta	Statistics of difference between ETA and the earliest pickup time (based on the creation time) of each order
	order_ct_minus_o_sta	Statistics of difference between current time and the order time of each order
	order_eta_minus_o_sta	Statistics of difference between ETA and the order time of each order

	order_ct_minus_oready_sta	Statistics of difference between current time and the earliest pickup time (based on the order time) of each order
	order_eta_minus_oready_sta	Statistics of difference between ETA and the earliest pickup time (based on the order time) of each order
	order_remaintime_sta	Statistics of remaining time of each order, which is the difference of ETA and current time
	order_value_sta	Statistics of the price of each order
	order_high_value_num	Number of orders whose price is larger than the threshold (100)
	order_high_value_portion	Portion of orders whose price is larger than the threshold
	order_real_dis_sta	Statistics of the real delivery distance (navigated by Global Positioning System) of each order
	order_euc_dis_sta	Statistics of the Euclidean delivery distance of each order
	order_total_real_dis_sta	Statistics of the total real delivery distance of each order, which consists of the Euclidean distance between the location of driver and the pickup location of the order, plus the real delivery distance of the order.
	order_total_euc_dis_sta	Statistics of the total Euclidean delivery distance of each order, which consists of the Euclidean distance between the location of driver and the pickup location of the order, plus the Euclidean delivery distance of the order.
	order_level_sta	Statistics of the level of each order
	order_speed_rec_sta	Statistics of the reciprocal of the speed of each order, which is the quotient of the remaining time divided by the total real delivery distance of each order
	order_speed_rec_euc_sta	Statistics of the reciprocal of the speed of each order, which is the quotient of the remaining time divided by the total Euclidean delivery distance of each order
	order_urgency_sta	Statistics of urgency of each order
Route	route_point_num	Number of all the location points
	route_pickpoint_num	Number of all the pickup location points
	route_pickpoint_portion	Portion of all the pickup location points
	route_center_dis_sta	Statistics of the Euclidean distance between each location point and the location center
	route_pick_center_dis_sta	Statistics of the Euclidean distance between each pickup location point and the pickup center
	route_delivery_center_dis_sta	Statistics of the Euclidean distance between each delivery location point and the delivery center
	route_current_dis_sta	Statistics of the Euclidean distance between each location point and the current location of the driver
	route_pick_current_dis_sta	Statistics of the Euclidean distance between each pickup location point and the current location of the driver
	route_delivery_current_dis_sta	Statistics of the Euclidean distance between each delivery location point and the current location of the driver

Cluster	cluster_num	Number of different clusters
	cluster_same_num	Number of location points that have the same cluster tag
	cluster_same_portion	Portion of location points that have the same cluster tag
	cluster_pick_in_same_num	Number of pickup location points that have the same cluster tag
	cluster_pick_in_same_portion	Portion of pickup location points that have the same cluster tag
	cluster_is_max_eta_in_same	It toggles 1 if the delivery location point of the order that has the maximal ETA is included in the location points that have the same cluster tag
	cluster_eta_minus_ct_same_sta	Statistics of the difference between ETA and current time of each order whose location points are included in the location points that have the same cluster tag

References

Bello, I., Pham, H., Le, Q. V., Norouzi, M., & Bengio, S. (2017). Neural combinatorial optimization with reinforcement learning. ArXiv:1611.09940 [Cs, Stat]. http://arxiv.org/abs/1611.09940.

Bengio, Y., Lodi, A., & Prouvost, A. (2020). Machine learning for combinatorial optimization: A methodological tour d'horizon. ArXiv:1811.06128 [Cs, Stat]. http://arxiv.org/abs/1811.06128.

Carrabs, F., Cordeau, J.-F., & Laporte, G. (2007). Variable neighborhood search for the pickup and delivery traveling salesman problem with LIFO loading. INFORMS Journal on Computing, 19(4), 618–632. https://doi.org/10.1287/ijoc.1060.0202.

Chen, T., & Guestrin, C. (2016, August). XGBoost: A scalable tree boosting system. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 785–794. San Francisco, CA, US. https://doi.org/10.1145/2939672.2939785.

Cooray, P. L. N. U., & Rupasinghe, T. D. (2017). Machine learning-based parameter tuned genetic algorithm for energy minimizing vehicle routing problem. Journal of Industrial Engineering, 1–13. https://doi.org/10.1155/2017/3019523.

Cordeau, J.-F., Laporte, G., & Ropke, S. (2008). Recent models and algorithms for one-to-one pickup and delivery problems. In The Vehicle Routing Problem: Latest Advances and New Challenges, 327–357. Boston, MA, US. https://doi.org/10.1007/978-0-387-77778-8_15.

Cordeau, J.-F., Iori, M., Laporte, G., & Salazar González, J. J. (2010). A branch-and-cut algorithm for the pickup and delivery traveling salesman problem with LIFO loading. Networks, 55(1), 46–59. https://doi.org/10.1002/net.20312.

Dhaliwal, S., Nahid, A.-A., & Abbas, R. (2018). Effective intrusion detection system using XGBoost. Information, 9(7), 149. https://doi.org/10.3390/info9070149.

Edelkamp, S., & Gath, M. (2014). Solving single vehicle pickup and delivery problems with time windows and capacity constraints using nested monte-carlo search. Proceedings of the 6th International Conference on Agents and Artificial Intelligence, 22–33. Loire Valley, FR. https://doi.org/10.5220/0004722300220033.

Hirschberg, C., Rajko, A., Schumacher, T., & Wrulich, M. (2016). The changing market for food delivery. https://www.mckinsey.com/industries/high-tech/our-insights/the-changing-market-for-food-delivery/ Accessed 5 September 2020.

Hosny, M. I., & Mumford, C. L. (2007). Single vehicle pickup and delivery with time windows: Made to measure genetic encoding and operators. Proceedings of the 9th Annual Conference Companion on Genetic and Evolutionary computation, 2489-2496. London, UK. https://doi.org /10.1145/1274000.1274015.

Hosny, M. I., & Mumford, C. L. (2010). The single vehicle pickup and delivery problem with time windows: Intelligent operators for heuristic and metaheuristic algorithms. Journal of Heuristics, 16(3), 417–439. https://doi.org/10.1007/s10732-008-9083-1.

Huang, Y.-H., & Ting, C.-K. (2010). Ant colony optimization for the single vehicle pickup and delivery problem with time window. 2010 International Conference on Technologies and Applications of Artificial Intelligence, 537–543. Hsinchu City, TW, CHN. https://doi.org/10.1109/TAAI.2010.90.

Jih, W. -R., Kao, C. -Y., & Hsu, J. Y. -J. (2002). Using family competition genetic algorithm in pickup and delivery problem with time window constraints. Proceedings of the IEEE Internatinal Symposium on Intelligent Control, 496–501. Vancouver, BC, CA. https://doi.org/10.1109/ISIC.2002.1157813.

Kalantari, B., Hill, A. V., & Arora, S. R. (1985). An algorithm for the traveling salesman problem with pickup and delivery customers. European Journal of Operational Research, 22(3), 377–386. https://doi.org/10.1016/0377-2217(85)90257-7.

Kerivin, H. L. M., Lacroix, M., & Mahjoub, A. R. (2012). Models for the single-vehicle preemptive pickup and delivery problem. Journal of Combinatorial Optimization, 23(2), 196–223. https://doi.org/10.1007/s10878-010-9349-z.

Landrieu, A., Mati, Y., & Binder, Z. (2001). A tabu search heuristic for the single vehicle pickup and delivery problem with time windows. Journal of Intelligent Manufacturing, 12(5/6), 497–508. https://doi.org/10.1023/A:1012204504849.

Little, J. D. C., Murty, K. G., Sweeney, D. W., & Karel, C. (1963). An algorithm for the traveling salesman problem. Operations Research, 11(6), 972–989.

Lokin, F. C. J. (1979). Procedures for travelling salesman problems with additional constraints. European Journal of Operational Research, 3(2), 135–141. https://doi.org/10.1016/0377-2217(79) 900997.

Mao, C., & Shen, Z. (2018). A reinforcement learning framework for the adaptive routing problem in stochastic time-dependent network. Transportation Research Part C: Emerging Technologies, 93, 179–197. https://doi.org/10.1016/j.trc.2018.06.001.

Martinovic, G., Aleksi, I., & Baumgartner, A. (2008). Single-commodity vehicle routing problem with pickup and delivery service. Mathematical Problems in Engineering, 1–17. https://doi.org/10.11 55/2008/697981.

Meituan, & CFLP. (2019). Report on the development of Chinese immediate delivery business in 2019. http://pdf.dfcfw.com/pdf/H3_AP202006011381522218_1.pdf. Accessed 4 October 2020.

Morgan Stanley Research. (2017, June). Is online food delivery about to get ’amazoned’? https://www.morganstanley.com/ideas/online-food-delivery-market-expands/ Accessed 18 May 2020.

Nazari, M., Oroojlooy, A., Snyder, L., & Takac, M. (2018). Reinforcement learning for solving the vehicle routing problem. Advances in Neural Information Processing Systems, 9839–9849.

Pan, B. (2018). Application of XGBoost algorithm in hourly PM2.5 concentration prediction. IOP Conference Series: Earth and Environmental Science, 113, 012127. https://doi.org/10.108 8/1755-1315/113/1/012127.

Parragh, S. N., Doerner, K. F., & Hartl, R. F. (2008). A survey on pickup and delivery problems: Part II: Transportation between pickup and delivery locations. Journal Für Betriebswirtschaft, 58(2), 81–117. https://doi.org/10.1007/s11301-008-0036-4.

Parsa, A. B., Movahedi, A., Taghipour, H., Derrible, S., & Mohammadian, A. (Kouros). (2020). Toward safer highways, application of XGBoost and SHAP for real-time accident detection and feature analysis. Accident Analysis & Prevention, 136, 105405. https://doi.org/10.1016/j.aap.2019.105 405.

Psaraftis, H. N. (2011). A multi-commodity, capacitated pickup and delivery problem: The single and two-vehicle cases. European Journal of Operational Research, 215(3), 572–580. https://doi.org/10.1016/j.ejor.2011.06.038.

Renaud, J., Boctor, F. F., & Laporte, G. (2002). Perturbation heuristics for the pickup and delivery traveling salesman problem. Computers & Operations Research, 29(9), 1129–1141. https://doi.org/10.1016/S0305-0548(00)00109-X.

Renaud, J., Boctor, F. F., & Ouenniche, J. (2000). A heuristic for the pickup and delivery traveling salesman problem. Computers & Operations Research, 27(9), 905–916. https://doi.or g/10.1016/S0305-0548(99)00066-0.

Van der Bruggen, L. J. J., Lenstra, J. K., & Schuur, P. C. (1993). Variable-depth search for the single-vehicle pickup and delivery problem with time windows. Transportation Science, 27(3), 298–311. https://doi.org/10.1287/trsc.27.3.298.

Wang, X., Wang, S., Wang, L., Zheng, H., Hao, J., He, R., & Sun, Z. (2020). An effective iterated greedy algorithm for online route planning problem. 2020 IEEE Congress on Evolutionary Computation (CEC), 1–8. Glasgow, UK. https://doi.org/10.1109/CEC48606.2020.9185864.

Wu, L., & Hifi, M. (2020). Discrete scenario-based optimization for the robust vehicle routing problem: The case of time windows under delay uncertainty. Computers & Industrial Engineering, 145, 106491. https://doi.org/10.1016/j.cie.2020.106491.

Wu, L., Hifi, M., & Bederina, H. (2017). A new robust criterion for the vehicle routing problem with uncertain travel time. Computers & Industrial Engineering, 112, 607–615. https://doi.org/10.1016/j.cie.2017.05.029.

Yu, J. J. Q., Yu, W., & Gu, J. (2019). Online vehicle routing with neural combinatorial optimization and deep reinforcement learning. IEEE Transactions on Intelligent Transportation Systems, 20(10), 3806–3817. https://doi.org/10.1109/TITS.2019.2909109.

Zheng, H., Wang, S., Cha, Y., Guo, F., Hao, J., & Sun, Z. (2019). A two-stage fast heuristic for food delivery route planning problem. INFORMS Annual Meeting. Seattle, WA, US.

Highlights

- Food delivery route planning problem is studied
- XGBoost-enhanced fast constructive algorithm is proposed
- Insertion-based heuristic with different sequencing rules is embedded
- Adaptive selection mechanism using XGBoost is proposed
- Effectiveness is validated on datasets from Meituan food delivery platform

Title: An XGBoost-enhanced fast constructive algorithm for food delivery route planning problem

Authors: Xing Wang[a], Ling Wang[a,*], Shengyao Wang[b], Jing-fang Chen[a], Chuge Wu[a]

Affiliation: [a]Department of Automation, Tsinghua University, Beijing 100084, China; [b]Meituan, Beijing 100102, China

Corresponding author: Ling Wang

Tel: +86-10-62783125 Fax: +86-10-62786911

Email: wang-x17@mails.tsinghua.edu.cn (Xing Wang), wangling@mail.tsinghua.edu.cn (Ling Wang), wangshengyao@meituan.com (Shengyao Wang), cjf17@mails.tsinghua.edu.cn (Jing-fang Chen), wucg15@mails.tsinghua.edu.cn (Chuge Wu)

Present/permanent address: Department of Automation, Tsinghua University, Beijing, 100084, China

No author associated with this paper has disclosed any potential or pertinent conflicts which may be perceived to have impending conflict with this work.

Xing Wang: Data curation, analysis, methodology, writing draft.

Ling Wang: Conceptualization, analysis, methodology, review, and funding acquisition.

Shengyao Wang: Conceptualization, methodology and funding acquisition

Jing-fang Chen: Methodology and review

Chuge Wu: Methodology and review

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.