# A Two-stage Algorithm for Fuzzy Online Order Dispatching Problem

Jie Zheng[1], Shengyao Wang[2], Ling Wang[1], Jing-fang Chen[1],
Li Wang[2], Jinghua Hao[2], Renqing He[2], Zhizhao Sun[2]
[1]*Department of Automation*, *Tsinghua University*, Beijing, China
[2]*Meituan-Dianping Group*, Beijing, China
j-zheng18@mails.tsinghua.edu.cn, wangshengyao@meituan.com,
wangling@mail.tsinghua.edu.cn, cjf17@mails.tsinghua.edu.cn
{wangli78, haojinghua, herenqing, sunzhizhao}@meituan.com

***Abstract*—Considering the uncertainty in the real world online food delivery applications, this paper addresses an online order dispatching problem with fuzzy preparation times (FOODP). According to the characteristics of the problem, the FOODP is decomposed into two sub-problems, an order assignment problem and a fuzzy traveling salesman problem with pickup and delivery. To deal with the two sub-problems efficiently, a two-stage algorithm is proposed by reasonably fusing a modified greedy search (mGS) and the fruit fly optimization algorithm (FOA). In the mGS phase, agreement index is employed in the multi-stage decision to search for robust optimal solutions. In the FOA-based search phase, a modified heuristic is used to generate the initial route. To enhance the exploration of the algorithm, an olfactory search is designed by the cooperation of several problem-specific search operators. Moreover, a specific local intensification is employed to further improve the performance of solutions. Numerical tests and statistical analysis demonstrate the effectiveness and efficiency of the proposed algorithm.**

***Index Terms*—fuzzy online order dispatching problem, the two-stage algorithm, modified greedy search heuristic, fruit fly optimization algorithm, robustness**

## I. Introduction

With the development of rapid economic and internet, the pace of life accelerates in recent years. To save time on dining, online food-delivery platforms have penetrated into the lives of young people, becoming an important choice for people's daily diet. The mode of online food delivery platforms has a wide market space, good development prospect, and fierce competition. Taking Meituan-Dianping (a major online food delivery platform in China) for example, the third quarter revenue was as high as $2.2 billion, with a growth rate of 39.4% and the number of transactions was 2.5 billion, with a growth rate of 38.1% [1]. Since the online order dispatching problem (OODP) is the key problem in real-time online food delivery platform, it is important to solve the problem.

The mode of the online food delivery platforms is shown in Fig. 1. When a customer orders food, the platform will push the order to the corresponding restaurant and dispatch it to a driver with a planned route at the same time. To improve word of mouth and gain benefits, the platform needs to provide customers with fast and reliable service experience.

Every day, millions of orders are generated across the whole country, and each order should be delivered within a short

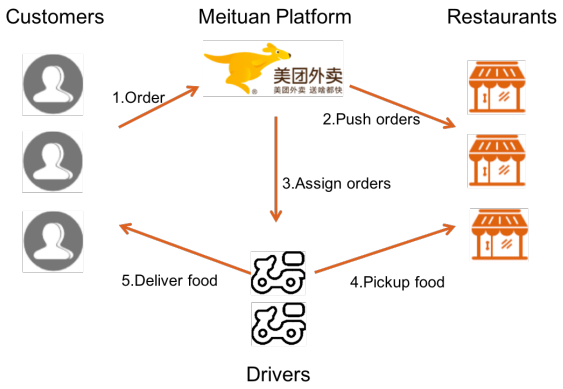


Fig. 1: The order dispatching process.

period of time (usually within 40 minutes). Therefore, it is significant to study the OODP with fast and stable algorithms. To the best of our knowledge, few research works have been carried out on OODP. And the pickup and delivery problems (PDP) and vehicle routing problems (VRP) are most relevant to the OODP. The main difference is that the OODP considers the complexity of real-world application scenario, and needs to react and calculate new orders in real time.

During recent years, extensive research has been carried out on the PDP and VRP. Lu and Dessouky [2] formulated the multiple vehicle PDP as an integer-programming problem and proposed a branch-and-cut algorithm to solve it. For the PDP with time window (PDPTW), Dumas [3] presented an exact algorithm based on column generation scheme. However, the exact algorithms are not suitable for large-scale problems due to explosive growth of computing time. Ropke and Pisinger D [4] presented an adaptive large neighborhood search algorithm, employing a number of competing sub-heuristics which are used with a frequency corresponding to their historic performance. As for dynamic VRP, Mavrovouniotis and Yang [5] designed an ant colony optimization (ACO) with the cooperation of three immigrants schemes, considering both the diversity and the adaptation capabilities of the population. Besides, Tchoupo et al. [6] also presented the ACO coupled with dedicated local search algorithms to minimize the number

978-1-7281-6929-3/20/$31.00 ©2020 IEEE

of vehicles and the total distance travelled of PDPTW. For the dynamic VRP with random customers, Guo et al. [7] designed a multi-objective particle swarm optimization. Although these evolutionary algorithms have good performs, the time consumption is unbearable for online platforms. Therefore, it is difficult but important to balance the solution quality and running speed of the algorithm for the OODP.

In real life food delivery platforms, uncertainty is an inevitable and inherent characteristic. The solutions obtained by the deterministic problem are usually not flexible and robust enough in unexpected cases such as traffic jam, special requests of customers, long food preparation time of restaurants and so on. Among them, the food preparation time of restaurants is the most important factor affecting the efficiency of food delivering, especially at peak hours. Ideally, the food is ready when the driver arrives at the restaurant, and he can leave for the customer immediately. Whereas, the preparation time is usually uncertain and difficult to estimate. Once the food is prepared slowly, the driver must wait in the restaurant until it is finished, which may cause subsequent orders timeout and make conflicts between the driver and the restaurant.

Therefore, it is important to study the OODP with uncertain preparation times. Fuzzy set theory is an important method to handle uncertainty and vagueness. Recently, there has been some research about the fuzzy VRP and fuzzy PDP. For dynamic VRP with uncertain service time, Brito et al. [8] proposed a fuzzy ant colony system combined with a cluster insertion algorithm. For fuzzy green PDPTW, Majidi et al. [9] presented an adaptive large neighborhood search heuristic by applying new removal and insertion operators with fuzzy credibility measure. Sifa et al. [10] simulated the fuzzy velocity based on history data and designed a Tabu Search to solve PDP with fuzzy velocity. For the VRP with fuzzy time window, Tang et al. [11] proposed a two-stage algorithm with different fuzzy membership functions.

This paper takes first attempt to address the online order dispatching problem with fuzzy preparation times (FOODP). To minimize the total assignment cost quickly and obtain a robust solution, we decompose the problem into two subproblems: an order assignment problem and a fuzzy traveling salesman problem with pickup and delivery (FTSPPD). To solve the former sub-problem, a modified greedy search (mGS) is proposed based on multi-stage decision. To solve the latter one, a fruit fly optimization algorithm (FOA) is presented with the cooperation of multiple problem-specific search operators. Besides, an improved heuristic method is employed to initialize the route. And the local intensification is applied to further improve the solution. By reasonably fusing the mGS and FOA, a two-stage algorithm (TSA) is proposed to solving the whole problem. Comparison of simulation results and statistic analysis demonstrate the effectiveness of the proposed algorithm.

The remainder of this paper is organized as follows. In Section II, operations of fuzzy set and the FOODP are described. In Section III, the two-stage algorithm is presented in details. Simulation results and statistic analysis are provided in Section IV. Finally, we end the paper with some conclusions and future work in Section V.

## II. The Fuzzy Online Order Dispatching Problem

### A. Operations on fuzzy sets

To deal with fuzzy sets in FOODP, the arithmetic operations of fuzzy numbers are used, including addition, maximum, and ranking method. For two triangular fuzzy numbers (TFNs) $A = (a_1, a_2, a_3)$, $B = (b_1, b_2, b_3)$ and a real number $c$, the addition, subtraction and maximum operations [12], [13] are shown as follows.

$$A + B = (a_1 + b_1, a_2 + b_2, a_3 + b_3); \tag{1}$$

$$A + c = (a_1 + c, a_2 + c, a_3 + c); \tag{2}$$

$$A - c = (a_1 - c, a_2 - c, a_3 - c); \tag{3}$$

$$max\{A, B\} \approx (max\{a_1, b_1\}, max\{a_2, b_2\}, max\{a_3, b_3\}); \tag{4}$$

$$max\{A, c\} = (max\{a_1, c\}, max\{a_2, c\}, max\{a_3, c\}); \tag{5}$$

The expected value of the TFN $A$ is given as follows [14].

$$E(A) = (a_1 + 2a_2 + a_3)/4; \tag{6}$$

Besides, the following criterion is adopted to compare TFNs [15].

Step 1: The greatest number $Z_1(A) = E(A)$ will be chosen as the first criterion to rank two TFNs.

Step 2: If two TFNs have the same $Z_1$, $Z_2(A) = a_2$ is used as the second criterion.

Step 3: If two TFNs have identical $Z_1$ and $Z_2$, $Z_3(A) = a_3 - a_1$ will be chosen as the third criterion.

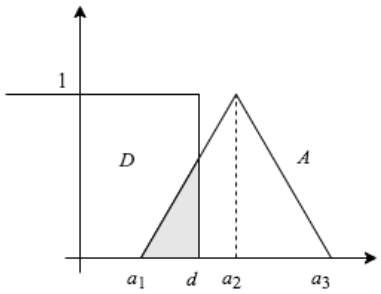


Fig. 2: Agreement Index

As shown in Fig. 2, the agreement index (AI) [16] of the TFN $A$ with respect to a due date $d$ is defined as the coincident area ratio of $D$ and the membership function of $A$, where $D$ is the step function on $d$. To be more explicit,

$$D(x) = \begin{cases} 1, & x \le d \\ 0, & otherwise \end{cases} \tag{7}$$

$$AI = [area(D \cap A)]/area(A)) \tag{8}$$

The AI can be seen as the percentage of $A$ that completed before the due date.

TABLE I: The notations of the FOODP

| Notation | Description |
|---|---|
| $n$ | the number of new orders |
| $m$ | the number of drivers |
| $i, j, l$ | the index of orders, drivers, and route points respectively |
| $nw_j(nw'_j)$ | the number of old (all) orders on the driver $v_j$ |
| $num_j(num'_j)$ | the number of route points of the old (new) route of $v_j$ |
| $V$ | the set of drivers, $V = \{v_1, \ldots, v_m\}$ |
| $W^{new}$ | the set of new orders, $W^{new} = \{w_1^{new}, \ldots, w_n^{new}\}$ |
| $W_j^{old}$ | the set of old orders on $v_j$, $W_j^{old} = \{w_1^{old}, \ldots, w_{nw_j}^{old}\}$ |
| $W^{old}$ | the set of all old orders, $W^{old} = \bigcup_{j=1}^{m} W_j^{old}$ |
| $W$ | the set of all orders, $W = W^{new} \cup W^{old}$ |
| $(i^+, i^-)$ | the node pair of order $w_i$; $i^+$ is the pickup node and $i^-$ is the delivery node |
| $P^+$ | the set of pickup nodes of the new orders $P^+ = \{i^+ \mid w_i \in W^{new}\}$ |
| $P^-$ | the set of delivery nodes of the new orders $P^- = \{i^- \mid w_i \in W^{new}\}$ |
| $P$ | the set of all nodes of the new orders $P = P^+ \cup P^-$ |
| $U^+$ | the set of pickup nodes of the old orders $U^+ = \{i^+ \mid w_i \in W^{old}\}$ |
| $U^-$ | the set of delivery nodes of the old orders $U^- = \{i^- \mid w_i \in W^{old}\}$ |
| $U$ | the set of all nodes of the old orders $U = U^+ \cup U^-$ |
| $H$ | the set of starting positions of all drivers, $H = \{h_j \mid v_j \in V\}$ |
| $PT_i$ | the TFN food preparation time of $w_i$, $PT_i = (pt_{i,1}, pt_{i,2}, pt_{i,3})$ |
| $d_i$ | the due date of $w_i$ |
| $R_{0,j}$ | the original route of $v_j$, $R_{0,j} = \{r_{0,0,j}, r_{0,1,j}, r_{0,2,j}, \ldots, r_{0,num_j,j}\}, r_{0,0,j} = h_j$ |
| $R_{i,j}$ | the new route of dispatching $w_i$ to $v_j$, $R_{i,j} = \{r_{i,0,j}, r_{i,1,j}, r_{i,2,j}, \ldots, r_{i,num'_j,j}\}, r_{i,0,j} = h_j$ |
| $t_{l_1,l_2}$ | the travel time of point $l_1$ and point $l_2$ |
| $D_{l_1,l_2}$ | the travel distance of point $l_1$ and point $l_2$ |
| $VT_{i^-}$ | the visiting time of the delivery node of order $w_i \in W$ |
| $OT_i$ | the overtime of order $w_i \in W$ |
| $TC_{i,j}$ | the time cost of dispatching $w_i$ to $v_j$ |
| $DC_{i,j}$ | the distance cost of dispatching $w_i$ to $v_j$ |
| $AC_{i,j}$ | the assignment cost of dispatching $w_i$ to $v_j$ |
| $AI_{i,j}$ | the agreement index of dispatching $w_i$ to $v_j$ |

### *B. Description of FOODP*

The notations of the FOODP are described in Table I.

The FOODP contains $n$ new orders and $m$ drivers, denoted as $W^{new} = \{w_1^{new}, \ldots, w_n^{new}\}$, $V = \{v_1, \ldots, v_m\}$ respectively. In addition to the new orders, the drivers also works on a number of old orders assigned previously, denoted as $W^{old} = \bigcup_{j=1}^{m} W_j^{old}$ where $W_j^{old} = \{w_1^{old}, \ldots, w_{nw_j}^{old}\}$ is the set of old orders assigned to $v_j$, and $nw_j$ is the number of old orders on $v_j$. Let $P^+ = \{i^+ \mid w_i \in W^{new}\}$ be the set of pickup nodes of the new orders, and $P^- = \{i^- \mid w_i \in W^{new}\}$ is the set of delivery nodes of the new orders. $|P^+| = |P^-| = n$. $P = P^+ \cup P^-$ represents the whole points associated to the new orders. Similarly, $U^+ = \{i^+ \mid w_i \in W^{old}\}$ and $U^- = \{i^- \mid w_i \in W^{old}\}$ are the sets of pickup points and delivery points of old orders, respectively. $U = U^+ \cup U^-$ represents all the points associated to old orders. Because some old orders have already been picked, $|U^+| <= |U^-| = \sum_{j=1}^{m} nw_j$. $W = W^{new} \cup W^{old}$ is the set of total orders. Each order $w_i \in W$ is represented by $(i^+, i^-)$ and has a TFN food preparation time $PT_i = (pt_{i,1}, pt_{i,2}, pt_{i,3})$ if it is unpicked. If the old order has already been picked, it is represented by $(0, i^-)$. Besides, each order $w_i \in W$ has a due date $d_i$ which is a promise to customers when the order is delivered. $H = \{h_j \mid v_j \in V\}$ is the set of starting positions of all drivers. Each driver $v_j$ has an original route, constituted by $h_j$ and the pickup and delivery points of the old orders on $v_j$, denoted as $R_{0,j} = \{h_j, r_{0,1,j}, r_{0,2,j}, \ldots, r_{0,num_j,j}\}$, where $r_{0,l,j}$ $(l = 1, \ldots, num_j)$ is the $l$th route point and $num_j$ is the number of route points of the original route. If order $w_i \in W^{new}$ is dispatched to $v_j$, a new route $R_{i,j} = \{h_j, r_{i,1,j}, r_{i,2,j}, \ldots, r_{i,num'_j,j}\}$ will be planed for $v_j$ where $r_{i,l,j}$ $(l = 1, \ldots, num'_j)$ is the $l$th route point, and $num'_j$ is the number of route points of the new route. For convenience, $h_j$ is replaced by $r_{i,0,j} (i = 0, \ldots, n)$ in the equation. Every driver has a maximum capacity Q, which is the maximum quantity that a driver can carry at a certain time.

The FOODP can be defined on a graph $G = (N, A)$ where $N = P \cup U \cup H$ is the set of all nodes. The set of arcs is $A = N \times N$. Each arc $(l_1, l_2) \in A$ is associated with a travel time $t_{l_1,l_2}$ and a travel distance $D_{l_1,l_2}$. The visiting time of the delivery node of order $w_i \in W$ is denoted as $VT_{i^-}$. The overtime $OT_i$ of order $w_i \in W$ is calculated as follows.

$$OT_i = max\{0, VT_{i^-} - d_i\} \tag{9}$$

A solution of FOODP includes the assignment of the new orders to each driver and the routes of each driver in which the sequence of the pickup and delivery points are determined.

The basic assumptions of the problem are as follows. a) Old orders of each driver can not be transformed to other drivers. b) Orders can only be picked up after the food being prepared by the restaurant. c) A driver must pick up food before delivering it, which means $i^-$ must be behind $i^+$ in a feasible route. d)

A new order can only be assigned to one driver. e) Orders on a driver cannot exceed capacity constraints all the time.

In conclusion, FOODP is a very complicated problem. It takes into account the characteristics of the real-world applications, including the variable position of drivers, old orders on drivers, uncertain food preparation times, capacity constraints and time limitation of the algorithm.

The cost of dispatching an order $w_i \in W^{new}$ to a driver $v_j$ includes two aspects: time cost (TC) and distance cost (DC). The TC is the difference of the expectation of total overtime between the new route and original route, and the DC is the difference of the distance between the two as follows.

$$TC_{i,j} = \sum_{l=1}^{nw'_j} E(OT_l) - \sum_{l=1}^{nw_j} E(OT_l), \forall i, j \tag{10}$$

$$DC_{i,j} = \sum_{k=0}^{num'_j} D_{r_{(i,k,j)},r_{(i,k+1,j)}} - \sum_{k=0}^{num_j} D_{r_{(0,k,j)},r_{(0,k+1,j)}}, \forall i, j \tag{11}$$

The objective function is to minimize the total assignment cost (AC), which is a weighted sum of TC and DC.

$$AC_{i,j} = \lambda_1 TC_{i,j} + \lambda_2 DC_{i,j}, \forall i, j \tag{12}$$

$$AC = \sum_{i=1}^{n}\sum_{j=1}^{m}(AC_{i,j} I_{i,j}) \tag{13}$$

$$I_{i,j} = \begin{cases} 0, \; if \; w_i \in W^{new} \; is \; dispatched \; to \; driver \; v_j \\ 1, \; else \end{cases} \tag{14}$$

where $\lambda_1$ and $\lambda_2$ are the weights of TC and DC respectively.

## III. Two-stage algorithm

To solve the FOODP, we decompose the original problem into two sub-problems. The upper-level sub-problem is to determine the assignment of each order to drivers. The lower-level one is a fuzzy traveling salesman problem with pickup and delivery.

A modified greedy search heuristic (mGS) is designed for solving the upper-level sub-problem quickly. By considering both the assignment cost and route robustness simultaneously, the multi-stage decision is employed in mGS. To solve the lower-level sub-problem, a FOA-based evolutionary algorithm is presented with the cooperation of several problem-specific operators. By reasonably fusing the mGS and the FOA-based search, a two-stage algorithm (TSA) is proposed. The framework of the TSA is shown in Fig. 3.

### A. modified greedy search phase

To determine the assignment of each new orders to drivers, a modified greedy search heuristic is presented. As a simple heuristic, the greedy search (GS) can get an approximation optimal solution for the scheduling quickly. The main idea of the GS is to assign the new order to the driver with lowest $AC_{i,j}$ at each iteration. However, it may be myopic and easy to be trapped in a local minimum. Therefore, we propose a modified GS, considering the robustness of the new routes by the multi-stage decision. The agreement index can reflect the possibility of order timeout with fuzzy preparation times. A route with $AI = 0$ means that no matter what the preparation time is, the order on this route will definitely time out. On the contrary, if $AI = 1$, it means that the order of this route will never time out. Therefore, AI can reflect the robustness of a route to some extent. $AI_{i,j}$ of the route $R_{i,j}$ is calculated as follows.

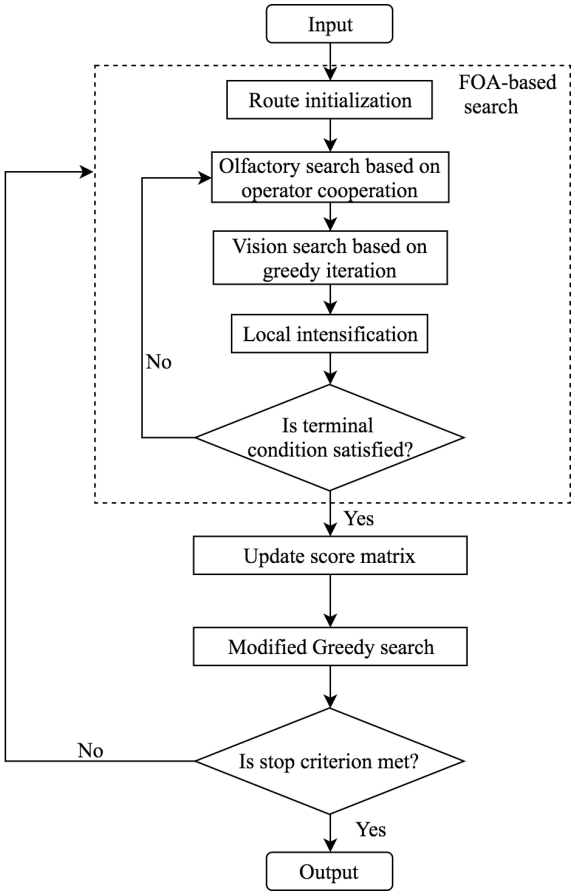

Fig. 3: The framework of the TSA.

$$AI_{i,j} = [area(VT_{i^-_{max}} \cap D_{i_{max}})]/area(VT_{i^-_{max}}) \tag{15}$$

where $i_{max}$ is the order with maximum overtime in the new route $R_{i,j}$, and $D_{i_{max}}$ is the step function on $d_{i_{max}}$. $AI_{i,j}$ reflects the timeout level of the order on $R_{i,j}$ which is most likely to be overtime.

In detail, a cost matrix $C = \{c_{i,j}\}$ is built in the mGS to describe the performance of order assignment. Each element $c_{i,j} = \{AC_{i,j}, AI_{i,j}\}$ in $C$ represents the assignment cost and agreement index of dispatching new order $w_i$ to driver $v_j$, obtained by solving the lower-level sub-problem. In each iteration, the assignment with lowest AC in $C$ is selected to build up a $\alpha$-min set $S$, in which the difference of the AC between any element is less than $\alpha$. Then new order will be

assigned to the driver with largest AI in the $S$. The procedure of mGS is as Algorithm 1.

**Algorithm 1** Pseudo-codes of mGS

**Require:** the set of unassigned orders $W^{new}$ and the set of drivers $V$ with old orders;

```
Calculate AC_{i,j} and AI_{i,j}, ∀i, j;
while the cost matrix C! = ∅ do
    Find the minimum AC_{i,j} in C;
    Let α-min set S = ∅;
    for l = 1 : n do
        if AC_{l,j} − AC_{i,j} <= α then
            Add pair (w_l, v_j) into S;
        end if
    end for
    Find the pair (w'_i, v_j) with minimum AI in S;
    Assign the order w'_i to driver v_j;
    Delete the i'th column of C;
    Update the jth row of C;
end while
```

For example, suppose the score matrix is shown as TABLE II and $\alpha = 2$. Firstly, we choose the pair $(w_2, v_2)$ which has the minimum AC=1.5 to build the $\alpha$-min set $S$. Since $|3.0 - 1.5| < \alpha = 2$ and $|4 - 1.5| > \alpha$, the $S$ is consisted of the pair $(w_2, v_2)$ and $(w_1, v_2)$. Then the pair $(w_2, v_2)$ will be selected at this iteration because it has the maximum AI in the $S$. That is, the new order $w_2$ will be dispatched to driver $v_2$ at this iteration. Afterwards, the cost value of $v_2$ should be updated and the $2nd$ column of $C$ should be deleted since $w_2$ has already been assigned.

TABLE II: An example of the cost matrix $C$

| $c_{i,j}$ | $v_1$ | $v_2$ | $v_3$ |
|---|---|---|---|
| $w_1$ | (4.0, 0.1) | (3.0, 0.5) | (2.0, 0.7) |
| $w_2$ | (2.6, 0.8) | (1.5, 0.6) | (5.0, 1.0) |
| $w_3$ | (3.0, 0.7) | (4.0, 1.0) | (8.0, 0.5) |

By introducing AI into the mGS, we will choose the most robust new route (less likely to be timeout) with satisfying assignment cost at each iteration, and assign the order to the corresponding driver. By this way, a solution of good quality will be obtained quickly.

### *B. FOA-based search phase*

The fruit fly optimization algorithm [17] is a relatively novel swarm intelligence optimization algorithm inspired by the foraging behavior of fruit flies. In recent years, the FOA has been used to solve complex optimization problems in many fields, including prediction, logistics, scheduling problems, power system and so on [18].

In this section, we design the FOA-based search according to the characteristics of the FTSPPD. The core of the FOA-based search includes encoding, decoding, route initialization, olfactory search based on operator cooperation, vision search based on greedy iteration and problem-specific local intensification.

*1) Encoding and decoding:* A solution in the FTSPPD is represented by $(1+q_p+q_d)$ permutation sequence $\Pi$, including a starting point 0, $q_p$ pickup points and $q_d$ delivery points. $i^+$ and $i^-$ means the pickup point and the delivery point of $w_i$, respectively. According to the constrains, $i^-$ must be placed after $i^+$ if $w_i$ is an unpicked order. If $w_i$ is already picked, $i^+ = 0$. For an example with 2 unpicked orders ($q_p = 2, q_d = 2$), a feasible solution is encoded as $0 - 1^+ - 2^+ - 1^- - 2^-$. The solution is shown in the Fig. 4, where (a) demonstrates the original route of the driver and (b) shows the new route obtained by dispatching $w_2$ to the driver. The fuzzy preparation times, due dates and travel times are also on the figures. For the old route, $OT_1 = max(0, [9, 11, 16] - 15) = [0, 0, 1]$. For the new route, $OT_1 = max(0, [11, 13, 18] - 15) = [0, 0, 3]$, $OT_2 = max(0, [13, 15, 20] - 15) = [0, 0, 5]$. Therefore, $TC = E([0, 0, 3]) + E([0, 0, 5]) - E([0, 0, 1]) = 1.75$, $DC = (10 + 18 + 20 + 15) - (10 + 32) = 21$, $AC = \lambda_1 TC + \lambda_2 DC = 5.95$ (if $\lambda_1 = 1, \lambda_2 = 0.2$), $AI = area(VT_{2^-} \cap D_2)/area(VT_{2^-}) = 2/7$.

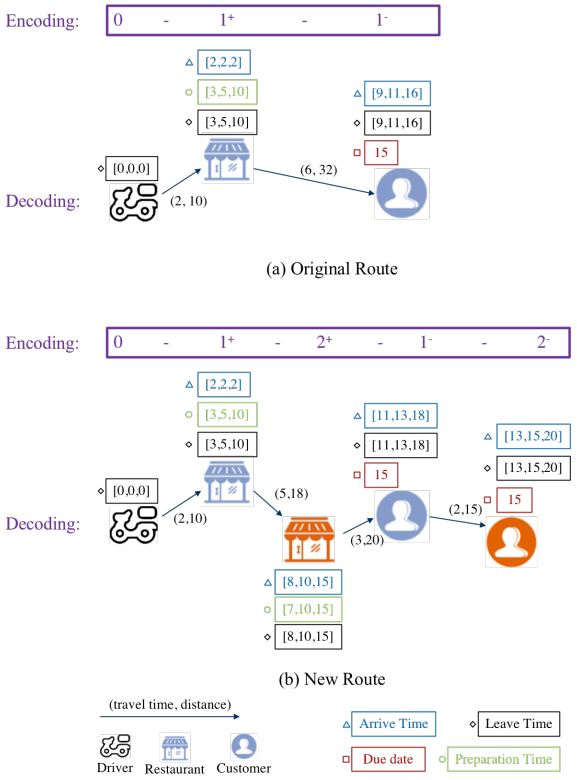


Fig. 4: An example of a fuzzy route

*2) Route initialization:* Considering the quality of initial solution to well guide the search, a modified Nawaz-Enscore-Ham [19] (mNEH) method is used to initialize the position of fruit fly population. The procedure of the heuristic is as follows.

Step 1: Calculate the capacity of all orders assigned to the driver $v_j$. If it is larger than $Q$, return NULL and set the AI

and AC as $\infty$. Otherwise, let initial route $\Pi = \emptyset$. Sort all the orders of driver $v_j$ in a descending order according to their due dates (including the new order $w_i$, totally $nw'_j$ orders), $\Omega' = (w_{(1)}, w_{(2)}, \ldots, w_{(nw'_j)})$. Let $l = 1$;

Step 2: Insert the pickup point (if exists) and delivery point of $w_{(l)}$ into all possible position of $\Pi$ sequentially due to precedence constraint and capacity constraints and calculate the $AI_{i,j}$ and $AC_{i,j}$ of the partial route. The one with lower $AC_{i,j}$ will be select to replace $\Pi$. If there exist ties, then choose the one with maximum $AI_{i,j}$ as $\Pi$.

Step 3: If $l < n, l = l + 1$, go to step 2; Otherwise output $\Pi$.

By applying this method, we formulate a feasible route with certain quality.

*3) Olfactory search and vision search:* Olfactory search is the core part of the FOA, having a great influence on the performance. In this paper, the olfactory search is designed by the cooperation of three neighborhood search operators as follows.

$Insert$: Randomly select a route point from $\Pi$ and insert it into all possible position. The best route will be chosen as new route $\Pi^*$.

$Swap\_N$: Randomly select a route point $p_i$ from $\Pi$ and swap the position of $p_i$ and the nearest feasible route point.

$Swap\_C$: Regarding continuous pickup points or continuous delivery points as an aggregation point. Randomly swap a pair of adjacent aggregation points and check feasibility, then choose the better route as $\Pi^*$.

The route with lower cost is regarded as the better route. If there exist ties, then the one with maximum AI is better.

With the cooperation of the above three operators, the fruit flies will fly to three different positions. Afterwards, the vision search based on greedy iteration will choose the best position of the three and all the fruit flies will gather to the that position.

*4) Local intensification:* To further improve the performance of the route, a local intensification is designed with the following two kinds of problem-specific neighborhood searches.

- Backward search: Find the delivery points with largest OT and move them backward to an optimal position.
- Forward search: Find the points with most sufficient time and move them forward to an optimal position

Once a better solution is found, the best solution will be replaced.

## IV. Experimental result

In this section, numerical experiments will be conducted to evaluate the performance of the proposed algorithm. The testing instances are sampled from real data in Shenzhen of Meituan-Dianping platform, and are grouped by the number of new orders $n$. Each group contains 10 different instances, so there are 100 instances in total. The preparation times of the orders are learned from extensive history data. That is, $PT_i = (pt_{i,1}, pt_{i,2}, pt_{i,3})$ where $pt_{i,1}$ and $pt_{i,3}$ are the minimum and maximum preparation times of the same food in the same restaurant, while $pt_{i,2}$ is the most frequent preparation time of the history data. $\lambda_1$ and $\lambda_2$ are set as 1/60, 1/10000 respectively according to the experience. All the algorithms are coded in Java SE8 and experiments are run on a MacBook Pro with 2.2 GHz processors / 16 GB RAM under Mac OS.

The TSA contains one key parameters: the value of $\alpha$ in $\alpha$-min set $S$, and is set as 2. To the best of our knowledge, there is no published work addressing the FOODP. Therefore, we combine the GS-based order assignment with variable-depth search (VDS) [21], simulated annealing (SA) [20] and genetic algorithm (GA) [22]. In the SA, exponential descent method is used where the initial temperature is 1500 and the annealing rate is 0.9. As for the GA, the size of the population is set as 10, and probabilities of crossover and mutation are 0.5 and 0.6, respectively. To compare the TSA with the above three state-of-art algorithms, we run each algorithm 5 times independently for each instance with $0.01 \times nw'_j$ seconds CPU time as the terminal condition for the FTSPPD. To demonstrate the effectiveness of the TSA without fuzzy theory (TSA-nF), we compare the TSA-nF with other three algorithms on deterministic instances, where the preparation times of unpicked orders are set as the expectation times according to history data. The following relative percentage deviation (RPD) is adopted as indicator to evaluate the results.

$$RPD = (alg - opt)/opt \times 100 \tag{16}$$

where $alg$ is the AC obtained by the algorithm, and $opt$ is the best AC obtained by the four algorithms. Obviously, a smaller value of $RPD$ means better performance of the corresponding algorithm.

TABLE III: The RPD on deterministic instances

| $n$ | TSA-nF | GA-nF | VDS-nF | SA-nF |
|---|---|---|---|---|
| $0<n \leq 10$ | **0.00** | 1.69 | 1.42 | 1.53 |
| $10<n \leq 20$ | 2.61 | **2.47** | 2.81 | 2.84 |
| $20<n \leq 30$ | **0.95** | 1.73 | 5.46 | 5.97 |
| $30<n \leq 40$ | **1.46** | 1.99 | 11.44 | 13.09 |
| $40<n \leq 50$ | **1.11** | 3.23 | 14.91 | 15.63 |
| $50<n \leq 60$ | **0.65** | 2.36 | 10.23 | 10.56 |
| $60<n \leq 70$ | **1.08** | 2.83 | 7.52 | 7.84 |
| $70<n \leq 80$ | **0.75** | 2.05 | 10.26 | 10.40 |
| $80<n \leq 90$ | **1.16** | 2.55 | 7.92 | 7.93 |
| $90<n \leq 100$ | **1.31** | 2.94 | 9.13 | 9.93 |
| Average | **1.11** | 2.38 | 8.11 | 8.57 |

As shown in Table III, the TSA-nF has smaller values of RPD in all groups of instances than the VDS-nF and SA-nF and is better than GA in most groups. Therefore, It can be concluded that TSA-nF performs significantly better than other algorithms, especially with large $n$. Besides, the standard deviations of the normalized AC obtained by the four algorithms is shown in Table IV. It can be seen that, the TSA-nF performs most steadily on average.

Table V-VI show the RPD and standard deviations on fuzzy instances respectively. From the Table V, we can see that the ACs of TSA are smallest in all groups of instances. And Table VI indicates that the robustness of the four algorithms is similar on average.

TABLE IV: The standard deviations on deterministic instances

| $n$ | TSA-nF | GA-nF | VDS-nF | SA-nF |
|---|---|---|---|---|
| $0<n \leq 10$ | **0.00** | 0.32 | 0.34 | 0.38 |
| $10<n \leq 20$ | 0.41 | 0.47 | 0.43 | **0.40** |
| $20<n \leq 30$ | **0.29** | 0.34 | 0.38 | 0.35 |
| $30<n \leq 40$ | 0.29 | **0.24** | 0.28 | 0.26 |
| $40<n \leq 50$ | **0.14** | 0.19 | 0.19 | 0.20 |
| $50<n \leq 60$ | **0.13** | 0.20 | 0.20 | 0.17 |
| $60<n \leq 70$ | **0.20** | 0.21 | 0.23 | 0.24 |
| $70<n \leq 80$ | **0.07** | 0.13 | 0.17 | 0.19 |
| $80<n \leq 90$ | **0.18** | 0.21 | 0.20 | 0.21 |
| $90<n \leq 100$ | 0.18 | 0.17 | 0.18 | **0.15** |
| Average | **0.19** | 0.25 | 0.26 | 0.26 |

TABLE V: The RPD on fuzzy instances

| $n$ | TSA | GA | VDS | SA |
|---|---|---|---|---|
| $0<n \leq 10$ | **0.20** | 0.21 | 0.70 | 1.56 |
| $10<n \leq 20$ | **0.17** | 0.23 | 0.22 | 1.13 |
| $20<n \leq 30$ | **2.24** | 3.79 | 3.19 | 4.15 |
| $30<n \leq 40$ | **2.32** | 2.64 | 5.01 | 5.00 |
| $40<n \leq 50$ | **1.58** | 1.87 | 5.14 | 5.67 |
| $50<n \leq 60$ | **1.79** | 1.81 | 2.93 | 2.71 |
| $60<n \leq 70$ | **1.98** | 2.08 | 3.58 | 3.39 |
| $70<n \leq 80$ | **1.32** | 1.48 | 3.36 | 3.47 |
| $80<n \leq 90$ | **1.78** | 2.13 | 3.39 | 3.42 |
| $90<n \leq 100$ | **1.64** | 1.79 | 2.36 | 2.09 |
| Average | **1.50** | 1.80 | 2.99 | 3.26 |

To show the effectiveness of fuzzy theory on robustness, we compare the four algorithms with and without fuzzy method respectively. For the assignment solutions and route planning obtained by the four algorithms, we generate 100 scenarios for each instance by randomly set preparation times for unpicked orders according to history data. Then, the average overtime (AOT) of the solutions can be calculated as follows. Obviously, an algorithm is of better robustness with smaller AOT.

$$AOT = 1/S_{num} \sum_{si=1}^{S_{num}} \sum_{j=1}^{m} \sum_{k=1}^{num'_j} OT_{k,si}/num'_j \qquad (17)$$

where $S_{num} = 100$ is the number of scenarios for each instance, and $OT_{k,si}$ is the $OT_k$ of $si$th scenario. The AOT indicates the average timeout caused by random preparation

TABLE VI: The standard deviations on fuzzy instances

| $n$ | TSA | GA | VDS | SA |
|---|---|---|---|---|
| $0<n \leq 10$ | 0.30 | **0.27** | 0.39 | 0.40 |
| $10<n \leq 20$ | 0.41 | 0.39 | 0.33 | **0.30** |
| $20<n \leq 30$ | 0.35 | 0.34 | 0.35 | **0.33** |
| $30<n \leq 40$ | 0.30 | **0.29** | 0.31 | 0.32 |
| $40<n \leq 50$ | 0.30 | **0.20** | 0.26 | 0.27 |
| $50<n \leq 60$ | 0.39 | 0.36 | **0.31** | 0.34 |
| $60<n \leq 70$ | 0.40 | 0.28 | **0.26** | 0.28 |
| $70<n \leq 80$ | **0.24** | 0.28 | 0.31 | 0.30 |
| $80<n \leq 90$ | **0.29** | 0.33 | 0.31 | 0.30 |
| $90<n \leq 100$ | 0.32 | **0.29** | **0.29** | 0.30 |
| Average | 0.33 | **0.30** | 0.31 | 0.31 |

times. Table VII shows the RPD of the AOT grouped by $n$. It can be seen that, for each algorithm, the solutions are of better robustness with fuzzy method, which implies that the fuzzy method is useful to obtain robust solutions. For the four $Algorithm-nFs$, the AOT of the TSA-nF is less than other three algorithms on average. Similarly, the solutions obtained by TSA are more robust than GA, SA and VDS on average. Therefore, it can be concluded that the TSA is able to obtain solutions with better robustness than other three algorithms.

In addition, we assume that the preparation times of each unpicked order are $\{2,4,6,8,10\}$ minutes later than the exception time, the average overtimes of the algorithms are shown in Fig. 5. It can be seen that the AOT of the solutions obtained by an algorithm with fuzzy method is less than the one without. That is, the fuzzy set theory can handle unexpected cases such as delayed preparation times and obtain a robust solution. And the AOT of the TSA is lowest in all the algorithms. As the delay time increases, the advantage of the TSA becomes more apparently. In general, we can conclude that the TSA is more effective than the state-of-art algorithms, and is more suitable for real world online food delivery platforms.

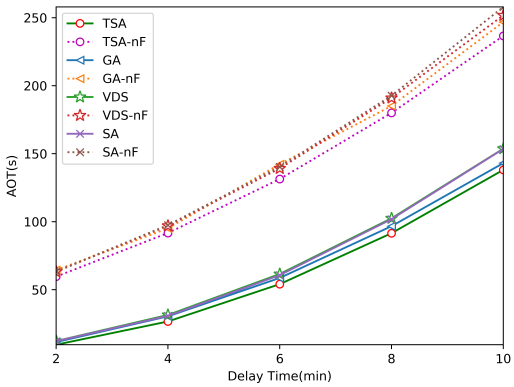


Fig. 5: The ROTs of the four algorithms with time delay.

## V. Conclusion And Future Works

In this paper, we addressed the fuzzy online food dispatching problem with fuzzy food preparation times for the first time. To optimize the assignment cost and find a robust solution, the original problem is decomposed into two subproblems: an order assignment problem and a fuzzy traveling salesman problem with pickup and delivery. A two-stage algorithm is proposed to solve the problem quickly by reasonably fusing the fruit fly algorithm into the framework of the modified greedy search heuristic. Comparative results and statistical analysis demonstrate the effectiveness and robustness of the proposed algorithm. The superiority of our algorithm mainly owes to the follows.

- Fusion of the modified greedy search heuristic and the fruit fly optimization algorithm to solve the problem effectively and efficiently.

TABLE VII: The RPD of the overtime

| n | TSA-nF | GA-nF | VDS-nF | SA-nF | TSA | GA | VDS | SA |
|---|---|---|---|---|---|---|---|---|
| $0<n \leq 10$ | 2.18 | 5.15 | 3.34 | 4.02 | 0.52 | 1.20 | 0.61 | **0.48** |
| $10<n \leq 20$ | 1.22 | 2.30 | 2.57 | 2.45 | 0.19 | 0.21 | 0.23 | **0.18** |
| $20<n \leq 30$ | 1.21 | 2.27 | 2.29 | 2.13 | **0.06** | 0.12 | 0.16 | 0.18 |
| $30<n \leq 40$ | 1.09 | 1.99 | 1.96 | 1.94 | **0.01** | 0.16 | 0.22 | 0.21 |
| $40<n \leq 50$ | 1.82 | 2.82 | 2.82 | 2.74 | **0.02** | 0.12 | 0.24 | 0.22 |
| $50<n \leq 60$ | 1.49 | 2.80 | 2.88 | 2.73 | **0.01** | 0.19 | 0.14 | 0.18 |
| $60<n \leq 70$ | 1.70 | 3.09 | 3.08 | 3.05 | **0.02** | 0.17 | 0.16 | 0.17 |
| $70<n \leq 80$ | 2.83 | 4.61 | 4.68 | 4.65 | **0.01** | 0.15 | 0.14 | 0.19 |
| $80<n \leq 90$ | 2.88 | 4.70 | 4.29 | 4.57 | **0.04** | 0.20 | 0.15 | 0.12 |
| $90<n \leq 100$ | 2.47 | 3.86 | 4.10 | 4.06 | **0.04** | 0.14 | 0.20 | 0.21 |
| Average | 1.90 | 3.37 | 3.23 | 3.27 | **0.09** | 0.26 | 0.22 | 0.22 |

- Utilization of multi-stage decision in mGS to search for robust optimal solutions.
- Utilization of the modified heuristic to produce a population with good quality.
- Design of the olfactory search based on the cooperation of several problem-specific search operators to enhance exploitation capability.
- Utilization of the specific local intensification to further improve the performance of solutions.

Since the real-world online food delivery applications require the algorithm to process a large number of orders in a short time, we will focus on the speed-up methods in our future work. Besides, it is interesting to solve the fuzzy online food dispatching problem with other types of uncertainties.

## ACKNOWLEDGEMENTS

This research is supported by the National Science Fund for Distinguished Young Scholars of China [No. 61525304], the National Natural Science Foundation of China [No. 61873328], and Meituan-Dianping Group.

## REFERENCES

[1] M.-D. Group. (2019, Nov.) Announcement of the Results for the Three Months Ended September 30, 2019. [Online]. Available: https://www1.hkexnews.hk/listedco/listconews/sehk/2019/1121/2019112100554.pdf.

[2] Q. Lu and M. Dessouky, "An exact algorithm for the multiple vehicle pickup and delivery problem," *Transportation Science*, vol. 38, no. 4, pp. 503-514, 2004.

[3] Y. Dumas, J. Desrosiers, and F. Soumis, "The pickup and delivery problem with time windows," *European Journal of Operational Research*, vol. 54, no. 1, pp. 7-22, 1991.

[4] S. Ropke and D. Pisinger, "An Adaptive Large Neighborhood Search Heuristic for the Pickup and Delivery Problem with Time Windows," *Transportation Science*, vol. 40, no. 4, pp. 455-472, 2006.

[5] M. Mavrovouniotis and S. Yang, "Ant algorithms with immigrants schemes for the dynamic vehicle routing problem," *Information Sciences*, vol. 294, pp. 456-477, 2015.

[6] M. N. Tchoupo, A. Yalaoui, L. Amodeo, F. Yalaoui and F. Lutz, "Ant colony optimization algorithm for pickup and delivery problem with time windows," in *International Conference on Optimization and Decision Science*. Springer, Cham, pp. 181-191, 2017.

[7] Y.-N. Guo, J. Cheng, S. Luo, D.-W. Gong, and Y. Xue, "Robust Dynamic Multi-objective Vehicle Routing Optimization Method," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 15, no. 6, pp. 1891-1903, 2018.

[8] J. Brito, F. J. Martínez, J. A. Moreno, and J. L. Verdegay, "An ACO hybrid metaheuristic for close–open vehicle routing problems with time windows and fuzzy constraints," *Applied Soft Computing*, vol. 32, pp. 154-163, 2015.

[9] S. Majidi, S. M. Hosseini-Motlagh, S. Yaghoubi, and A. Jokar, "Fuzzy green vehicle routing problem with simultaneous pickup–delivery and time windows," *RAIRO-Operations Research*, vol. 51, no. 4, pp. 1151-1176, 2017.

[10] S. Zheng, J. Cao, X. Lian, and K. Li, "Urban pickup and delivery problem considering time-dependent fuzzy velocity," *Computers & Industrial Engineering*, vol. 60, no. 4, pp. 821-829, 2011.

[11] J. Tang, Z. Pan, R. Y. K. Fung, and H. Lau, "Vehicle routing problem with fuzzy time windows," *Fuzzy Sets and Systems*, vol. 160, no. 5, pp. 683-695, 2009.

[12] S. Chanas and A. Kasperski, "On two single machine scheduling problems with fuzzy processing times and fuzzy due dates," *European Journal of Operational Research*, vol. 147, no. 2, pp. 281-296, 2003.

[13] J. J. Palacios, I. González-Rodríguez, C. R. Vela, and J. Puente, "Robust multiobjective optimisation for fuzzy job shop problems," *Applied Soft Computing*, vol. 56, pp. 604-616, 2017.

[14] B. Liu and Y. K. Liu, "Expected value of fuzzy variable and fuzzy expected value models," *IEEE Transactions on Fuzzy Systems*, vol. 10, no. 4, pp. 445-450, 2002.

[15] M. Sakawa and R. Kubota, "Fuzzy programming for multiobjective job shop scheduling with fuzzy processing time and fuzzy duedate through genetic algorithms," *European Journal of Operational Research*, vol. 120, no. 2, pp. 393-407, 2000.

[16] M. Sakawa, and T. Mori, "An efficient genetic algorithm for job-shop scheduling problems with fuzzy processing time and fuzzy duedate," *Computers & Industrial Engineering*, vol. 36, no. 2, pp. 325-341, 1999.

[17] W. T. Pan, "A new fruit fly optimization algorithm: taking the financial distress model as an example," *Knowledge-Based Systems*, vol. 26, pp. 69-74, 2012.

[18] L. Wang, and X. L. Zheng, "Advances in fruit fly optimization algorithms," *Control Theory & Applications*, vol. 34, no. 5, pp. 557-563, 2017.

[19] M. Nawaz, E. E. J. Enscore, I. Ham, "A heuristic algorithm for the m-machine, n-job flow shop sequencing problem," *Omega*, vol. 11, no. 1, pp. 91-95, 1983.

[20] M. I. Hosny, C. L. Mumford, "The single vehicle pickup and delivery problem with time windows: intelligent operators for heuristic and metaheuristic algorithms," *Journal of Heuristics*, vol. 16, no. 3, pp. 417-439, 2010.

[21] L. J. J. Van der Bruggen, J. K. Lenstra, and P. C. Schuur, "Variable-depth search for the single-vehicle pickup and delivery problem with time windows," *Transportation Science*, vol. 27, no. 3, pp. 298-311, 1993.

[22] Z. Al Chami, H. Manier, M.-A. Manier, and C. Fitouri, "A hybrid genetic algorithm to solve a multi-objective pickup and delivery problem," *IFAC-PapersOnLine*, vol. 50, no.1, pp. 14656-14661, 2017.