OCD: Online Crowdsourced Delivery for On-Demand Food

Article *in* IEEE Internet of Things Journal · July 2019

DOI: 10.1109/JIOT.2019.2930984

CITATIONS
5

READS
1,772

6 authors, including:

Wei Tu
Shenzhen University
81 PUBLICATIONS **1,340** CITATIONS
SEE PROFILE

Tianhong Zhao
Shenzhen University
8 PUBLICATIONS **47** CITATIONS
SEE PROFILE

Baoding Zhou
Shenzhen University
33 PUBLICATIONS **616** CITATIONS
SEE PROFILE

Jincheng Jiang
Shenzhen institute of advanced technology
16 PUBLICATIONS **94** CITATIONS
SEE PROFILE

OCD: Online Crowdsourced Delivery for On-Demand Food

Wei Tu, *Member, IEEE*, Tianhong Zhao, Baoding Zhou, Jincheng Jiang, Jizhe Xia, and Qingquan LI

***Abstract*—Online-to-offline (O2O) commerce connecting service providers and individuals to address daily human needs is quickly expanding. In particular, on-demand food, whereby food orders are placed online by customers and delivered by couriers, is becoming popular. This novel urban food application requires highly efficient and scalable real-time delivery services. However, it is difficult to recruit enough couriers and route them to facilitate such food ordering systems. This article presents an online crowdsourced delivery approach for on-demand food. Facilitated by Internet-of-Things and 3G/4G/5G technologies, public riders can be attracted to act as crowdsourced workers delivering food by means of shared bicycles or electric motorbikes. An online dynamic optimization framework comprising order collection, solution generation, and sequential delivery processes is presented. A hybrid metaheuristic solution process integrating the adaptive large neighborhood search and tabu search approaches is developed to assign food delivery tasks and generate high-quality delivery routes in a real-time manner. The crowdsourced riders are dynamically shared among different food providers. Simulated small-scale and real-world large-scale on-demand food delivery instances are used to evaluate the performance of the proposed approach. The results indicate that the presented crowdsourced food delivery approach outperforms traditional urban logistics. The developed hybrid optimization mechanism is able to produce high-quality crowdsourced delivery routes in less than 120 seconds. The results demonstrate that the presented online crowdsourced delivery approach can facilitate city-scale on-demand food delivery.**

***Index Terms*—On-demand food delivery, crowdsourcing, hybrid optimization, heuristics, urban logistics, smart cities**

I. Introduction

THE prevalence of online-to-offline (O2O) commerce using smartphones has led to increasing popularity of smart urban living [1]–[6]. In particular, the on-demand food business pools large numbers of small catering businesses, customers, and couriers to provide ubiquitous and scalable food services. Catering shops accept orders placed by users on smartphones or computers and prepare personalized meals and drinks. Then, couriers deliver the food to the corresponding customers as soon as possible. The on-demand food market in the United States reached nearly 17 billion dollars in 2017 [7]. In China, 300 million people have placed at least one on-demand food order via an O2O service provider [8]. It is reported that the number of daily on-demand food orders in metropolitan areas such as Beijing and Shenzhen has exceeded 500,000. Consequently, flexible and efficient food delivery is required to support this booming sector.

Traditional urban catering businesses, such as McDonald's and Domino's Pizza, rely on professional logistics services to deliver food. In general, a catering shop may maintain a self-owned delivery team or may cooperate with a third-party logistics service provider to provide point-to-point food delivery service [5], [9]. Traditional urban logistics is reliable but also expensive [10], and it cannot adequately address on-demand food delivery. From a long-term perspective, many small catering shops frequently enter and leave the on-demand food market. Maintaining a fleet of delivery men would be very expensive for these small catering shops. From a short-term perspective, the volume of food orders placed at catering shops fluctuates, and resizing the delivery team accordingly is not easy. On the other hand, customers are time sensitive; they may switch to other food modes rather than order food online if food delivery is subject to long delays.

Crowdsourcing is an efficient approach to taking advantage of collaboration among individuals to cooperatively complete tasks [11]–[14]. In particular, using Internet-of-Things (IoT) and 3G/4G/5G technologies, a large number of workers can be rapidly attracted and outsourced for various types of spatio-temporal tasks [13], [15], e.g., mobile sensing [16], [17], indoor localization [18]–[20], road surface monitoring [21], or air quality monitoring [22]. A few studies have focused on crowdsourced urban logistics using ubiquitous urban travel modalities [23]–[28], such as taxi travel [29] and private vehicle travel [30]. For example, Chen et al. [10] developed a novel crowdsourced logistics system to deliver packages by leveraging existing taxis, thereby reducing the time sensitivity of the task. This pioneering study demonstrated the feasibility of crowdsourcing in the last-mile delivery domain. However, real-time on-demand food delivery based on crowdsourcing has not been well investigated.

Here, we present an online crowdsourced delivery (OCD) approach for food that coordinates among restaurants, customers, and crowdsourced riders. Fig. 1 displays an overview of the crowdsourced on-demand food delivery system. The customers place their food orders with the catering shops via the food cloud. The catering shops accept the customer orders and produce personalized food packages, which are then

W. Tu, T. Zhao, B. Zhou, J. Jiang, J. Xia, and Q. Li are with the Guangdong Key Laboratory of Urban Informatics, the Shenzhen Key Laboratory of Spatial Smart Sensing and Services, and the Research Institute of Smart Cities, Department of Urban Informatics, School of Architecture and Urban Planning, Shenzhen University, Shenzhen 518060, China. W. Tu is also with the Senseable City Laboratory, Massachusetts Institute of Technology. (e-mail: tuwei@szu.edu.cn, zhaoteanhong@foxmail.com, bdzhou@szu.edu.cn, j.jiang@szu.edu.cn, jzxia@szu.edu.cn, liqq@szu.edu.cn). Corresponding author: W. Tu.

This work was jointly supported by the Basic Research Program of the Shenzhen Science and Technology Innovation Commission (Nos. JCJY20180305312511388 3 and JCYJ20170412105839889) and the National Natural Science Foundation of China (Nos. 71961137003 and 41401444)

delivered by crowdsourced riders rather than the restaurants' own delivery staff. The crowdsourced riders log into the OCD framework, accept their assigned delivery tasks, go to the restaurants, pick up the food packages and deliver them to the corresponding customers. Members of the public are attracted to act as delivery riders via the food cloud. Facilitated by IoT and 3G/4G/5G technologies, such public riders can perform delivery tasks by means of shared bicycles or electric motorbikes. Therefore, the presented OCD approach can elastically expand to satisfy dynamic customer order demands. From the caterers' perspective, the OCD approach can decrease the cost of recruiting delivery teams. From the city's perspective, the OCD approach will benefit part-time laborers and reduce both traffic congestion and the consequent carbon emissions.

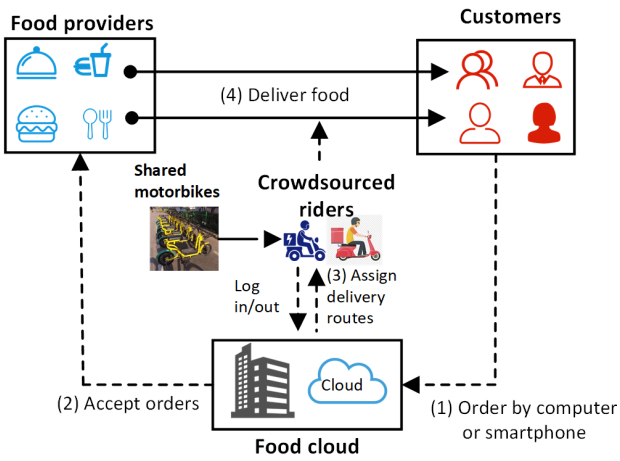

Fig. 1. The crowdsourced on-demand food delivery system.

In contrast to traditional urban logistics, the presented crowdsourced on-demand food delivery system has three distinctive properties: (1) The crowdsourced riders are floating resources. Crowdsourced riders can serve various catering establishments and customers, rather than being bound to a single restaurant as in traditional urban logistics. The riders are highly flexible because they log into/out of the crowdsourced food delivery system based on their own schedules. (2) The crowdsourced food delivery system is dynamic. In contrast to traditional urban logistics based on precollected information, orders and riders enter the crowdsourced food delivery system over time and are processed in a real-time manner. (3) On-demand food delivery is a time-sensitive task. A customer might reject a food delivery if it is severely delayed. Therefore, achieving efficient crowdsourced delivery is the key to success. These properties highlight the necessity of efficient and effective OCD services.

Focusing on the growing on-demand food business, we have developed an online dynamic optimization framework for assigning food delivery tasks to crowdsourced riders. The delivery of one food order is treated as a crowdsourced task. Urban residents are recruited as floating crowdsourced riders. Considering the time windows of the customers and the available time windows of the crowdsourced riders, three parallel processes, i.e., order collection, solution generation, and route navigation, are performed to facilitate on-demand food delivery. A mathematical model of the crowdsourced delivery problem is formulated to minimize the total travel cost and delivery delay. The problem is solved with a hybrid metaheuristic algorithm integrating the adaptive large neighborhood search (ALNS) and tabu search (TS) approaches. Both simulated and real-world applications in Shenzhen demonstrate the effectiveness and efficiency of the presented approach and the dynamic optimization framework.

This study makes the following contributions:

- **A crowdsourced delivery approach** is presented for on-demand food service. To the best of our knowledge, this is the first study to apply the crowdsourcing principle in the on-demand food delivery domain.
- **An online dynamic optimization framework** is developed to address the time preferences of the customers and crowdsourced riders. The presented framework is elastic and robust and thus is able to facilitate on-demand food delivery in a real-time manner.
- **A hybrid solution method** is proposed to address the complexity of the OCD problem. The ALNS and TS methods are hybridized to balance search intensification and diversification to achieve improved routing performance.
- **An intensive experiment** using simulated small- to medium-scale and real-world large-scale food delivery instances is conducted for performance evaluation. The results suggest that the presented approach is superior to traditional logistics and baseline algorithms.

The remainder of this paper is organized as follows: Section 2 defines the components of the problem and introduces the mathematical model. Section 3 presents the online dynamic optimization framework. Section 4 describes the hybrid solution method. Section 5 describes the conducted experiment, reports the experimental results and evaluates the performance of the presented framework. Section 6 concludes this study and discusses future work.

II. Related Work

In this section, we present our investigation on crowdsourced task allocation and urban logistics.

A. Task allocation based on spatial crowdsourcing

Spatial crowdsourcing has been used to leverage the power of the public for many different applications, from spatial sensing [17], [22], [31] to problem solving [9], [20], [32], [33]. The key research issue related to spatial crowdsourcing is how to allocate spatial tasks to crowdsourced workers subject to certain constraints, such as worker count, incentive cost, and quality of service [17]. Two approaches have been developed for the allocation of crowdsourced tasks: the pull mode and the push mode [12], [34]. In the pull mode, the crowdsourcing platform collects and publishes tasks, and the workers select tasks in accordance with their personal preferences [35], [36]. In the push mode, tasks are assigned to maximize the system's overall goal, e.g., a certain total working time or total budget [9], [37]. The solutions for the push mode fall into two

categories: task matching and task scheduling [12]. In a task matching method, a bipartite graph is constructed to represent the crowdsourced workers and tasks, which are then matched using the Hungarian algorithm [38] or the K-M matching algorithm [11]. Only one task is assigned to any worker. In a task scheduling method, multiple tasks may be assigned to a single worker by coordinating multiple workers. Both single-objective and multiobjective allocation models have been proposed for task scheduling. The greedy approach has been employed to find near-optimal spatial crowdsourcing allocation solutions [16], [17]. Recently, heuristics such as local search [9], [10], [30] and genetic optimization [39], [40] have been adopted to find high-quality solutions for crowdsourced task scheduling problems within a reasonable computing time.

B. Urban logistics

Urban logistics refers to the management of the pickup and delivery of goods and parcels so as to fulfill human demands in a city. In general, urban logistics relies on professional transportation services with self-owned vehicles and couriers; therefore, this approach is high in cost and difficult to scale [10].

With the development of information and communication technology (ICT) and IoT technology, alternative urban logistics modes based on ubiquitous devices have been developed that can improve efficiency and reduce logistics costs; examples include smart lockers, pick-own-parcel stations (POPstations), and crowdsourced urban logistics [16], [41]. Crowdsourced urban logistics leverages current elements of urban mobility (e.g., commuters and taxis) to provide ubiquitous delivery services. Recently, Devari et al. [30] presented a novel crowdsourced last-mile delivery mode that exploits the social networks of retail store customers. They reported that retailers could reduce their truck mileage by 57%. Wang et al. [16] developed an effective large-scale mobile crowd-tasking model to coordinate citizen workers to perform last-mile package delivery. These studies demonstrate the effectiveness of crowdsourcing in urban logistics. Regarding food delivery, Liu et al. [9] presented a food delivery network (FDN) based on ride sharing between passengers and food deliveries using taxis, in which the taxi drivers are assumed to deliver food opportunistically. Here, we present an alternative crowdsourced food delivery approach that leverages ubiquitous urban travel via bicycles or motorbikes, which are more attractive in urban environments. The riders are floating resources that are shared by many food providers. A dynamic optimization framework based on hybrid metaheuristics is developed to assign delivery tasks by considering the time preferences of customers and travelers on the road.

III. Problem definitions and mathematical model

A. Definitions

The proposed crowdsourced on-demand food delivery system consists of three components: customers, food providers, and crowdsourced riders.

Definition 1 (Customer): A *customer* is a person who places a food order using his/her computer or smartphone. A customer is represented by his/her ID (i) and address. The address can be geocoded to a real-world position (l_i), i.e., longitude and latitude. A customer can place a food order with a preferred food provider and for a preferred delivery time t_i^e.

Definition 2 (Food provider): A *food provider* accepts customers' orders based on its food producing capabilities. The food providers in the market consist of restaurants, drink shops, cake shops, etc. Each food provider must register with the crowdsourced on-demand food delivery system, in which it is represented by an ID j and a position l_j.

Definition 3 (Crowdsourced rider): A *rider* is a crowdsourced worker who performs assigned food delivery tasks. Following the provider and order sequence along a delivery route, a rider picks up food and delivers it to customers. A rider is represented by his/her ID (k), location (l_k), capacity (q_k), and logged-on time (t_k^{on}), as follows: $r \equiv< k, l_k, q_k, t_k^{on} >$. A logged-on rider is available to accept delivery tasks. When the rider logs out, he/she becomes inactive, meaning that he/she automatically rejects food delivery tasks.

Definition 4 (Delivery task): A *delivery task* is a task consisting of the delivery of food from a particular food provider to a particular customer. At the most basic level, a delivery task (O_{ji}) can be represented in terms of the corresponding provider (j) and customer (i). By incorporating the provider location, the customer location, and the customer's preferred time, a delivery task can be represented by $O_{ji} \equiv< j \rightarrow i, l_j, l_i, t_i^e >$.

Definition 5 (Delivery route): A *delivery route* is a sequence of segments connecting riders, food providers, and customers, represented as $r \equiv< j_1, i_1, i_2, ..., j_2, i_n, ... >$. Such a route prescribes how a rider should travel from his/her current position to providers j_1 and j_2, pick up food, and deliver that food to the corresponding customers $i_1, ..., i_n$. The sequence of riders, food providers and customers represents the pickup and delivery of orders.

The goal of OCD is to assign crowdsourced riders to deliver food from providers to customers. Fig. 2 shows an example of on-demand OCD for food services. Three routes are generated for three riders ($k1$, $k2$, and $k3$) delivering food from three providers ($j1$, $j2$, and $j3$) to ten customers ($i1, ..., i10$). Rider $k1$ travels to food provider $j1$, picks up 3 orders of food, and delivers them to customers $i1$, $i2$, and $i3$. Rider $k2$ travels to food provider $j2$, picks up 3 orders of food, and delivers them to customers $i4$, $i5$, and $i6$; then, rider $k2$ picks up 2 more food orders from provider $j3$ and delivers them to customers $i7$ and $i8$. Rider $k3$ delivers food from provider $j3$ to customers $i9$ and $i10$. Note that the riders can pick up food from multiple different food providers and thus are not restricted to any particular provider or customer, as demonstrated by the example of rider $k2$, which makes the proposed crowdsourced delivery technique quite different from traditional urban logistics.

B. Mathematical model

The goal of the presented OCD problem is to assign customer orders to crowdsourced riders and simultaneously design

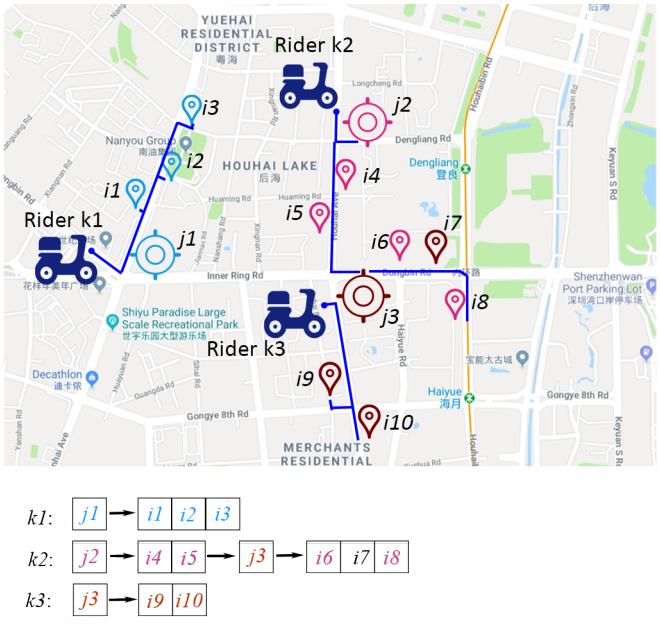

Fig. 2. Crowdsourced food delivery.

delivery routes. This type of problem belongs to the family of dial-a-ride problems [42], in which customer demands are fulfilled by dispatching vehicles. Without loss of generality, we adopt the following assumptions regarding the OCD problem: (1) The riders are willing to accept the food delivery tasks assigned by the system because they can be rewarded for doing so. (2) All accepted food orders must be delivered, as riders can deliver several times. In other words, the crowdsourced food delivery system does not reject any food orders. Formally, the OCD problem is formulated as follows.

The following are given:

a food order set $I \equiv< i_1, i_2, \ldots, i_x >$,

a food provider set $J \equiv< j_1, j_2, \ldots, j_y >$, and

a rider set $K \equiv< k_1, k_2, \ldots k_z >$.

Considering real-world food delivery scenarios, two constraints are imposed:

(1) Each accepted food order must be fulfilled and can be fulfilled only once. In other words, all food orders in the set I will be produced by the corresponding food providers and delivered by the crowdsourced riders.

(2) The capacity of a rider is limited. Based on the size of the crowdsourced rider's bicycle, motorcycle, or car, the rider's load should be no more than his/her capacity q_k at any given time.

The OCD problem can be represented as a directed graph $G \equiv (A, E)$. The node set $A \equiv < I, J, K > \equiv < 1, ..., |I| + |J| + |K| >$ is the union of the food orders, food providers, and crowdsourced riders. Each element in A corresponds to a food provider, a food order, or a crowdsourced rider. $|I|$, $|J|$, and $|K|$ denote the numbers of elements in the corresponding sets. The edge set E represents route segments traveling from one node to another node. Each edge e_{ab} denotes a route segment from node a to node b, with travel cost c_{ab} and travel time t_{ab}. Three types of decision variables are employed, as follows:

x_{ab}^k, a binary variable that is equal to 1 if and only if crowdsourced rider k travels from node a to node b;

T_a^k, the time of delivery at node a by crowdsourced rider k; and

Q_a^k, the load of crowdsourced rider k when leaving node a.

The goal of OCD is to generate routes for delivering food to the correct customers as soon as possible at the lowest cost. Consequently, the objectives of the OCD problem are two-fold: (1) The total travel cost of all crowdsourced riders during the delivery process should be minimized. (2) The delivery delay to customers should be minimized. Because of road traffic, the arrival time T_a^k of rider k at customer a may be later than the preferred time, t_a^e. The difference between the predefined preferred time and the delivery time, $T_a^k - t_a^e$, is defined as a delay if T_a^k is later than t_a^e. The mathematical model of the OCD problem is defined as follows:

Objectives:

$$\textbf{min} \quad F = \beta_1 \sum_{k\in K}\sum_{a\in A}\sum_{b\in A} x_{ab}^k * c_{ab} + \beta_2 \sum_{a\in I} \max(0, T_a^k - t_a^e) \tag{1}$$

s.t. :

$$\sum_{a\in I\cup J}\sum_{k\in K} x_{ab}^k = 1 \quad \forall b \in I, \tag{2}$$

$$\sum_{b\in I\cup J}\sum_{k\in K} x_{ab}^k = 1 \quad \forall a \in I, \tag{3}$$

$$\sum_{a\in I\cup J} x_{ab}^k = \sum_{a\in I\cup J} x_{ba}^k \quad \forall b \in I, \forall k \in K, \tag{4}$$

$$T_a^k + t_{ab} = T_b^k \leftarrow x_{abk} = 1 \quad \forall a \in I, \tag{5}$$

$$Q_a^k \le q_k \leftarrow \sum_{b\in I} x_{ab}^k = 1 \quad \forall k \in K, \tag{6}$$

$$x_{ab}^k \in \{0, 1\} \quad \forall a, b \in I \cup J \forall k \in K. \tag{7}$$

The objectives are the weighted travel cost and delivery delay, as shown in Eq. 1, where $\beta_1, \beta_2 \in [0, 1]$ are weight parameters. The constraints defined in Eqs. 2 and 3 require that all accepted customer orders must be produced and delivered. Eq. 4 requires a crowdsourced rider to leave a customer's location after delivering that customer's order. Eq. 5 states that the arrival time at customer b is equal to the arrival time at the previous customer a plus the travel time from node a to node b, t_{ab}. Note that the travel time t_{ab} can be updated by means of dynamic transportation analysis techniques [43]. For example, by obtaining the average travel speeds from an advanced traffic information system (ATIS) or simulation [44], [45], the time-dependent travel times on different roads can be estimated [46]. Eq. 6 guarantees that the total amount delivered by crowdsourced rider k is no greater than his/her capacity q_k. Eq. 7 defines the decision variables.

The solution to the OCD problem is a set of delivery routes representing food pickup, travel, and delivery sequences, similar to what is illustrated in Fig. 2. Such a solution will help crowdsourced riders navigate to deliver food in a complex urban environment.

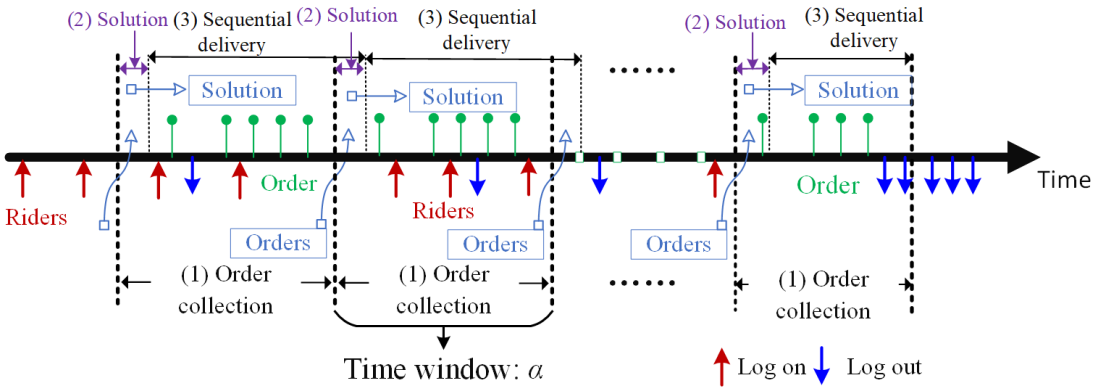

Fig. 3. The dynamic optimization framework for crowdsourced food delivery. Riders log into/out of the crowdsourcing food delivery system in accordance with their schedules. Three processes, namely, order collection (1), solution generation (2), and route navigation (3), are iteratively implemented for on-demand crowdsourced food delivery.

IV. Dynamic Optimization Framework

In traditional urban logistics, all information about warehouses, customers, and delivery men is required to be collected in advance to plan the delivery of packages or food [47]. In contrast, in a crowdsourced on-demand food delivery system, customers place their food orders via smartphones or computers, thus generating orders in a near-real-time manner. Therefore, the OCD problem is highly dynamic. Here, we present an online dynamic optimization framework based on a short-sight strategy for food delivery task allocation. Fig. 3 illustrates the presented framework, which includes three processes: order collection, solution generation, and route navigation.

Order collection. This is the process of collecting the order and rider information. Customer orders are accepted by food providers in accordance with their food production capabilities. The active riders are recorded, and their current positions are updated. If an active rider logs out of the OCD system, that rider will be labeled as inactive and will not be considered in the subsequent solution generation process.

Solution generation. This is the process of allocating food delivery tasks to active riders. The optimal food delivery task allocation is generated to guide riders to pick up food orders and deliver them to the corresponding customers. Considering the system performance, the total travel cost and delivery delay are minimized using the hybrid metaheuristics described in the next section to coordinate the crowdsourced riders.

Sequential delivery. Once the solution has been obtained, each rider will receive a personalized delivery sequence describing his/her visits to food providers and customers. Each rider will visit a set of food providers and customers following the specified sequence. The riders will navigate using online map services such as Google Maps or Open Street Maps to ensure effective food delivery in a complex urban environment.

Note that a sliding time window α is used to begin the solution generation process. From the life-cycle perspective, the length of this time window corresponds to the degree of dynamism of the OCD problem. Here, we define $\frac{1}{\alpha}$ as the degree of dynamism. A longer time window α means a longer order collection period and thus implies a lower degree of dynamism. In this case, more customer orders will be accepted during the order collection process, and the solution generation process will be faster. By contrast, a smaller α implies a higher degree of dynamism and requires the solution generation process to be performed more frequently.

V. Hybrid Solution Method

The OCD problem is a variant of the classical dial-a-ride problem, which has the characteristic of nondeterministic polynomial-time (NP) hardness. Hence, it is impossible to find the globally optimal solution for a real-world large-scale OCD instance [48], [49]. However, heuristic algorithms are able to find high-quality solutions for many complex problems within a reasonable time. Here, we propose a hybrid metaheuristic method for assigning delivery tasks to crowdsourced riders. The workflow of the developed hybrid metaheuristic solution method is presented in Fig. 4. An initial solution is first generated using a simple heuristic algorithm. Then, the solution is iteratively improved to reduce the travel cost and possible delivery delay. The generated optimal solution is then reported and passed to the route navigation process.

A. Generating an initial solution

A good initial solution will significantly benefit the subsequent solution improvement procedure. In essence, the initial solution generation process consists of two steps: assigning active riders to food providers and generating initial delivery routes.

Active riders are first assigned to food providers because each rider will first need to travel to a food provider. This can be regarded as a problem of matching between the riders and food providers. In general, a food provider may need many riders because it may accept many food orders. Rider-provider matching is achieved by means of a cost-based matching algorithm. The objective of the matching procedure is to minimize the total travel cost for the riders traveling to the food providers. The matching procedure is described in Alg. 1. The travel costs from each rider to each provider are first calculated. Here, we consider either the network travel distance

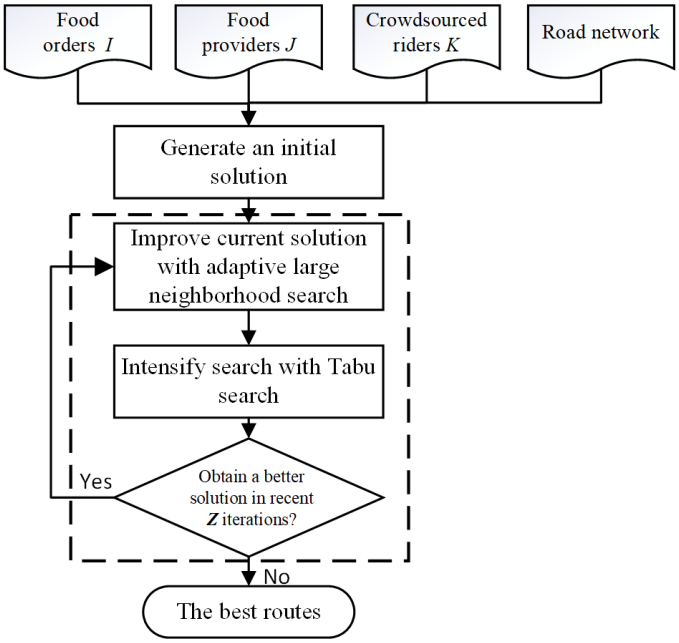

Fig. 4. The workflow of the presented hybrid metaheuristic method.

or travel time as the travel cost. All possible rider-provider pairs are stored in a priority queue (*lines 0-1*). Then, the rider-provider pairs are sequentially popped to examine whether the current total capacity of the rider at the food provider, q_j, is less than that provider's total number of orders, Q_j (*line 6*). If so, the rider is assigned to the food provider, and the rider's current total capacity at the provider is updated (*lines 7-8*). This process continues until all rider-provider pairs have been popped.

Algorithm 1 Rider-provider matching

Input: J: the provider set; K: the rider set
Output: the initial routes $k(p)$

1: $c_{kj} \leftarrow calculate\_travel\_cost(K, J)$
2: store c_{kj} in a queue L
3: initialize rider status: $s_k =$ **False**
4: initialize current total load: $q_j = 0$
5: **for all** $c_{kj} \in L$ **do**
6: **if** $s_k =$ **False** **and** $q_j + q_k < Q_j$ **then**
7: $s_k =$ **True**, $q_j += q_k$
8: update the rider's provider: $k_j = j$
9: **end if**
10: **end for**

Route generation is performed by means of an insertion heuristic, as shown in Alg. 2. For each active rider, an initial route is created with the first travel segment being from the rider's current location to the matched food provider. All orders are sorted in accordance with their preferred times t_i^e. Following the sorted sequence, each order is inserted at the best route point with the minimal insertion cost, which is calculated as the corresponding increase in the objective value.

If order i is inserted after an order a placed to the same provider, then the insertion cost is estimated as shown in Eq. 8, where b is the node subsequent to a along the route if such a subsequent node exists or $c_{ib} = 0$ and $c_{ab} = 0$ otherwise. a^- is the subsequent order set after a in the route into which the order is inserted.

$$\Delta f_i^a = \beta_1(c_{ai} + c_{ib} - c_{ab}) + \beta_2 \sum_{z \in a^-} \max(0, T_z^k - t_z^e). \quad (8)$$

If order i was placed to a different food provider j, then provider j must first be inserted before i. Hence, the insertion cost is equal to the sum of the increases in the objective value corresponding to the order i and its food provider j, as shown in Eq. 9, where a^+ denotes the nodes before order a, x denotes the route point where provider j is to be inserted, z denotes an order after x, and x^- is the set of nodes subsequent to point x along the route into which the order is inserted.

$$\Delta f_i^a = \beta_1(c_{ai} + c_{ib} - c_{ab}) + \beta_1 \min_{x \in a^+ \cup i}(c_{xj} + c_{jy} - c_{xy}) + \beta_2 \sum_{z \in x^-} \max(0, T_z^k - t_z). \quad (9)$$

Considering the load capacity of a rider, an insertion will be rejected if the rider's current load exceeds his/her capacity q_k. Once all orders have been sequentially inserted, the initial routes are generated.

Algorithm 2 Insertion heuristic algorithm

Input: I: the order set; J: the provider set; K: the rider set; c_{ab}: the travel costs; t_{ab}: the travel times
Output: the initial routes K

1: $sort\_orders(I)$
2: inserted node set: $D = \varnothing$
3: **for all** $i \in I$ **do**
4: $f_{min} \leftarrow \infty$, $i_{best} = -1$
5: **for all** $a \in D$ **do**
6: initialize route for $a : k \leftarrow k(a)$
7: **if** $f_{min} > insert\_cost(a, i)$ **and** $Q_k < q_k$ **then**
8: $f_{min} = insert\_cost(a, i)$
9: $i_{best} = a$
10: **end if**
11: $insert\_node(i_{best}, i)$
12: $D \leftarrow D \cup i$
13: **end for**
14: **end for**

B. Solution improvement

The initial solution is a feasible assignment of the food delivery tasks to the riders. However, severe time window violations and improper delivery task assignments might exist; therefore, the initial solution should be further improved. Solution improvement is achieved by changing parts of the food delivery assignment through a combination of ALNS and TS. Generally, the ALNS process greatly improves the current solution by destroying and repairing part of the current solution. Because of the large affected solution neighborhood,

ALNS represents search diversification in the solution space. By contrast, in the TS process, the current solution is adjusted by relocating a node or exchanging the positions of two nodes. Thus, TS represents search intensification in the solution space. The details of the proposed hybrid method are described as follows.

(1) Destroying routes

First, a set of current routes is destroyed. Five destruction operators for removing a set of nodes from the current routes are proposed below.

- **Random removal operator (RRO)**. This operator randomly removes N nodes to perturb the current solution.
- **Time window operator (TWO)**. Many orders may be delayed because of possibly improper delivery task assignments in the initial routes. This operator checks the delivery times of all orders to identify time window violations. These violations are sorted, and the top-N violating orders are ejected from the current routes.
- **Longest-segment removal operator (LRO)**. The current routes may contain long travel segments, which will strongly affect the travel cost. This operator identifies the longest travel segments in the current routes and removes their endpoints.
- **Spatio-temporal similarity operator (STO)**. This operator removes a set of orders in close spatio-temporal proximity to adjust local route segments. An order a is randomly selected as the seed. The top-N nearest orders to the seed are identified on the basis of their spatio-temporal similarity as expressed in Eq. (10), where $ST(a,b)$ denotes the similarity between orders a and b as indicated by the travel cost c_{ab} and the time difference $|t_a^e - t_b^e|$.

$$ST(a,b) = \beta_1 c_{ab} + \beta_2 |t_a^e - t_b^e|. \tag{10}$$

- **Clustering operator (CLO)**. In contrast to the STO, this operator removes a set of clustered orders. All accepted orders are pregrouped using the *K*-means algorithm [50] on the basis of the spatio-temporal similarity defined in Eq. 10. The value of *K* is set to $\left\lceil \frac{|I|}{N} \right\rceil$, where $|I|$ is the number of orders and N is the expected size of a cluster. The CLO removes a group of orders from the current routes.

In each iteration, we select one destruction operator and use it to eject nodes from the current solution.

(2) Repairing routes

The destroyed routes are then repaired by reinserting the ejected orders. In essence, the insertion process is similar to the initial route generation process. We have developed two repair operators, as follows.

- **Greedy insertion operator (GIO)**. For one order to be inserted, this operator selects the best position such that the insertion cost, as calculated using Eq. 8 or Eq. 9, is minimized.
- **Regret insertion operator (RIO)**. One problem with the GIO is that it may postpone the insertion of certain removed orders. Because each insertion will affect the subsequent insertion of the next order, the best insertion position may not be satisfactory for all ejected nodes. This operator attempts to circumvent this problem by incorporating a type of look-ahead information when selecting the insertion position. Let Δf_i^j denote the total change in the objective value incurred with the insertion of order i into its best position in the j-th cheapest route for this order. For each insertion, the RIO chooses to insert order i at the position that maximizes $\Delta f_i^2 - \Delta f_i^1$. In other words, the difference between the costs of inserting order i into its best position and its second-best position is maximized. We iteratively reinsert all removed orders to reconstruct new delivery routes.

In each iteration, we select one repair operator to reinsert all ejected orders. Notably, the reinsertion performance depends on the reinsertion sequence. Therefore, we randomly sort all ejected orders and then reinsert them into their chosen positions one by one. Without loss of generality, we repeat this process of sorting and reinsertion N times, where N is the number of ejected nodes, and finally select the best insertion result that minimizes the increase in the total objective value.

(3) Adaptive weighting strategy

The performance of the ALNS process relies on the destruction and repair operators. We employ an adaptive roulette wheel mechanism [51] to select the pair of implemented destruction-repair operators. Initially, all operators have the same probability of implementation. After one round of removal and reinsertion, the increase Δf_o in the objective value incurred with the applied operator o during the search is recorded. The weight of operator o in iteration z, ω_z^o, is then updated according to Eq. 9. The factor ρ controls how quickly the weight adjustment mechanism reacts to improvements in the objective value. $\rho = 1$ means that the roulette wheel selection is based only on the improvement in the most recent iteration, while with $\rho < 1$, the historical improvement is considered.

$$\omega_z^o = \rho \frac{\Delta f^o}{\sum_{o \in O} \Delta f^o} + (1-\rho)\omega_{z-1}^o. \tag{11}$$

(4) Acceptance strategy

The normal destruction and repair operations will cause the improvement process to quickly become trapped in local optima. Following the approach used by Pisinger and Popke [52] for the vehicle routing problem, we employ the simulated annealing strategy to accept temporary solutions that are worse than the current solution using the Metropolis criteria. An initial temperature H and a cooling ratio η are set such that the temperature will decrease in accordance with $H_z = \eta H_{z-1}$. A worse solution will be accepted with probability $\exp -\frac{\Delta f}{H_z}$, and thus, the current solution will converge to a high-quality optimal solution with decreasing temperature.

(5) Tabu search

To further improve the routes, the TS metaheuristic is incorporated for search intensification in the solution space. Instead of focusing on a set of ejected nodes, the TS metaheuristic operates on only one or two nodes per iteration. Two neigh-

borhood search operators, namely, the 1-0 and 1-1 exchange operators [47], as illustrated in Fig. 5, are used to adjust the local route segments. The 1-0 exchange operator relocates one order from its current position to another position. The 1-1 exchange operator swaps the positions of a pair of orders. Because both the 1-0 and 1-1 exchange operators involve two orders, we randomly select one order and evaluate its k-th Voronoi neighbor following the strategy of Tu et al. [47]. The order pair with the best solution improvement will be operated upon. The tabu list is restricted to recently operated-on orders. The length of the tabu list is set to be no greater than $\sqrt{|I|+|J|}$.

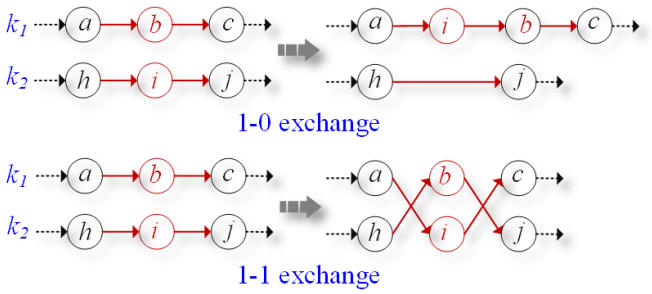

Fig. 5. Local route operators.

C. Predispatching idle crowdsourced riders

Both food orders and crowdsourced riders enter the system dynamically. Therefore, a mismatch may occur when too many orders are sent to one provider when few active riders are available nearby, which will significantly increase the delivery delay. Here, we propose a rider predispatching strategy to address this issue. After solution generation, the diversity of the riders visiting a given food provider is measured in terms of the visiting entropy, as defined in Eqs. 12 and 13, where v_k^j is the number of visits of rider k at provider j and r_k^j is the visiting ratio of rider k at provider j during solution generation. The larger the entropy at a provider and the greater the diversity in rider visitation are, the greater is that provider's popularity with customers. By weighting the food providers in terms of their entropies, we can improve the rider-provider matching process described in Section IV.A to more effectively dispatch idle riders and newly active riders, thereby reducing delivery delays.

$$r_k^j = \frac{v_k^j}{\sum_{k \in K} v_k^j}. \tag{12}$$

$$Entropy(j) = \sum_{k \in K} r_k^j log(r_k^j). \tag{13}$$

VI. Experiment and Results

A. Experimental setup

We designed an experiment to examine the performance of the proposed OCD approach. Two datasets were generated to evaluate the performance in terms of the solution quality, the solution efficiency and the effects of various factors.

- **The simulation dataset S** contains 15 instances. Each instance consists of a set of providers, orders, and riders. The locations of the providers, orders, and riders have been randomly generated from within the range [(0, 0), (100, 100)]. The time windows are also randomly set within the time span T=[1, 100].The number of orders per instance ranges from 5 to 100 to represent small- to medium-scale OCD instances. The travel speed is set to 1. The details of the instances in **S** are listed in Tab. I. This simulation dataset was used to evaluate the solution quality.
- **The real-world OCD dataset SZ** consists of 7 instances from Shenzhen, China. The dataset covers the central business district of Shenzhen. Each instance consists of 917-1390 orders along with a set of providers and crowdsourced riders. The customers' preferred delivery times span the lunch period, from 11:00 AM to 2:00 PM. The travel speed is set to the average of the speeds of a bicycle and a motorcycle. The details of **SZ** are listed in Tab. VI-A.

The presented dynamic crowdsourced food delivery framework was implemented in Python. The experiment was conducted on a personal computer equipped with an Intel(R) Core(TM) i7-4790 CPU @3.60 GHz and 8 GB of RAM. Two baseline algorithms were employed to solve the OCD instances for comparison with the presented hybrid solution method. The first of these algorithms was the mixed-integer linear programming (MILP) algorithm implemented in IBM ILOG CPLEX [53], the most effective solver for complex combinatorial optimization problems [49]. Because of the NP-hard nature of the problem of interest, CPLEX was employed only to solve the small- and medium-scale instances in **S**. The computing time for CPLEX was limited to 7200 seconds. The second algorithm used for comparison was the normal TS metaheuristic implemented in Python. This TS implementation is similar to that in the presented hybrid algorithm.

Regarding the setting of the parameters, after an investigation of food delivery pricing strategies in Shenzhen, the weights in the overall objective function (Eq. 1) were set in accordance with the real-world costs. The weight of the travel cost term, β_1, was set to the average travel cost per km, 3.33 Chinese Yuan (CNY). The delivery delay penalty, β_2, was set to 0.3 CNY per minute. For the ALNS procedure, the parameters were set following Tu et al. [47]. The initial temperature, H, was set to the average length of a route segment. The cooling ratio, η, was set to 0.95. The maximum number of ejected nodes was set to $\sqrt{|I|+|J|}$. For the TS procedure, the length of the tabu list was also set to $\sqrt{|I|+|J|}$. The parameter settings for the ALNS part and the TS part in the presented hybrid solution method were similar.

B. Results for small- to medium-scale OCD instances

The small- to medium-scale instances in dataset **S** were solved with CPLEX, TS, and the presented hybrid solution method. The total objective value Obj and the computing time $Time$ (in seconds) are reported. The best CPLEX, TS, and hybrid solutions are summarized. The gap between the hybrid

TABLE I
RESULTS FOR SMALL- TO MEDIUM-SCALE OCD INSTANCES

Name	Providers	Customers	Riders	CPLEX		TS		Best	Hybrid method		
				Obj	Time (s)	Obj	Time (s)		Obj	Time (s)	Gap
S1	3	5	3	29.76	0.23	29.76	0.18	29.76	29.76	0.07	0.0
S2	3	5	3	32.64	0.38	32.64	0.24	32.64	32.64	0.06	0.0
S3	3	5	3	19.52	0.28	19.52	0.14	19.52	19.52	0.06	0.0
S4	3	10	3	44.16	335.65	45.44	4.02	44.16	45.44	3.87	2.9
S5	3	10	3	46.4	68.82	46.4	4.26	46.4	46.4	4.08	0.0
S6	3	10	3	36.48	233.45	36.48	4.54	36.48	36.48	4.97	0.0
S7	3	20	5	105.28	7200	101.52	13.92	101.52	99.6	6.53	**-1.9**
S8	3	20	5	161.84	7200	79.12	10.92	79.12	72.4	18.8	**-8.5**
S9	3	20	5	75.36	7200	70.76	10.88	70.76	65.44	7.09	**-7.5**
S10	5	50	10	-	-	163.24	49.13	163.24	154.6	79.15	**-5.3**
S11	5	50	10	-	-	161.96	46.6	161.96	152.48	50.45	**-5.9**
S12	5	50	10	-	-	196.88	38.69	196.88	195.04	33.59	**-0.9**
S13	5	100	20	-	-	298.08	418.91	298.08	279.8	169.2	**-6.1**
S14	5	100	20	-	-	272.52	132.72	272.52	250.84	175.77	**-8.0**
S15	5	100	20	-	-	312.52	124.49	312.52	291.96	208.62	**-6.6**
Mean							**57.3**			**50.8**	**-3.2**

TABLE II
RESULTS FOR REAL-WORLD FOOD DELIVERY INSTANCES

Instance	Providers	Customers	Riders	alpha	TS						Hybrid method						
					Obj	TTime (min)	Dist	Delay	Time_all (s)	Time (s)	Obj	TTime (min)	Dist	Delay	Time_all (s)	Time (s)	Gap
SZ1	26	1250	61	10	5894.83	6797.32	1699.33	768	1696.08	94.23	5369.05	6300.6	1575.15	395.2	1722.88	95.72	-8.92
SZ2	20	926	46	10	4520.64	5325.4	1331.35	276.06	800.25	44.46	4195.1	4968.76	1242.19	181.52	822.21	45.68	-7.2
SZ3	26	1273	62	10	5884.6	6883.32	1720.83	495.04	1404.52	78.03	5430.37	6396.68	1599.17	332.71	1462.14	81.23	-7.72
SZ4	28	1318	65	10	6005.1	7046.76	1761.69	442.65	1674.12	93.01	5499.51	6494.32	1623.58	291.98	1673.38	92.97	-8.42
SZ5	25	1339	66	10	5684.85	6727.88	1681.97	261	1818.87	101.05	5271.54	6241.92	1560.48	233.16	1826.13	101.45	-7.27
SZ6	36	1686	83	10	7075.6	8408.56	2102.14	228.21	1818.55	101.03	6620.32	7831.76	1957.94	312.85	1794.46	99.69	-6.43
SZ7	26	1238	59	10	5724.61	6680	1670	526.5	1198.07	66.56	5288.8	6203.96	1550.99	396.12	1181.97	65.67	-7.61
Mean					**5827.18**	**6838.48**	**1709.62**	**428.21**	**1487.21**	**82.62**	**5382.1**	**6348.28**	**1587.07**	**306.22**	**1497.6**	**83.2**	**-7.65**

solution and the better of the CPLEX and TS solutions is calculated as $Gap = (V - Best)/Best$, where V denotes the objective value of the hybrid solution. A negative gap indicates that the hybrid solution is better than the baseline solution, while a positive gap suggests that it is worse. The obtained results are reported in Tab. I.

The results indicate that the CPLEX solver obtains the best solutions for small-scale OCD instances with no more than 10 customers within a computing time of 340 seconds. Within the maximum allowed time of 7200 seconds, the CPLEX solver reports feasible solutions only for OCD instances with 20 or fewer customers; CPLEX cannot solve the OCD instances with 50 or 100 customers in the allotted time. The TS metaheuristic achieves high-quality OCD solutions for these small- and medium-scale instances within 500 seconds, which is acceptable for real-world on-demand food delivery. Compared to these baselines, the presented hybrid method is more effective and efficient. The hybrid method produces high-quality solutions within at most 210 seconds. The computing effort necessary to generate the hybrid solutions increases approximately linearly with the number of customers. In terms of solution quality, the average gap with respect to the better of the two baseline solutions is -3.2%, which suggests that the hybrid solutions outperform the CPLEX and TS solutions, except for instance S4, for which there is a 2.9% gap.

C. Results for real-world instances

We solved the real-world large-scale on-demand food delivery instances using the TS and hybrid solution methods. The sliding time window α was set to 10 minutes. In other words, a new hybrid solution was generated every 10 minutes. CPLEX was not used because it cannot achieve high-quality solutions in a reasonable computing time for instances of this size. The obtained results are described in Tab. VI-A. The gap is measured as the deviation with respect to the TS solution. TTime (min) denotes the total travel time of all active riders. Time_all (s) denotes the total computing time, whereas Time (s) denotes the average computing time for one solution process.

These results demonstrate that the presented hybrid solution method outperforms the TS method. For the example of instance SZ1, the hybrid solution requires 1575.15 km and results in delivery delays totaling 395.20 minutes, corresponding to approximately 19 seconds per customer. In terms of solution quality, all hybrid solutions are better than the TS solutions. On average, the hybrid solutions are superior to the TS solutions by 7.65%. Regarding efficiency, for the example of instance SZ1, the hybrid solution method requires an average of 95.72 seconds, approximately 15.8% of the data collection period, to report the final solution. Thus, this method can satisfy the requirements for fast and effective real-world OCD. Compared

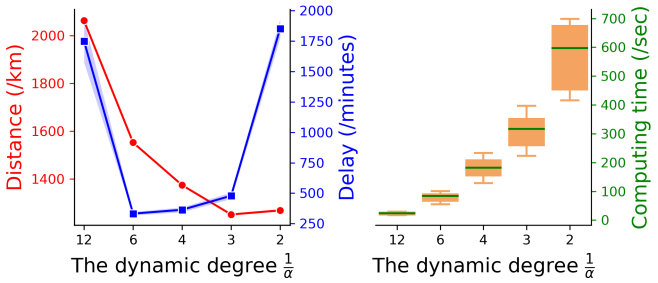

Fig. 6. The effects of the degree of dynamism.

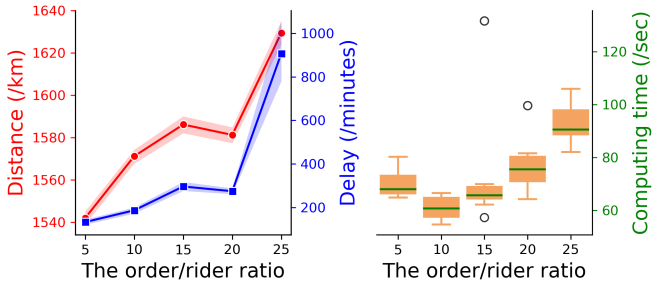

Fig. 7. The effects of crowdsourced rider recruitment.

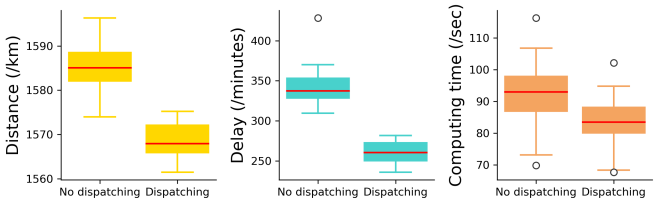

Fig. 8. The effects of rider predispatch.

to the TS metaheuristic, the hybrid solution method requires a similar average computing time of 83.2 seconds.

D. Effects of the degree of dynamism

The sliding time window α controls the implementation frequency of the hybrid solution process. The smaller the interval is, the higher the degree of dynamism ($\frac{1}{\alpha}$) of the OCD service, corresponding to greater flexibility in responding to customer orders. Taking instance SZ1 as an example, we varied $\frac{1}{\alpha}$ from 12 to 2 to evaluate its effects on the OCD service performance. The obtained results are displayed in Fig. 6. These results indicate that the total travel distance is reduced as the degree of dynamism decreases because the delivery routes of the riders will be better coordinated when more orders are delivered during the same period. However, the total delivery delay to customers initially decreases but then increases as the degree of dynamism is decreased to 2. In addition, the median of computing time increases to 597.63 seconds, approximately 33% of the data collection process. This is because as more orders are collected, greater computational effort will be required to find a high-quality OCD solution.

E. Effects of crowdsourced rider recruitment

The success of crowdsourced food delivery depends on the ability to recruit crowdsourced riders. By adding and removing riders in instance SZ1, we evaluated the effects of rider recruitment. A higher order/rider ratio indicates that fewer riders have been recruited. The obtained results are presented in Fig. 7. These results indicate that when fewer riders are recruited, the quality of service of crowdsourced food delivery degrades. When the ratio increases from 5 to 25, the median of total travel distance increases from 1542.46 km to 1627.95 km, and the median of total delivery delay increases from 131.17 minutes to 834.02 minutes. Note that the median of computing time also increases from 67.9 seconds to 90.6 seconds because the competition for food delivery by any one rider increases.

F. Effects of rider predispatch

Predispatching riders between consecutive solution generation processes may improve the OCD performance. We tested the effects of rider predispatch by removing the predispatching procedure from the main hybrid solution algorithm. The resulting differences in the objective values and computing times are reported in Fig. 8. These results demonstrate that the predispatching procedure significantly improves the total travel distance and delivery delay. By predispatching idle crowdsourced riders, the median of total travel distance is decreased from 1585.1 km to 1567.94 km, and the median of total delivery delay to customers is improved from 337.47 minutes to 260.54 minutes. Moreover, with the predispatching procedure, the median computing time for the hybrid solution decreases from 93 seconds to 83.5 seconds.

G. *Crowdsourced food delivery* vs. *traditional logistics*

Another key feature of the proposed OCD method is that the crowdsourced riders can float among food providers rather than being bound to one provider. By restricting each rider to serve a single provider, we modeled traditional urban logistics for food delivery and compared the results. To ensure a fair comparison, we left the travel speed unchanged. The findings, which are displayed in Fig. 9, indicate that crowdsourced food delivery is superior to traditional logistics. Because the riders are shared among food providers, the median of total travel distance is decreased by 28.8%, from 2202.01 km to 1567.94 km, by reducing the need for return travel to specific food providers. Simultaneously, the median of total delivery delay is improved from 2088.62 minutes to 260.54 minutes. Although the median of computing effort increases from 59.36 seconds to 83.51 seconds, the computation remains sufficiently fast for real-world crowdsourced food delivery applications.

H. *Single-mode* vs. *multimode transportation*

The recruited workers may use various modes of transportation. By considering crowdsourced workers with different transportation modes, such as bicycles (10 km/h), motorbikes (15 km/h), and cars (28 km/h), we simulated the OCD problem

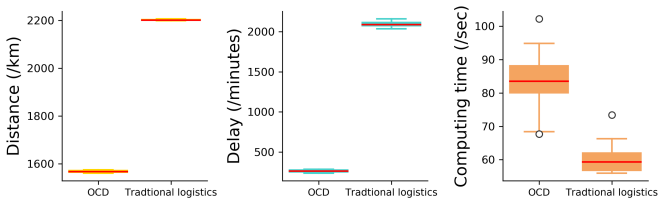

Fig. 9. The effects of crowdsourced delivery.

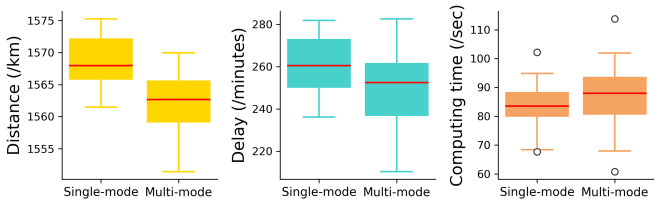

Fig. 10. The effects of multimode transportation.

with multimode transportation. The percentages of the three types of crowdsourced workers were set to 20%, 40%, and 40%, respectively. The results shown in Fig. 10 demonstrate that OCD is more effective with multimode transportation. Because traveling by car is much faster, the delivery efficiency is improved. The median of total travel distance decreases from 1567.94 km to 1562.67 km, and the total delivery delay is simultaneously improved from 260.645 minutes to 252.422 minutes, while the computing effort remains similar.

I. *Static* vs. *dynamic transportation*

The dynamics of urban transportation have a great influence on urban travel. Dynamic urban transportation can be modeled by means of a traffic simulation [43]–[45] or ATIS [46]. By varying the travel speeds in the OCD problem with multimode transportation, the effects of dynamic transportation were evaluated. Thus, we simulated dynamic transportation by extending the multimode transportation test. Time-dependent travel speeds were set in accordance with historical traffic information [46]. The results presented in Fig. 11 show the effects of dynamic traffic compared with static transportation. The allocation of food delivery tasks changes to avoid heavy delivery delays. The median of total travel distance decreases from 1562.67 km to 1586.93 km. The median of total delivery delay increases from 252.42 minutes to 365.77 minutes. The computing effort also slightly increases because of the need to compute the time-dependent travel times.

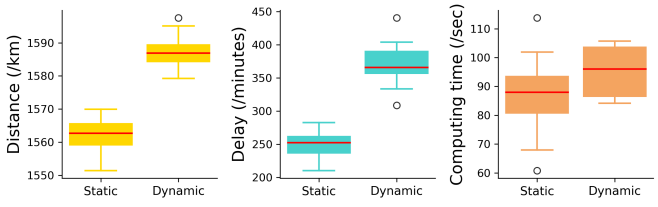

Fig. 11. The effects of dynamic transportation.

VII. Conclusion and Future Works

A. *Conclusion*

The on-demand food business highlights the need for fast and efficient food delivery services. This study presents an OCD approach for satisfying real-time food demands. Unlike in traditional urban logistics, crowdsourced riders are dynamically recruited to deliver food to customers. An online dynamic optimization framework with parallel order collection, solution generation, and sequential delivery processes is presented for near-real-time food delivery. Considering travel costs and time windows, the solution generation process uses a hybrid metaheuristic that combines the ALNS and TS approaches to allocate food delivery tasks to crowdsourced riders and schedule the riders' travel. Simulated small- and medium-scale as well as real-world large-scale on-demand food delivery instances are presented to evaluate the performance of the proposed approach. The results indicate that the presented approach outperforms traditional urban logistics. The developed hybrid optimization mechanism is able to produce high-quality routes for 1558 orders within 1818 seconds. The degree of dynamism of the OCD problem, the success of rider recruitment, and the dynamic nature of the transportation conditions all affect the total travel distance and the total delivery delay to customers.

B. *Future work*

There are still some limitations to be overcome in the future. First, recruiting crowdsourced riders is important. A worker recruitment mechanism should be developed to enhance the presented approach. The recruitment cost will be integrated into the objectives of the presented OCD model. Second, multiple types of crowdsourced tasks [34] might be simultaneously integrated into a new mobile crowdsensing platform based on motorbikes. Third, the presented OCD model can be extended to incorporate dynamic transportation. Real-time traffic information and short-time traffic prediction should also be incorporated into the model to accurately estimate travel times. In this way, the delivery delay to customers may be further reduced. Finally, the preferences of food providers can be considered and incorporated into the presented OCD model.

References

[1] M. Roh and K. Park, "Adoption of O2O food delivery services in south korea: The moderating role of moral obligation in meal preparation," *International Journal of Information Management*, 2018.

[2] J. C. Correa, W. Garzón, P. Brooker, G. Sakarkar, S. A. Carranza, L. Yunado, and A. Rincón, "Evaluation of collaborative consumption of food delivery services through web mining techniques," *Journal of Retailing and Consumer Services*, vol. 46, pp. 45 – 50, 2019.

[3] F. B. A. Abid and A. N. M. R. Karim, "Cross-platform development for an online food delivery application," in *2017 International Conference on Computing Networking and Informatics (ICCNI)*, Oct 2017, pp. 1–4.

[4] N. R. Dayama and M. Krishnamoorthy, "Facility location and routing decisions for a food delivery network," in *2016 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)*, Dec 2016, pp. 94–98.

[5] C. Selma, D. Tamzalit, N. Mebarki, O. Cardin, L. Bruggeman, and D. Thiériot, "Industry 4.0 and service companies: The case of the french postal service," *Studies in Computational Intelligence*, vol. 803, pp. 436–447, 2019.

[6] Z. Fang, S.-L. Shaw, W. Tu, Q. Li, and Y. Li, "Spatiotemporal analysis of critical transportation links based on time geographic concepts: a case study of critical bridges in wuhan, china," *Journal of Transport Geography*, vol. 23, pp. 44 – 59, 2012.

[7] EServices, "EServices Report 2018 - Online Food Delivery," https://www.statista.com/study/40457/food-delivery/, in Chinese. Accessed at Nov 12, 2018.

[8] Meituan-Dianping Research Institute, "Takeout development report 2017." https://mp.weixin.qq.com/s/eTyDmhwjEZyjs41vc9RPlw, in Chinese. Accessed at Nov 12, 2018.

[9] Y. Liu, B. Guo, C. Chen, H. Du, Z. Yu, D. Zhang, and H. Ma, "Foodnet: Toward an optimized food delivery network based on spatial crowdsourcing," *IEEE Transactions on Mobile Computing*, pp. 1–1, 2018.

[10] C. Chen, D. Zhang, X. Ma, B. Guo, L. Wang, Y. Wang, and E. Sha, "Crowddeliver: Planning city-wide package delivery paths leveraging the crowd of taxis," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 6, pp. 1478–1496, June 2017.

[11] L. Kazemi and C. Shahabi, "Geocrowd: Enabling query answering with spatial crowdsourcing," in *SIGSPATIAL '12*. New York, NY, USA: ACM, 2012, pp. 189–198.

[12] S. R. B. Gummidi, X. Xie, and T. B. Pedersen, "A survey of spatial crowdsourcing," *ACM Trans. Database Syst.*, vol. 44, no. 2, pp. 8:1–8:46, Mar. 2019.

[13] J. Wang, Y. Wang, D. Zhang, J. Goncalves, D. Ferreira, A. Visuri, and S. Ma, "Learning-assisted optimization in mobile crowd sensing: A survey," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 1, pp. 15–22, Jan 2019.

[14] W. Tu, P. Santi, T. Zhao, X. He, Q. Li, L. Dong, T. J. Wallington, and C. Ratti, "Acceptability, energy consumption, and costs of electric vehicle for ride-hailing drivers in beijing," *Applied Energy*, vol. 250, pp. 147 – 160, 2019.

[15] J. An, X. Gui, Z. Wang, J. Yang, and X. He, "A crowdsourcing assignment model based on mobile crowd sensing in the internet of things," *IEEE Internet of Things Journal*, vol. 2, no. 5, pp. 358–369, Oct 2015.

[16] L. Wang, D. Zhang, Y. Wang, C. Chen, X. Han, and A. M'hamed, "Sparse mobile crowdsensing: challenges and opportunities," *IEEE Communications Magazine*, vol. 54, no. 7, pp. 161–167, July 2016.

[17] J. Wang, Y. Wang, D. Zhang, F. Wang, H. Xiong, C. Chen, Q. Lv, and Z. Qiu, "Multi-task allocation in mobile crowd sensing with individual task quality assurance," *IEEE Transactions on Mobile Computing*, vol. 17, no. 9, pp. 2101–2113, Sep. 2018.

[18] B. Zhou, Q. Li, Q. Mao, W. Tu, X. Zhang, and L. Chen, "ALIMC: Activity landmark-based indoor mapping via crowdsourcing," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 5, pp. 2774–2785, Oct 2015.

[19] B. Zhou, Q. Li, Q. Mao, and W. Tu, "A robust crowdsourcing-based indoor localization system," *Sensors*, vol. 17, no. 4, 2017.

[20] Y. Li, Z. He, Z. Gao, Y. Zhuang, C. Shi, and N. El-Sheimy, "Towards robust crowdsourcing-based localization: A fingerprinting accuracy indicator enhanced wireless/magnetic/inertial integration approach," *IEEE Internet of Things Journal*, pp. 1–1, 2018.

[21] A. S. El-Wakeel, J. Li, A. Noureldin, H. S. Hassanein, and N. Zorba, "Towards a practical crowdsensing system for road surface conditions monitoring," *IEEE Internet of Things Journal*, vol. 5, no. 6, pp. 4672–4685, Dec 2018.

[22] J. Huang, N. Duan, P. Ji, C. Ma, F. Hu, Y. Ding, Y. Yu, Q. Zhou, and W. Sun, "A crowdsource-based sensing system for monitoring fine-grained air quality in urban environments," *IEEE Internet of Things Journal*, 2018.

[23] H. Akeb, B. Moncef, and B. Durand, "Building a collaborative solution in dense urban city settings to enhance parcel delivery: An effective crowd model in paris," *Transportation Research Part E: Logistics and Transportation Review*, vol. 119, pp. 223 – 233, 2018.

[24] M. Schreieck, C. Pflügler, C. Dehner, S. Vaidya, S. Bönisch, M. Wiesche, and H. Krcmar, "A concept of crowdsourced delivery for small local shops," *Informatik 2016*, 2016.

[25] E. Mohamed and M. Ndiaye, "Optimal routing and scheduling in e-commerce logistics using crowdsourcing strategies," in *2018 7th International Conference on Industrial Technology and Management (ICITM)*, March 2018, pp. 248–253.

[26] H. B. Rai, S. Verlinde, J. Merckx, and C. Macharis, "Crowd logistics: an opportunity for more sustainable urban freight transport?" *European Transport Research Review*, vol. 9, no. 3, p. 39, 2017.

[27] A. Mladenow, C. Bauer, and C. Strauss, "Crowdsourcing in logistics: concepts and applications using the social crowd," in *Proceedings of the 17th International Conference on Information Integration and Web-based Applications & Services, iiWAS 2015, Brussels, Belgium, December 11-13, 2015*, 2015, pp. 30:1–30:8.

[28] J. Xia, K. M. Curtin, J. Huang, D. Wu, W. Xiu, and Z. Huang, "A carpool matching model with both social and route networks," *Computers, Environment and Urban Systems*, vol. 75, pp. 90 – 102, 2019.

[29] Y.Chen, D. Guo, M. Xu, G. Tang, T.Zhou, and B. Ren, "PPtaxi: Non-stop package delivery via multi-hop ridesharing," *Arxiv*, 2018. [Online]. Available: https://arxiv.org/abs/1809.04733

[30] A. Devari, A. G. Nikolaev, and Q. He, "Crowdsourcing the last mile delivery of online orders by exploiting the social networks of retail store customers," *Transportation Research Part E: Logistics and Transportation Review*, vol. 105, pp. 105 – 122, 2017.

[31] C. Yi, Y. Chuang, and C. Nian, "Toward crowdsourcing-based road pavement monitoring by mobile sensing technologies," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 4, pp. 1905–1917, Aug 2015.

[32] J. Xia, C. Yang, and Q. Li, "Using spatiotemporal patterns to optimize earth observation big data access: Novel approaches of indexing, service modeling and cloud computing," *Computers, Environment and Urban Systems*, vol. 72, pp. 191 – 203, 2018.

[33] W. Tu, Q. Li, Q. Li, J. Zhu, B. Zhou, and B. Chen, "A spatial parallel heuristic approach for solving very large-scale vehicle routing problems," *Transactions in GIS*, vol. 21, no. 6, pp. 1130–1147, 2017.

[34] B. Guo, Y. Liu, L. Wang, V. O. K. Li, J. C. K. Lam, and Z. Yu, "Task allocation in spatial crowdsourcing: Current state and future directions," *IEEE Internet of Things Journal*, vol. 5, no. 3, pp. 1749–1764, June 2018.

[35] M.-R. Ra, B. Liu, T. F. La Porta, and R. Govindan, "Medusa: A programming framework for crowd-sensing applications," in *MobiSys '12*. New York, NY, USA: ACM, 2012, pp. 337–350.

[36] D. Deng, C. Shahabi, and U. Demiryurek, "Maximizing the number of worker's self-selected tasks in spatial crowdsourcing," in *SIGSPATIAL'13*. New York, NY, USA: ACM, 2013, pp. 324–333.

[37] H. To, C. Shahabi, and L. Kazemi, "A server-assigned spatial crowdsourcing framework," *ACM Trans. Spatial Algorithms Syst.*, vol. 1, no. 1, pp. 2:1–2:28, Jul. 2015.

[38] H. W. Kuhn, "The hungarian method for the assignment problem," *Naval Research Logistics Quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955.

[39] W. Tu, Q. Li, Z. Fang, S. Shaw, B. Zhou, and X. Chang, "Optimizing the locations of electric taxi charging stations: A spatial–temporal demand coverage approach," *Transportation Research Part C: Emerging Technologies*, vol. 65, pp. 172 – 189, 2016.

[40] B. Guo, Y. Liu, W. Wu, Z. Yu, and Q. Han, "Activecrowd: A framework for optimized multitask allocation in mobile crowdsensing systems," *IEEE Transactions on Human-Machine Systems*, vol. 47, no. 3, pp. 392–403, June 2017.

[41] Z. Yu, H. Xu, Z. Yang, and B. Guo, "Personalized travel package with multi-point-of-interest recommendation based on crowdsourced user footprints," *IEEE Transactions on Human-Machine Systems*, vol. 46, no. 1, pp. 151–158, Feb 2016.

[42] S. C. Ho, W. Szeto, Y.-H. Kuo, J. M. Leung, M. Petering, and T. W. Tou, "A survey of dial-a-ride problems: Literature review and recent developments," *Transportation Research Part B: Methodological*, vol. 111, pp. 395 – 421, 2018.

[43] C. Fiandrino, A. Capponi, G. Cacciatore, D. Kliazovich, U. K. Sorger, P. Bouvry, B. Kantarci, F. Granelli, and S. Giordano, "Crowdsensim: a simulation platform for mobile crowdsensing in realistic urban environments," *IEEE Access*, vol. 5, pp. 3490–3503, 2017.

[44] D. Krajzewicz, J. Erdmann, M. Behrisch, and L. Bieker, "Recent development and applications of sumo – simulation of urban mobility," *International Journal on Advances in Systems and Measurements*, vol. 5, no. 3, pp. 128–138, 2012.

[45] S. Uppoor, O. Trullols-Cruces, M. Fiore, and J. M. Barcelo-Ordinas, "Generation and analysis of a large-scale urban vehicular mobility dataset," *IEEE Transactions on Mobile Computing*, vol. 13, no. 5, pp. 1061–1075, 2014.

[46] W. Tu, Z. Fang, and Q. Li, "Exploring time varying shortest path of urban od pairs based on floating car data," in *2010 18th International Conference on Geoinformatics*, June 2010, pp. 1–6.

[47] W. Tu, Q. Li, Q. Li, J. Zhu, B. Zhou, and B. Chen, "A spatial parallel heuristic approach for solving very large-scale vehicle routing problems," *Transactions in GIS*, vol. 21, no. 6, pp. 1130–1147, 2017.

[48] C. Lin, "A vehicle routing problem with pickup and delivery time windows, and coordination of transportable resources," *Computers and Operations Research*, vol. 38, no. 11, pp. 1596 – 1609, 2011.

[49] B. L. Golden, S. Raghavan, and E. A. Wasil, *The vehicle routing problem: latest advances and new challenges*. Springer Science & Business Media, 2008, vol. 43.
[50] A. K. Jain, "Data clustering: 50 years beyond k-means," *Pattern Recognition Letters*, vol. 31, no. 8, pp. 651 – 666, 2010, award winning papers from the 19th International Conference on Pattern Recognition (ICPR).
[51] R. A. Santana, M. R. Pontes, and C. J. A. Bastos-Filho, "A multiple objective particle swarm optimization approach using crowding distance and roulette wheel," in *2009 Ninth International Conference on Intelligent Systems Design and Applications*, Nov 2009, pp. 237–242.
[52] D. Pisinger and S. Ropke, "A general heuristic for vehicle routing problems," *Computers and Operations Research*, vol. 34, no. 8, pp. 2403 – 2435, 2007.
[53] IBM, "IBM ILOG CPLEX," http://www.ibm.com/analytics/cplex-optimizer, accessed at Jan 6, 2019.

Wei Tu received the Ph.D. degree in photogrammetry and remote sensing from Wuhan University, Wuhan, China, in 2013. He is currently a senior associate research fellow at Shenzhen Key Laboratory of Spatial Smart Sensing and Service and Research Institute of Smart Cities, Department of Urban Informatics, Shenzhen University. He is also a visiting scholar at Seneseable City Laboratory, Massachusetts Institute of Technology. His research interests include urban informatics, big data driven human activity and mobility, and trajectory analytic.

Tianhong Zhao is currently pursuing the Ph.D. degree with the Department of Urban Informatics, School of Architecture and Urban Planning, Shenzhen University, Shenzhen, China. His research interests focus on spatio-temporal optimization algorithm, spatio-temporal big data analysis.

Baoding Zhou received the Ph.D. degree in photogrammetry and remote sensing from Wuhan University, Wuhan, China, in 2015. He is currently an Assistant Professor with the College of Civil and Transportation Engineering, Shenzhen University, Shenzhen, China. His research interests include indoor localization and mapping, mobile computing, and intelligent transportation.

Jincheng Jiang received recieved his Ph.D degree in geographic information science from Beijing Normal University in 2016. He is currently an associate research professor of Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen, Guangdong, China. His research interests include urban computing, spatial-temporal data analysis, high-performance computing, human mobility, and emergency response.

Jizhe Xia received his Ph.D. degree in Earth Systems and Geoinformation Sciences from George Mason University in 2015. From 2015 to 2017, he was a research assistant professor in the National Science Foundation (NSF) spatio-temporal invitation center. In 2017, he joined the department of urban informatics, Shenzhen University, China as an assistant professor. His research interests include spatial data indexing, spatiotemporal optimization and geographic information applications.

Qingquan Li received the M.S. degree in survery engineering and the Ph.D degree in photogrammetry and remote sensing from Wuhan Technical University of Surveying and Mapping, Wuhan, China, in 1988 and 1998, respectively. He is the President of Shenzhen University, Shenzhen, China, and the Director of the Guangdong Key Laboratory of Urban Informatics, Shenzhen University. His research interests include photogrammetry, remote sensing, mobile mapping, and intelligent transportation systems.