European Journal of Operational Research 323 (2025) 139–152

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/eor

Production, Manufacturing, Transportation and Logistics

Stochastic scheduling and routing decisions in online meal delivery platforms with mixed force

Yanlu Zhao [a], Laurent Alfandari [b], Claudia Archetti [c,*]

[a] Durham University Business School, Durham, UK
[b] Department of Information Systems, Decision Sciences and Statistics, ESSEC Business School, Cergy-Pontoise, France
[c] Department of Economics and Management, University of Brescia, Brescia, Italy

ARTICLE INFO

Keywords:
Routing
Online meal delivery
Scheduling and routing decisions
Mixed delivery force
Markov decision process

ABSTRACT

This paper investigates stochastic scheduling and routing problems in the online meal delivery (OMD) service. The huge increase in meal delivery demand requires the service providers to construct a highly efficient logistics network to deal with a large-volume of time-sensitive and fluctuating fulfillment, often using inhouse and crowdsourced drivers to secure the ambitious service quality. We aim to address the problem of developing an effective scheduling and routing policy that can handle real-life situations. To this end, we first model the dynamic problem as a Markov Decision Process (MDP) and analyze the structural properties of the optimal policy. Then we propose four integrated approaches to solve the operational level scheduling and routing problem. In addition, we provide a continuous approximation formula to estimate the bounds of required fleet size for the inhouse drivers. Numerical experiments based on a real dataset show the effectiveness of the proposed solution approaches. We also obtain several managerial insights that can help decision makers in solving similar resource allocation problems in real-time.

1. Introduction

The business of delivering restaurant meals to individual homes is undergoing rapid growth as the emerging online platforms (e.g., *UberEats*, *Deliveroo*, and *Meituan*) race to capture worldwide markets in the past few years. The COVID-19 pandemic strengthened this trend as we saw a significant demand increase for this service in 2022. For example, the business surge renders the London-based online meal delivery (OMD) unicorn Deliveroo launching its debut in March 2021, then raising additional funds to continue the rival with competitors across Atlantic. The underlying reason is that the operational cost in the delivery force is expensively high (Korosec & Wilhelm, 2020), thus the OMD practitioners are making every endeavor to reduce costs through effective strategies. Nonetheless, the practical features of OMD services, such as fluctuating seasonal demand and stringent delivery requirements, renders this business rather complicated to attain operational excellence (Hirschberg et al., 2016).

Seasonal demand. Our study is motivated by analyzing the OMD operations of the Chinese platform Meituan. The regional data analysis (shown in Fig. 1, left panel) reveals a significant variation in platform demand throughout the day. That is, approximately 40% of the orders occur during lunchtime (10:00–14:00), while 30% happen during dinner time (18:00–20:00). We refer to this heterogeneous demand arrival pattern in OMD operations as *seasonal demand* due to its periodic nature with peaks and valleys occurring in each day. To further characterize this pattern, we identify the peak periods as *rush hours* and the low-demand periods as *no-rush hours*. The heterogeneous nature of demand creates fluctuating needs for delivery personnel, making OMD operations more complex. As depicted in the right panel of Fig. 1, there is also a *seasonal* distribution of drivers availability throughout the day, with 30 drivers available during no-rush periods and 130 drivers during rush periods on average. Managing such variability in both demand and driver availability poses significant challenges that must be tackled to ensure a high service level.

The seasonal demand is not specific to OMD platforms. With the wide usage of mobile devices, more and more consumers find it convenient to access online platforms and make immediate requests for products and services, triggering the demand surge in a specific timing or region. For example, Afèche et al. (2023) studied the ride-hailing platform operation and intended to reduce the spatial supply and demand mismatch during rush hours through demand-side admission control and supply-side capacity repositioning policies: the former allows the platform to accept or reject rider requests while the latter allows the platform to direct drivers to the highest demand places.

* Corresponding author.
E-mail addresses: yanlu.zhao@durham.ac.uk (Y. Zhao), alfandari@essec.edu (L. Alfandari), claudia.archetti@unibs.it (C. Archetti).

https://doi.org/10.1016/j.ejor.2024.11.028
Received 16 December 2022; Accepted 18 November 2024
Available online 4 December 2024

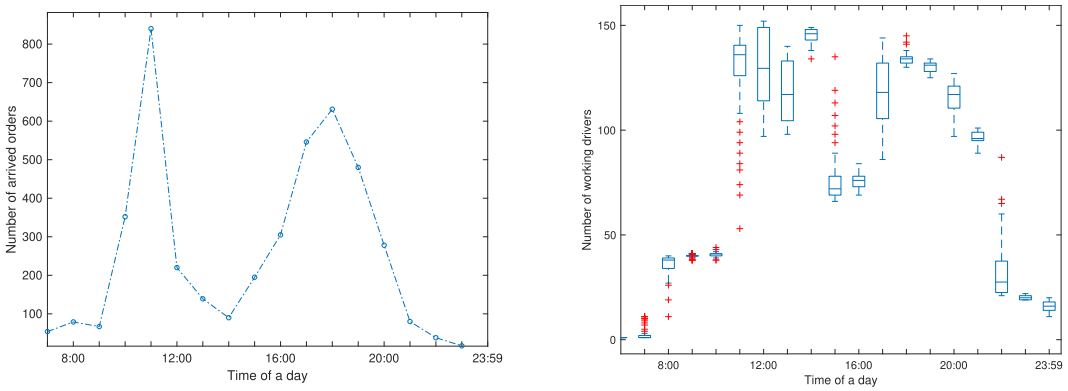

Fig. 1. The average arrival pattern of OMD platform agents (orders and drivers) per day.

However, these control policies seem not suitable for the OMD operations. What is specific to the OMD is the high variance in the temporal distribution of demand, real-time response to the customer requests and high penalties for rejected and delayed service. As a result, the admission control through requests rejection would cause significant loss of goodwill and the repositioning control is deemed as less effective to alleviate the temporal capacity shortage. Instead, the OMD platform is recommended to construct a highly *flexible* logistics network to deal with the large volume of time-sensitive and fluctuating demand.

Practice dilemma. Fundamentally, the platform needs to strategically determine an appropriate capacity of required drivers to satisfy the seasonal demand per day. To ensure a high quality of service, the platform typically establishes its own dedicated delivery team, which is a tactical decision that remains fixed for a specific period, such as six months (Dai & Liu, 2020; Tao et al., 2023; Zhao et al., 2024). The most straightforward way is to employ as many drivers as needed to satisfy the demand in rush hours. Accordingly, customers' service quality would be highly rated as they can always receive the ordered meals timely. However, this strategy is clearly not cost-effective because it incurs an expensive fixed cost (e.g., training fees, insurances, and paid holidays) to maintain a large fleet size which is redundant for the no-rush hours. By contrast, if the platform seeks to reduce the operational cost, it might recruit a number of drivers sufficient to cover the demand during no-rush period. However, this might incur service delays within rush hours due to the inadequate supply of delivery force. As a result, the OMD practitioners are *de facto* facing a dilemma. By retrospecting Fig. 1, we also observe that the number of busy drivers during rush hours in Meituan are approximately 4 times higher than that of no-rush hours, i.e., 150 vs. 40, and the *busy rate* (number of loading orders per driver) also increases by up to 8 times in rush hours. Therefore, it is rather challenging for managers to tackle this flexible capacity design issue while balancing the service quality and cost-effectiveness targets.

Inhouse vs. crowdsourced drivers. In recent years, the self-scheduling delivery service provided by crowdsourced drivers (e.g., *Amazon Flex, UberEats*) has been gradually introduced into the online marketplace (Archetti & Bertazzi, 2021; Taylor, 2018). The self-scheduling flexibility allows the drivers to decide when and how often to offer their delivery service with certain degree of freedom (Hall & Krueger, 2018), and the OMD platform can take such an advantage to satisfy the variable demands rather than maintain a large fleet all the time. Thus, the platform either disposes of its own delivery force or crowdsources individual drivers. Hereafter, the self-own delivery force is denoted as *inhouse drivers* and work as full-time employees. On the contrary, the self-scheduling individual drivers (*crowdsourced drivers*) work part-time and are allowed to join or leave with personal freedom, depending on whether deliveries are beneficial to them. Crowdsourced drivers add high flexibility on capacity management, on one side. However, on the other side, they insert a new degree of uncertainty related to their availability to provide service. Additionally, the inhouse drivers incur higher fixed costs while crowdsourced drivers incur higher variable costs (see Table 1 for detailed characteristics comparison of Meituan drivers).

Table 1
Characteristics of inhouse and crowdsourced drivers in Meituan.

	Inhouse driver	Crowdsourced driver
Employment	Platform	Self-scheduling
Contract	Contract-based, full-time	No contract, part-time
Fixed wage	3000 yuan per month	No fixed wage
Compensation	6 yuan per order	6~9 yuan per order

Key challenge. Consequently, determining a fleet of delivery force, effective scheduling and dispatching of drivers from both delivery modes (inhouse and crowdsourced) is a critical activity to secure the operational cost performance (Yildiz & Savelsbergh, 2019a, 2019b). This is particularly difficult due to the uncertain arrival of incoming orders and the unpredictable availability of crowdsourced drivers. The platform must, therefore, devise effective dispatching routes on-the-fly and allocate new orders to the drivers based on their availability and geographical locations. The dynamic nature of dispatching routes, which may need to be altered as new orders arrive even while a driver is already en route of other delivery tasks, adds a significant level of complexity to the problem. In this regard, existing studies on routing and scheduling optimization problems for meal delivery are still developing. Reyes et al. (2018) designed a rolling-horizon repeated-matching algorithm to solve the dynamic meal delivery problem in nearly real-time, and Ulmer et al. (2021) studied the postponement strategies to schedule drivers in the dynamic pickup and delivery meal problem. Nonetheless, in both studies, the size of the delivery fleet is exogenously known, and the effects of uncertain driver availability on dynamic dispatching decisions have not been thoroughly explored. The complex interplay between fleet management and the optimization of dynamic dispatching necessitates further detailed study, which is exactly our focus here.

Our contributions. In this article, we take the perspective of the OMD service provider which connects meal providers (e.g., restaurants) and customers through a team of mixed delivery force. We analyze the one-period (i.e., a day) scheduling and dispatching problem in which the providers sell products (meals) to customers from different locations (restaurants). The objective of the service provider is to minimize the total delivery cost with a timely service level target. We are seeking to answer the following research questions: (i) Given an inhouse driver fleet of size, how to develop an effective yet computationally feasible scheduling and routing policy that can handle real-life situations? (ii) What is the impact of the inhouse driver fleet on the cost of the operational plan? The first question is at an operational level for a given number of inhouse drivers, while the second is about determining the

best fleet size and, thus, is at the tactical level, but requires first to be able to compute the best policy for any given size. The contributions of this paper are as follows:

- *Dynamic models and optimal policy analysis for a given fleet size:* We formulate the operational level dynamic scheduling and dispatching problem as a Markov Decision Process (MDP) and derive structural properties of the optimal policy where possible, i.e., we analytically provide the recursive formulation of the optimal policy.
- *Solution approaches leveraging future information and fast-effective routes generation:* Given the huge size of state space that renders the problem too complex to be solved optimally, we propose two customized algorithms to take future information into account (see Soeffker et al., 2021; Thomas & White, 2004, for the importance of exploiting information when solving dynamic vehicle routing problems), and exploit two fast-effective strategies to adapt route construction. Accordingly, this leads to four solution approaches. We first compare the performance of the designed algorithms versus optimal policies through a set of simple instances to highlight their advantages. Then we conduct extensive numerical experiments with the dataset from Meituan to investigate the performance of the developed approaches in real-life business environments.
- *Managerial insights for real-world OMD applications:* We present an adaptation of the approximation formula by Daganzo (1987) to determine the range of inhouse driver fleet size, and perform a binary search to determine the capacity that minimizes total cost. Then, we show the importance of incorporating future information and refining routes generation to improve the operational performances, by reducing the total cost up to 6% compared to the policy without these features. We also present the impact of key factors, such as the fixed wage and variable compensation cost of different drivers, on the optimal inhouse drivers fleet size. Note that our proposed solution approach can be easily implemented in the OMD and other similar platforms to address real-time resource allocation problems with mixed delivery force.

2. Literature review

Our research lies at the intersection of several related streams of literature: online meal delivery, real-time vehicle routing and scheduling and crowdshipping. In this section, we review the key contributions of each stream of literature and discuss how we extend them.

Research on online meal delivery. Our work is related to the literature about online meal delivery, which has received ample attention by researchers in the last few years (Liu et al., 2021; Reyes et al., 2018; Yildiz & Savelsbergh, 2019a; Zhao et al., 2024). The online meal delivery problem is one of the most challenging in last mile delivery, due to the strict service level requested by the customers (Yildiz & Savelsbergh, 2019a). It is also similar to the same day delivery problem described in Voccia et al. (2019), but requires quick-response decisions to respect the short delivery time, rendering the required solution approach fundamentally different from the solution approaches proposed in this research stream.

We only review previous studies investigating the stochastic online meal delivery problem in this article. Liu et al. (2021) propose a data-driven framework to model the online meal delivery problem and optimize the assignment decisions based on orders delivery prediction with uncertain service time, while they regard the routing decision as a black-box and do not provide an explicit route for each delivery person. Ulmer et al. (2021) consider a stochastic dynamic pickup and delivery problem in which a fleet of drivers delivers meals from a set of restaurants to ordering customers. The authors present an anticipatory customer assignment (ACA) policy to address the stochasticity, postpone the assignment decisions for selected customers, and introduce time buffers to account for the uncertainty in the meal ready times. Tao et al. (2023) developed two machine learning models to design personalized dispatching schemes for drivers, integrating information such as the order and driver's characteristics in the order assignment and routing decisions. In our paper, to tackle the dynamic scheduling and routing problem in online meal delivery service, we develop an anticipatory solution approach that incorporates information about future requests into routing decisions.

Research on real-time vehicle routing and scheduling. Our problem is also related to the dynamic vehicle routing problem (DVRP) (Powell et al., 1995). A vehicle routing problem is dynamic (i.e., real-time or online) if information about the problem is unknown in advance to the decision maker and arrives in real time during the routing horizon. Dynamic information may include customer demand, travel times, service time or customer requests (Ehmke et al., 2015).

In the literature, two major classes of solution approaches have been reported to solve such kind of problems, which differ mainly in the way they process the dynamic information. The first class of methods is called the myopic approach (Powell et al., 2000) or the local approach (Chen & Xu, 2006), which optimizes and reoptimizes the dispatching routes solely based on known information without considering future information. At each period, a static model consisting of known orders up to the current time point is solved. These methods do not need any advance information about future events and can be used for situations where future orders are difficult to predict. Recent studies on myopic approaches for DVRP include insertion algorithms (Ichoua et al., 2000), nearest neighbor (Naccache et al., 2018), recursive approaches (Arslan et al., 2019), column generation (Chen & Xu, 2006), and others. The second class of methods is called the anticipatory approach (Berbeglia et al., 2010) or the look-ahead approach (Chen & Xu, 2006). This approach tries to incorporate the probabilistic or forecasted information about the future into the static problem. These methods require anticipated information about future events and can be used for situations where at least some probabilistic information about future events is known in advance. With the recent developments in information and communication technologies, it becomes easier and more affordable to explore historical data and to extend anticipatory approaches. We refer the reader to the extensive review on the stochastic DVRP by Soeffker et al. (2021).

All of the above research, however, concern the dynamic scheduling and routing problem with only a professional delivery fleet, i.e., inhouse drivers. With the adoption of on-demand crowdsourced drivers, the DVRP needs to be able to integrate more stochastic factors, under which the choice of drivers is a crucial element to determine the ultimate operational costs.

Research on crowdshipping. The significant increase in the online meal delivery service requires the platform to construct a highly efficient logistics network tackling a large-volume of time-sensitive and fluctuating requests. Therefore, relying only on inhouse drivers is no longer a viable strategy. In recent years, the adoption of crowdsourced drivers gives the service providers more flexibility in facing the demand fluctuations (Cachon et al., 2017). As a result, the online platforms are able to reduce costs and fixed assets through crowdshipping.

We can notice a growing interest in routing with crowdshipping in the literature, and we refer to Archetti and Bertazzi (2021) for a detailed review. Archetti et al. (2016) first introduced the crowdshipping in routing problems under the name vehicle routing problem with occasional drivers (VRPOD). The authors studied the basic setting in which each occasional driver (OD) could serve one customer request at most and they provided insights on the impact of different compensation schemes on the solution quality. Later, Arslan et al. (2019) studied a problem in which dedicated vehicles are considered as an option for the crowdsourced drivers and studied the case in which both the customers' orders and the ODs' availability are uncertain and dynamic. Dayarian and Savelsbergh (2020) designed two rolling horizon dispatching approaches to address a stochastic same-day delivery

problem and quantified the potential benefits of crowdshipping for same-day delivery. Fatehi and Wagner (2021) investigated the labor planning and pricing for crowdsourced last-mile delivery systems to satisfy on-demand orders. They developed a model to combine crowdsourcing, robust queueing, and robust routing theories to derive the optimal delivery assignments to available independent crowdsourced drivers given their optimal hourly wage. Zehtabian et al. (2022) provided a reliable estimation of arrival times in a crowd-shipping context with both uncertain requests and (future) occasional drivers participation, among which, they modeled the dynamic pickup and delivery problem as an MDP and developed two look-ahead heuristic methods to address it.

Most of the above studies assume that the fleet sizes are exogenously given or only focus on the crowdshipping force. Therefore, we conclude that the crowdshipping delivery mode still needs further exploration, especially in the routing optimization if the delivery force is mixed and heterogeneous.

Detailed comparison with close literature. Finally, we would like to compare our work with studies closely related to ours to better position our contribution in the literature. On the one hand, this article shares a similar assumption about the arrival of crowdsourced drivers in Ulmer and Savelsbergh (2020). Starting from this, we address a different real-time scheduling and routing problem in the OMD service with a cost-minimal objective and develop online solution approaches to effectively solve this challenging problem. On the other hand, our findings also correspond to an analysis of the effect of crowdsourcing versus hiring in-house on the total cost, thereby determining an optimal fleet size. To achieve this, we adopt continuous approximation to estimate the required size. In this regard, Yildiz and Savelsbergh (2019b) estimate the service region (radius of a circle) and capacity level for crowdsourced delivery from the viewpoint of a single restaurant, which diverges significantly from our purpose. Our main emphasis lies in adopting an overall perspective that not only investigates fleet size decisions at the tactical level but also prioritizes operational aspects such as uncertain driver supply, order assignment, and route optimization (with pickup and delivery constraints). Consequently, our study contributes extensively to the OMD field by offering a systematic plan to address these crucial business problems, including planning, scheduling, and routing decisions.

3. Scheduling and dispatching operational problem in OMD

In this section and the next section, we focus on the operational problem of finding the best request assignment and routing policy for a given number of M inhouse drivers and uncertain occurrence of crowdsourced drivers.

3.1. OMD operational process

The OMD operations of the company that motivated our research consist of *order scheduling* and *meal dispatching* activities. The whole OMD process depicted in Fig. 2 involves customers, drivers, meal providers, and the company managing the platform (i.e., the decision maker). As a *customer* visits an online meal *platform* webpage to order a meal, the webpage first displays a few shops (or restaurants) close to her location. Given the displayed information, the customer then chooses her favorite dish from a particular shop and places an order for it. After confirmation with the customer, the platform notifies the *provider* (e.g., restaurant, shop, etc.) to process the required meal accordingly. Meanwhile, the platform assigns a *driver* to collect the meal at the shop and deliver it to the customer location. The meal will be picked up when it is ready at the restaurant, and dispatched to the customers. Once a driver finishes a delivery task, she can get the corresponding reward for that order. The *service time* of a platform is defined as the total duration from an order arrival until the customer receives her meal, which is subject to a service time target defined by

Fig. 2. Order scheduling and meal dispatching process in an OMD platform (Vasi, 2019).

the platform. Usually, the target is limited by a range of 30~60 minutes. The platform assigns orders in real-time to the drivers and proposes a dispatching route for each driver given her holding tasks.

In the perspective of the decision maker, she seeks to fulfill all the delivery tasks at a minimum total cost, i.e., the *fixed cost* associated with inhouse drivers and the *variable cost* (compensation) paid to both inhouse and crowdsourced drivers, which is related to the number of served orders considering travel time and drivers types, plus the *penalty cost* due to the lost-sales of unserved orders.

In the following section we formally define the problem as a Markov Decision Process (MDP). For ease of exposition, all notations are summarized in Table 2.

3.2. MDP assumptions and notations

Let $\mathcal{L} = \{0, 1, \dots, L\}$ be the set of all potential locations in the problem, where the depot for inhouse drivers is 0 and the locations of customers, en route drivers and restaurants are denoted by $\{1, \dots, L\}$. We define, within set $\mathcal{L}$, the restaurants set $\mathcal{P}$ and the customers set $\mathcal{C}$, such that $\mathcal{P} \cap \mathcal{C} = \emptyset, \mathcal{P} \cup \mathcal{C} \cup \{0\} \subseteq \mathcal{L}$. The drivers could be located at any place in $\mathcal{L}$. Let d_{ij} denote the deterministic travel time between any two locations $i, j \in \mathcal{L}$. The estimation of these parameters are further discussed in Section 5.1.

The exogenous stimulus of the platform is fed by a sequence of order requests. Formally, it is represented by a sequence of increasing real numbers t, which denotes the arrival time of a request. Upon the request at time t, its characteristics are revealed as a tuple $I(t) \equiv (t, i, j)$, where $i \in \mathcal{C}$ and $j \in \mathcal{P}$ are the customer and restaurant locations, respectively. We call (i, j) an O-D pair hereafter. For ease of modeling, we assume the meal's processing time for each request is equal to the same value $\overline{v}$, and we relax this hypothesis in the numerical experiments. The production capacity is assumed to be infinite in all restaurants. Therefore, the earliest time for a driver to pick up a meal is $\overline{v}$ minutes after the request arrival. As a consequence, (t, i, j) completely characterizes a realized order information.

Now, we further clarify the definition of arrival time t for a request. We assume that requests are placed by the customers during the time horizon $[0, T]$. A random variable T_{ij}, representing the time at which the order request is placed, is associated with every customer $i \in \mathcal{C}$ and restaurant $j \in \mathcal{P}$. For all O-D pairs (i, j), we have

$$T_{ij} = \begin{cases} \overline{T}_{ij} & \text{with probability } \theta_{ij}, \\ T + 1 & \text{with probability } 1 - \theta_{ij}. \end{cases} \quad (1)$$

Table 2
Notation in the MDP modeling.

Notation	Meaning
Parameters:	
T	The planning horizon
$\mathcal{L}$	All potential locations in the service area
$\mathcal{P}$	The restaurant locations
$\mathcal{C}$	The customer locations
d_{ij}	Traveling time between locations i and j
θ_{ij}	The probability that a customer i demands food from a restaurant j
Ω	All potential realizations of crowdsourced drivers arrival events
ω	A possible realization of the crowdsourced driver arrival events
r_0	The fixed amount of wage paid to an inhouse driver
r_1, r_2	The unit cost to accept and serve an order by an inhouse or crowdsourced driver
r_3	The unit cost to reject an order (resembling to assign it to a specialized costly driver)
n	Index of an interim stage, $n \in \{1, 2, \dots, \}$
$(T, \emptyset, \emptyset)$	The final stage
$\mathcal{M}$	The set of inhouse drivers
$\mathcal{J}_n$	The set of active crowdsourced drivers at stage n
Q	The set of potential O-D pairs (i.e., customer-restaurant paired locations)
I_n	A tuple of an order arrival indicating the arrival time, customer and restaurant locations
Functions:	
$f_{ij}, \overline{F}_{ij}$	The PDF and CDF of an order's arrival time for customer i demanding food from restaurant j within T
F_{ij}	The CDF of an order's arrival time for customer i demanding food from restaurant j
V_n^{π}	The cost-to-go function from stage n for a policy π given a crowdsourced drivers realization ω
$\mathcal{V}$	The expected total cost given all possible crowdsourced driver realizations
Random variables:	
T_{ij}	A random variable indicating an order's arrival time for customer i demanding food from restaurant
δ_j	The service duration of a crowdsourced driver
$\mathbf{S}_n$	The state vector at stage n
$\mathbf{U}_n$	The state vector of active drivers at stage n
$\mathbf{R}_n$	The set of rejected orders until stage n
$\mathbf{Y}_n^u$	The open (unfinished) orders of a driver u at stage n
$\mathbf{p}_n^u$	The scheduled path of a driver u to deliver the open orders
l_n^u	The real time locations of driver along p_n^u at stage n
Decision variables:	
$\mathbf{a}_n$	The action taken at stage n
z_n	The accept or reject action indicator
k_n	The driver index assigned to the order

where $\overline{T}_{ij}$ is a continuous random variable with known Cumulative Distribution Function (CDF) $\overline{F}_{ij}$ and Probability Density Function (PDF) $\overline{f}_{ij}$. Thus, we have

$$\overline{F}_{ij}(t) = \begin{cases} 0 & \text{if } t \le 0, \\ \int_0^t \overline{f}_{ij}(x)dx & \text{if } 0 < t \le T, \\ 1 & \text{if } t > T. \end{cases} \tag{2}$$

For a potential request of O-D pair (i, j), θ_{ij} is the probability that a request arises from a customer i demanding meal provided by a restaurant j within T; we formally define $T + 1$ as a conventional time point when "no-show" requests are placed. The CDF of T_{ij} can thus be written as

$$F_{ij}(t) = \theta_{ij}\overline{F}_{ij} + (1 - \theta_{ij})\mathbf{1}(t \ge T + 1) \tag{3}$$

where $\mathbf{1}(t)$ is an indicator function. We assume that requests arise independently from different paired locations.

The delivery force is composed of a set $\mathcal{M} = \{1, 2, \dots, M\}$ of inhouse drivers, and unaffiliated crowdsourced drivers. Suppose active busy drivers move as the designed routes such that it is possible to keep track of their locations at anytime. We also assume each idle driver acts as a random walker in the region when she does not provide service, and the probability of moving towards four directions is evenly distributed. One may argue that the idle driver keep staying in a restaurant location, which is just a special case of the current assumption. Inhouse drivers traverse routes starting and ending at the depot, and keep active within the planning horizon.

As a supplement to the inhouse drivers, the platform also hires a certain amount of crowdsourced drivers, whose availability is revealed as an exogenous stimulus. The set of all possible crowdsourced driver arrivals is denoted by Ω, i.e., each element $\omega \in \Omega$ is a realization of crowdsourced driver arrivals and each realization ω is assumed to be equally likely to occur. The realization ω consists of a sequence of crowdsourced driver arrivals, $\omega = \{(u_j^{\omega}, t_j^{\omega}, l_j^{\omega}, \delta_j^{\omega})\}, j = 1, \dots, |\omega|$, where the number of drivers $|\omega|$ varies for different realizations ($|\cdot|$ denotes the cardinality of a set). More precisely, each crowdsourced driver u_j^{ω} is characterized by a start-time $t_j^{\omega} \in [0, T]$, a start-to-work location $l_j^{\omega} \in \mathcal{L}$ and a working duration $\delta_j^{\omega} \in [0, T - t_j^{\omega}]$. We only become aware of a crowdsourced driver at her starting time t_j^{ω}. Furthermore, δ_j^{ω} represents the duration that a crowdsourced driver will work, indicating that she leaves the system at time $t_j^{\omega} + \delta_j^{\omega}$. After this, she will not accept any future requests. A crowdsourced driver can leave only when she is idle. Hereafter, we denote the active crowdsourced drivers at time t as a set $\mathcal{J}_t$. This setting for crowdsourced drivers resembles the configuration studied in Ulmer and Savelsbergh (2020). For the sake of readability, in the following we describe the MDP associated with each realization ω of crowdsourced driver arrivals, avoiding the use of index ω in the notation.

The rewards of inhouse drivers consist of two parts: on the one hand, they receive a fixed amount of wage r_0, independent of their daily delivery performance; on the other hand, they also acquire a bonus of r_1 per delivery task completed. For the crowdsourced drivers, they are paid a reward r_2 by completing a delivery task. Since dynamic pricing is out of scope for current study, we assume that r_1 and r_2 are constant values, representing the average rewards for inhouse and crowdsourced drivers, respectively. We further assume $r_2 \ge r_1$, consistently with the realistic situation reported in the Table 1. Note that, we rule out the uncertainties in traveling and service times, therefore each order is supposed to be delivered on time if the driver follows the designed route.

We keep track of the status of *open* orders, that is, the order arrival, pick-up at a restaurant and delivery to a customer. Once the meal is received by the customer, its delivery task is completed, the service is finished and the order is *closed*. If we define the service time target as κ, it means that each request should be served within κ units of time, say minutes. In case an order cannot be served within κ minutes by any active driver, we assume that the platform assigns the request to an external operator, asking for an extremely high reward r_3. This is equivalent to decline the request at a high lost-sales value r_3 and we refer to this action as *rejecting* the request in the following.

Given the above assumptions, the scheduling and routing problem can be modeled as an MDP in continuous-time scale. We sketch the key elements of the MDP in the following.

3.3. MDP description

At each *stage* a decision is taken based on the orders information available on the platform. Then two types of stages should be considered in the MDP: *interim stage* and *final stage*. An interim stage is the arrival of a request at time $t \in [0, T]$, and the final stage is the achievement of the time horizon T. In each interim stage, the decision is about rejection or acceptance-scheduling of a request. An interim stage is described as a request tuple (t, i, j), where t is the time of occurrence, i is the customer location asking for meal and j is the restaurant. The final stage is defined as $(T, \emptyset, \emptyset)$. Since every customer may request meal from any potential restaurant at most once, the system has at most $|C||\mathcal{P}|$ requests, and the set of potential O-D pairs is $\mathcal{Q} = \{(i, j) : i \in C, j \in \mathcal{P}\}$. We denote $n \in \{1, 2, \dots, h\}$ $(h \le |C||\mathcal{P}| + 1)$ as the stage counter and t_n as the time at which stage n takes place.

The *state* of the system has to include all the information about the current status of drivers and their allocated orders. Given the inhouse drivers and a crowdsourced driver realization $\omega \in \Omega$, we know that upon the arrival of request $I_n = (t_n, i_n, j_n)$, the set of active drivers circulating in the service area at stage n can be denoted as $\mathbf{U}_n = \mathcal{M} \cup \mathcal{J}_n$. For each active driver $u \in \mathbf{U}_n$, we define a tuple $(\mathbf{Y}_n^u, \mathbf{p}_n^u, l_n^u)$, where set $\mathbf{Y}_n^u$ includes u's open (unfinished) orders, vector $\mathbf{p}_n^u$ represents u's scheduled path at t_n to deliver these open orders of $\mathbf{Y}_n^u$, and scalar l_n^u tracks u's location along $\mathbf{p}_n^u$ at time t_n. Notably, if $\mathbf{Y}_n^u = \emptyset$, then the driver u is idle at time t_n; otherwise, $\mathbf{Y}_n^u$ records the holding orders' relevant information and indicates whether these orders are picked-up or not. Thus, we describe the state of the system at the nth stage as $\mathbf{S}_n = \bigcup_{u \in \mathbf{U}_n} (\mathbf{Y}_n^u, \mathbf{p}_n^u, l_n^u)$. In the initial stage, only the inhouse drivers are active and the initial state is $\mathbf{S}_0 = \bigcup_{u \in \mathcal{M}} (\emptyset, \emptyset, 0)$. At the end of the nth stage, we define a set $\overline{\mathcal{Q}}_n$ to represent appeared O-D pairs, i.e., the customer and restaurant paired locations occurred until the nth stage, and $\overline{\mathcal{Q}}_0 = \emptyset$.

Decision epochs are associated with any new order request arrival. An *action* is made upon the decision epoch, which comprises two components: reject/accept the request and select a driver if accept. Hence, we can represent the action for each decision epoch as a tuple $\mathbf{a}_n = (z_n, k_n)$, where z_n is a binary rejection/acceptance indicator and k_n indicates the unique index for the selected driver. If $z_n = 1$, a driver of index k_n will be selected from the drivers set and assigned to the request. Moreover, since the active drivers set at stage n is $\mathcal{M} \cup \mathcal{J}_n$, we know that when $k_n \le M$, an inhouse driver is selected to serve the request; otherwise, k_n represents a crowdsourced driver. A driver can be assigned to a request if there exists a route for the driver to dispatch all on-hand orders including the new one. Note that, due to computational efficiency requirements, it is too cumbersome to search for optimal routes in practical environments. Thus, routes are generated in a fast-effective heuristic way in this paper and are illustrated in Section 4.2.

Once an action is chosen in an interim stage, we have a *state transition*. The transition associated with the nth interim stage, identified by the request tuple (t_n, i_n, j_n), is $\mathbf{S}_n = \bigcup_{u \in \mathcal{M} \cup \mathcal{J}_n} (\mathbf{Y}_n^u, \mathbf{p}_n^u, l_n^u)$ where the values are updated as follows. First of all, whatever $z_n = 0$ or $z_n = 1$, the state components are updated as follows, for all $u \in \mathcal{M} \cup \mathcal{J}_n$,

$$\mathbf{p}_n^u = \mathbf{p}_{n-1}^u \setminus \zeta_{n-1,n}^u, \quad l_n^u = \eta_n^u, \quad \mathbf{Y}_n^u = \mathbf{Y}_{n-1}^u \setminus \xi_{n-1,n}^u. \tag{4}$$

Here, we need to clarify several details regarding the state update: assume a driver u moves as the designed route $\mathbf{p}_{n-1}^u$ between stage $n-1$ and n. At the beginning of nth stage, η_n^u is its real-time location, $\zeta_{n-1,n}^u$ are the visited restaurants and customers locations within the interval, and $\xi_{n-1,n}^u$ denotes the set of finished orders by the driver. The route $\mathbf{p}_{n-1}^u$ is derived by a route generation approach given the unvisited locations related to the driver u's open orders at the beginning of stage $n-1$, as shown in Section 4.2.

Furthermore, if $z_n = 1$, given the determined route, the attributes $\mathbf{Y}_n^u$ and $\mathbf{p}_n^u$ are updated as follows:

$$\mathbf{Y}_n^u, \mathbf{p}_n^u = \begin{cases} \mathbf{Y}_n^u \cup \{(t_n, i_n, j_n)\}, \mathbf{p}_n^u \oplus \{(t_n, i_n, j_n)\} & u = k_n \quad \text{(a)} \\ \mathbf{Y}_n^u, \mathbf{p}_n^u & u \ne k_n \quad \text{(b)} \end{cases} \tag{5}$$

In (5a), $\mathbf{p}_n^k \oplus \{(t_n, i_n, j_n)\}$ represents the new route obtained by the aforementioned generation method when incorporating the newly accepted order.

For ease of exposition, we also define an operator Γ to describe the above state transition process, that is, $\mathbf{S}_n = \Gamma(\mathbf{S}_{n-1}, t_n, i_n, j_n, \mathbf{a}_n)$, meaning that the state $\mathbf{S}_n$ is determined by state $\mathbf{S}_{n-1}$, a request arrival tuple (t_n, i_n, j_n) and the corresponding action $\mathbf{a}_n$.

Lastly, minimizing the total cost is the *objective* of the delivery platform. It is measured by the fixed wages for inhouse drivers plus compensation cost for delivered orders. Given the strategic decision associated with the inhouse fleet size M, the fixed wages are exogenous to the operational level. Therefore, focusing on the operational decisions, we define $V_n^\pi(\mathbf{S}_{n-1}, t_n, i_n, j_n)$ to represent the expected compensation cost-to-go function produced by a given policy π when the interim stage (t_n, i_n, j_n) occurs in the system state $\mathbf{S}_{n-1}$. We write π^* for the optimal policy that achieves $V_n^* = \min_\pi \{V_n^\pi\}$ where:

$$\begin{aligned} V_n^*(\mathbf{S}_{n-1}, t_n, i_n, j_n) = \min_{\mathbf{a}_n} \Big\{ & r_1 \cdot \mathbf{1}(z_n = 1, k_n \le M) + r_2 \cdot \mathbf{1}(z_n = 1, k_n > M) \\ & + r_3 \cdot \mathbf{1}(z_n = 0) + \mathbb{E}\left[V_{n+1}(\Gamma(\mathbf{S}_{n-1}, t_n, i_n, j_n, \mathbf{a}_n), \right. \\ & \left. t_{n+1}, i_{n+1}, j_{n+1})\right] \Big\}. \end{aligned} \tag{6}$$

$$V_h^*(\mathbf{S}_{h-1}, T, \emptyset, \emptyset) = 0 \tag{7}$$

Eq. (7) indicates the state cost associated with the final stage $\mathbf{S}_h = (T, \emptyset, \emptyset)$ under the optimal policy, which is zero. Backwardly, Eq. (6) is derived through the induction from the final stage, where $\Gamma(\mathbf{S}_{n-1}, t_n, i_n, j_n, \mathbf{a}_n)$ represents the transition function to the new state $\mathbf{S}_n$ with the aforementioned action $\mathbf{a}_n = (z_n, k_n)$.

The above recursively cost-to-go function V^* is derived given a specific crowdsourced drivers realization $\omega \in \Omega$, as illustrated in Ulmer and Savelsbergh (2020). For notation convenience, let us denote such an optimal compensation cost to serve all orders (from the initial stage to the end of planning horizon) as $V_0^*(\mathbf{S}_0 | \omega, M)$ for a given fleet size M. As a realization $\omega \in \Omega$ occurs with even probability, we thus have an expected compensation cost given uncertain crowdsourced drivers scenarios as

$$\mathcal{V}^*(\mathbf{S}_0, \Omega, M) = \frac{\sum_{\omega \in \Omega} V_0^*(\mathbf{S}_0 | \omega, M)}{|\Omega|} \tag{8}$$

3.4. MDP optimal policy analysis

In this section, we discuss the structural properties of optimal policies for the problem described above. Obviously, since ω is an exogenous factor to the cost-to-go function and each realization ω is equally likely to occur, we only need to characterize the properties

for the recursive function (6) for one realization. For the sake of conciseness, all proofs are moved to Appendix 7.1.

We start by deriving the optimal policy that minimizes the cost-to-go Eq. (6), on the basis of the transition functions between stages. At the final stage $(T,\emptyset,\emptyset)$, the cost-to-go function associated with any policy π is deterministic because of no further request and incurring cost, we thus have $V_h^{\pi}(\mathbf{S}_{h-1},T,\emptyset,\emptyset)=0$. For an interim stage (t,i,j), that is, at stage $n\in[0,h)$, we need to compute the transition probabilities to the next stage in order to derive the expected cost-to-go value. We first analyze the cost-to-go function for the interim stage $(t_{n+1},i_{n+1},j_{n+1})$ when the state is $\mathbf{S}_n$. This means that, at time t_{n+1}, a request of O-D pair (i_{n+1},j_{n+1}) is placed and we aim to estimate the expected compensation cost-to-go function for the current stage.

Lemma 3.1. *Assume that the interim stage $(t_{n+1},i_{n+1},j_{n+1})$ occurs with system at state $\mathbf{S}_n$. Then, the probability that stage $n+2$ is an interim stage occurring at time $t>t_{n+1}$, and involves an O-D pair $(i,j)\in\mathcal{Q}\setminus\overline{\mathcal{Q}}_{n+1}$, is described as*

$$p_{\min}((i,j)|t_{n+1},\overline{\mathcal{Q}}_{n+1})=\int_{t_{n+1}}^{T}\psi^{t}_{ij|t_{n+1},\overline{\mathcal{Q}}_{n+1}}(t)dt \tag{9}$$

where $\overline{\mathcal{Q}}_n$ represent the already requested O-D pairs in $\mathbf{S}_n$, $\overline{\mathcal{Q}}_{n+1}=\overline{\mathcal{Q}}_n\cup\{(i_{n+1},j_{n+1})\}$ and

$$\psi^{t}_{ij|t_{n+1},\overline{\mathcal{Q}}_{n+1}}(t)=\left[\prod_{(\hat{i},\hat{j})\in\mathcal{Q}\setminus\overline{\mathcal{Q}}_{n+1}\cup\{(i,j)\}}\left(\frac{1-\theta_{\hat{i}\hat{j}}\overline{F}_{\hat{i}\hat{j}}(t)}{1-\theta_{\hat{i}\hat{j}}\overline{F}_{\hat{i}\hat{j}}(t_{n+1})}\right)\right]\frac{\theta_{ij}\overline{f}_{ij}(t)}{1-\theta_{ij}\overline{F}_{ij}(t_{n+1})}. \tag{10}$$

Moreover, the probability that stage $n+2$ is the final stage is

$$\psi^{T}_{\emptyset|t_{n+1},\overline{\mathcal{Q}}_{n+1}}=p_{\min}(\emptyset|t_{n+1},\overline{\mathcal{Q}}_{n+1})=\prod_{(i,j)\in\mathcal{Q}\setminus\overline{\mathcal{Q}}_{n+1}}\left(\frac{1-\theta_{ij}}{1-\theta_{ij}\overline{F}_{ij}(t_{n+1})}\right). \tag{11}$$

From Lemma 3.1, we know the transition probabilities from $n+1$ to $n+2$. With the transition probabilities to next stages, we derive the corresponding optimal expected compensation cost-to-go function.

Proposition 3.2. *The expected compensation cost-to-go function $V^{\pi}_{n+1}(\mathbf{S}_n,t_{n+1},i_{n+1},j_{n+1})$ under the optimal policy π^* can be expressed recursively as:*

$$\begin{aligned}V^{*}_{n+1}(\mathbf{S}_n,t_{n+1},i_{n+1},j_{n+1})=\min_{\mathbf{a}_{n+1}}\Bigg\{&r_1\cdot\mathbf{1}(z_n=1,k_n\le M)\\&+r_2\cdot\mathbf{1}(z_n=1,k_n>M)\\&+r_3\cdot\mathbf{1}(z_n=0)+\sum_{(i,j)\in\mathcal{Q}\setminus\overline{\mathcal{Q}}_{n+1}}\int_{t_{n+1}}^{T}V^{*}_{n+2}(\mathbf{S}_{n+1},t,i,j)\psi^{t}_{ij|t_{n+1},\overline{\mathcal{Q}}_{n+1}}(t)dt\\&+V^{*}_{n+2}(\mathbf{S}_{n+1},T,\emptyset,\emptyset)\psi^{T}_{\emptyset|t_{n+1},\overline{\mathcal{Q}}_{n+1}}\Bigg\}\end{aligned} \tag{12}$$

where $\mathbf{S}_{n+1}=\Gamma(\mathbf{S}_n,t_{n+1},i_{n+1},j_{n+1},\mathbf{a}_{n+1})$ represents the state transition outcome given the current state $\mathbf{S}_n$, request $(t_{n+1},i_{n+1},j_{n+1})$ and action $\mathbf{a}_{n+1}=(z_n,k_n)$, following Eqs. (4), (5a), or (5b).

Proposition 3.2 states the optimal policy for the expected compensation cost-to-go function. Theoretically, following Eq. (12), we are able to identify the optimal action for each state through backward induction. However, it is obvious that, except for trivial cases, an exact evaluation of the formula in Proposition 3.2 is intractable. Even though it is computationally prohibitive to derive the optimal policy backwardly from the final stage, it still provides some hints: mimicking the transition behavior within limited stages enables to include future information into a fast-effective algorithm development, as explained in Section 4.

Proposition 3.3. *The optimal expected compensation cost $\mathcal{V}^*(\mathbf{S}_0,\Omega,M)$ of (8) is non-increasing in the inhouse driver size M.*

Proposition 3.3 is straightforward. Once an additional inhouse driver is available, it is always possible to mimic the optimal policy in the case without this driver. Furthermore, given that the variable compensation for an inhouse driver is less than that of a crowdsourced driver (i.e., $r_1<r_2$), the overall compensation cost $V_0^*(\mathbf{S}_0|\omega,M)$ will not increase. As $\mathcal{V}^*(\mathbf{S}_0,\Omega,M)$ is a convex combination of $V_0^*(\mathbf{S}_0|\omega,M)$ in (8), it also implies that $\mathcal{V}^*(\mathbf{S}_0,\Omega,M)$ is non-increasing in M.

Since the aforementioned MDP suffers from the curses of dimensionality in the states, we now develop four fast-effective solution approaches to solve the problem in practical environments. Furthermore, to assess the effectiveness of any developed policy π, we define a metric Gaps(%) measuring the average performance deviation of the policy π from the optimal policy π^*, i.e., $\text{Gaps}(\%)=\left(\frac{\mathcal{V}^{\pi}-\mathcal{V}^{*}}{\mathcal{V}^{*}}\right)\times100$. Such metric is particularly useful when dealing with small-scale state spaces due to its computationally viability (as shown in Section 4.4). For large-scale state spaces situations, obtaining $\mathcal{V}^*$ can be intractable and it is often necessary to resort to approximations or heuristic approaches.

4. MDP solution approaches

In this section we present fast-effective policies for the scheduling and routing decisions of the MDP operational problem of Section 3, for a given fleet size M. We introduce dynamic procedures to tackle the information involvement and routes generation decisions, and later integrate these approaches together leading to four algorithms for the operational level problem.

The idea behind each approach is to take one decision at each stage, which is to determine the rejection/acceptance of a new request and the assignment of the new order to a specific driver, more precisely, to an inhouse driver or a crowdsourced driver. These approaches incorporate two sequential components: *information involvement* (`Info`) and *routes generation* (`Route`). For each component we provide two alternatives and integrate them together pair-wisely, thus obtaining four algorithms to solve the scheduling and routing problem. We now explain the details of `Info` and `Route` strategies in Sections 4.1 and 4.2, respectively. Note that all approaches are applied to a given realization ω of crowdsourced driver arrivals.

4.1. Information strategies

We propose two strategies to tackle the *information involvement* issue, which are mainly different in whether the future requests information gets involved into the decision process. These two strategies are named as *Myopic*- and *Monte-Carlo*-strategy.

Myopic-strategy (`MY`). This approach, summarized in Algorithm 1 in Appendix 7.3, makes a decision by using the information available at the time a decision is made, without considering future information. That is, upon each request arrival, the decision maker searches for the most suitable driver to serve the request on the basis of the realized information only. The driver selection is guided by a hierarchically greedy rule. When the nth request arrives, the active drivers set is $\mathcal{M}\cup\mathcal{J}_n$. On the top-level (line 5 of the Algorithm 1 in Appendix 7.3), the decision maker chooses a (or a set of) driver $\widetilde{U}\subseteq\mathcal{M}\cup\mathcal{J}_n$ with the minimum cost to dispatch this request. Note that, as we assume that $r_1\le r_2$ and the compensation is the same for all drivers of the same type, then either $\widetilde{U}$ is composed by inhouse drivers only, or it is composed by crowdsourced drivers only (in case there is no inhouse driver who can feasibly serve the request). The decision is dependent on the selected drivers' type and the corresponding rewards paid to serve the order. Given the chosen driver(s), on the low-level, the `Route` strategies (see Section 4.2) are adopted to generate a route $\mathbf{p}_n^u$ for each driver candidate $u\in\widetilde{U}$ and the additional travel time ΔTT_n^u is determined. At last, the driver with the minimum additional travel time is chosen. This strategy is simple and intuitive, since it assigns an order to the 'cheapest' and 'near-by' driver.

Monte-Carlo-strategy (MC). This approach uses Monte-Carlo simulation to predict the expected performance over the scheduling and routing decision. In particular, upon a new request arrival, a number of scenarios about future possible order arrivals are generated according to the request arrival probability distribution function. Given each scenario, the value obtained by a policy is evaluated.

Specifically, the probabilities of the future possible requests are computed as follows. Recall that, when the nth request $I_n \equiv (t_n, i_n, j_n)$ arises at time t_n associated with a customer i_n and a restaurant j_n, the state $\mathbf{S}_{n-1}$ records the requests already received (both accepted and rejected). We also know the already appeared O-D pairs set $\overline{\mathcal{Q}}_{n-1}$ and thus have $\overline{\mathcal{Q}}_n = \overline{\mathcal{Q}}_{n-1} \cup \{(i_n, j_n)\}$. The probability θ^n_{ij} that a request will arise from an O-D pair $(i,j) \in \mathcal{Q} \setminus \overline{\mathcal{Q}}_n$ can be computed using the CDF $F_{ij|t_n}$ on the random variable T_{ij}, given that no request has occurred for the O-D pair (i,j) within $[0, t_n]$. For all $t \in (t_n, T)$, according to the probability function Eq. (3), we get

$$F_{ij|t_n}(t) = \frac{\mathbb{P}(t_n < T_{ij} \le t)}{\mathbb{P}(T_{ij} > t_n)} = \frac{F_{ij}(t) - F_{ij}(t_n)}{1 - F_{ij}(t_n)} = \frac{\theta_{ij}(\overline{F}_{ij}(t) - \overline{F}_{ij}(t_n))}{1 - \theta_{ij}\overline{F}_{ij}(t_n)} \tag{13}$$

Thus, we have

$$\theta^n_{ij} = F_{ij|t_n}(T) = \frac{\theta_{ij}(1 - \overline{F}_{ij}(t_n))}{1 - \theta_{ij}\overline{F}_{ij}(t_n)}, \forall (i,j) \in \mathcal{Q} \setminus \overline{\mathcal{Q}}_n \tag{14}$$

which represents the probability to have a request for the pair (i,j) after t_n.

Given the request arrival probability distributions, the MC strategies sample the future requests arrival events thereby providing the prospected information. At each stage, the decision procedure consists of three key steps, *Monte-Carlo time sampling*, *Evaluation* and *Decision*. These components are organized as in Algorithm 2 in Appendix 7.3 and their details are described as follows:

1. *Monte-Carlo time sampling*. Given the nth stage request I_n, we randomly generate a set S of scenarios, where for each potential order request, the arrival time is determined. More precisely, each scenario $s \in S$ is a sequence of stages $(t_{ij,s}, i, j)$ with the O-D pair $(i,j) \in \mathcal{Q} \setminus \overline{\mathcal{Q}}_n$, and $t_n < t_{ij,s} < T$. Note that an arrival time is generated for every O-D pair $(i,j) \in \mathcal{Q} \setminus \overline{\mathcal{Q}}_n$ according to the CDF Eq. (3) with new θ^n_{ij} of Eq. (14), but only requests with arrival time less than T are kept in scenario s.
2. *Evaluation*. Given the active drivers set $\mathcal{M} \cup \mathcal{J}_n$ at stage n, for each scenario s, MC evaluates $1 + M + |\mathcal{J}_n|$ different possible actions. In particular, (i) it rejects the request I_n, (ii) it assigns the request I_n to an inhouse driver in $\mathcal{M}$, (iii) it assigns the request I_n to a crowdsourced driver in $\mathcal{J}_n$, and then schedules all future requests of s through the MY-based algorithm (see Section 4.3). Notably, in practical implementation, we can shorten the evaluation process by only processing a restricted number of future requests, rather than scheduling all requests in s, thus leading to a rollout algorithm (RA). In particular, we restrict the evaluation to the first χ requests (temporarily-wise) appearing in each scenario s, where χ represents the roll out period (Goodson et al., 2017). Theoretically, according to the recursive equation in Proposition 3.2, this procedure can evaluate all possible future events occurrence by letting $\chi \to \infty$ and choose the decision that minimize the expected total cost, and this would correspond to the optimal policy.
3. *Decision*. After evaluating the potential decision regarding the request I_n on all scenarios in S, the MC-based approach computes $\sum_{s \in S} \sigma_u(s)/|S|$ for all $u \in \mathcal{M} \cup \mathcal{J}_n \cup \{0\}$ and selects the driver with minimum total cost as the final decision (or rejects the request). Note that, we can also prioritize either inhouse or crowdsourced drivers during this decision step. However, the performance of this approach has been found to be less effective than the general selection process without prioritization (The detailed analysis is available in Appendix 7.4).

Obviously, the MC-based algorithm should achieve a better performance than MY but at the expense of intensive computational efforts.

In addition to the Info incorporation decision, another important issue regarding the decision procedure is the *route generation* for each candidate driver given the collected orders information. Specifically, we need to provide fast-effective routing guidelines for the chosen driver to deliver the unfinished orders. This is discussed in the following subsection.

4.2. Routing strategies

In this section, we introduce two routes generation strategies to build routes efficiently, namely, *Non-adaptive* and *Adaptive* routes generation approaches. The difference between the two is that the former inserts new locations into the original route without modifying the sequence of visits of other locations, while the latter reshuffles the current route and designs a completely new route given all provided locations.

Non-adaptive routes generation (NR). This method inserts the new request's locations into the original route without modifying the previous visiting sequence. The scheme of the approach is presented in Algorithm 3 in Appendix 7.3. The idea is to insert the locations associated with the new request in the current route, without modifying the visiting sequence of the vertices already included in the route. Given the request $I_n = \{t_n, i_n, j_n\}$ to be inserted, vertices i_n and j_n are inserted in the current route in the position leading to the *least additional traveling time* associated with the detour. Note that index n is discarded in the Algorithm 3 for the ease of reading. The insertion position needs to satisfy the following feasibility conditions:

1. *Time windows condition (TW)*: Each visited node in the route should satisfy the corresponding time windows, i.e., (earliest time) the meal can be fetched only if it is ready at the restaurant, and (latest time) each customer should receive the meal before the service time target. These requirements will be checked for all the nodes in the route. Note that the earliest time condition is checked only for nodes i and j associated with the new request as they are satisfied by construction for the other nodes. Operator $TW(r,i,j)$ in the Algorithm 3 checks whether vertex i can be feasibly inserted in route r after vertex j, i.e., it checks whether time windows constraints are satisfied for i and for all vertices following j once i is inserted in r.
2. *Pickup-delivery condition (PD)*: Node j (the customer location associated with the new request) has to be visited after node i (the restaurant location associated with the new request). This condition is checked through operator $PD(r,i,j)$ where vertex i is inserted in route r after vertex j.

As described in the algorithm, the routes generated by NR does not modify the visiting sequence of vertices already present in the route. Such a procedure is efficient (i.e., fast), however, the solution quality might be penalized as it lacks flexibility and does not explore for better sequences to connect the unvisited locations. Motivated by this drawback, we design a more effective routes generation scheme as follows.

Adaptive routes generation (AR). Compared to the NR approach, the AR method reshuffles all nodes inside the original route. Therefore, it shows more adaptive characteristics when including the new information. In detail, this method first constructs a graph $G_n = (V_o \cup V_I, A_n)$ where V_o are the unvisited nodes, V_I are the pickup and delivery vertices of the new request, and A_n are the directed arcs between the nodes. The immediate destination of the original route is taken as the starting (depot) position. Then a completely new route is generated from scratch according to the *cheapest insertion* (Rosenkrantz et al., 1974) procedure considering the *TW* and *PD* conditions. We present the AR details in Appendix 7.3 Algorithm 4. In principle, the AR

Table 3
Integrated algorithms for operational scheduling and routing decisions.

Route \ Info	Myopic (MY)	Monte-Carlo (MC)
Non-adaptive (NR)	Myopic Non-adaptive (MY−NR)	Monte-Carlo Non-adaptive (MC−NR)
Adaptive (AR)	Myopic Adaptive (MY−AR)	Monte-Carlo Adaptive (MC−AR)

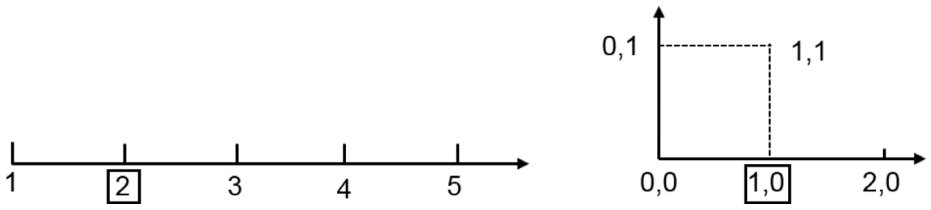

Fig. 3. The 1-D (left) and 2-D (right) geographical locations for shop (inside box) and customers.

Table 4
Gaps and Opt between heuristic and optimal policies for simple instances.

Policies	1-D				2-D			
	Inhouse driver		Mixed drivers		Inhouse driver		Mixed drivers	
	Gaps (%)	Opt (%)	Gaps (%)	Opt (%)	Gaps (%)	Opt (%)	Gaps (%)	Opt (%)
MY−NR	7.4	72.3	3.6	71.7	8.7	70.2	2.6	68.8
MY−AR	2.2	91.7	0.5	91.6	0.5	98.3	0.6	96.9
MC−NR	6.1	76.8	3.3	76.2	8.7	70.2	2.4	67.9
MC−AR	0.1	99.5	0.1	99.4	0.3	99.1	0.6	97.6

method should produce better solutions than NR with a sacrifice of computational efficiency. Compared with Ulmer and Savelsbergh (2020) and Zehtabian et al. (2022), where the cheapest insertion method is used, our AR method incorporates additional constraints and introduces more adaptability through reshuffling, thereby resulting in the generation of higher-quality solutions.

4.3. Integrated algorithms

Given the aforementioned Info and Route strategies, we integrate them together pair-wisely and derive four algorithms to solve the operational level scheduling and routing problems, which are the MY−NR, MY−AR, MC−NR and MC−AR algorithms (see Table 3).

4.4. Comparison of algorithms on simple instances

To give an illustration of the algorithms' advantage, we compare them on simple instances and shed light on the optimal policy. The setting of these instances are summarized as follows: time horizon $T = 7$ units, food processing time $\nu = 1$ unit, service time target $\kappa = 4$ units, and food quantity $q = 1$ unit. There is only one restaurant corresponding to the depot (where all routes, for both inhouse and crowdsourced drivers, start from) and 4 customers, which are distributed as one-dimensional (1-D) and two-dimensional (2-D) space as in Fig. 3. If we set the requests arising probabilities for all O-D pairs as 1, then there are 840 (i.e., A_7^4) order events scenarios.

We consider two cases of driver arrivals: a case with only one inhouse driver circulating in the system, and a case with mixed drivers (i.e., an inhouse driver and a crowdsourced driver). We set unit costs as $(r_1, r_2, r_3) = (6, 8, 18)$. The performance comparison of the proposed algorithms and the optimal policy π^* are summarized in Table 4. The performance of policy π^* is determined by enumerative analysis for each order events scenario. The column Gaps (%) has been defined in Section 3.4. In addition, given the optimal result known for each order arrival scenario, we also measure in column Opt (%) the percentage of scenarios where the heuristic algorithm π achieves the same value as the optimal policy π^*.

The results show that the strategies of incorporating both future information and adaptive route design are indeed advantageous. In particular, comparing the MC−AR with MY−NR among all scenarios composed by 1-D/2-D and different number of drivers, the overall average gaps to optimality are reduced by 5.3% and the proportion of instances reaching the value of the optimal policy π^* is increased by 28.2%. Moreover, the MC−AR policy outperforms the remaining ones, indicating the benefits of considering future information and invest efforts to produce better routes.

5. Numerical experiments

In this section, we present computational tests with a real dataset to verify the efficiency of the algorithms presented in Section 4, and also to identify whether the performance of different solution approaches depends on problem characteristics. We first describe the instances on which the tests have been performed in Section 5.1, and then show the computational results in the following sections.

5.1. Data preprocessing

We use the dataset from the OMD platform Meituan for our numerical experiments. We had access to a monthly dataset, from 3rd to 30th July 2017, including detailed orders and drivers information. The order information contains: Order ID, order arrival time, restaurant ID, locations of the restaurant and the customer, food processing time and delivery starting and ending time. Driver information contain minute-wise number of idle and busy drivers in the network.

We now clarify the process of deriving key system parameters from the real dataset. Specifically, each location within the dataset is identified through a coordinate pair of latitude and longitude. By eliminating duplicate coordinate pairs, we construct the set of locations $\mathcal{L}$, which includes coordinates for all customers and restaurants over the month. The two subsets are denoted as $\mathcal{C} \subsetneq \mathcal{L}$ for customers and $\mathcal{P} \subsetneq \mathcal{L}$ for restaurants, respectively. Furthermore, our computational experiments are primarily focused on two scenarios: *no-rush* (T_o) and *rush* (T_r). T_o includes all orders placed between 7:00-10:00 during the month, whereas T_r encompasses orders from 10:00-14:00. Consequently, we can approximate the order request probability θ_{ij} for each scenario, which represents the probability in the Bernoulli distribution. Specifically, by analyzing orders in T_r, we define sets

Table 5
Accuracy test of the approximation model: Comparison with LKH-3.

Instance	Nodes	Area (km^2)	δ (/km^2)	ρ (km)	C	m	T (h)	w_1	S_1	l_1	d_1	w_2	S_2	l_2	d_2	LKH	M	Gaps (%)
CMT01H	50	36.5	1.4	4	30	8	4	8.4	30	2.6	3.0	8.4	25.8	2.3	3.0	2.5	2	20.73
CMT03T	100	48.1	2.1	5	30	8	4	6.8	30	2.1	2.5	6.8	31.8	2.3	2.5	2.2	4	12.79
CMT04Q	150	48.1	3.1	5	30	8	4	5.5	30	1.7	2.1	5.5	38.9	2.3	2.0	2.6	5	19.14
CMT05H	200	48.1	4.2	5	30	8	4	4.8	30	1.5	1.9	4.8	45.0	2.3	1.8	1.7	7	12.15
SR200	200	76	2.6	6	30	2	1.3	3.0	30	3.8	1.3	3.0	23.4	2.9	1.4	1.1	9	24.17
SR400	400	76	5.3	6	30	2	1.7	2.1	30	2.7	1.0	2.1	42.1	3.8	0.9	0.9	14	13.57
SR600	600	76	7.9	6	30	3	2.0	2.1	30	1.8	1.0	2.1	49.4	2.9	0.9	1.0	20	10.12
SR800	800	76	10.5	6	30	4	2.1	2.1	30	1.3	1.1	2.1	52.3	2.3	0.9	0.9	27	16.64
SR1000	1000	76	13.2	6	30	4	2.4	1.9	30	1.2	1.0	1.9	67.3	2.7	0.8	0.9	34	14.69
$SR_{T_o^3}$	320	76	4.2	6	30	6	3	3.7	30	1.5	1.6	3.7	43.8	2.3	1.5	–	10	–
$SR_{T_r^3}$	3516	76	46.3	6	30	8	4	1.6	30	0.5	0.9	1.6	139.2	2.3	0.6	–	118	–
$SR_{T_o^6}$	390	76	5.1	6	30	6	3	3.7	30	1.5	1.6	3.7	43.8	2.3	1.5	–	13	–
$SR_{T_r^6}$	2904	76	38.2	6	30	8	4	1.6	30	0.5	0.9	1.6	139.2	2.3	0.6	–	97	–
$SR_{T_o^9}$	462	76	6.1	6	30	6	3	3.7	30	1.5	1.6	3.7	43.8	2.3	1.5	–	15	–
$SR_{T_r^9}$	4158	76	54.7	6	30	8	4	1.6	30	0.5	0.9	1.6	139.2	2.3	0.6	–	139	–

*$v = 18$ km/h, $\tau = 0, \delta_p = \delta_c = \delta/2, S_1 = C, S_2 = \frac{vT}{v\tau+(4m/3(\delta_p+\delta_c))^{1/2}}$, see Appendix 7.2 for more details about the notations.

of customer locations $\mathcal{C}^r$ and restaurant locations $\mathcal{P}^r$. The request probability θ^r_{ij} is estimated based on the frequency of orders from customer $i \in \mathcal{C}^r$ to restaurant $j \in \mathcal{P}^r$, calculated as the total number of orders for the pair (i, j) during the month divided the number of days observed (28 days from 3rd to 30th of July), where the probability $0 \leq \theta^r_{ij} \leq 1$ is guaranteed since a customer typically places at most one order per mealtime. Additionally, we assume that the inter-arrival time of requests within rush hours adheres to a uniform distribution. The parameters for this distribution are determined from the average total number of orders placed during rush hours per day, based on historical data. This approach to approximating θ^r_{ij} and assuming a uniform distribution for inter-arrival times facilitates the simulation of daily order arrivals in a stationary manner, excluding fluctuations by focusing on shorter periods such as rush hours, as demonstrated in Fig. 1. In a similar vein, for the *no-rush* scenario, T_o, we determine the customer locations ($\mathcal{C}^o$), restaurant locations ($\mathcal{P}^o$), request probabilities (θ^o_{ij}), and the distribution of inter-arrival times.

Without loss of generality, we simplify our approach by selecting order data from three specific days (July 3rd, 6th, and 9th) and using this information to simulate order streams. This simulation replicates the actual historical data regarding order arrival times, customer and restaurant locations, and food processing time. For clarity, let us denote the orders during the rush hour on July 3rd as T^3_r, with analogous labels for the other days and for no-rush periods. For these chosen days, the volume of orders in rush and no-rush periods are $|T^3_r| = 1758, |T^6_r| = 1452, |T^9_r| = 2079$ and $|T^3_o| = 160, |T^6_o| = 195, |T^9_o| = 231$, respectively. We use these detailed scenarios, T_o and T_r, as initial data points to estimate the lower- and upper bounds for the required inhouse driver fleet size M, as described in Section 5.2.

5.2. Determining the range of fleet size M by continuous approximation

In this section, we aim to provide a range of the optimal number of inhouse drivers engaged for the delivery service, given the estimated set of orders to serve. With the notations of Section 3.2, the total cost function (to minimize) associated with a fleet size M is $C(M) = \mathcal{V}^*(\mathbf{S}_0, \Omega, M) + r_0 M$, where r_0 is the wage of an inhouse driver. Note that $C(M)$ can be decomposed into a strictly increasing term $r_0 M$ and a non-increasing term $\mathcal{V}^*(\mathbf{S}_0, \Omega, M)$ (see Proposition 3.3), hence motivating a binary search on optimal M^*. Even though we cannot mathematically prove that the marginal compensation cost decreases when M increases, we indeed observed such a trend in our tests. Thus, we now aim at finding good lower and upper bounds of the value of M to start the binary search. The upper bound $\overline{M}$ is estimated through a continuous approximation method inspired by Daganzo (1987) as shown in Appendix 7.2 with the orders information in rush hours T_r. Similarly, the lower bound $\underline{M}$ is obtained with the orders information in no-rush hour T_o. Given the values of $\underline{M}$ and $\overline{M}$, we then estimate the number of required inhouse drivers $M^* \in [\underline{M}, \overline{M}]$ which minimizes the total cost by a dichotomic search, using the methods described in Section 4 to solve the operational problem for each value of M examined during that search.

As a followup, we conducted numerical tests to verify the accuracy of the approximation method. Specifically, we applied the method on benchmark instances for the vehicle routing problem with pickup and delivery and on instances derived from our dataset. Then, we compare the average travel time per point computed by the state-of-art vehicle routing problem solvers and the approximation method. The LKH-3 (Helsgaun, 2017; Lin & Kernighan, 1973) heuristic is adopted here as the state-of-art solver for benchmark cost computation.

The parameters required in the approximation formula and results are summarized in Table 5. The testbed consists of several standard CMT-instances from Salhi and Nagy (1999) and SR-instances generated from our real dataset. The CMT-instances cover 50∼200 nodes with time windows and pickup-delivery information. The SR-instances own the same information but cover 200∼1000 nodes. We do not report the accuracy comparison with over 1000 nodes because the LKH-3 has difficulties in tackling instances over this scale. In Table 5, we first use the approximation model to derive the number of vehicles M required to cover the whole area. Such M is taken as an input parameter to the LKH-3 heuristic to compute the benchmark average travel time. Column Gaps (%) measures the difference of average travel time per point computed between the approximation formula and the LKH-3 solution, i.e., $|d_1 - \text{LKH}|/\text{LKH}\times 100$ or $|d_2 - \text{LKH}|/\text{LKH}\times 100$, depending on which approximation formula d_1 or d_2 in Appendix 7.2 is tight. The results show that such gaps are fairly acceptable (on average 16%). Therefore, we conclude that the approximation model performs well in estimating the number of required vehicles for delivery service. Finally, we determine that the upper bound $\overline{M} = 139$ and the lower bound $\underline{M} = 10$, where the upper bound represents the highest value of M calculated from the scenarios of T_r over three selected days, while the lower bound is derived from the lowest value of M in scenarios of T_0. Consequently, our subsequent analyses will explore the fleet size of in-house drivers M within the range between 10 and 139.

5.3. Policies performance

Given the estimated range of inhouse drivers fleet size M, we now test the routing and scheduling algorithms presented in Section 4 with extensive computational tests. All performances are calculated as the average values under the selected three days' order streams.

Default parameters. We set the default configuration of parameters as follows. First of all, we assume that the arrival of crowdsourced drivers follows a Poisson process with rate μ, dependent on the time

of a day. More precisely, the hourly arrival rates are μ_r in the *rush* scenario and μ_o in the *no-rush* scenario. According to our observation from Fig. 1, we set $\mu_r^1 = 25$ for 10:00–12:00, $\mu_r^2 = 10$ for 12:00–14:00, and $\mu_o = 5$ for 7:00–10:00. Each crowdsourced driver appears randomly in the region and will leave after 3∼4 hours working duration, following an uniform distribution $U(3, 4)$. With these assumptions, we generated 10 different realizations of the crowdsourced drivers arrivals as the set Ω (i.e., $|\Omega| = 10$). Each driver is restricted to hold at most 9 unfinished orders in parallel during the delivery. Moreover, the service time target is set as $\kappa = 60$ minutes, indicating that each received order should be completed within an hour. Finally, the algorithms are tested with order arrivals during both *no-rush* and *rush* hours, i.e., 7:00–14:00. We assume the fixed wage $r_0 = 60$ yuan (i.e., ∼10 USD), and variable compensation $(r_1, r_2, r_3) = (6, 8, 18)$ yuan per order, i.e., 0.9, 1.2, 2.8 USD. Note that, even though the rewards parameters are constant values here, the proposed approaches are quite adaptive to the dynamic pricing environments if the r_1 and r_2 are dynamically determined as functions of distances, busy rate and other factors. In the following simulation, without specific clarification, we only vary the investigated parameter for sensitivity analysis and keep the rest as the default.

Performances comparison across policies. To maintain our focus, we concentrate on the MC−AR policy and utilize the other algorithms as benchmarks to evaluate its effectiveness. Fig. 4 shows that the MC−AR policy dominates the other policies, demonstrating the benefits of incorporating future information and generating adaptive routes in each decision epoch. On average, the total cost is reduced by 6% when comparing MC−AR with MY−NR, which values around USD 200 million for the company if it used the latter policy, because the delivery related cost is around USD 35 billion in 2022 (Meituan, 2023). Notably, the effectiveness of the adaptive route generation algorithm appears to outperform the impact of information involvement. Specifically, while the MY−AR policy achieves a total cost reduction of 4.8%, the MC−NR policy achieves a marginal reduction of 1.5%. This indicates that the adaptive route generation plays a more significant role in enhancing performance compared to information involvement alone. However, incorporating future information can yield even greater benefits if the number of rollout horizons is expanded, although this enhancement comes at the expense of exponentially increasing computational efforts. The 3-periods rollout strategy is embedded inside the MC−AR and MC−NR algorithms in our computational tests. Additionally, our findings suggest that when the capacity is sufficiently large (for example, $M \geq 80$), the benefit of incorporating future information tends to diminish rapidly. This outcome is intuitive since a lack of delivery force ceases to be a bottleneck, and the platform primarily needs to focus on optimizing route decisions, resulting in the AR policies maintaining their superiority over the NR policies.

An inhouse driver size M^* that minimizes total cost. Given the range of M provided by continuous approximation in Section 5.2, we first determine the fleet size of inhouse drivers. Fig. 4 also presents the impact of inhouse driver size M on the total cost when applying the four algorithms for the operational decision presented in Section 4.3. Clearly, the total cost decreases first and increases second in M. Even though we cannot demonstrate this observation by a rigorous mathematical analysis, we still think that it provides key insights for practitioners. The numerical results indicate that the platform has to balance the trade-off related to seasonal demand: If the platform recruits more inhouse drivers, it certainly helps to reduce the compensation cost but also incurs higher fixed wages, which is a kind of redundant capacity waste for the no-rush hours. By contrast, if the delivery capacity of inhouse drivers is limited, this causes huge delay penalties due to the lack of enough delivery force during the rush hours. Consequently, an optimal intermediate size of inhouse drivers is required, thereby minimizing the total cost of fixed wages and variable compensations. In the end, one may argue that the continuous approximation to estimate the capacity size is unnecessary as we can still find the optimal M^* by enumeration. However, adopting an approximation scheme restricts the searching space into a known range and thus reduces the computational efforts. Otherwise, the decision-maker needs to enumerate from zero to an arbitrary upper bound. Given that the continuous approximation requires a negligible computing time and provides a good estimation of the required number of drivers, it definitely pays-off using it to obtain lower and upper bounds.

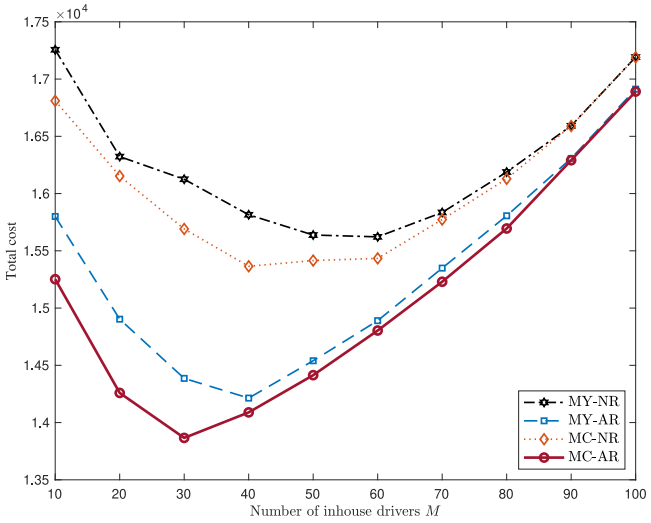

Fig. 4. Performance comparison for different policies.

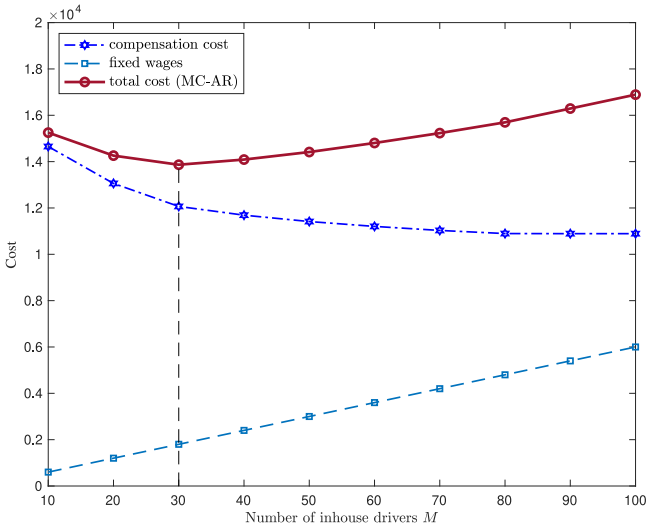

Fig. 5. Cost decomposition of the MC−AR policy.

The optimal fleet size of inhouse drivers is $M^* = 30$ under the MC−AR policy. By decomposing the total costs into different components (Fig. 5), we see that the variable compensation cost is decreasing in M (as shown in Proposition 3.3), while the fixed wages are constantly increasing in M. Note that, the marginal compensation cost decreases in a diminishing way, verifying the viability to determine an optimal M^* by dichotomic search. The best value M^* achieves a good balance between the fixed wages and the variable compensation cost. Let us take a further look at the optimal decision details for simulation results on July 3rd: The platform assigns 1290 orders to 30 inhouse drivers, 627 orders to 52 crowdsourced drivers and rejects no request. On average, an inhouse driver delivered 43 orders and a crowdsourced driver 12 orders. However, if the company decreases the value of M, several requests are rejected due to the shortage of drivers, while if the platform increases M, a higher percentage of orders are allocated to

Table 6
Impact of the crowdsourced driver arrival rate μ_r^1 on operational level performances.

M	μ_r^1	$\lvert\mathcal{J}\rvert$	$\lvert\mathcal{J}_{max}\rvert$	$\mathcal{V}^{\texttt{MC-AR}}$	$\overline{TT}$ (s)	%_in	%_out	%_rej
20	10	274	67	15 614	283.16	51.12	37.29	11.59
20	20	316	85	14 108	273.86	50.23	46.22	3.55
20	30	352	104	13 398	263.61	51.02	48.98	0.00
30	10	284	77	13 886	267.91	67.50	26.66	5.84
30	20	326	95	12 850	265.05	66.86	32.81	0.33
30	30	362	114	12 778	259.22	67.19	32.81	0.00
40	10	294	87	12 856	259.30	75.50	16.92	7.47
40	20	336	105	12 352	248.10	78.30	21.80	0.00
40	30	372	124	12 350	230.04	78.35	21.65	0.00

[a] $\lvert\mathcal{J}\rvert$ is the total number of drivers appeared in system through whole planning horizon.
[b] $\lvert\mathcal{J}_{max}\rvert$ is the maximal number of drivers appeared in system for the peak time.

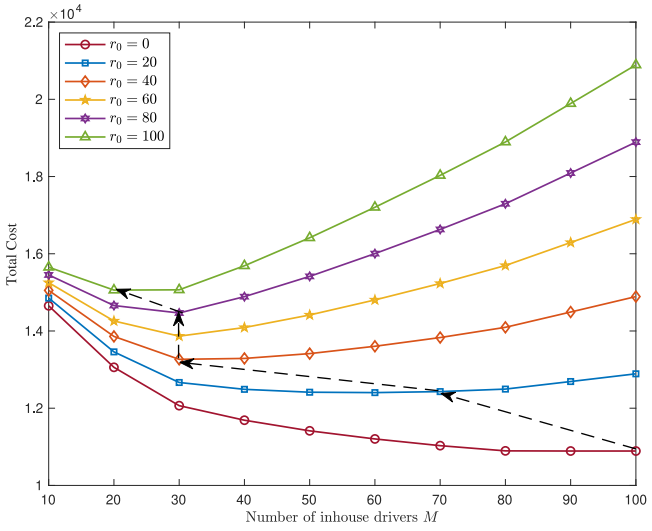

Fig. 6. Trace (dash arrow line) of optimal inhouse driver size M^* with respect to the fixed wage r_0.

inhouse drivers, but the increment of fixed wages is not compensated by the reduction of variable cost. In the following, we focus on the MC−AR policy with $M^* = 30$ and explore additional sensitivity results.

Impact of the crowdsourced drivers availability μ_r^1. Given a fixed size of inhouse drivers, we would like to know the impact of crowdsourced drivers participation on the total cost. Therefore, we vary the parameter μ_r^1 to investigate the cost changes. The results are summarized in Table 6. First of all, crowdsourced drivers are indeed helpful to reduce the requests rejection rate (column '%_rej') especially when the inhouse drivers resources are limited (e.g., $M = 20$). In this case, the percentage of orders allocated to the crowdsourced drivers (column '%_out') also increases in μ_r^1 since more crowdsourced drivers are available in the platform to provide service. Sometimes, it even slightly reduces the percentage '%_in' of requests served by inhouse drivers, in order to exploit the comparatively beneficial (e.g., nearby) crowdsourced resources. On the other hand, if the platform already recruited a large amount of inhouse drivers, the benefits of using crowdsourced drivers diminish as more orders are assigned to inhouse drivers. Besides, we find that the average travel time (column '$\overline{TT}(s)$') is decreasing in μ_r^1 due to more available drivers. In summary, through our experiments, we conclude that the adoption of crowdsourced drivers is beneficial to address the seasonal demand in OMD platforms, while such benefits are diminishing as the inhouse driver fleet capacity is expanded.

Impact of the inhouse driver fixed wage r_0. From the total cost definition, the optimal value of M is also affected by the fixed wage r_0. Note that if r_0 is very large, the total cost increases in M, given that the increasing wage cost $r_0 \cdot M$ is not compensated by the decreasing compensation cost $\mathcal{V}^*$. However, for the range [0,100] of r_0 considered

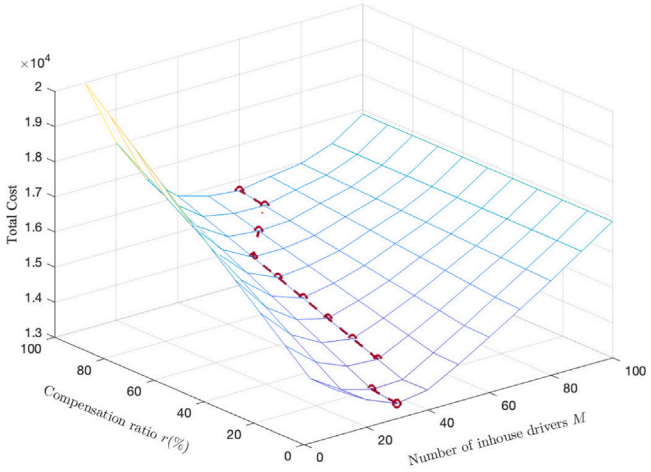

Fig. 7. Impact of the compensation ratio $\tilde{r}$ and the inhouse driver size M on total cost.

in our tests, we observe that the total cost is decreasing first and increasing second in the inhouse drivers fleet size. As indicated in Fig. 6, the optimal value of M (non-strictly) decreases in r_0. In detail, if r_0 is small, the platform prefers to recruit as many inhouse drivers as possible, since it is more expensive to compensate a crowdsourced driver than an inhouse driver. By contrast, if r_0 is large, the company prefers to adopt more crowdsourced drivers because the fixed wages paid to the inhouse drivers accounts for a high percentage of total cost. This is shown in Fig. 6 (top curve), where the incremental of total cost is dominantly caused by the fixed wages which are almost linearly increasing in the number of employed inhouse drivers.

Impact of the compensation ratio $\tilde{r}$. From the expected compensation cost-to-go function, we know that a key factor for the resource allocation (scheduling) decision is the compensation ratio $\tilde{r} = [(r_2/r_1)-1]\cdot 100$. Thus, we also investigate how this ratio affects the optimal capacity planning for inhouse fleet size and the order scheduling solutions. Intuitively, the percentage of orders allocated to crowdsourced drivers should be non-increasing in $\tilde{r}$. Given this trend, the optimal value of M is non-decreasing in $\tilde{r}$ (see the dash red line of Fig. 7) as the additional fixed cost is covered by the reduction of the crowdsourced drivers compensation. Therefore, the inhouse drivers and crowdsourced drivers are complementary in offering the meal delivery service. This is a useful insight for the platform managers, as they can build a team of inhouse drivers by determining the optimal size at the beginning and manipulate the compensation cost parameter to balance the orders allocation for total cost minimization. After all, the fleet size is a long-term decision and it is not altered easily in daily decisions, while the compensation cost can be adapted dynamically, as dynamic (surge) pricing studies indicated in this field (see for example Cachon et al., 2017).

Impact of the anticipatory assignment strategy. We conclude this section by analyzing the impact of the anticipatory assignment strategy (e.g., MC−AR) on scheduling outcomes, particularly in comparison to the myopic insertion policy (e.g., MY−NR). First, we found that the anticipatory assignment strategy results in shorter average travel times (y-axis '$\overline{TT}(s)$' in the left panel of Fig. 8). This occurs because the planner can group nearby delivery tasks when future order arrivals are anticipated, reducing overall travel time. However, the adaptive routing strategy can sometimes increase travel time, as it balances additional travel with the potential to serve more orders, in contrast to the non-adaptive routing approach which focuses on least travel time. Moreover, as shown in Fig. 8 (right panel), the anticipatory assignment strategy helps assign more orders to inhouse drivers ('%_in' of MC−AR) and reduces the number of rejected orders ('%_rej' of MC−AR). This preference for inhouse drivers is primarily driven by their

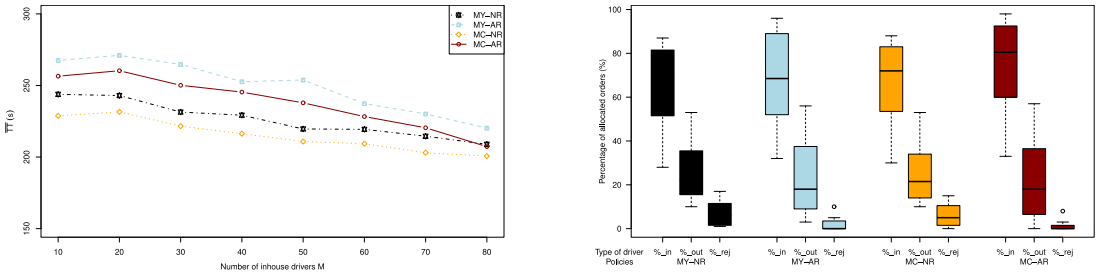

Fig. 8. Comparing the average travel time (in seconds) and the allocation of orders (in percentage) across different algorithms.

lower unit costs, encouraging planners to use them whenever available. Additionally, factors like order consolidation and non-urgent service levels also favor the use of inhouse drivers. For instance, our analysis shows that orders assigned to crowdsourced drivers have an average food processing time (FPT) of 10 minutes, while those assigned to inhouse drivers have an FPT of 20 minutes. This insight provides valuable guidance for planners in designing assignment strategies for OMD practitioners. When FPT is short and the associated time windows are tight, crowdsourced drivers can be a cost-efficient resource, allowing planners to reserve inhouse drivers for nearby or consolidated orders, which are more cost-effective.

In summary, we evaluate the performance of several developed policies for scheduling and routing orders in the OMD platform. By examining key design parameters, such as fleet size and the unit costs of using inhouse or crowdsourced drivers, we further gain valuable insights for practitioners that are helpful when making decisions.

6. Conclusions

In this paper, we studied the operational scheduling and dispatching problem in OMD platforms. The seasonal demand phenomena including rush and no-rush hours render the operational costs control rather challenging in practice: a large force secures the high service quality but also incurs high cost, while a small team reduces the cost and harms the service level, thus an appropriate capacity decision significantly impacts the operational cost. We are motivated by the practical operations of the Meituan platform in China. Meituan addresses the service quality and operational cost trade-off problem under seasonal demand by adopting the mixed delivery force (inhouse and crowdsourced drivers), which is a widely accepted strategy in most OMD platforms. We contribute to existing OMD studies by hierarchically addressing the planning, scheduling and routing problems. More precisely, given a known inhouse driver capacity, we described the operational level dynamic scheduling and dispatching problem as a Markov Decision Process and we presented the recursive formulation of the optimal compensation cost. We then developed simple-yet-efficient solution approaches to solve the scheduling and routing problem, through exploitation of the future information and adaptive routes generation procedures. Moreover, determining the inhouse driver fleet size that can minimize total cost under the uncertain crowdsourced drivers participation is yet another open question to consider. Thus, we tackled this issue by estimating the required capacity range through a continuous approximation model under the rush and no-rush scenarios, and further performing a dichotomic search for an appropriate capacity level. Through extensive numerical experiments based on a real dataset from Meituan, we observed that the designed policy is effective to solve the operational problems, confirming the benefits of incorporating future information, designing adaptive routes and determining appropriate capacity, thereby leading to the total cost reduction by 6% (at best) compared to the policy without these procedures.

By leveraging the system parameters for sensitivity analysis, our research leads to several important managerial insights. First of all, we find that it is possible to estimate the upper- and lower-bound of the required fleet size for online platforms endowed with the seasonal demand phenomenon, which provides a range for the managers to build (mixed) delivery force for service requirements. Moreover, through computational tests on the real-time scheduling and routing policies, we conclude that incorporating both future information and smart routes design is indeed beneficial to improve the operational performances. More importantly, a skillful routes generation scheme seems to be more effective in improving the performance compared to the information incorporation strategy, enlightening that if the practical computational resources are sufficient, the `MC-AR` policy is certainly the first choice, otherwise, deploying these limited resources into the route design optimization would be more advantageous to acquire an acceptable performance improvement even under the myopic strategy. Finally, the proposed hierarchical decision scheme to solve the strategic and operational problems supports the managers to determine the optimal size of inhouse force given the uncertain participation of the crowdsourced counterparts. Interestingly, we find that an intermediate size of inhouse fleet is optimal to minimize the total cost, which is also robust in the compensation ratio between inhouse and crowdsourced drivers, enabling the decision maker to set a dynamic compensation fee.

Our proposed policies and insights can be implemented in other platforms facing similar resource allocation challenges with mixed delivery force. For instance, Amazon builds its *Prime* team, together with *Flex* drivers to fulfill the growing e-commerce delivery tasks (Archetti & Bertazzi, 2021). These drivers allow the company to tackle the peak demand during Black Friday and Christmas holidays, without maintaining a large self-owned delivery team. Another example is related to retailers such as *Walmart* grocery, who also need to address similar strategic and operational problems with mixed delivery force.

We see several venues for future research. Our model could be extended to include the uncertain service and travel times of delivery force (Liu et al., 2021). Another venue would be studying the benefits of delaying the scheduling timing with an intentional postponement so as to create a thicker marketplace and provide more advantageous scheduling opportunities (Zhao et al., 2024). Lastly, a theoretically significant but challenging extension would be to develop performance guarantees for the proposed policies, as suggested in prior works (see, e.g., Ledvina et al., 2022; Simchi-Levi et al., 2005).

CRediT authorship contribution statement

Yanlu Zhao: Conceptualization, Data curation, Formal analysis, Investigation, Methodology, Software, Validation, Writing – original draft, Writing – review & editing. **Laurent Alfandari:** Conceptualization, Formal analysis, Methodology, Supervision, Validation, Writing – original draft, Writing – review & editing. **Claudia Archetti:** Conceptualization, Formal analysis, Methodology, Supervision, Validation, Writing – original draft, Writing – review & editing.

Acknowledgments

The authors sincerely thank the Editor and three anonymous referees for their valuable comments and suggestions, which significantly improved the quality of the paper.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.ejor.2024.11.028.

References

Afèche, P., Liu, Z., & Maglaras, C. (2023). Ride-hailing networks with strategic drivers: the impact of platform control capabilities on performance. *Manufacturing & Service Operations Management*, *25*(5), 1890–1908.

Archetti, C., & Bertazzi, L. (2021). Recent challenges in routing and inventory routing: E-commerce and last-mile delivery. *Networks*, *77*(2), 255–268.

Archetti, C., Savelsbergh, M., & Speranza, M. (2016). The vehicle routing problem with occasional drivers. *European Journal of Operational Research*, *254*(2), 472–480.

Arslan, A. M., Agatz, N., Kroon, L., & Zuidwijk, R. (2019). Crowdsourced delivery—A dynamic pickup and delivery problem with ad hoc drivers. *Transportation Science*, *53*(1), 222–235.

Berbeglia, G., Cordeau, J., & Laporte, G. (2010). Dynamic pickup and delivery problems. *European Journal of Operational Research*, *202*(1), 8–15.

Cachon, G., Daniels, K., & Lobel, R. (2017). The role of surge pricing on a service platform with self-scheduling capacity. *Manufacturing & Service Operations Management*, *19*(3), 368–384.

Chen, Z., & Xu, H. (2006). Dynamic column generation for dynamic vehicle routing with time windows. *Transportation Science*, *40*(1), 74–88.

Daganzo, C. F. (1987). Modeling distribution problems with time windows: Part I. *Transportation Science*, *21*(3), 171–179.

Dai, H., & Liu, P. (2020). Workforce planning for O2O delivery systems with crowdsourced drivers. *Annals of Operations Research*, *291*, 219–245.

Dayarian, I., & Savelsbergh, M. (2020). Crowdshipping and same-day delivery: Employing in-store customers to deliver online orders. *Production and Operations Management*, *29*(9), 2153–2174.

Ehmke, J., Campbell, A., & Urban, T. (2015). Ensuring service levels in routing problems with time windows and stochastic travel times. *European Journal of Operational Research*, *240*(2), 539–550.

Fatehi, S., & Wagner, M. R. (2021). Crowdsourcing last-mile deliveries. *Manufacturing & Service Operations Management*.

Goodson, J. C., Thomas, B. W., & Ohlmann, J. W. (2017). A rollout algorithm framework for heuristic solutions to finite-horizon stochastic dynamic programs. *European Journal of Operational Research*, *258*(1), 216–229.

Hall, J., & Krueger, A. (2018). An analysis of the labor market for uber's driver-partners in the United States. *ILR Review*, *71*(3), 705–732.

Helsgaun, K. (2017). *An extension of the Lin-Kernighan-Helsgaun TSP solver for constrained traveling salesman and vehicle routing problems* (pp. 24–50). Roskilde: Roskilde University.

Hirschberg, C., Rajko, A., Schumacher, T., & M, W. (2016). The changing market for food delivery. https://www.mckinsey.com/industries/technology-media-and-telecommunications/our-insights/the-changing-market-for-food-delivery.

Ichoua, S., Gendreau, M., & Potvin, J. (2000). Diversion issues in real-time vehicle dispatching. *Transportation Science*, *34*(4), 426–438.

Korosec, K., & Wilhelm, A. (2020). Huge delivery demand fails to stop uber's revenue from shrinking in Q3. https://uk.movies.yahoo.com/huge-delivery-demand-fails-stop-220819252.html.

Ledvina, K., Qin, H., Simchi-Levi, D., & Wei, Y. (2022). A new approach for vehicle routing with stochastic demand: Combining route assignment with process flexibility. *Operations Research*, *70*(5), 2655–2673.

Lin, S., & Kernighan, B. W. (1973). An effective heuristic algorithm for the traveling-salesman problem. *Operations Research*, *21*(2), 498–516.

Liu, S., He, L., & Shen, M. Z.-J. (2021). On-time last-mile delivery: Order assignment with travel-time predictors. *Management Science*, *67*(7), 4095–4119.

Meituan, D. (2023). ANNUAL report 2022. https://media-meituan.todayir.com/20230425215252173578660 4_en.pdf.

Naccache, S., Côté, J., & Coelho, L. (2018). The multi-pickup and delivery problem with time windows. *European Journal of Operational Research*, *269*(1), 353–362.

Powell, W., Jaillet, P., & Odoni, A. (1995). Stochastic and dynamic networks and routing. *Handbooks in Operations Research and Management Science*, *8*, 141–295.

Powell, W., Snow, W., & Cheung, R. (2000). Adaptive labeling algorithms for the dynamic assignment problem. *Transportation Science*, *34*(1), 50–66.

Reyes, D., Erera, A., Savelsbergh, M., Sahasrabudhe, S., & O'Neil, R. (2018). The meal delivery routing problem. *Optimization Online*.

Rosenkrantz, D. J., Stearns, R. E., & Lewis, P. M. (1974). Approximate algorithms for the traveling salesperson problem. In *15th annual symposium on switching and automata theory* (pp. 33–42). IEEE.

Salhi, S., & Nagy, G. (1999). A cluster insertion heuristic for single and multiple depot vehicle routing problems with backhauling. *Journal of the Operational Research Society*, *50*(10), 1034–1042.

Simchi-Levi, D., Chen, X., & Bramel, J. (2005). The logic of logistics: Theory. *Algorithms, and Applications for Logistics and Supply Chain Management*, 294–305.

Soeffker, N., Ulmer, M. W., & Mattfeld, D. C. (2021). Stochastic dynamic vehicle routing in the light of prescriptive analytics: A review. *European Journal of Operational Research*.

Tao, J., Dai, H., Chen, W., & Jiang, H. (2023). The value of personalized dispatch in O2O on-demand delivery services. *European Journal of Operational Research*, *304*(3), 1022–1035.

Taylor, T. A. (2018). On-demand service platforms. *Manufacturing & Service Operations Management*, *20*(4), 704–720.

Thomas, B. W., & White, C. C., III (2004). Anticipatory route selection. *Transportation Science*, *38*(4), 473–487.

Ulmer, M. W., & Savelsbergh, M. (2020). Workforce scheduling in the era of crowdsourced delivery. *Transportation Science*, *54*(4), 1113–1133.

Ulmer, M. W., Thomas, B. W., Campbell, A. M., & Woyak, N. (2021). The restaurant meal delivery problem: dynamic pickup and delivery with deadlines and random ready times. *Transportation Science*, *55*(1), 75–100.

Vasi, P. (2019). Real-time delivery order assignment policy: modeling and optimization. https://techweblabs.com/food-delivery-app-development/.

Voccia, S. A., Campbell, A. M., & Thomas, B. W. (2019). The same-day delivery problem for online purchases. *Transportation Science*, *53*(1), 167–184.

Yildiz, B., & Savelsbergh, M. (2019a). Provably high-quality solutions for the meal delivery routing problem. *Transportation Science*, *53*(5), 1372–1388.

Yildiz, B., & Savelsbergh, M. (2019b). Service and capacity planning in crowd-sourced delivery. *Transportation Research Part C (Emerging Technologies)*, *100*, 177–199.

Zehtabian, S., Larsen, C., & Wohlk, S. (2022). Estimation of the arrival time of deliveries by occasional drivers in a crowd-shipping setting. *European Journal of Operational Research*.

Zhao, Y., Papier, F., & Teo, C.-P. (2024). Market thickness in online food delivery platforms: The impact of food processing times. *Manufacturing & Service Operations Management*.