Crowdshipping and Same-day Delivery: Employing In-store Customers to Deliver Online Orders

Iman Dayarian
Culverhouse College of Business, University of Alabama, Tuscaloosa, USA,
idayarian@cba.ua.edu

Martin Savelsbergh
School of Industrial & Systems Engineering, Georgia Institute of Technology, Atlanta, USA,
martin.savelsbergh@isye.gatech.edu

Same-day delivery of online orders is becoming an indispensable service for large retailers. We explore an environment in which in-store customers supplement company drivers and deliver online orders on their way home. We consider a highly dynamic and stochastic same-day delivery environment in which online orders as well as in-store customers willing to make deliveries arrive throughout the day. Studying settings in which delivery capacity is uncertain is novel and practically relevant. Our proposed approaches are simple, yet effective and can be employed in practice. We develop two rolling horizon dispatching approaches: a myopic one that considers only the state of the system when making decisions, and one that also incorporates probabilistic information about future online order and in-store customer arrivals. We quantify the potential benefits of a novel form of crowdshipping for same-day delivery and demonstrate the value of exploiting probabilistic information about the future. We explore the advantages and disadvantages of this form of crowdshipping and show the impact of changes in environment characteristics, e.g., online order arrival pattern, company fleet size, and in-store customer compensation on its performance, i.e., service quality and operational cost.

Key words: Same-day delivery, Crowdshipping, Dynamic decision-making, Sample-scenario planning, Vehicle routing and scheduling

1. Introduction

Business-to-consumer (B2C) e-commerce sales worldwide reached $1.9 trillion in 2014, representing a doubling in sales compared to 2011 (Ecommerce-Foundation 2015). In this highly competitive environment, large retailers often seek to attract online buyers by providing fast delivery services, even when online shoppers are not willing to pay large fees for these delivery services. Not surprisingly, the B2C e-commerce growth has lead to renewed interest in last-mile delivery as the delivery to an online customer's home is one of the most expensive components of the order fulfillment cost. Last-mile delivery typically takes place in urban areas, where carriers face many challenges, such as traffic congestion and access limitations imposed by city administrations. Such challenges will only increase given the continuing urbanization; by 2050, 66% of the worlds population is expected to live in cities (Savelsbergh and Van Woensel 2016). Combining this trend with the desire to provide faster and faster delivery services to satisfy consumers' need for instant gratification in the online channel, large retailers have started to explore nontraditional last-mile delivery systems to provide cost-effective same-day (or even 2-hour) delivery services, see for example Banjo (2013), Barr and Wohl (2013), and Bensinger (2015).

At the same time, a plethora of start-ups is helping people share their underused assets, such as a spare room, a car, or even a cocktail dress, with others in the hope of making money or reducing their cost of ownership. This concept is referred to as *collaborative consumption* or *distributed economy*; some of the most successful companies in this space are Uber, Lyft, and AirBnB. In collaborative consumption, efficiency is generated by connecting the owner of an under-used asset and a user willing to pay for the use the asset. Walmart suggested a same-day last-mile delivery system based on these principles, in which in-store customers deliver one or more orders placed by online shoppers on their way home (for a small fee). That is, Walmart asks in-store customers to make available their time and vehicle to deliver packages to other shoppers. The system suggested by Walmart would be an example of *crowdshipping*, where organizations or persons who need to transport certain items are connecting with (being connected to) individuals willing to do so (Sampaio et al. 2017). In Walmart's proposal, these individuals are those customers going to one of their stores for their own shopping, but who are willing to deliver orders of online shoppers on their way home (or, more generally, to their next destination). If successful, such a system may allow retailers to provide more and faster delivery services at an acceptable cost. This form of crowdshipping would supplement existing delivery systems (whether operated by the retailers themselves or outsourced to a third-party), because retailers need to be sure that the level of service offered/guaranteed to their customers can be achieved. However, the incorporation of this form of crowdshipping may reduce cost, may reduce the company fleet size, and may allow for an expansion of service offerings.

Another form of crowdshipping is offered by Uber Freight and Amazon Flex, where individuals agree to transport items for a certain period of time and a certain fee, i.e., drivers sign up in advance for prefixed time-slots in return for a minimum pay guarantee. Meal delivery companies such as Doordash and Grubhub employ similar schemes. More recently, third party companies such as Deliv, Roadie, and PiggyBee created platforms for both individuals and retailers to ship packages using crowdsourced delivery capacities of independent shippers. We do not consider this form of crowdshipping in this paper, but we will elaborate on it and contrast it to the form of crowdshipping in which in-store customers are used to deliver orders of online shoppers towards the end of the paper.

1.1. Literature review

In this section, to better position our paper, we review recent work with similarities to the problem studied or our solution approach.

The considered crowdshipping studied in this paper as a variant of collaborative consumption has the features of a two-sided market (Weyl 2010, Rochet and Tirole 2006), which connects the online buyers to in-store customers for the task of goods delivery. In order to implement a successful and functional crowd-based business model, the companies face several challenges. The main challenge of any crowdsourcing platform is to incentivize and engage a large enough crowd who is willing to participate in the practice and also capable of adding value to the system (Kohler 2015). Furthermore, the two-sided platform must be balanced, as the value created for one entity strongly depends on participation on the other side (Kohler 2015, Hagiu and Wright 2015, Eisenmann et al. 2006, Evans 2003). An important difference between the considered crowdshipping setting and two-sided markets is that the two sides are not distinct. In crowdshipping, being either an online buyer or an in-store customer is a decision that customers of the store make, with more online buyers implying fewer in-store customers, and vice versa.

Archetti et al. (2016) conducted a preliminary investigation of a static version of the form of crowdshipping considered in this paper, which they refer to as the vehicle routing problem with occasional drivers. To the best of our knowledge, their paper is the only prior work considering the use of in-store customers to deliver orders of online shoppers. They observe that the compensation scheme, i.e., how much in-store customers get paid for their services, has a significant impact on the cost effectiveness of the (integrated) system. In a similar vein, a crowdsourced delivery system, modeled as a dynamic pickup and delivery problem aiming to utilize the excess capacity of the existing traffic flow in urban areas, was introduced and addressed by Arslan et al. (2018). The authors focus on a local peer-to-peer delivery platform that matches delivery tasks and ad-hoc drivers. Similar to the context of Archetti et al. (2016), the platform also operates a set of backup vehicles to serve tasks for which the use of an available ad-hoc driver is not feasible or not efficient. As such, the crowdsourcing provider needs to assign delivery tasks to ad-hoc drivers and backup vehicles and determine the associated delivery routes. The setting is designed for a same-day delivery context in which both tasks and drivers dynamically arrive over time. The proposed solution approach is based on a rolling horizon framework and repeatedly solving a matching formulation corresponding to various versions of the off-line problem.

Same-day delivery has been receiving much attention lately. Klapp et al. (2016) use a simple setting to study the core trade-off encountered in same-day delivery: whether to dispatch a vehicle to deliver known orders now or whether to wait for additional orders to arrive. Waiting for additional orders may lead to lower cost routes (i.e., with a lower cost per order delivered), but, at the same time, decreases the delivery flexibility for the known orders, as less time will be available for their delivery, which, in turn, may lead to higher cost routes. Three types of solution approaches are proposed and compared: constructing an a priori plan, a roll-out strategy in which a new "a priori" plan is constructed at certain points in time to take advantage of additional information, and a more involved strategy engaging approximate linear programming. Their results show that the, relatively simple, roll-out approach is almost as effective as the approach using approximate linear programming. Voccia et al. (2017) introduce a multi-vehicle dynamic one-to-many pickup and delivery problem with time constraints – in the form of delivery time windows. A sample-scenario planning method is proposed in which different scenarios are constructed based on random samples of customer request arrivals. A specifically designed consensus function takes advantage of the sampled information to guide vehicle route construction by identifying when waiting at the depot in anticipation of future requests is beneficial. Azi et al. (2012) consider a same-day delivery setting in which online orders can be accepted or rejected, i.e., an online order will be accepted, in real-time, based on the ability of the fleet to accommodate the request – guarantee same-day delivery. Each online order has an associated revenue and service time and the objective is to maximize profit, i.e., total revenue minus delivery cost. An acceptance rule is proposed that involves evaluating a number of possible future order arrival scenarios.

The primary difference with previous research on same-day delivery is that we study a setting in which there is not only uncertainty about future demand, but also uncertainty about future delivery capacity. Furthermore, we focus on the even more restrictive service offering in which delivery of an order has to take place within a fixed amount of time after the order is placed. We assume that the goal is to deliver all orders that are placed, which means that we have to simultaneously consider delivery cost and service quality (due to uncertainty of demand and delivery capacity, it is impossible to guarantee that the promised service is always achieved). Another important focus of the research is to assess the benefits of (and challenges associated with) incorporating probabilistic information on future demand and delivery capacity. We propose a sample-scenario planning method to do so.

1.2. Contributions

In summary, our contributions are as follows:

- We introduce a highly dynamic and stochastic routing problem in which not only the demand, in the form of online orders, arrives over time, but also part of the delivery capacity, in the form of in-store customers willing to make deliveries, arrives over time.
- We consider three variants of the problem, differing in the knowledge assumed available and exploited concerning future online order and in-store customer arrivals. In the static variant, full knowledge about future events (demand and delivery capacity) is available. In the (two) dynamic variants, no information or distributional information about future events is available. For each of these setting, we propose a decision making framework which concurrently advises (1) the assignment of online orders to the two sets of delivery capacities (company vehicles and in-store customers); and (2) the routing decisions for both company vehicles and in-store customers (including the sequence of visits to different locations, and dispatch times). We show that when distributional information about future demand and capacity is available, the service quality can be improved significantly.
- We study the interaction between service guarantee, permanent delivery capacity (e.g., company-owned vehicles), and temporary delivery capacity (e.g., in-store customers willing to make deliveries). We show that while an increase in company-owned delivery capacity can improve the service quality (i.e., on-time delivery), it does not necessarily reduce the operational costs. Obviously, a larger company fleet size involves higher fixed costs (vehicle acquisition costs and driver salary and benefit costs). On the other hand, we observe that

if the temporary delivery capacity is enlarged by offering higher compensation to in-store customers, the service quality may increase and the operational costs may decrease.

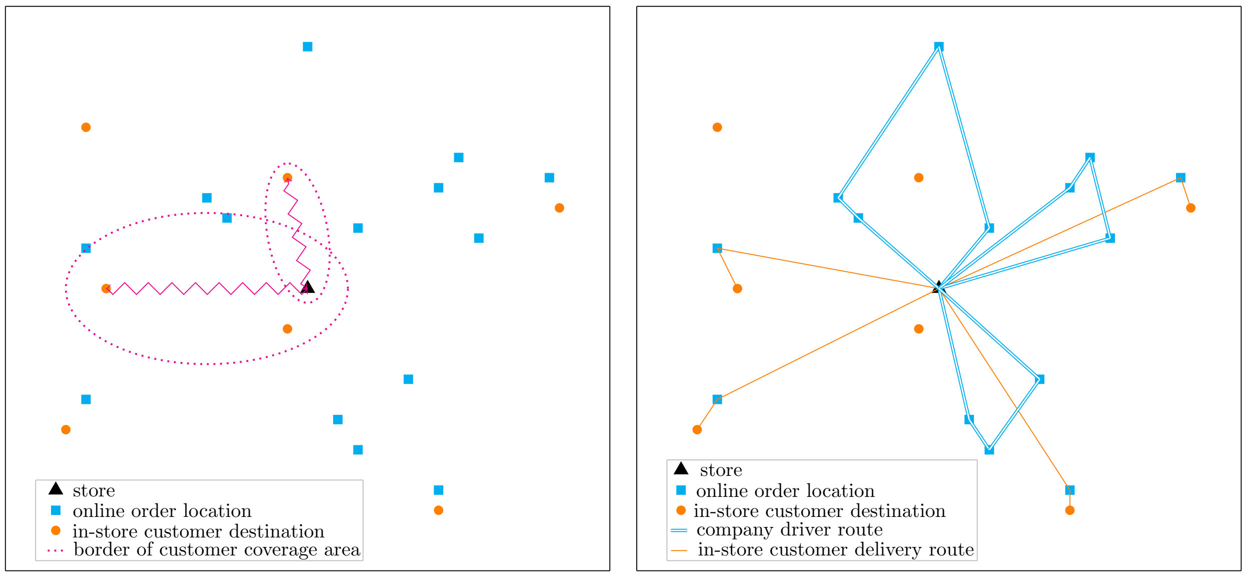
- We study the price-sensitivity of in-store customers, i.e., their willingness to participate as a function of the compensation offered. Higher compensation results in more in-store customers participating, but also in in-store customers becoming a less attractive option. When the temporary delivery capacity becomes less attractive from a cost point of view compared to the permanent delivery capacity, the system tends to use the temporary delivery capacity only when achieving the same level of customer service and operational cost efficiency is not possible with permanent delivery capacity. A high compensation to in-store customers pushes the system to a state of saturation where at times available temporary delivery capacity may not be fully used. As a result, rather than being the solution of choice, crowdshipping may become simply the back-up plan.

1.3. Paper structure

The remainder of the paper is organized as follows. Section 2 provides a detailed description of the problem. In Section 3, we introduce our solution approach for the static version of the problem, which forms a critical component of the rolling horizon dispatching approaches developed to address the dynamic version of the problem, which are elaborated in Section 4. The results of an extensive computational study are presented in Section 5. A comprehensive discussion of the advantages, limitations and possible extensions of our approaches is given in Section 6.

2. Problem statement

We consider a same-day home delivery problem in which a (single) store serves both as the location where in-store customers come to do their shopping and as the location where inventory is kept to fulfill orders from online customers. The novel aspect of the problem is that to fulfill orders from online customers, either company drivers or in-store customers willing to make a delivery can be used. Upon arrival at the store (or prior to arrival), a participating in-store customer notifies the store (using an app on his smartphone) about his willingness to deliver an online order and his destination after finishing shopping at the store. While the in-store customer is busy with his own shopping, the decision maker can decide to assign one or more online orders to the in-store customer, ensuring that the extra travel required to deliver the orders by the in-store customer is acceptable (what is


▲ store
■ online order location
● in-store customer destination
border of customer coverage area
▲ store
■ online order location
● in-store customer destination
company driver route
in-store customer delivery route

(a) Coverage area of in-store customers (b) A complete solution

Figure 1 A representation of a possible situation at a specific point in time

acceptable for the in-store customer has been specified at the time the in-store customer signs up for the program). If the decision maker decides to assign any online orders to the in-store customer, the in-store customer is notified (using the app on his smartphone) and the online orders will be waiting for the in-store customer in a designated area to be picked upon leaving the store. (More details will be provided later in this section.)

The set of locations where online orders need to be delivered as well as the set of locations where the in-store customer willing to make a delivery will go when they finish shopping are assumed to be known (and may overlap). Note that each location may potentially represent several customers (e.g., a potential location may represent a subdivision or an apartment complex).

More formally, the problem is defined on a graph $\mathcal{G} = (\mathcal{V}, \mathcal{A})$, where the node set $\mathcal{V}$ represents locations and the arc set $\mathcal{A}$ represents the connections between locations. Let $\mathcal{V} = \{d\} \cup \mathcal{V}^o \cup \mathcal{V}^c$ with d the store location, $\mathcal{V}^o$ the set of potential online order locations, and $\mathcal{V}^c$ the set of in-store customer home locations ($\mathcal{V}^o$ and $\mathcal{V}^c$ may overlap or even coincide). We assume that the travel time t_{uv} between two locations $u, v \in \mathcal{V}$ is deterministic. The online order placement rate at location $v \in \mathcal{V}^o$ is λ_v^o and the arrival rate at the store of in-store customers with home location $u \in \mathcal{V}^c$ is λ_u^c. Each location on the network (i.e., each node of graph $\mathcal{G} : v \in \mathcal{V} \backslash d$) may represent delivery location for multiple orders and home location of multiple in-store customers over a given time horizon. Let $\mathcal{N}^o$ and $\mathcal{N}^c$

represent a realization of demand and capacity, i.e., sets of actual online orders placed and in-store customers (willing to perform deliveries) arriving over the planning horizon T. Each online order $j \in \mathcal{N}^o$ has a delivery location in $\mathcal{V}^o$ denoted by v_j^o, and, similarly, each in-store customer $i \in \mathcal{N}^c$ has a home location in $\mathcal{V}^c$, denoted by u_i^c. Note that multiple online order delivery locations and multiple in-store home locations may be mapped to the same locations in $\mathcal{V}^o$ and $\mathcal{V}^c$, respectively. Hereafter, with a slight abuse of notation, we let t_{ij} denotes the travel time between the locations associated with i and j, e.g., the delivery location for an online order and the home location of an in-store customer. The placement time of online order j and the arrival time of in-store customer i at the store d are denoted by τ_j^o and τ_i^c, respectively. The service guarantee, representing the maximum time between the time an order is placed and the time the order is delivered at the buyer's location, is denoted by S. Thus, online order j has to be delivered no later than $\rho_j^o = \tau_j^o + S$, where we have implicitly assumed that no online orders are accepted after T. Note that in a more general setting, the service guarantee may be location dependent, by considering S_v to be the service guarantee at location $v \in \mathcal{V}^o$. The in-store customer availability is characterized by two values: r, the minimum time, after announcing his willingness to participate, an in-store customer requires at the store before being ready to make a delivery, and R, the maximum time, after announcing his willingness to participate, an in-store customer is willing to wait to make a delivery (otherwise he/she will leave the store).

The maximum number of orders that can be delivered by an in-store customer is denoted by Q^c. The inconvenience for an in-store customer is controlled by means of the in-store customer's coverage area. We assume that each in-store customer has an associated coverage area, comprising of a set of locations where they are willing to make deliveries. In our computational study, this set includes locations in a region, defined based on a *maximum detour rule*, encompassing the direct path between the store and the in-store customer's home location. Any intermediary visit to a location within this region would increase the travel time of the customer by a factor of $\gamma \geq 0$ or less of the direct path between the store and the customer's home location. Based on this rule, the coverage area of in-store customer i corresponds to the surface of an ellipse representing the locus of a moving point v such that $t_{dv} + t_{vi} \leq \gamma t_{di}$ (see Figure 1a). This definition of coverage area of in-store customers reflects our belief that individuals are more willing to deliver an online order in the periphery of the direct path between the store and their home locations. It is easy to adapt our approach to different definitions of coverage area (as mentioned above, we assume the coverage area for an in-store customer is determined at the time he signs up for the program). Note too that in-store customer parameters Q^c, γ, R, and r can be personalized to each in-store customer without affecting the solution approaches introduced in this paper.

Relying solely on in-store customers to deliver online orders is unlikely to be a feasible, and certainly a risky strategy. Customer loyalty is important to retailers and therefore it is critical that promised service to customers is met. Consequently, a retailer has to supplement (anticipated) in-store customer delivery capacity with company-owned delivery capacity, i.e., a set of company drivers and vehicles, to be able to provide same-day delivery of online orders. Let $\mathcal{K} = \{1, \ldots, K\}$ be the fleet of company vehicles available for deliveries at the store. While an in-store customer delivers a small number ($\leq Q^c$) of online orders in the vicinity of the path between the store and his home location, company drivers can, and likely will, deliver multiple online orders on a single trip (starting and ending at the store).

In addition to their vehicle capacity, the in-store customers differ from the corporate drivers in terms of their cost functions. Compensating in-store customers making deliveries of online orders in the form of store credits rather than actual payments, may be one creative way in which this is realized. Additionally, using in-store customers is less of a burden to a retailer than hiring company drivers (which implies paying a guaranteed salary and fringe benefits, on top of the purchase of the truck to be operated by the driver). Once a truck is purchased and a corporate driver is hired, the only costs to optimize are those variable costs related to the daily operations of the vehicles, namely their routing costs. Therefore, we assume that the cost of using company drivers is proportional to the total travel time of the routes they execute.

The compensation to in-store customers, however, is assumed to include two components. The first component, denoted by c^c_f, represents the fixed compensation paid to any in-store customer who delivers at least one order. The second component is assumed to be proportional to the extra time they need to travel to deliver the online orders assigned to them on the way to their next destination, with c^c being the per extra-minute compensation paid to an in-store customer. Assuming that the route of in-store customer i is denoted by Σ_i and function $T(\Sigma_i)$ returns the total travel time of the customer's route, then the compensation paid to in-store customer i is

$$c_i^c = c_f^c + c^c(T(\Sigma_i) - t_{di}). \tag{1}$$

In-store customer's route, Σ_i, and consequently its total travel time $T(\Sigma_i)$ depend on the set of orders delivered by the customer. The term $T(\Sigma_i) - t_{di}$ captures the extra travel time the in-store customer undergoes to deliver the assigned orders. Figure 1b depicts a solution in which some of the online orders are served by in-store customers and the remaining online orders are served by company drivers.

At a decision point h, the critical decision is whether to dispatch one or more company drivers to deliver some of the online orders that have not been assigned to in-store customers for delivery or whether to wait in anticipation of future placements of online orders and arrivals of in-store customers. That is, at each decision point, we have to decide which, if any, of the online orders that are not assigned to in-store customers are going to be delivered by company vehicles departing at that point in time (and, of course, also the assignment of those online orders to specific company vehicles and the routes of those vehicles). Postponing the delivery of order j at decision epoch h to decision epoch $h+1$ is done in the hope that between decision epochs h and $h+1$ either an in-store customer that can serve order j arrives in the store, or one or more online orders with delivery addresses close to that of order j are placed (or both), offering an opportunity to serve order j with a lower delivery cost (than initially thought). In the second case, delivering the order by a company driver may be less costly as it may be combined with other deliveries to addresses that are geographically close. However, postponing the delivery of order j also decreases its flexibility as it becomes more urgent (the next decision point will be closer to its due time, ρ_j^o).

The goal is to develop technology that decides at each decision epoch, based on the state of the system, (1) which active online orders to serve using available in-store customers, (2) which active online orders to serve using available company drivers, and, if any, how to assign them to the company vehicles and how to route the company vehicles, and (3) which active orders to postpone (to be considered again at the next decision epoch). As a decision will have to be made in near real-time, the efficiency of the technology is a critical concern.

We develop a technology for two variants of the problem: a static variant and a dynamic variant.

- **Static variant:** In this variant, complete information about the arrivals of in-store customers and placements of online orders is known *a priori.* We are interested in this variant for several reasons. First and foremost, the dispatching algorithm developed for the static variant forms a critical component of the dispatching algorithm developed for the dynamic variant. In addition, a solution to the static variant provides a baseline for comparison and allows us to assess the value of information.
- **Dynamic variant:** In this variant, information about the arrivals of in-store customers and placements of online orders is revealed over time. As a result, decisions have to be made dynamically through time as new information becomes available. We assume distributional information regarding future arrivals of in-store customers and placements of online orders is available and compare a dispatching algorithm in which this information is ignored to a dispatching algorithm in which this information is used.

Since limited time is available for decision making in the dynamic variant, we have chosen to focus on the design and implementation of heuristics that provide high-quality solutions in an acceptable amount of time.

3. Static variant

In the deterministic, static variant decisions are made *a priori.* It is assumed that the decision maker has *a priori* access to full information regarding the arrivals of in-store customers and the placements of online orders throughout the time horizon T. In such a situation, all of the decisions can be made once at the beginning of the time horizon. The decisions consist of (1) the assignment of online orders to in-store customers, taking into account the service guarantee of the orders, the availability, capacity, and coverage area of in-store customers, and the routing of the assigned orders, and (2) the company vehicle routes and dispatch times to serve the remaining orders. In the following sections, we discuss how these decisions are made in the case of the static variant.

3.1. Online order assignments and vehicle routing

For each order j, we define θ_j to be the latest dispatch time from the store, based on a direct delivery. The value of θ_j can simply be obtained as $\theta_j := \rho^o_j - t_{dj}$. Also, for each in-store customer $i \in \mathcal{N}^c$, let $I(i)$ be the set of online orders that can potentially be served by in-store customer i. For all in-store customer $i \in \mathcal{N}^c$, set $I(i)$ contains all orders $j \in \mathcal{N}^o$ for which the following three conditions are satisfied: (1) $t_{dj} + t_{ji} \leq \gamma t_{di}$, (2) $\tau_j^o \leq \tau_i^c + R$, and (3) $\tau_i^c + r \leq \theta_j$. Note that these conditions are necessary for an online order to be assigned to an in-store customer. However, since in-store customers can potentially deliver multiple orders on their routes, these conditions are not sufficient to identify the subset of orders that can possibly be served on a route by an in-store customer.

The assignment of the orders to the two types of delivery capacities, namely in-store customers and company drivers, and the routing decisions are made concurrently by solving a special variant of the vehicle routing problem (VRP). This variant of the VRP is characterized by (1) a heterogeneous fleet comprising of company vehicles and in-store customers' vehicles; and (2) order release and due times.

The two available types of vehicle differ in terms of:

- *Cost function:* While the operational (variable) cost of a company vehicle is proportional to the total travel time of the vehicle, the compensation paid to an in-store customer includes a fixed and a variable component as detailed in Eq. (1);
- *Route structure:* Company routes take the form of multi-trip routes, each trip starting and ending at the store. In-store customer routes, however, consist of a single trip starting at the store and ending at the home location of the in-store customer;
- *Vehicle capacity:* The number of orders that an in-store customer can deliver is limited to Q^c, while the capacity of company vehicles is not binding (due to high turnover of orders and relatively short service guarantee);
- *Scheduled late delivery:* Due to the potential lack of capacity, an order may be delivered late. However, we do not allow late delivery of an order by an in-store customer. This ensures that an in-store customer does not have to deal with potentially unhappy customers.

The goal is to identify a set of least-cost routes (both company vehicle and in-store customer routes) that minimizes the total delivery lateness (as an indication of service quality). That is, if in a given situation late deliveries are unavoidable, we aim at minimizing operational costs while keeping the total unavoidable lateness at its minimum (best achievable service quality). We have developed an efficient and effective tabu search heuristic to construct vehicle routes building on modern neighborhood search concepts, e.g., Nagata and Bräysy (2009), Vidal et al. (2013), and Cattaruzza et al. (2016). Note that we have chosen to develop a metaheuristic because in the dynamic environment decisions have to be made in near real-time (and a heuristic producing high-quality solutions in a short amount of time is what's needed). Details of our tabu search can be found in the online supplement.

A solution representation can be given using the following definitions:

- *Partial path (κ):* A sequence of visits to different locations by a vehicle;
- *Trip (σ):* A company vehicle trip consists of a path starting and ending at the store, delivering one or more orders, whereas an in-store customers trip starts at the store, ends at the customer's home location and delivers a maximum of Q^c orders;
- *Route (Σ):* For company vehicles, we denote a multi-trip route Σ as $(\sigma_1, \dots, \sigma_{M^\Sigma})$ with M^Σ being the number of trips. For convenience, we associate a *temporal* copy d_m of the store with each trip σ_m (from which the route departs) and refer to these as "intermediate" stores. Accordingly, each route consists of visits to online customer locations (to deliver orders) and visits to intermediate stores $\{d_1, \dots, d_{M^\Sigma}\}$. Because the in-store customer routes consist of one trip, the definition of in-store customer route coincides with the definition of an in-store customer trip;
- *Complete routing solution (ϕ):* A set of company vehicle and in-store customer routes delivering all orders.

In addition to producing the routes, the tabu search also generates information about partial paths, which is used to make decisions in other parts of our solution approach (to be discussed in the subsequent sections). The tabu search generates the following information:

Notation	Description
$T(.)$	Total travel time of a partial path, trip, route or complete solution
$LT(.)$	Total lateness of a partial path, trip, route or complete solution
$C(\phi)$	Utility function used for solution evaluation with $C(\phi) = T(\phi) + \eta LT(\phi)$, where η is a penalty term
$E(\kappa), L(\kappa)$	The earliest and latest possible visits to the first location of κ minimizing total route duration and delivery lateness.

The interested reader is referred to the online supplement for a more detailed description of how this information is generated in the course of the neighborhood search.

Note that the sequence of customer locations visited on a trip as well as the dispatch time of a trip are chosen so as to minimize the total due time violation (primary objective) and minimize the transportation cost (secondary objective). Balancing these two objectives is one of the challenges and one that is not typically seen in the routing and scheduling literature.

In the routing solution obtained through the tabu search, the active online orders are partitioned into two sets:

${}^{I}\mathcal{N}^o \subseteq \mathcal{N}^o$: the set of online orders assigned to a subset of available in-store customers $\mathcal{N}^c$;

${}^{R}\mathcal{N}^o \subseteq \mathcal{N}^o \setminus {}^{I}\mathcal{N}^o$: the set of online orders to be delivered on routes performed by company drivers.

4. Dynamic variant

In the dynamic version of the problem, decisions are made at various times during the planning horizon T, referred to as decision epochs, $h \in \mathcal{H}$. The state of the system at each decision epoch h can be represented by a tuple $(t^h, \mathcal{N}^o_h, \mathcal{N}^c_h, \mathcal{K}_h)$, where t^h is the time of the decision epoch, $\mathcal{N}^o_h$ is the set of active online orders, $\mathcal{N}^c_h$ is the set of available in-store customers, and $\mathcal{K}_h$ is the next availability of the company vehicles. The next availability of a company vehicle specifies the next time the vehicle can be dispatched from the store. For a company vehicle that is at the store at decision epoch h, the next availability is t^h, while for en-route company vehicles, the next availability is strictly greater than t^h. The set of available vehicles at the store at decision epoch h, is denoted by $\mathcal{K}^A_h := \{k \in \mathcal{K} | \mathcal{K}_h[k] \leq t^h\}$, and it defines the maximum number of company routes departing from the store at time h.

In our solution approaches, the set $\mathcal{H}$ comprises two types of decision epochs: (1) fixed times throughout the planning horizon (e.g., every δ minutes), and (2) upon every vehicle return to the store. Thus, $\mathcal{H}$ is a dynamic set and the number of decision times can vary based on the dispatch decisions made (for a given planning horizon T).

The goal of our proposed solution approaches is to provide a decision maker with a series of actions to be taken at epoch h given the system state at time h. The set of actions provided at each decision epoch h consists of the following sets:

${}^{I}\mathcal{N}^o_h \subseteq \mathcal{N}^o_h$: the subset of active online orders to be assigned to available in-store customers $\mathcal{N}^c_h$ (with information identifying the in-store customer making the delivery),

${}^{R}\mathcal{N}_h^o \subseteq \mathcal{N}_h^o \setminus {}^{I}\mathcal{N}_h^o$: the subset of active online orders to be delivered on routes performed by company drivers (with information on the delivery sequences of the routes);

${}^{P}\mathcal{N}_h^o = \mathcal{N}_h^o \setminus {}^{I}\mathcal{N}_h^o \cup {}^{R}\mathcal{N}_h^o$: the subset of active orders to be postponed to the next epoch. (This set is implied by the two previous sets.)

The above decisions are made such that all online orders are delivered and with the goal of minimizing the total lateness of deliveries and the overall cost of routes performed by the company drivers and compensation paid to in-store customers. Note that due to the uncertainty associated with the arrivals of online orders and in-store customers and the limited size of the fleet of company vehicles, it is impossible to guarantee that service is met for all online orders and lateness may occur.

We propose two solution approaches:

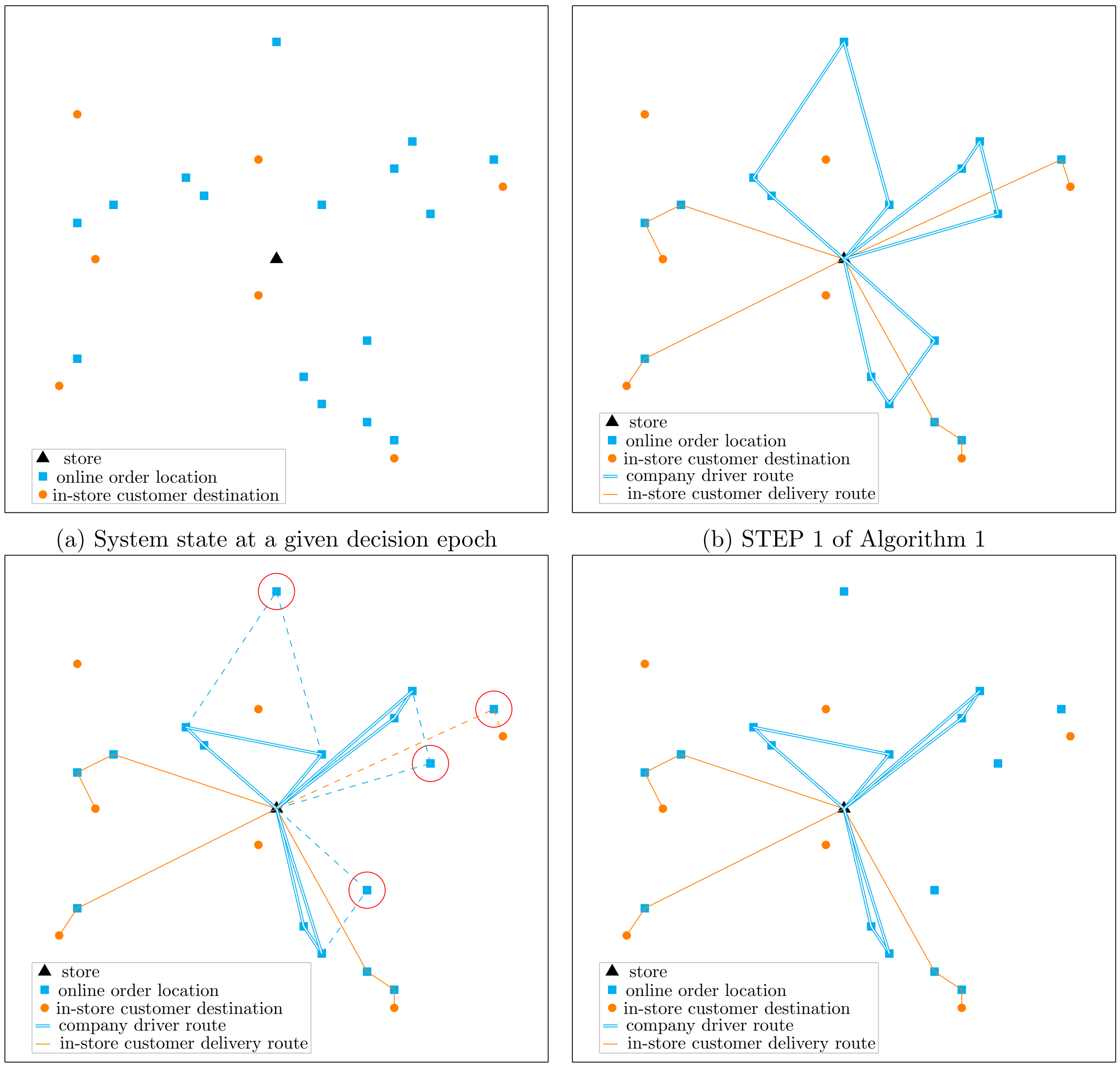
- **Myopic:** This solution approach ignores any information regarding future events; decisions are made solely based on the system state at every decision point.
- **Sample-scenario planning (SSP):** This solution approach uses the arrival rates of in-store customers and placement rates of online orders in addition to the system state at every decision point.

Both approaches use the same decision making framework but differ in the information about future events that is taken into account while making dispatching decisions. The decision making at a given epoch h follows the steps described in Algorithm 1 and is depicted in Fig. 2.

Algorithm 1: Decision logic at a decision epoch h

```
1 Given: $h$, $\mathcal{N}_h^o$, $\mathcal{N}_h^c$, and $\mathcal{K}_h$
2 STEP 1: Construct routes to serve all active online orders, $\mathcal{N}_h^o$, using a fleet including
    available in-store customers $\mathcal{N}_h^c$ and vehicles available at the store $\mathcal{K}_h^A$ (Fig. 2b)
  // Tabu Search
3 STEP 2: Release less urgent active online orders, ${}^{P}\mathcal{N}_h^o$, from the constructed routes
    (Fig. 2c)
4 STEP 3: Dispatch the company vehicles and in-store customers based on the
    resulting routes (Fig. 2d).
```

The main difference between the myopic and SSP-based approaches is the way the subset of orders to release from the current routes and postponed to the next decision epoch is


▲ store
■ online order location
● in-store customer destination
company driver route
in-store customer delivery route

(a) System state at a given decision epoch (b) STEP 1 of Algorithm 1

(c) STEP 2 of Algorithm 1 (d) STEP 3 of Algorithm 1

Figure 2 Decision making at each decision epoch of dynamic approaches

identified (STEP 2 of Algorithm 1). In the following sections, we discuss in detail each of these approaches.

4.1. Myopic decision making

In the myopic decision making approach, at every decision epoch $h \in \mathcal{H}$, we take into account only system state information to make decisions, i.e., $(h, \mathcal{N}_h^o, \mathcal{N}_h^c, \mathcal{K}_h)$. At each epoch, the goal is to make decisions regarding the set of orders to be served by the available in-store customers as well as vehicle routing decisions.

Let $I^h(i) \subseteq \mathcal{N}_h^o$ be the set of orders that can be served by in-store customer $i \in \mathcal{N}_h^o$. At any decision epoch h, a set of routes is generated to deliver all active online orders $\mathcal{N}_h^o$, using the set of available company vehicles $\mathcal{K}_h^A$ and the available in-store customers $\mathcal{N}_h^c$ (**STEP 1**). The routes are generated using a variant of the tabu search designed to solve the routing problem of the static variant. The only difference is that at any epoch h, only the subset of outstanding online orders, and the set of in-store customers that we are aware of are considered. However, not all active orders need to be dispatched immediately, as a subset of them can be postponed in the hope of finding a cheaper way to deliver them in the next decision epochs. More precisely, we identify a subset of orders j with $\theta_j \geq h+1$ whose dispatch will be postponed to decision epoch $h+1$ in the hope that one of the following scenarios plays out in the time interval $(h, h+1]$:

1. An in-store customer $\hat{i}$ with $j \in I^{h+1}(\hat{i})$ arrives in the store,
2. Online orders with delivery addresses close to the delivery address of order j are placed, so that the delivery cost of order j is reduced, if it is delivered together with (some of) these orders on a route.

The downside of postponing the delivery of order j to a later decision epoch is that its flexibility is decreased. In fact, delivering an order in a later decision epoch may result in a late delivery, even if $h+1 \leq \theta_j$.

In the myopic approach, decisions regarding which online orders to release from the routes are made based on a set of thresholds. The decision process is described next. (Pseudo-code is provided in Algorithm 2 in the online supplement.)

We denote the cost of the detour performed to serve online order j in the current routing solution ϕ at decision point h by $cost_j^h(\phi)$. Assuming order j is currently assigned to route Σ based on a given routing solution ϕ, the detour cost is obtained as

$$cost_j^h(\phi) : \begin{cases} T(\Sigma) - T(\Sigma_{(j)}), & \text{if } j \text{ is assigned to a company vehicle,} \\ \frac{c_f^c}{|\Sigma|} + c^c[T(\Sigma) - T(\Sigma_{(j)})], & \text{if } j \text{ is assigned to an in-store customer.} \end{cases} \tag{2}$$

In Eq. (2), $\Sigma_{(j)}$ represents the route after removing online order j and $|\Sigma|$ represents the number of online order delivered on route Σ. Depending on the type of route order j is assigned to in solution ϕ, the cost of serving it is calculated. If order j is assigned to a company vehicle route, cost of serving it is simply proportional to the detour made to deliver the order on the route. This can be calculated by taking the difference between the duration of the route with and without order j. This difference can be written as $cost_j^h(\phi) = t_{pred(j),j} + t_{j,succ(j)} - t_{pred(j),succ(j)}$. If order j is assigned to an in-store customer route, the cost of serving it depends on the compensation paid to the customer (Eq. (1)). The term $c_f^c/|\Sigma|$ indicates the share of each of those orders from the fixed component of the in-store customer compensation. Additionally, the term $c^c[T(\Sigma) - T(\Sigma_{(j)})]$ captures the variable component of in-store customer compensation associated with the delivery of online order j.

Let $cost_j^{h^+}$ represent an estimate of the cost of serving online order j at a decision epoch later that h. A simple estimate of the cost of serving online order j at a later decision epoch is $cost_j^{h^+} \approx 2t_{dj}$. Note that $2t_{dj}$ represents the cost of delivering order j on with an out-and-back direct delivery route of a company vehicle. Using this future cost estimate, an order j is released from its current route at epoch h if the value of $\Gamma_j = cost_j^h(\phi)/cost_j^{h^+}$ is greater than a predefined threshold. To better take the urgency of orders into account, orders are divided into two categories:

- *Semi-urgent:* Orders j that will become urgent soon ($t^{h+1} \leq \theta_j < t^{h+2}$),
- *Non-urgent* Orders j that will become urgent later ($\theta_j \geq t^{h+2}$).

Note that an order with $\theta_j < t^{h+1}$ (called an *urgent order*) must be dispatched at decision point h. Let α_1 and α_2, with $\alpha_1 \geq \alpha_2$, be the thresholds associated with semi-urgent and non-urgent orders, respectively. Then a semi-urgent order j is removed from its current route if $\Gamma_j \geq \alpha_1$ and a non-urgent order j' is released from its current route if $\Gamma_{j'} \geq \alpha_2$. (Parameters α_1 and α_2 are tuned using a procedure described in Section 5.2.) Note that $h+1$ is identified considering vehicle availability. That is, if at the next decision epoch of type “fixed time”, no vehicle will be available at the store, that decision epoch is skipped ($h+1 = \min h'_{h' \in \mathcal{H}} | h' > h \quad \text{and} \quad |\mathcal{K}_{h'}^A| \geq 1$). Obviously, the number of available vehicles at $h+2$ may depend on routing decisions made at h and $h+1$. Every time a vehicle is dispatched, its return to the store is inserted in $\mathcal{H}$ in its proper position ($\mathcal{H}$ is maintained in ascending order). Details of order release logic are provided in Algorithm 2 in Appendix EC.2.

We use the tabu search heuristic to generate routes in **STEP 1**. The routes may contain more than one trip. If a trip σ_i of a route generated at epoch h has a latest departure time $L(\sigma_i) \geq t^{h+1}$, then all orders on trip σ_i are released. It is worth mentioning that $L(\sigma_i) \geq t^{h+1}$

implies that the same vehicle to which orders on σ_i are currently assigned at epoch h will be available at the store to start trip σ_i at time t^{h+1}. Releasing such orders increases the chance of serving them by in-store customers arriving in time interval $(t^h, t^{h+1}]$ or possibly serving one or more orders placed in time interval $(t^h, t^{h+1}]$ on trip σ_i when dispatched at epoch $h+1$.

4.2. Sample-scenario planning (SSP)

In the sample-scenario planning approach, at every decision epoch $h \in \mathcal{H}$, we take into account not only system state information to make decisions, i.e., $(h, \mathcal{N}_h^o, \mathcal{N}_h^c, \mathcal{K}_h)$, but also the probabilistic information about the arrival rates of in-store customers and the placement rates of online orders. The sample-scenario-based approach follows the same steps as in Algorithm 1, but the actions taken in **STEP 2** are influenced by anticipated future events.

At a given decision epoch h, once a set of routes, both company vehicles and in-store customer routes, are generated (**STEP 1**), online orders are considered for potential release from their current route, to be postponed to a later decision epoch (**STEP 2**). Active online orders $j \in \mathcal{N}_h^o$ are considered to be released in a descending order of θ_j. The larger θ_j is, the less urgent order is.

In the myopic approach, thresholds were used to decide on the release of orders, because only a crude estimate of the cost of serving an order in the future, $cost_j^{h^+}$, was available. A more accurate estimate of the cost of serving order j in a future decision epoch allows a decision based on a direct comparison of the cost now and the (estimated) cost in the future. Moreover, at a decision epoch h, one may also be able to estimate the lateness caused by serving an order in a decision epoch later than h. Cost and lateness are the two elements to take into consideration when deciding whether to dispatch an order or to postpone its dispatch.

We note that in addition to the extra layer of complexity related to uncertain delivery capacity (associated with in-store customers), our proposed SSP differs fundamentally from the one proposed by Voccia et al. (2017) in the way the consensus between the information obtained from sampled scenarios is achieved. While their consensus function is based on a score function counting the number of times a partial plan is generated among the scenario plans, our SSP generates scenario plans to calculate the expected cost and lateness of delivering an order that is postponed to a later decision epoch. Those estimates are then compared to the current cost and lateness of delivering that order based on the routes generated in **STEP 1**, and a decision to release the order is made based on this comparison.

To estimate $cost_j^{h^+}$ and $\mathcal{L}_j^{h^+}$, i.e., the cost and lateness that can be attributed to online order $j \in \mathcal{N}_h^o \setminus {}^I\mathcal{N}_h^o$ if it is dispatched in a decision epoch later than h, we anticipate future events. Specifically, we generate a set of scenarios, each representing a realization of the future, using the arrival rates of in-store customers and the placement rates of online orders, and use these scenarios when estimating $cost_j^{h^+}$ and $\mathcal{L}_j^{h^+}$ for order j. Note that rather than generating a set of scenarios at each decision epoch, one can generate a set of scenarios for the entire planning horizon and use the relevant portions of these scenarios at each decision epoch. Let $\mathcal{S}$ be the set of scenarios generated.

In a decision epoch later than h, an order j may be assigned to an in-store customer or it will be delivered on a vehicle route dispatched after h. The cost of serving order j after h in scenario $s \in \mathcal{S}$, denoted by $cost_j^{sh^+}$, is equal to the compensation paid to an in-store customer if a match can be found, or the cost of serving j on a route dispatched after h, otherwise.

We now estimate the cost of serving order j in a decision epoch later than h with respect to a set of scenarios $\mathcal{S}$ as

$$cost_j^{h^+} = \frac{1}{|\mathcal{S}|} \sum_{s \in \mathcal{S}} cost_j^{sh^+}. \tag{3}$$

Similarly, the lateness that can be attributed to order j if it is served on a vehicle route departing at a decision epoch later than h with respect to a set of scenarios $\mathcal{S}$ is estimated as

$$\mathcal{L}_j^{h^+} = \frac{1}{|\mathcal{S}|} \sum_{s \in \mathcal{S}} \mathcal{L}_j^{sh^+}, \tag{4}$$

where $\mathcal{L}_j^{sh^+}$ is the lateness that can be attributed to order j in scenario s. Note that computing $\mathcal{L}_j^{sh^+}$ requires knowledge of the route on which j is served. That is, the solution to the routing problem for online orders postponed at time h or placed after time h, based on scenario s, needs to be explicitly constructed. Let $\mathcal{N}_h^{so}$ and $\mathcal{N}_h^{sc}$ denote the set of online orders placed and in-store customers arrived in the store after t^h according to scenario s, respectively $\mathcal{N}_h^{so} = \{j \in \mathcal{N}^{so} | \tau_j^o \geq t^h\}$ and $\mathcal{N}_h^{sc} = \{i \in \mathcal{N}^{sc} | \tau_i^c \geq t^h\}$).

Next, we discuss in more depth how $cost_j^{sh^+}$ and $\mathcal{L}_j^{sh^+}$ are computed. Using the tabu search described in Section 3, we solve the routing problem associated with each scenario s at decision epoch h, taking into account the set of online orders $\mathcal{N}_h^{so} \cup \{j\}$, the set of in-store customers $\mathcal{N}_h^{sc}$ and the set of available company vehicles $\mathcal{K}_h^A$. Note that the set of available vehicles at decision epoch h is identical for all scenarios, as it solely depends on the system state at that time.

The cost $cost_j^{sh^+}$ of delivering online order j if scenario $s \in \mathcal{S}$ realizes in a decision epoch later than h, is calculated based on Eq. (2). Accordingly, the cost estimate $cost_j^{h^+}$ of delivering online order j in a decision epoch later than h is calculated using Eq. (3). The lateness that can be attributed to j, $\mathcal{L}_j^{sh^+}$, is determined as follows. First, note that based on the rules we set in Section 3, an order is assigned to an in-store customer only if on-time delivery is guaranteed. Let Σ_j^{sh+} be the vehicle route delivering order j and let $\Sigma_{(j)}^{sh+}$ be the vehicle route obtained by removing j from Σ_j^{sh+}. Then the lateness that can be attributed to j is taken to be $LT(\Sigma_j^{sh+}) - LT(\Sigma_{(j)}^{sh+})$.

Observe that for small values of h (early stages of the planning horizon T), the set $\mathcal{N}_h^{so}$ may be quite large, in which case the time required by the tabu search algorithm to solve routing problems associated with all scenarios may be prohibitively large. However, the orders that are placed much later than j are likely to have little or no impact on the cost of serving j. Therefore, we accelerate the tabu search algorithm by limiting the set of events considered (with restricted lookahead). Specifically, we ignore all events after time β^h, where we set β^h to the latest decision epoch before $\hat{\theta} = \max\{\theta_j | j \in \mathcal{N}_h^o\}$, i.e., $\beta^h = \max\{h^v, h^\delta\}$, where h^v and h^δ are, respectively, the latest vehicle return and the latest fixed decision epoch before $\hat{\theta}$ (see Fig. 3):

$$h^v = \max\{t_v \in \mathcal{K}_h \mid t_v > h \text{ and } t_v \leq \hat{\theta}\},$$
$$h^\delta = \max\{\bar{h} \mid \bar{h} = \lfloor \frac{h}{\delta} \rfloor + z\delta, z \in \mathbb{Z}_+ \text{ and } \bar{h} \leq \hat{\theta}\}.$$

The idea behind the restricted lookahead β^h is related to the fact that for a given online order j, the latest departure time (based on a direct delivery path) to secure on-time delivery is given by θ_j. Moreover, decisions are only made at discrete decisions epochs, and therefore, the latest possible departure time of a vehicle delivering online order j would be at the latest decision epoch before θ_j. Accordingly, we consider $\mathcal{N}_h^{so}(\beta^h)$ instead of $\mathcal{N}_h^{so}$, where $\mathcal{N}_h^{so}(\beta^h) := \{j \in \mathcal{N}_h^{so} \mid t^h \leq \tau_j^{so} \leq \beta^h\}$, and $\mathcal{N}_h^{sc}(\beta^h)$ instead of $\mathcal{N}_h^{sc}$, where $\mathcal{N}_h^{sc}(\beta^h) := \{i \in \mathcal{N}_h^{sc} \mid t^h \leq \tau_i^{sc} \leq \beta^h\}$.

At each decision epoch h, we first solve the routing problem for the set of online orders $\mathcal{N}_h^o$, considering routes associated with available company vehicles $\mathcal{K}_h^A$ and in-store customers $\mathcal{N}_h^c$. Assume that ϕ is the obtained routing solution, then as described in Algorithm

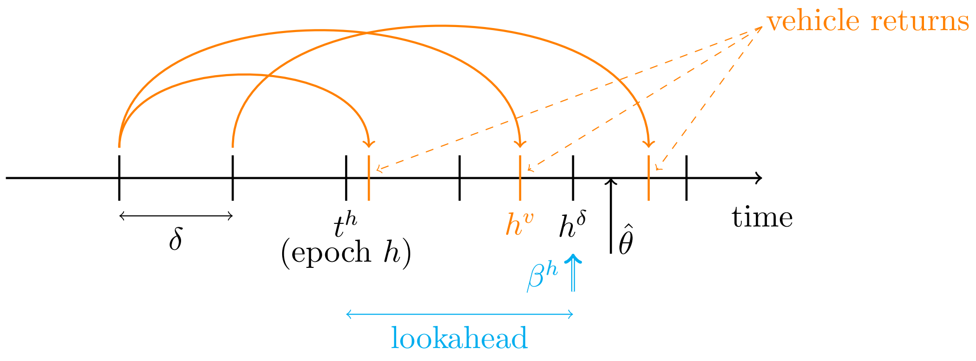


Figure 3 **Example of lookahead in decision epoch** h

3, Appendix EC.2, for each order $j \in \mathcal{N}_h^o$ we calculate $cost_j^h(\phi)$ and $\mathcal{L}_j^h(\phi)$ the cost of delivering online order j, and the total lateness caused by delivering online order j, respectively. On the other hand, we solve the routing problem associated with scenario $s \in \mathcal{S}$, for the set of online orders $\mathcal{N}_h^{so}(\beta^h)$. The routes generated for scenario s consider unused available company vehicles in decision epoch h and vehicles returning to the store in time interval $(t^h, \beta^h]$ in addition to in-store customers $\mathcal{N}_h^{sc}(\beta^h)$. Online orders $j \in \mathcal{N}_h^o$ are considered in descending order of θ_j for potentially being released from their current routes to be delivered on a route dispatched in a later decision epoch. Online order j is then tentatively inserted in the best positions in the routing solutions associated with all scenario $s \in \mathcal{S}$, allowing us to compute $cost_j^{sh^+}$ and $\mathcal{L}_j^{sh^+}$, and consequently $cost_j^{h^+}$ and $\mathcal{L}_j^{h^+}$. Order j is released from its current route in ϕ if $cost_j^h(\phi) + \pi\mathcal{L}_j^h(\phi) \geq cost_j^{h^+} + \pi\mathcal{L}_j^{h^+}$. Parameter π captures the relative importance given to service quality (lateness) versus the delivery cost. If we decide to release an order from ϕ, then it is kept in its inserted position in routing solutions of the scenarios; otherwise, it is kept in ϕ and instead it is removed from the routing solutions of the scenarios. These steps are repeated until no more orders for which a release is beneficial can be identified. Then, before committing to the routes in epoch h, the routes are reoptimized using tabu search.

Note that releasing an order impacts the cost to serve orders in the future in two ways: 1) the vehicle that served the order will return to the depot earlier, and 2) one more order needs to be served in the future. To save computation time, we only adjust vehicle returns once at the beginning of each iteration, and not after each order release in an iteration. (See Algorithm 3 in the online supplement.)

5. Computational Study

To study the performance of our proposed algorithms, we conduct a series of computational experiments. We first describe how instances are generated. Then we provide details on the training procedure designed to tune the thresholds used in the myopic approach. Next, we provide a set of results comparing the performance of the static and dynamic (myopic, SSP) approaches. Finally, we provide insights regarding sensitivity to problem characteristics, such as the company fleet size, the flexibility of in-store customers, the frequency of decision making, and the service guarantee.

5.1. Instances

We generate a large set of instances comprised of several instance classes. Each class of instances is characterized by the following attributes:

1. The delivery locations for online orders and the home locations for in-store customers,
2. The placement rates of online orders and the arrival rates of in-store customers.

For each instance in a class, we generate $|\mathcal{S}|+1$ realizations: $|\mathcal{S}|$ scenarios that are used to anticipate the future in the sample-scenario variants, and one scenario representing the future that actually materializes. Each realization (scenario) of a given instance has the following attributes:

- A temporal list of online orders with the coordinates of their delivery location and their placement time (τ^o_j),
- A temporal list of in-store customers with the coordinates of their home location and their arrival times (τ^c_i) in the store.

Moreover, the following parameters are set to the following values (unless stated otherwise) before solving each instance:

- Time horizon: $T = 480$ minutes,
- Service guarantee: $S = 60$ minutes,
- In-store customers' availability: $R = 30$ minutes,
- Minimum time given to in-store customers to complete their shopping in the store: $r = 5$ minutes,
- In-store customer maximum detour ratio: $\gamma = 1.25$,
- In-store customer compensation per extra minute travel: $c^c = \$0.5$,
- In-store customer fixed compensation: $c^c_f = \$2$,
- Maximum number of orders an in-store customer can deliver: $Q^c = 2$,

- Company fleet size: $|\mathcal{K}| = 7$,
- Time between fixed decision epochs: $\delta = 10$ minutes.

We generate 12 classes of instances based on 4 sets of locations and 3 sets of arrival rates.

5.1.1. Locations In our instances, the online orders and in-store customers are randomly located in a geographical region represented by a 100 × 100 square with the store is located at the center of the square. The delivery locations of online orders and the home locations of in-store customers are generated randomly in the region, favoring locations closer to the store. That is, a randomly generated candidate location j is kept with probability $exp(-\varphi \bar{d}_j)$, where $\bar{d}_j$ is the Euclidean distance from location j to the store and where we set $\varphi = 0.05$. Locations are generated until N locations have been kept. Note that the same set of locations is used for delivery locations of online orders and for home locations of in-store customers. We generate 4 sets of locations, each set containing $N = 50$ locations, referred to as Loc 1, Loc 2, Loc 3, and Loc 4. The travel times are obtained by normalizing the Euclidean distances in such a way that the farthest location in a location set can be reached from the store in 60 minutes.

5.1.2. Arrival Rates We consider fixed placement rate of online orders and fixed arrival rate of in-store customers, i.e., $\lambda_j^o = \lambda^o, \forall j \in \mathcal{V}^o$, and $\lambda_i^c = \lambda^c, \forall i \in \mathcal{V}^c$, following independent Poisson processes. To better study the interaction between the placement rate of online orders and the arrival rate of in-store customers, we generate three sets of rates according to the following rules:

- Rate1: $\lambda^o/\lambda^c = 2$
- Rate2: $\lambda^o/\lambda^c = 1$
- Rate3: $\lambda^o/\lambda^c = 0.5$

Note that in the first case, more online orders are placed than in-store customers arrive, which means that the effective use of company vehicles will be critical, and, the fleet size will likely have a significant impact. On the other hand, in the third case, more in-store customers arrive than online orders are placed, which means that the effective use of company vehicles will be less critical.

Each of the four location sets generated previously are matched with each of the above rates, creating 12 instance classes.

5.1.3. Scenario generation As mentioned before, each instance consists of one actual scenario and $\mathcal{S}$ scenarios. Considering the set of N locations and the online order placement and in-store customer arrival rates of an instance class, a realization of an instance is generated as follows.

Step 1: From each location $j \in \{1, \ldots, N\}$, the inter-arrival times of in-store customers and online orders from each location j are generated in two distinct time series. The inter-arrival times are distributed following exponential distributions $exp(\lambda^c)$ and $exp(\lambda^o)$. The generation of times in the time series of each location j is stopped when the horizon length T is exceeded.

Step 2: The final sets of order placements and in-store customer arrivals are created by combining the time series from each location j, for $j \in N$, and they are sorted in an ascending order of event times.

Note that multiple online orders or in-store customers can materialize from the same location during the planning horizon.

For each of the 12 instance classes, we generate 50 instances, constituting 600 instances. The first 20 instances in each class are categorized as the *test set* and the other 30 instances are categorized as the *training set*. In the myopic approach, the values of the thresholds are set based on a tuning strategy ran on the training set, for each instance class separately. These values are then used for the instances in the test sets.

5.2. Threshold tuning for the myopic approach

A pair of thresholds α_1 and α_2 (with $\alpha_1 \geq \alpha_2$) are generated for each instance class. We use a heuristic approach for parameter tuning. For each instance in the training set of a given instance class, we initially set $\alpha_1 := \alpha_2 := 0.1$. Next, α_1 is iteratively incremented with a step size of 0.1 up to 1.0. For each value of α_1, the value of α_2 is incremented in the interval $[0.1, \alpha_1]$ with a step-size 0.2. The performance of the algorithm for each threshold pair (α_1, α_2) is evaluated based on $C(\phi)$, with $\eta = 4$, where ϕ represents the routing solution obtained for the instance. The actual value of η is application dependent (how much importance one gives to on-time delivery of orders versus the transportation cost). Setting $\eta = 0$ gives no importance to the total lateness of a solution and focuses solely on transportation cost. The value of $\eta = 4$ was obtained through an intensive training procedure, considering different potential values for η, to scale the solution's total lateness versus total cost. The best threshold pair (α_1^*, α_2^*) for each instance class is identified as the average of those pairs minimizing $C(\phi)$ for all instances in the training set of that instance class.

5.3. Comparing the results of static and dynamic approaches

Table 1 compares the results obtained using different solution approaches on the instances in the test set of each instance class. Each line of the table corresponds to the average over 20 instances in the test set of each instance class. We present three categories of results. The first category, referred to as "Full information", corresponds to the results of the static approach in which perfect information regarding future events is assumed to be available a priori. The second category corresponds to the myopic approach, in which no information regarding future events is assumed to be available. The third category corresponds to the situation in which information regarding future events is available in the form of in-store customer arrival and order placement rates. The approaches are compared based on the following metrics:

cost (D): The total cost of the routes performed by company vehicles.

cost (I): The total compensation paid to in-store customers.

TC: The total delivery cost (cost (D) + cost (I)).

lateness: The total delivery lateness incurred.

# used in-cust: The total number of in-store customers used during the planning horizon.

time (s): The total time required to solve an instance for the static strategy.

time/epoch (s): The average time spent per decision epoch for an instance for the dynamic strategies.

gap1 %: The total cost (cost (D) + cost (I)) of a given solution approach compared to the total cost of the static approach (as a percentage).

Figure 4 shows a radar chart in which the three approaches (static, myopic, and SSP) are compared in terms of the key performance metrics.

Note that because the routing problem that has to be solved in the static version can be quite large, for some instances (especially Rate 1 instances) the tabu search was unable to find solutions with zero total lateness given the limit on the number of iterations. This may also mean that the considered fleet size (7 vehicles) is not large enough to ensure on-time delivery of all orders for those specific instances.

We see, as expected, that when more in-store customers are available (Rate2 and Rate3) a higher number of online orders is served by in-store customers, directly translates into lower total costs (fewer and/or shorter company vehicle routes). It also resulted in fewer online orders delivered per route, which led to a higher service quality (i.e., lower total lateness).

Comparing the results for the dynamic approaches reveals the value of anticipating future events. The SSP variant reduces the total lateness by 61%, on average. However, the total cost (company vehicles routing cost + in-store customer compensation) increases by 9%, on average. The primary reason that the SPP variant has smaller total lateness is its ability to better anticipate future capacity (i.e., in-store customers and company vehicles) and by identifying and releasing orders with a high chance to be served with lower lateness in the future.

A comparison of the time spent per decision epoch for the dynamic approaches reveals the added complexity introduced when anticipating future events. Since in the myopic approach, the release thresholds are set in advance (based on the analysis carried using the training set), the release decisions take little time whereas in the SSP variants at each decision epoch several routing problems are solved in order to decide on the release of orders. These results provide useful managerial insights by highlighting the trade-offs between the average time/epoch, the service quality and the routing cost.

In the remaining computational experiments, we use SSP because of its better quality.

5.4. Sensitivity to problem characteristics

In this section, we provide insights regarding sensitivity to problem characteristics, namely the company fleet size, the flexibility of in-store customers, the frequency of decision making, and the service guarantee.

5.4.1. The impact of fleet size One of the main challenges for companies considering the use of in-store customers to deliver online orders is deciding the size of the company fleet. To be able to maintain a high level of service, it is clear that the company fleet cannot be eliminated, but how much smaller can it be? A larger fleet can absorb unexpected fluctuations in the placement of orders and the arrival of in-store customers. However, a larger fleet also comes at a higher cost. Table 2 compares results obtained for different fleet sizes with our reference configuration (7 vehicles). As expected, a larger fleet size

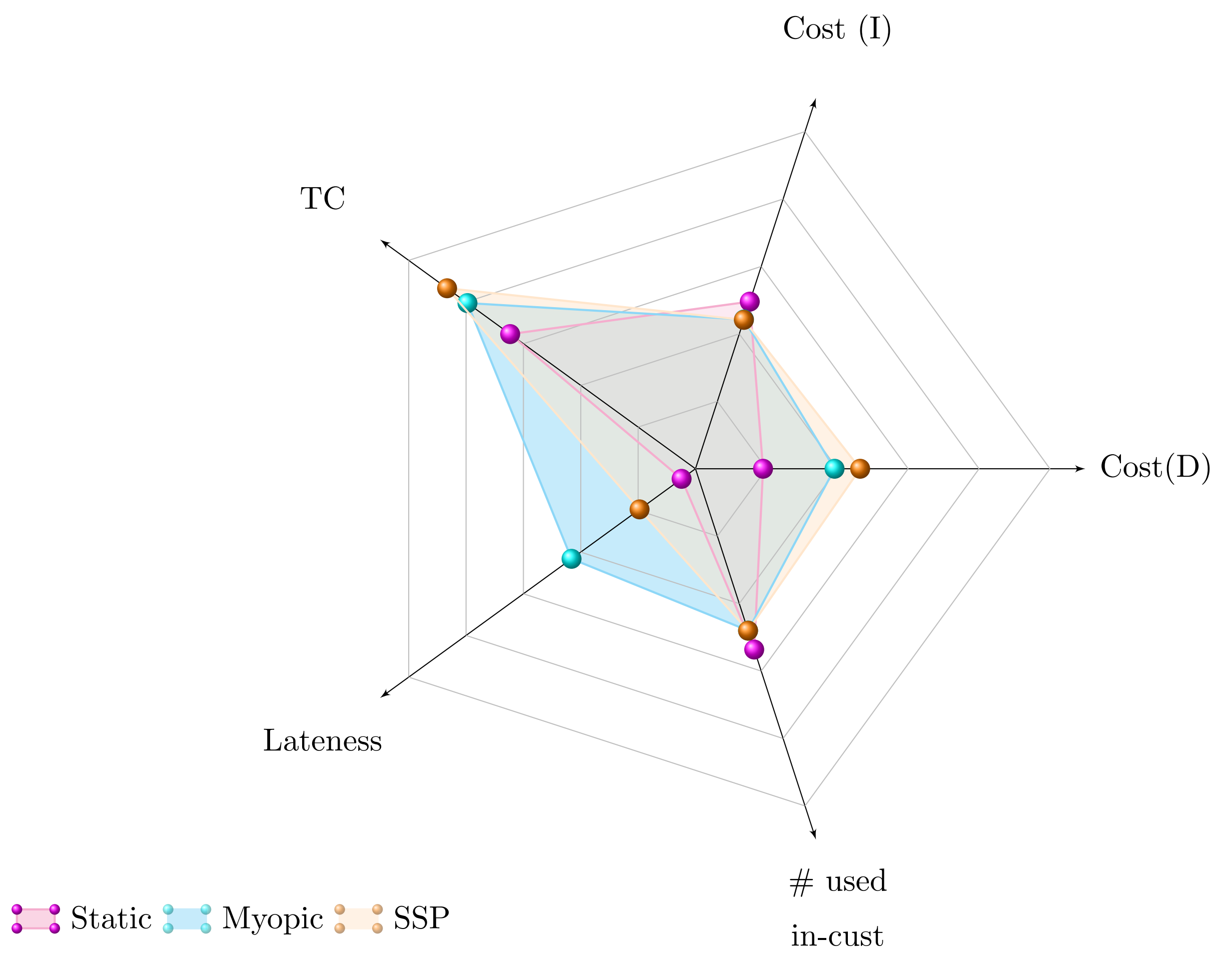


Figure 4 **Comparing Performances of different solution approaches w.r.t. different metrics**

improves the service quality by decreasing average total delivery lateness. The reported gaps represent the percentage change in cost relative to the lowest total cost among the cases with 8, 9, and 10 vehicles. Accordingly, when the average gap reported for an option is small, that variant more often resulted in the lowest total cost. Somewhat surprisingly, a change in the fleet size does not have a significant impact on the operational cost (i.e., total length of the vehicle routes plus the incentives paid to in-store customers). The reason is that, mostly, the same online orders are delivered by company vehicles on the same routes, but the dispatch time of the routes is different (earlier), which reduces the lateness of the orders delivered on the routes.

5.4.2. The impact of decision making frequency Table 3 reports results when the time between consecutive fixed decision epochs (δ) varies from 5 minutes to 30 minutes, represented by EP5, EP10, EP20, and EP30. The gaps reported for EP5, EP20, and EP30 show the change in cost relative to the minimum total cost among the options with EP5, EP10, EP20, and EP30.

The natural expectation is that more frequent decision making is beneficial (given the dynamic nature of order placements and in-store customer arrivals, the tight service guarantee for online orders, and the short availability of in-store customers). Therefore, in our tests we chose EP10 as default.

The frequency of decision making may exhibit dual effects. On the one hand, when the system state is checked more frequently, there is a higher chance to identify matches between available in-store customers and online orders. On the other hand, the decision making is more fact-based rather than estimation-based when the inter-epoch time period is longer. More time between consecutive decisions implies more real data is being collected between consecutive decisions and decisions are influenced more by these data, than by anticipated future events.

Generally, a higher decision making frequency may result in more “unfortunate estimation-based decisions”, but will also give the decision maker more opportunities to recover from “unfortunate decisions” made at earlier decision epochs. Our experiments, however, show that the best results are obtained with EP30. Note that, since vehicle returns are also considered as decision points, very often the system state is checked, and new decisions are made more frequently than once every 30 minutes. Moreover, given that $R = 30$, all in-store customers are considered at least once. Additionally, $Q^c = 2$, even in the case of Rate 1, creates a buffer for the decision maker despite the decreased flexibility in terms of available time to make deliveries. In summary, we observe that fewer decision points may in fact result in better performance due to the fact that accumulating/batching information to improve consolidation is valuable even in a highly dynamic environment.

5.4.3. The impact of service guarantee The service guarantee has a direct impact on the difficulty of the problem. The more time available between the placement of an online order and the time the delivery has to take place, the more time the decision maker has to match orders with in-store customers or to consolidate orders and dispatch them together on vehicle routes, resulting in lower costs. Table 4 provides results for cases where the service guarantee is doubled compared to the baseline setting, i.e., $S = 120$ as opposed to $S = 60$. Column “gap” represents the change in total cost TC compared to the baseline setting (SSP in Table 1). As expected, by increasing the service guarantee the total lateness tends towards zero, while the transportation cost is decreased. Increasing the service guarantee from one hour to two hours reduced the total cost by 21%, due to more time to make better routing decisions by accumulating orders with delivery addresses that are close and dispatching them together on a vehicle route. A looser service guarantee can result in multiple postponements of online orders to later decision epochs to increase the chance of consolidation. Moreover, as a direct result of loosening the service guarantee, the latest possible dispatch time for online orders, θ_j, are pushed later, resulting in an increase in β^h. Due to these two factors, the set of online orders $\mathcal{N}_h^o$ for which routing solutions are generated at a given decision epoch tends to be larger, resulting in more required computational effort at each epoch. This can be observed from the almost 100% increase in the average time spent per epoch when $S = 120$ in Table 4, compared to the baseline setting.

5.4.4. The impact of number of scenarios considered in SSP The quality of SSP may depend on the number of sampled scenarios. Table 4, under column $|\mathcal{S}| = 100$, provides results for the tests ran with twice more scenarios than the baseline setting. A larger set of scenarios translates into a larger number of routing problems solved per decision epoch (**STEP 2**). This resulted in a significant solve time per epoch (160 vs. 74 seconds/epoch). Despite a significantly higher computational effort, the payoff is quite limited in this case. We observe a total cost improvement of 0.1%, while the service quality did not improve. Note that a larger set of scenarios increases the level of variability in the future events. Our SSP is inherently sensitive to variability estimated for the future demand and delivery capacity. A higher level of total lateness in a small subset of the scenarios may result in not releasing online orders from their currents routes, and therefore, not taking advantage of the capacity materialized in upcoming decision epochs.

5.4.5. The algorithmic contribution of releasing online orders (STEP 2) In Table 4, under column “No order postponing”, we report results obtained for a variant of our algorithm in which no order postponing is considered. Specifically, orders are dispatched as soon as delivery capacity becomes available, and vehicles depart the store in the first decision epoch following their availability in the store (if there are any outstanding online order). In such a context, the decision maker does not necessarily try to increase the chance of online order consolidation. To evaluate the algorithmic contribution of releasing orders, we disable this feature and run all the tests. As expected, disabling this feature simplifies the algorithm significantly compared to the baseline setting. This is evident from the fact that the time spent per epoch dropped from 74 seconds to 1.4 seconds. However, on the downside, the solutions deteriorated both in terms of total cost and service quality. While the total cost increased by 17%, the total lateness went from 77 minutes up to 231 minutes, representing an increase of almost 300%. These results show the benefit of "waiting" based on a smartly designed framework when demand and supply (capacity) materialize in a dynamic fashion over time.

5.4.6. In-store customer availability We analyze the effect of in-store customer availability in four different ways. First, we varied in-store customer availability by considering three different arrival rates, Rate 1, Rate 2, and Rate 3, in the tests reported on so far. These tests represent scenarios in which, given the default compensation scheme, different numbers of in-store customers accept to participate in crowdshipping. Note that the arrival rates of in-store customers in these tests implicitly account for the probabilistic ratio of participation in crowdshipping. Second, we varied the in-store customer coverage area, given a fixed compensation scheme. Specifically, in these tests, we consider scenarios in which customers are willing to serve a larger area while the compensation remains the same. Third, we varied the in-store customer capacity. The baseline tests assumed a capacity of two orders per in-store customer. Here, we study the effect of reducing the in-store customer capacity to one order, as shown in Table 5. Fourth, we perform a sensitivity analysis to study the price elasticity of in-store customers. In these tests, we assume that the compensation thresholds for participation of in-store customers – threshold for the per extra-minute compensation – follow a Normal distribution. That is, an in-store customer participates only if the offered compensation exceed his/her threshold and these thresholds are drawn from a Normal distribution (most in-store customers have almost the same threshold, but some have a lower, and some have a higher threshold).

As can be seen in Tables 1- 3, a larger set of participating in-store customers (e.g., Rate 3 vs. Rate 1) results in a higher level of service quality (lower total lateness). In several cases, a higher service quality is achieved even at a lower total cost. In these tests, the compensation paid to the in-store customers is fixed. That is, a larger set of participating in-store customers is the result of non-monetary incentives (e.g., receiving one free delivery of a future order, after delivering an order).

Table 5 reports results obtained when the coverage area of in-store customers is enlarged by increasing the value of parameter γ from 1.25 to 1.5. We observe that as a results of this change, the total lateness decreased by more than 50%, while the total cost also decreased by 2.7%. Similarly, we study the impact of varying the maximum number of orders that an in-store customer can possibly deliver on a single trip. Our baseline constitutes tests in which Q^c is set to two. In Table 5, we report additional results for the case where this capacity is reduced to 1. This change results in a significant deterioration of service quality (250% increase in total lateness), and 26% total cost increase. A decrease in the capacity of in-store customers leads into a potential increase in the per delivery cost of delivering an order using in-store customers (due to the fixed incentive paid c_0^c). This explains why a smaller subset of in-store customers are used when Q^c is set to 1.

Finally, We perform a sensitivity analysis on compensation paid to in-store customers. For this study, we consider three cases w.r.t. the in-store customer thresholds versus the compensation offered by the company. For these tests, we only focus on instances of type Loc 1 and assume that the set of in-store customers in each class of instances (Rate 1, Rate 2, and Rate 3) represents all the in-store customers who arrive to the store, while only those who find the offered compensation by the company appealing would participate in crowdshipping. The three cases considered are:

Case 1: The company offers a per extra-minute of travel time compensation (c^c) of \$0.5, and this compensation is high enough to incentivize all arriving in-store customers in the store to participate in crowdshipping. Specifically, the in-store customers threshold of c^c is \$0.5 or less.

Case 2: The company offers a per extra-minute of travel time compensation (c^c) of \$0.5, while the in-store customer threshold for c^c follows a Normal distribution $N(0.35, 0.25)$. Each arriving in-store customer is assigned a threshold by drawing a random number $\tilde{c^c}$ from distribution $N(0.35, 0.25)$, and that customer is considered for crowdshipping only if $\tilde{c^c} \leq 0.5$.

Case 3: The company offers a per extra-minute of travel time compensation (c^c) of \$0.6, while the in-store customer threshold for c^c follows a Normal distribution $N(0.35, 0.25)$. Each arriving in-store customer is assigned a threshold by drawing a random number $\tilde{c^c}$ from distribution $N(0.35, 0.25)$, and that customer is considered for crowdshipping only if $\tilde{c^c} \leq 0.6$.

Case 1 corresponds to the tests reported on in Section 5.3. For Case 2 and Case 3, each instance is run five times, and in each run based on a different random number generator seed, in-store customers are assigned thresholds $\tilde{c}^c$. Table 6 reports the results obtained for these tests. The comparison of the results obtained for Case 1 and Case 2, reveals the fact that convincing a larger set of in-store customers to participate in crowdshipping can significantly improve the overall service quality, while in most cases it results in a lower total cost too. Comparing the results of Case 3 with Case 1 and Case 2 reveals the two effects of increasing the compensation to in-store customers: On the one hand, a higher compensation convinces a larger subset of in-store customers to participate in crowdshipping, which results in a larger delivery capacity. On the other hand, increasing the compensation paid to in-store customers turns this delivery option into a more costly one, resulting in potentially using them less frequently. As can be seen from the results, in the case of Rate 1 and Rate 2, increasing c^c from \$0.5 to \$0.6 (Case 2 vs. Case 3) led into a higher participation of in-store customers in crowdshipping, resulting in improving the service quality, even though the overall cost increases mainly due to the cost of deliveries by in-store customers. As for Rate 3, we observe that although in Case 3, potentially a larger subset of in-store customers agreed to participate in crowdshipping, the algorithm decided to use a smaller number of them and instead benefit from the capacity of company drivers. These results show the importance of proper pricing of in-store customers' service in order to incentivize the largest subset of arriving customers to participate in crowdshipping program, while keeping the paid compensation low enough for the system to find it beneficial to employ them.

6. Discussion

We have explored a form of crowdshipping in which in-store customers supplement company drivers to fulfill online orders, an innovative concept in last-mile delivery which large retailers may consider in their quest to provide fast, low-cost home delivery service for online orders. The idea is to use in-store customers to deliver an online order on their way home from the store (or, more generally, on the way to their next destination). The concept builds on other examples of collaborative consumption, in which people share their under-used assets with others. The assets shared in this form of crowdshipping are spare time and spare vehicle capacity. We have focused on employing in-store customers to deliver online orders as opposed to outsourcing deliveries to independent drivers. The primary reason to be at the store for such an in-store customer is shopping. The in-store availability time reflects the time that the customer is doing this shopping, after which he or she either goes home or delivers one or more online orders on the way home. This perspective also explains why the compensation to an in-store customer for delivering an online order can be small, e.g., in the form of a store credit. Outsourcing deliveries to independent drivers would be more expensive.

To be able to focus on the core characteristics of this type of environment, we have made a number of simplifying assumptions, most of which can be easily relaxed. These include:

- The capacity of the vehicles is assumed to be a non-binding constraint when making routing decisions for the company drivers.
- The loading time at the fulfillment center is assumed to be zero.
- The service time at a delivery address is assumed to be zero.

Another critical aspect of the setting is the coverage area of an in-store customer willing to make delivery. In practice, this area will be customer dependent and, highly likely, also a function of the compensation offered for performing the delivery. In practice, compensation may have to depend on the location of the delivery and the detour incurred by the in-store customer on his way home. It is even possible to envision a situation, in which the compensation varies depending on the system state, e.g., the set of active orders, the set of available in-store customers, and the state of the company vehicles (number of vehicles available in store or en-route with known return times).

One of the main limitations of our approach is that it assumes knowledge about the (next) destination of in-store customers, based on which their routes are constructed. In practice, some in-store customers may be reluctant to reveal their destinations. If, instead, in-store customers are willing to provide a coverage area, the routing problem of in-store customers becomes an open vehicle routing problem. Another limitation is the fact that travel times are assumed to be deterministic. In the real-world, it is rare to be able to exactly predict the return time of the vehicles. However, the availability of GPS and tracking technologies allows regular plan updates.

Another form of crowdshipping, which has revolutionized passenger transportation and may do the same with freight transportation, is popularized by companies like Uber and Lyft. These companies have developed a (phone) app that matches, in real-time, demand for transportation with individuals willing to provide that transportation. Indeed, in a setting that is only slightly different from the one studied in this paper, company drivers are supplemented with Uber-Freight drivers (rather than in-store customers). It is even possible to envision a setting in which all three transportation options are available, i.e., in-store customers, Uber-Freight drivers, and company drivers. From a company perspective, the main differences between these three options are their cost and their availability/reliability/control, with in-store customers being the cheapest and riskiest (uncertain availability) and company drivers the most expensive and safest (guaranteed availability).

One of the fundamental questions facing large retailers that are considering crowdsourced delivery options is how to ensure service quality. They are the one making service promises to their (online) customers, and they will be the one "blamed" if these service promises are not met. On the other hand, crowdsourced delivery can provide a (relatively) cheap way to expand home and same-day delivery offerings. Our study provides some (initial) insights into the pros and cons of incorporating crowdsourced delivery options.

Table 1 Results for instances in the 12 instance classes using different solution approaches - Fleet size: 7.

		Full information						**Ignoring future**							**Anticipating future**						
		Static						Myopic							SSP						
		cost (D)	cost (I)	TC	lateness	# used in-cust	time(s)	cost (D)	cost (I)	TC	lateness	# used in-cust	time/epoch (s)	gap (%)	cost (D)	cost (I)	TC	lateness	# used in-cust	time/epoch (s)	gap (%)
Rate 1	Loc1	1419.8	1560.5	2980.3	13.2	192.8	2986.8	2691.4	1169.7	3861.1	594.0	145.9	0.5	29.7	2895.6	1273.3	4169.0	107.4	163.9	33.2	40.0
	Loc2	1388.9	1653.7	3042.6	0.1	199.7	4358.6	2619.1	1239.5	3858.6	533.1	156.1	0.4	27.0	3029.2	1303.8	4332.9	235.5	167.0	43.1	42.8
	Loc3	1608.2	1880.1	3488.2	34.3	209.7	4910.9	2867.9	1410.8	4278.7	870.5	158.2	0.4	23.0	3273.7	1517.1	4790.8	407.0	171.6	31.5	37.8
	Loc4	1316.7	1581.7	2898.4	2.3	199.0	3182.1	2384.4	1155.4	3539.8	219.7	148.4	0.7	22.4	2823.0	1210.3	4033.3	104.5	159.0	30.4	39.4
Rate 2	Loc1	640.0	2220.7	2860.8	0.0	276.8	6625.4	1696.3	1854.6	3551.0	19.7	213.6	1.3	24.2	2100.0	1864.4	3964.4	10.7	214.0	58.0	38.6
	Loc2	664.6	2316.1	2980.7	0.0	281.7	6627.8	1851.5	1923.9	3775.4	27.1	216.5	1.3	26.7	2315.0	1897.7	4212.6	18.1	214.0	59.3	41.3
	Loc3	697.2	2567.3	3264.5	0.0	274.0	10580.6	2027.4	2131.9	4159.3	64.3	213.4	1.5	27.6	2485.0	2093.9	4578.9	20.6	207.3	45.9	40.3
	Loc4	573.9	2287.9	2861.9	0.0	288.6	7423.8	1591.3	1898.3	3489.6	23.3	216.4	1.9	22.0	2050.1	1871.1	3921.1	6.2	212.7	43.5	37.1
Rate3	Loc1	230.8	2681.3	2912.1	0.0	320.3	21148.5	845.7	2749.4	3595.2	0.0	290.7	4.8	23.5	1124.1	2647.2	3771.3	0.0	279.6	151.1	29.5
	Loc2	161.9	2802.2	2964.1	0.0	343.7	16408.7	950.4	2743.3	3693.8	0.6	288.3	5.1	24.7	1184.5	2699.8	3884.3	0.0	281.1	117.4	31.1
	Loc3	167.0	3117.8	3284.9	0.0	338.4	16608.7	962.5	3079.5	4042.0	0.8	287.4	5.8	23.2	1298.1	2972.1	4270.2	0.1	276.9	135.3	30.1
	Loc4	166.4	2671.2	2837.6	0.0	337.3	13708.4	651.1	2811.0	3462.1	0.2	304.2	6.6	22.0	896.0	2744.1	3640.1	0.2	294.8	144.5	28.3
average		753.0	2278.4	3031.3	4.2	271.8	9547.5	1761.6	2014.0	3775.5	196.1	219.9	2.5	24.7	2122.8	2007.9	4130.7	75.9	220.1	74.4	36.4

Table 2 Results for instances in the 12 instance classes based on different fleet sizes.

		7 Veh						8 Veh						9 Veh						10 Veh					
		cost (D)	cost (I)	TC	lateness	# used in-cust	gap%	cost (D)	cost (I)	TC	lateness	# used in-cust	gap%	cost (D)	cost (I)	TC	lateness	# used in-cust	gap%	cost (D)	cost (I)	TC	lateness	# used in-cust	gap%
Rate 1	Loc1	2895.6	1273.3	4169.0	126.6	163.9	6.9	2896.8	1128.0	4024.9	44.3	143.0	3.2	2879.2	1103.0	3982.1	9.2	137.6	2.1	2896.5	1059.3	3955.8	1.0	132.2	1.5
	Loc2	3029.2	1303.8	4332.9	235.5	167.0	6.9	3012.4	1173.9	4186.3	56.8	149.2	3.2	3069.1	1133.1	4202.2	19.7	142.2	3.7	3026.1	1070.1	4096.2	2.1	132.9	1.0
	Loc3	3273.7	1517.1	4790.8	407.0	171.6	5.6	3362.1	1371.5	4733.6	132.6	153.4	4.3	3438.2	1234.5	4672.7	57.0	137.5	3.0	3415.7	1179.0	4594.8	19.2	129.4	1.3
	Loc4	2823.0	1210.3	4033.3	104.5	159.0	6.2	2869.5	1098.4	3967.9	50.2	140.2	4.4	2854.9	1051.1	3906.0	6.2	132.5	2.8	2826.2	1014.4	3840.6	1.6	127.5	1.1
Rate 2	Loc1	2100.0	1864.4	3964.4	10.7	214.0	3.7	2112.3	1786.6	3898.9	0.9	207.0	2.1	2133.4	1765.3	3898.8	0.0	205.9	2.0	2115.2	1772.7	3887.9	0.0	204.5	1.8
	Loc2	2315.0	1897.7	4212.6	18.1	214.0	4.2	2295.0	1835.6	4130.6	2.1	208.2	2.2	2277.2	1817.6	4094.7	0.7	206.5	1.3	2328.4	1781.6	4109.9	0.6	203.2	1.7
	Loc3	2485.0	2093.9	4578.9	20.6	207.3	4.7	2483.7	1984.6	4468.4	2.3	196.3	2.1	2468.8	1976.4	4445.2	1.7	197.2	1.6	2484.7	1964.2	4448.9	0.6	194.5	1.7
	Loc4	2050.1	1871.1	3921.1	6.2	212.7	3.7	2053.5	1833.9	3887.4	1.9	209.8	2.9	2019.6	1807.3	3826.9	0.2	206.2	1.3	2053.6	1787.2	3840.8	0.0	204.1	1.6
Rate 3	Loc1	1124.1	2647.2	3771.3	0.0	279.6	1.1	1099.0	2664.3	3763.3	0.0	280.9	0.9	1102.0	2665.5	3767.5	0.0	281.9	1.0	1135.3	2639.5	3774.9	0.0	278.8	1.2
	Loc2	1184.5	2699.8	3884.3	0.0	281.1	1.0	1190.8	2698.5	3889.3	0.3	281.2	1.2	1206.1	2683.0	3889.1	0.2	281.0	1.2	1208.9	2674.7	3883.6	0.2	280.1	1.0
	Loc3	1298.1	2972.1	4270.2	0.1	276.9	1.9	1282.4	2971.8	4254.2	0.3	277.9	1.5	1279.9	2959.8	4239.7	0.1	277.8	1.1	1288.9	2953.7	4242.5	0.0	277.0	1.2
	Loc4	896.0	2744.1	3640.1	0.2	294.8	1.0	899.6	2745.5	3645.1	0.0	294.2	1.2	914.1	2717.8	3632.0	0.2	291.8	0.8	894.8	2749.4	3644.2	0.2	295.5	1.2
average		2122.8	2007.9	4130.7	77.5	220.1	3.9	2129.8	1941.1	4070.8	24.3	211.8	2.4	2136.9	1909.5	4046.4	7.9	208.2	1.8	2139.5	1887.2	4026.7	2.1	204.9	1.4

Table 3 The impact of decision making frequency (fixed epoch length).

		EP5						EP10						EP20						EP30					
		cost (D)	cost (I)	TC	lateness	# used in-cust	gap%	cost (D)	cost (I)	TC	lateness	# used in-cust	gap%	cost (D)	cost (I)	TC	lateness	# used in-cust	gap%	cost (D)	cost (I)	TC	lateness	# used in-cust	gap%
Rate 1	Loc 1	2894.1	1327.3	4221.4	137.1	170.2	9.6	2895.6	1273.3	4169.0	126.6	163.9	8.1	2852.4	1147.0	3999.4	93.5	147.9	3.8	2815.6	1067.7	3883.3	110.7	138.3	0.7
	Loc 2	3006.8	1359.3	4366.1	206.5	173.1	8.7	3029.2	1303.8	4332.9	235.5	167.0	7.9	2964.5	1190.4	4154.9	175.8	153.8	3.4	2934.3	1129.1	4063.5	200.4	147.9	1.1
	Loc 3	3249.8	1601.8	4851.6	447.0	181.7	8.8	3273.7	1517.1	4790.8	407.0	171.6	5.0	3233.7	1396.8	4630.5	479.8	165.4	3.8	0.0	0.0	0.0	0.0	0.0	0.5
	Loc 4	2864.7	1295.0	4159.7	109.5	169.9	10.9	2823.0	1210.3	4033.3	104.5	159.0	7.5	2838.8	1104.5	3943.3	83.5	149.2	5.0	2719.5	1043.5	3763.0	74.9	140.3	0.2
Rate 2	Loc 1	2047.3	1960.1	4007.3	13.2	224.4	10.2	2100.0	1864.4	3964.4	10.7	214.0	9.0	2120.8	1676.0	3796.8	3.5	195.9	4.4	2036.5	1602.0	3638.5	6.1	188.7	0.1
	Loc 2	2293.1	2011.4	4304.6	33.0	224.5	10.2	2315.0	1897.7	4212.6	18.1	214.0	8.0	2294.3	1735.5	4029.8	16.3	198.4	3.3	2285.9	1667.9	3953.8	13.3	189.8	1.2
	Loc 3	2426.0	2251.4	4677.4	32.2	219.5	10.5	2485.0	2093.9	4578.9	20.6	207.3	8.1	2385.0	2007.8	4392.8	31.2	199.5	3.7	2406.2	1843.8	4250.0	36.9	187.5	0.3
	Loc 4	2023.8	1981.7	4005.5	11.5	223.5	11.6	2050.1	1871.1	3921.1	6.2	212.7	9.3	2048.6	1724.9	3773.5	5.4	199.1	5.1	1977.3	1628.8	3606.0	6.3	188.5	0.2
Rate 3	Loc 1	1082.7	2833.3	3916.0	0.0	291.1	12.9	1124.1	2647.2	3771.3	0.0	279.6	8.8	1224.8	2397.4	3622.2	0.0	263.2	4.4	1229.2	2249.1	3478.3	1.8	249.6	0.2
	Loc 2	1175.7	2817.7	3993.4	1.2	286.8	12.9	1184.5	2699.8	3884.3	0.0	281.1	9.8	1274.2	2453.8	3728.1	2.5	264.8	5.5	1265.0	2321.2	3586.1	1.2	257.4	1.4
	Loc 3	1197.4	3140.0	4337.4	0.3	285.3	10.8	1298.1	2972.1	4270.2	0.1	276.9	9.1	1364.2	2717.7	4081.9	1.4	261.8	4.2	1363.9	2557.6	3921.5	6.1	255.6	0.1
	Loc 4	907.8	2880.1	3787.9	0.0	300.1	14.1	896.0	2744.1	3640.1	0.2	294.8	9.7	981.1	2491.2	3472.3	0.5	277.1	4.7	1005.9	2327.1	3333.0	0.1	265.0	0.4
average		2097.4	2121.6	4219.0	82.6	229.1	10.9	2122.8	2007.9	4130.7	77.5	220.1	8.3	2131.9	1836.9	3968.8	74.4	206.3	4.3	1836.6	1619.8	3456.4	38.2	184.0	0.5

Table 4 Sensitivity to algorithmic components (Service guarantee (S), Scenario set size ($|\mathcal{S}|$), and Order postponing)

		$S = 120$							$|\mathcal{S}| = 100$							No order postponing (no waiting)						
		cost (D)	cost (I)	TC	lateness	# used in-cust	time/epoch (s)	gap (%)	cost (D)	cost (I)	TC	lateness	# used in-cust	time/epoch (s)	gap (%)	cost (D)	cost (I)	TC	lateness	# used in-cust	time/epoch (s)	gap (%)
Rate 1	Loc 1	2034.7	911.5	2946.2	0.0	115.7	127.2	-29.4	2928.8	1245.7	4174.5	123.9	161.0	69.7	0.2	3462.4	1417.0	4879.4	1564.6	180.1	0.4	17.0
	Loc 2	2142.8	927.9	3070.7	2.6	117.3	107.1	-29.1	3015.1	1297.3	4312.4	258.7	165.7	60.3	-0.5	3466.6	1433.7	4900.2	1815.1	182.5	0.3	13.1
	Loc 3	2396.7	959.2	3355.8	1.2	112.3	105.5	-30.0	3222.3	1503.0	4725.3	489.0	169.8	56.2	-1.3	3629.5	1627.0	5256.4	2671.4	182.4	0.3	9.7
	Loc 4	2003.3	895.6	2898.9	0.1	114.7	89.7	-28.1	2850.4	1196.0	4046.5	103.8	158.0	39.0	0.5	3433.1	1375.9	4809.0	1309.3	181.7	0.4	19.2
Rate 2	Loc 1	1597.8	1545.7	3143.6	0.0	182.8	233.1	-20.7	2110.0	1861.9	3971.9	8.4	215.1	126.5	0.1	2784.7	2083.1	4867.8	207.1	231.5	1.1	22.8
	Loc 2	1715.6	1596.8	3312.4	0.0	181.7	179.1	-21.4	2315.8	1889.8	4205.6	24.9	213.1	155.6	-0.1	2985.2	2122.3	5107.5	397.6	232.5	0.8	21.2
	Loc 3	1903.5	1659.8	3563.2	0.0	171.6	143.3	-22.2	2463.2	2120.7	4583.9	26.9	208.5	194.4	0.1	3103.2	2403.8	5507.0	507.8	231.7	0.7	20.3
	Loc 4	1496.5	1576.3	3072.8	0.0	183.8	232.1	-21.6	2023.8	1871.3	3895.1	5.2	213.7	79.2	-0.6	2746.2	2090.1	4836.4	231.3	234.2	1.0	23.3
Rate 3	Loc 1	816.3	2448.2	3264.5	0.0	265.2	344.3	-13.4	1112.5	2655.5	3768.0	0.0	280.7	354.9	-0.1	1676.9	2712.3	4389.2	7.0	276.7	3.8	16.4
	Loc 2	849.4	2554.2	3403.7	0.0	273.2	268.8	-12.4	1209.9	2685.9	3895.8	0.0	280.3	314.2	0.3	1702.9	2769.4	4472.3	13.0	280.3	3.0	15.1
	Loc 3	973.8	2772.0	3745.8	0.0	264.2	382.0	-12.3	1311.9	2959.7	4271.6	0.1	275.7	244.2	0.0	1827.2	3056.5	4883.7	17.8	275.7	2.6	14.4
	Loc 4	646.9	2605.4	3252.2	0.0	283.8	378.7	-10.6	891.8	2758.1	3649.9	0.2	296.2	229.5	0.3	1440.9	2724.1	4165.0	5.4	285.1	2.6	14.4
average		1548.1	1704.4	3252.5	0.3	188.8	215.9	-20.9	2121.3	2003.7	4125.0	86.8	219.8	160.3	-0.1	2688.2	2151.3	4839.5	728.9	231.2	1.4	17.2

Table 5 Sensitivity to in-store customer characteristics (In-store coverage area (γ), and in-store capacity (Q^c))

		SSP (Baseline)					$\gamma = 1.5$						$Q^c = 1$					
		cost (D)	cost (I)	TC	lateness	# used in-cust	cost (D)	cost (I)	TC	lateness	# used in-cust	gap%	cost (D)	cost (I)	TC	lateness	# used in-cust	gap%
Rate 1	Loc 1	2895.6	1273.3	4169.0	126.6	163.9	2609.0	1436.0	4045.1	52.8	174.9	-3.0	3273.4	1430.6	4704.0	417.6	120.9	13.1
	Loc 2	3029.2	1303.8	4332.9	235.5	167.0	2726.9	1541.2	4268.1	103.3	193.4	-1.5	3383.9	1469.5	4853.4	847.7	124.2	12.3
	Loc 3	3273.7	1517.1	4790.8	407.0	171.6	2889.0	1782.6	4671.6	217.4	191.4	-2.5	3531.2	1730.0	5261.2	1212.3	132.8	10.0
	Loc 4	2823.0	1210.3	4033.3	104.5	159.0	2602.0	1412.3	4014.3	48.7	179.6	-0.5	3207.7	1388.1	4595.9	521.4	119.6	14.1
Rate 2	Loc 1	2100.0	1864.4	3964.4	10.7	214.0	1616.6	2212.4	3829.0	1.4	244.4	-3.4	2713.6	2344.6	5058.2	37.0	167.8	27.8
	Loc 2	2315.0	1897.7	4212.6	18.1	214.0	1781.3	2255.3	4036.5	6.9	249.1	-4.2	2838.7	2461.5	5300.2	108.8	173.9	25.9
	Loc 3	2485.0	2093.9	4578.9	20.6	207.3	1611.1	2690.9	4302.0	2.5	254.5	-6.0	2996.1	2794.1	5790.2	105.5	175.8	26.5
	Loc 4	2050.1	1871.1	3921.1	6.2	212.7	1548.7	2230.2	3779.0	1.6	247.5	-3.6	2656.9	2347.8	5004.7	58.1	169.3	27.7
Rate 3	Loc 1	1124.1	2647.2	3771.3	0.0	279.6	737.6	2948.8	3686.4	0.0	306.5	-2.3	1692.1	3601.3	5293.4	1.1	227.7	40.4
	Loc 2	1184.5	2699.8	3884.3	0.0	281.1	958.7	2878.1	3836.8	0.0	301.2	-1.2	1736.8	3660.6	5397.4	1.1	236.2	39.0
	Loc 3	1298.1	2972.1	4270.2	0.0	276.9	602.8	3535.6	4138.4	0.2	324.9	-3.1	1854.6	4089.1	5943.7	4.5	231.6	39.2
	Loc 4	896.0	2744.1	3640.1	0.2	294.8	516.0	3060.2	3576.2	0.0	323.4	-1.7	1469.0	3737.9	5206.9	0.6	242.7	43.2
average		2122.8	2007.9	4130.7	77.5	220.1	1683.3	2332.0	4015.3	36.2	249.2	-2.7	2612.8	2587.9	5200.7	276.3	176.8	26.6

Table 6 **The In-store customer compensation sensitivity.**

	Case	Threshold c^c	Paid	cost (D)	cost (I)	TC	lateness	# used in-cust
Rate 1	Case 1	$ 0.5	$ 0.5	2895.6	1273.3	4169.0	126.6	163.9
	Case 2	N(0.35, 0.25)	$ 0.5	3207.0	919.6	4126.6	714.9	119.6
	Case 3	N(0.35, 0.25)	$ 0.6	3120.7	1204.3	4325.0	398.2	130.2
Rate 2	Case 1	$ 0.5	$ 0.5	2100.0	1864.4	3964.4	10.7	214.0
	Case 2	N(0.35, 0.25)	$ 0.5	2642.1	1373.6	4015.7	349.0	159.3
	Case 3	N(0.35, 0.25)	$ 0.6	2514.3	1846.5	4360.8	205.0	178.6
Rate 3	Case 1	$ 0.5	$ 0.5	1124.1	2647.2	3771.3	0.0	279.6
	Case 2	N(0.35, 0.25)	$ 0.5	1946.3	1982.3	3928.6	374.6	210.5
	Case 3	N(0.35, 0.25)	$ 0.6	2159.3	2173.3	4332.6	193.6	200.9
average	Case 1	$ 0.5	$ 0.5	2039.9	1928.3	3968.2	45.8	219.1
	Case 2	N(0.35, 0.25)	$ 0.5	2598.5	1425.2	4023.6	479.5	163.1
	Case 3	N(0.35, 0.25)	$ 0.6	2598.1	1741.4	4339.5	265.6	169.9

References

Archetti C, Savelsbergh M, Speranza MG (2016) The vehicle routing problem with occasional drivers. *European Journal of Operational Research* 254(2):472 – 480, ISSN 0377-2217.

Arslan AM, Agatz N, Kroon L, Zuidwijk R (2018) Crowdsourced delivery-a dynamic pickup and delivery problem with ad-hoc drivers. *Transportation Science* 53(1):222–235.

Azi N, Gendreau M, Potvin JY (2012) A dynamic vehicle routing problem with multiple delivery routes. *Annals of Operations Research* 199(1):103–112.

Banjo S (2013) Home depot to offer same day delivery. *Wall Street Journal* .

Barr A, Wohl J (2013) Exclusive:Walmart may get customers to deliver packages to online buyers. *Reuters* .

Bensinger G (2015) Amazons next delivery drone: You. *Wall Street Journal* .

Bräysy O, Gendreau M (2005) Vehicle routing problem with time windows, part II: Metaheuristics. *Transportation Science* 39(1):119–139.

Cattaruzza D, Absi N, Feillet D (2016) The multi-trip vehicle routing problem with time windows and release dates. *Transportation Science* 50(2):676–693.

Ecommerce-Foundation (2015) Global B2C e-commerce report 2015. URL `http://www.ecommercefoundation.org`.

Eisenmann T, Parker G, Van Alstyne MW (2006) Strategies for two-sided markets. *Harvard business review* 84(10):92.

Evans DS (2003) Some empirical aspects of multi-sided platform industries. *Review of Network Economics* 2(3).

Glover F, Laguna M (1997) *Tabu Search* (Norwell, MA, USA: Kluwer Academic Publishers), ISBN 079239965X.

Hagiu A, Wright J (2015) Multi-sided platforms. *International Journal of Industrial Organization* 43:162–174.

Klapp MA, Erera AL, Toriello A (2016) The one-dimensional dynamic dispatch waves problem. *Transportation Science* 52(2):402–415.

Kohler T (2015) Crowdsourcing-based business models: how to create and capture value. *California Management Review* 57(4):63–84.

Nagata Y, Bräysy O (2009) A powerful route minimization heuristic for the vehicle routing problem with time windows. *Operations Research Letters* 37(5):333–338, ISSN 0167-6377.

Rochet JC, Tirole J (2006) Two-sided markets: a progress report. *The RAND journal of economics* 37(3):645–667.

Sampaio A, Savelsbergh M, Van Woensel T, Veelenturf L (2017) Crowd-based city logistics. *Optimization Online* URL `http://www.optimization-online.org/DB_HTML/2017/11/6346.html`.

Savelsbergh M, Van Woensel T (2016) 50th anniversary invited articlecity logistics: Challenges and opportunities. *Transportation Science* 50(2):579–590.

Vidal T, Crainic TG, Gendreau M, Prins C (2013) A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time-windows. *Computers & Operations Research* 40(1):475 – 489.

Voccia SA, Campbell AM, Thomas BW (2017) The same-day delivery problem for online purchases. *Transportation Science* 53(1):167–184.

Weyl EG (2010) A price theory of multi-sided platforms. *American Economic Review* 100(4):1642–72.

Online Supplement

EC.1. Tabu Search

We use a Tabu Search (TS) algorithm (see Glover and Laguna 1997, Bräysy and Gendreau 2005) to optimize the routes. TS is an iterative local search metaheuristic that explores the solution space by moving from the current solution ϕ to the best solution in its neighborhood. Anti-cycling rules must be implemented as the current solution may deteriorate during the search.

Move Evaluation Each move is evaluated based on a utility function defined as $C(\phi) = T(\phi) + \eta LT(\phi)$, where $T(\phi)$ and $LT(\phi)$ represent the total travel time (transportation cost) and total lateness, respectively. Moreover, η is a penalty term which is updated adaptively based on the performance of neighborhood search. The parameter η is initially set to 1. After each block of $Iter^{adj}$ iterations, we multiply η by 2 if the number of solutions with non-zero lateness in the last $Iter^{his}$ iterations is greater than δ_{max}, and we divide it by 2 if the number of such solutions is less than δ_{min}.

With each routing solution ϕ is associated an attribute set $g(\phi) = \{(j,k)$: order j is served by vehicle $k\}$. The neighborhood $N(\phi)$ of solution ϕ is defined by a move which incorporates replacing an attribute (j,k) from $g(\phi)$ with (j,k'), where $k \neq k'$. We also use these attributes to implement our tabu list. Our tabu list is updated as follows: When customer j is removed from route k, a tabu status is assigned to the attribute (j,k). To control the length of the tabu list, the number of iterations for which a move is recognized as tabu, called tabu tenure, is a random variable between $[minTabu,\ maxTabu]$ where $minTabu$ and $maxTabu$ are fixed by the user. As long as attribute (j,k) is considered as tabu, the reinsertion of customer j in route k is forbidden. Through an aspiration criterion, the tabu status of an attribute can be revoked. That is, a tabu move may still be performed if that would allow obtaining a smaller cost than that of the best solution identified having that attribute.

For a partial path κ, comprising a sequence of order deliveries and intermediate store visits, let $T(\kappa)$ be the duration (including waiting times at intermediate stores) of a partial path κ, and $E(\kappa)$ and $L(\kappa)$ be the earliest and latest visits to the first vertex of κ minimizing total waiting time at the intermediate stores and delivery lateness. Each move can be viewed as splitting routes into some partial paths which are concatenated into new routes. Concatenating (symbolized by $\oplus$) two partial paths $\kappa = (\kappa_u, \ldots, \kappa_v)$ and $\kappa' = (\kappa'_{u'}, \ldots, \kappa'_{v'})$ can be efficiently evaluated using the following formulations (Vidal et al. 2013).

$$T(\kappa \oplus \kappa') = T(\kappa) + T(\kappa') + t_{\kappa_v \kappa'_{u'}} + \Delta_{WT}; \tag{EC.1}$$

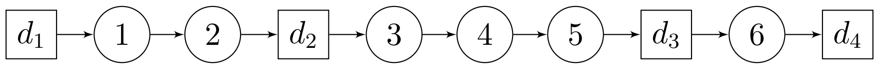
$$E(\kappa \oplus \kappa') = \max\{E(\kappa') - \Delta, E(\kappa)\} - \Delta_{WT}; \tag{EC.2}$$

$$L(\kappa \oplus \kappa') = \min\{L(\kappa') - \Delta, L(\kappa)\} + \Delta_{TW}; \tag{EC.3}$$

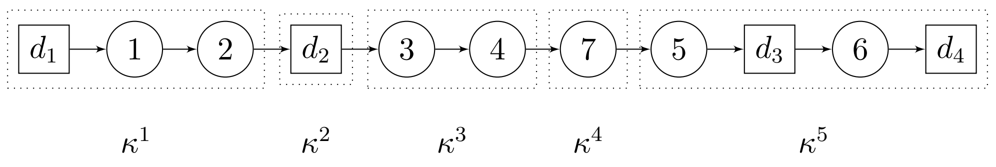
where $\Delta = T(\kappa) + t_{\kappa_v \kappa'_{u'}}$, $\Delta_{WT} = \max\{E(\kappa') - \Delta - L(\kappa), 0\}$, and $\Delta_{TW} = \max\{E(\kappa) + \Delta - L(\kappa'), 0\}$. The definition of Δ in our case is slightly different from the one used by Vidal et al. (2013) since they consider hard time windows and do not report solutions with lateness. Any dispatch within the time interval $[E(\kappa), L(\kappa)]$ will have minimum waiting time at the intermediate stores (shortest total route duration $T(\kappa)$) and minimum delivery lateness for the orders delivered on κ. Obviously, if the total lateness of a partial path κ is non zero, $E(\kappa)$ and $L(\kappa)$ coincide. Moreover, the time interval $[E(\kappa), L(\kappa)]$ when partial path κ contains one single node, corresponds to the node's time window. Notice that $T(\kappa)$ may be different than $T(\kappa)$.

For a given route Σ, evaluating a move with respect to potential change in $LT(\Sigma)$ is done in $O(n^{\Sigma})$, with n^{Σ} being the number of orders delivered on route Σ and by taking $E(\Sigma)$ as the initial dispatch time from the store. On the other hand, evaluating $T(\Sigma)$ following a potential move is calculated in a time $O(1)$ by considering the detour cost (saving) as a result of the insertion (removal) of an order in (from) Σ. Calculation of $T(\Sigma)$ requires updating $E(\Sigma)$ for each potential move. Each insertion can be represented by the concatenation of at most 5 partial paths and each removal by the concatenation of at most 4 partial paths. Figure EC.1a provides an example of a route with three trips (d_1 and d_4 being the initial and final visits to the store, while d_2 and d_3 are the intermediate visits to the store). In order to update the information of the route following the insertion of node 7 between nodes 4 and 5, the five partial paths κ^1-κ^5 are concatenated. These 5 partial paths can be described as below (see also Figure EC.1b):

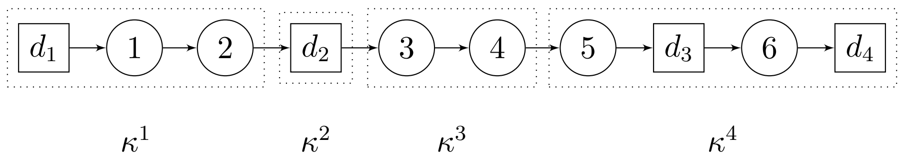
κ^1*:* The partial path starting form d_1 to the predecessor of the intermediate store corresponding to the trip in which the order is inserted, d_2.



(a) Initial route



(b) Route after inserting node 7



(c) Route after removing node 7

Figure EC.1 Concatenations to update route information after an insertion and a removal

κ^2*:* The updated intermediate store to which corresponds the trip in which the order is inserted, d_2. The update consists of updating the beginning of its time window following Eq. (EC.4) ($[a][b]$ stands for the partial path from node a to node b):

$$E[d_2][d_2] := \max\{E[d_2][d_2], \tau_7^o\} = \max\{\tau_3^o, \tau_4^o, \tau_7^o, \tau_5^o\}. \tag{EC.4}$$

κ^3*:* The partial path starting from the first delivered order after the intermediate store of the trip in which the node is inserted to the predecessor of the newly inserted order.

κ^4*:* The partial path corresponding to the order being inserted.

κ^5*:* The partial path from the successor of the inserted order to the last store visit, d_4.

Note that an order removal move evaluation differs from an order insertion in terms of the set of partial paths to concatenate and also in the way the time window of the associated intermediate store is updated. Updating route information following an order removal consists of concatenating four partial paths as shown in Figure EC.1c. The time window of intermediate store tied to the trip from which the order is removed can be updated as follows, where succ(.) and pred(.) denote the successor and the predecessor of a node in the current route, respectively:

$$E[d_2][d_2] := \max\{E[d_2][pred(7)], E[succ(7)][pred(d_3)]\} = \max\{\tau_3^o, \tau_4^o, \tau_5^o\}. \tag{EC.5}$$

EC.2. Releasing orders from routes in myopic and SSP approaches

Algorithms 2 and 3 provide the details of **STEP 2** of Algorithm 1 in the case of myopic and SSP approaches, respectively. In a given decision epoch h, once all outstanding online orders are tentatively assigned to the available delivery capacity (both company drivers and in-store customers), a subset of online orders are released from their current routes and are postponed to a later decision epoch. In both methods, our decision making in broken down into three blocks. In **BLOCK 1**, we identify any online order served on a trip σ of a company vehicle with a latest possible departure time $L(\sigma)$ on or later than the next decision epoch, $h+1$. If any such trips are identified, all orders on these trips are released and postponed to the next epoch. In **BLOCK 2**, we examine all in-store customer routes, to identify those that are assigned a number of orders strictly less than Q^c. If such an in-store customer exists and in case the in-store customer and all orders currently assigned to it can wait until the next epoch, all those orders are released from the customer's route and postponed to $h+1$. Algorithms 2 and 3 mainly differ in terms of their **BLOCK 3**. In myopic approach (Algorithms 2), unreleased online orders are categorized in two groups of non-urgent and semi-urgent. First, all non-urgent orders meeting the removal condition (ll. 30-37) are released, then we consider semi-urgent online orders for removal (ll. 38-46). If in the entire process of **BLOCK 1**-**BLOCK 3**, any order is released, we reoptimize the routing solution and restart from **BLOCK 1**. The procedure stops when no more order can be released. As for SSP approach, **BLOCK 3** involves considering online orders in a descending order of their $\theta_{\hat{j}}$ (less urgent orders are considered first for potential removal). Each online order is tentatively inserted in all routing solutions of the scenarios, ϕ_s and $\mathcal{L}_{\hat{j}}^{h^+}$ and $cost_{\hat{j}}^{h^+}$ are calculated (ll. 36-39, Algorithm 3). The future cost and lateness are compared with the current ones and if beneficial, the order is released from its current route (ll. 40-44). Otherwise, it is kept in its current route and instead it is removed from scenario routes (ll. 46-48). After considering all online orders in $\mathcal{N}_h^o$ once for a potential release, if at least one order is released, we reoptimize current routing solution ϕ and the scenario routing solutions ϕ_s and restart the procedure. The algorithm stops when no more online order is released.

Algorithm 2: Releasing orders from routes in myopic approach

```
Algorithm Release($\phi, \alpha_1, \alpha_2$)
    initialize bool $order\_removed := 0$
    do
        $order\_removed := 0$
        while ($|\phi| > 0$) do // $|\phi|$: the number of orders served on vehicle routes
            BLOCK 1:
            foreach $\sigma_i \in \phi$ do // $\sigma_i \in \phi$: all trips $\sigma_i$ in routing solution $\phi$
                if $L(\sigma_i) \geq t^{h+1}$ then
                    foreach $\hat{j} \in \sigma_i$ do
                        $\phi \leftarrow \phi \setminus \{\hat{j}\}$
                        ${}^{P}\mathcal{N}^{o}_{h} := {}^{P}\mathcal{N}^{o}_{h} \cup \{\hat{j}\}$
                        $order\_removed := 1$
                    end
                end
            end
            BLOCK 2:
            foreach $\Sigma_i \in \phi$ do // in-store customer routes
                if $Q_{\Sigma_i} < Q^c$ then // in-store customer capacity not attained
                    if $\tau^c_i + R \geq t^{h+1}$ & $\theta_{\hat{j}} \geq t^{h+1}$ then // customer and order can wait to next epoch
                        foreach $\Sigma_i \in \phi$ do
                            $\phi \leftarrow \phi \setminus \{\hat{j}\}$
                            ${}^{P}\mathcal{N}^{o}_{h} := {}^{P}\mathcal{N}^{o}_{h} \cup \{\hat{j}\}$
                            $order\_removed := 1$
                        end
                    end
                end
            end
            BLOCK 3:
            foreach $\hat{j} \in \phi$ do
                if $\theta_{\hat{j}} \geq t^{h+2}$ then // non-urgent orders
                    if $\Gamma_{\hat{j}} \geq \alpha_2$ then
                        $\phi \leftarrow \phi \setminus \{\hat{j}\}$
                        ${}^{P}\mathcal{N}^{o}_{h} := {}^{P}\mathcal{N}^{o}_{h} \cup \{\hat{j}\}$
                        $order\_removed := 1$
                    end
                end
            end
            foreach $\hat{j} \in \phi$ do
                if $t^{h+1} \leq \theta_{\hat{j}} < t^{h+2}$ then // semi-urgent orders
                    if $\Gamma_{\hat{j}} \geq \alpha_1$ then
                        $\phi \leftarrow \phi \setminus \{\hat{j}\}$
                        ${}^{P}\mathcal{N}^{o}_{h} := {}^{P}\mathcal{N}^{o}_{h} \cup \{\hat{j}\}$
                        $order\_removed := 1$
                    end
                end
            end
        end
        Reoptimize($\phi$) // reoptimize $\phi$ using tabu search
    while ( $order\_removed\ != 0$)
    return $\phi$
```

Algorithm 3: Releasing orders from routes in SSP

```
Algorithm Release($\phi, \mathcal{S}$)
    initialize bool $update := 0$
    initialize bool $order\_removed := 0$
    $\hat{\theta} = \{\max \theta_j | j \in \mathcal{N}_h^o\}$
    Compute $\beta^h$ given $\hat{\theta}$
    foreach $s \in \mathcal{S}$ do
        $\phi_s \leftarrow$ Solve routing problem of scenario $s$ given $\beta^h$
    end
    do
        $order\_removed := 0$
        while $(|\phi| > 0)$ do // $|\phi|$: the number of orders served on vehicle routes
            BLOCK 1:
            foreach $\sigma_i \in \phi$ do // $\sigma_i \in \phi$: all trips $\sigma_i$ in routing solution $\phi$
                if $L(\sigma_i) \geq t^{h+1}$ then
                    foreach $\hat{j} \in \sigma_i$ do
                        $\phi \leftarrow \phi \setminus \{\hat{j}\}$
                        ${}^P\mathcal{N}_h^o := {}^P\mathcal{N}_h^o \cup \{\hat{j}\}$
                        $order\_removed := 1$
                    end
                end
            end
            BLOCK 2:
            foreach $\Sigma_i \in \phi$ do // in-store customer routes
                if $Q_{\Sigma_i} < Q^c$ then // in-store customer capacity not attained
                    if $\tau_i^c + R \geq t^{h+1}$ & $\theta_{\hat{j}} \geq t^{h+1}$ then // customer and order can wait to next epoch
                        foreach $\Sigma_i \in \phi$ do
                            $\phi \leftarrow \phi \setminus \{\hat{j}\}$
                            ${}^P\mathcal{N}_h^o := {}^P\mathcal{N}_h^o \cup \{\hat{j}\}$
                            $order\_removed := 1$
                        end
                    end
                end
            end
            BLOCK 3:
            foreach $\hat{j} \in \phi$ do // orders considered in a descending order of $\theta_{\hat{j}}$
                foreach $s \in \mathcal{S}$ do
                    $\phi_s \leftarrow \phi_s \cup \{\hat{j}\}$ // insert $\hat{j}$ in best position in $\phi_s$
                end
                Compute $\mathcal{L}_{\hat{j}}^{h+}$ and $cost_{\hat{j}}^{h+}$
                if $cost_{\hat{j}}^h(\phi) + \pi\mathcal{L}_{\hat{j}}^h \geq cost_{\hat{j}}^{h+} + \pi\mathcal{L}_{\hat{j}}^{h+}$ then
                    $\phi \leftarrow \phi \setminus \{\hat{j}\}$
                    ${}^P\mathcal{N}_h^o := {}^P\mathcal{N}_h^o \cup \{\hat{j}\}$
                    $update := 1$
                    $order\_removed := 1$
                else
                    foreach $s \in \mathcal{S}$ do
                        $\phi_s \leftarrow \phi_s \setminus \{\hat{j}\}$
                    end
                end
            end

            if $update$ != 0 then
                Reoptimize($\phi$) // reoptimize $\phi$ using tabu search
                foreach $s \in \mathcal{S}$ do
                    Reoptimize($\phi_s$) // reoptimize $\phi_s$ using tabu search
                end
            end
            if $order\_removed := 0$ then
                break
            end
            $update = 0$
        end
    while ( $order\_removed$ != 0)
```