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

Iman Dayarian, Martin Savelsbergh
H. Milton Stewart School of Industrial & Systems Engineering, Georgia Institute of Technology, Atlanta, USA,
{iman.dayarian, 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 can take on the task of delivering online orders on their way home. Because online orders as well as in-store customers willing to make deliveries arrive throughout the day, it is a highly dynamic and stochastic environment. 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. The results of our computational study provide insights into the benefits for same-day delivery of this form of crowdshipping, and demonstrate the value of incorporating and exploiting probabilistic information about the future.

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

1. Introduction

Business-to-consumer (B2C) e-commerce sales worldwide reached $1.9 trillion in 2014, representing a doubling in sales compared to 2011 (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 world's population is expected to live in cities (Savelsbergh and 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. Recently, 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 would be asking in-store customers to make available their time and their vehicle to deliver packages to other shoppers. This 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. 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 cheaper and more flexible delivery services. 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 corporate 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., hourly paid independent drivers sign up in advance for prefixed time-slots. 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 employed to deliver orders of online shoppers towards the end of the paper.

Archetti et al. (2016) conducted a preliminary investigation of a static version of this form of crowdshipping, 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. We will assume that delivering online orders using in-store customers is always cheaper than using corporate drivers. Accordingly, a primary objective is to cover as many online orders as possible using in-store customers (as opposed to delivering them using corporate vehicles). Corporate vehicles are used as a “backup” to deliver orders of online shoppers when it is determined that the set of available in-store customers will not be able to provide the desired service level. 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. (2016). 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. (2015) 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. Apart from that, we assume that all orders placed have to be delivered, 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 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. Given that delivery capacity has two components, company-owned vehicles and an uncertain number of in-store customers willing to make deliveries, we investigate the choice of company fleet size.

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 present effective dispatching strategies that differ in the way that probabilistic information about future demand and delivery capacity is exploited.

- We study the interaction between service level, permanent delivery capacity (e.g., company-owned vehicles), and temporary delivery capacity (e.g., in-store customers willing to make deliveries).

- We provide insights resulting from an extensive and comprehensive computational study.

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

More formally, the problem is defined on a graph $G = (V, A)$, where the node set $V$ represents locations and the arc set $A$ represents the connections between locations. Let $V = \{d\} \cup V^o \cup V^c$ with $V^o$ the set online order locations and $V^c$ the set of in-store customer home locations ($V^o$ and $V^c$ may overlap or even coincide), and with $d$ the store location. We assume that the travel time $t_{uv}$ between two locations $u, v \in V$ is deterministic. The online order placement rate at location $v \in V^o$ is $\lambda^o_v$ and the arrival rate at the store of in-store customers with home location $u \in V^c$ is $\lambda^c_u$. Let $N^o$ and $N^c$ represent a realization of demand and capacity, i.e., sets of actual online orders placed and in-store customers arriving over the planning horizon $T$. Each online order $j \in N^o$ has a delivery location in $V^o$, and, similarly, each in-store customer $i \in N^c$ has a home location in $V^c$. Hereafter,
with a slight abuse of notation, $V^o_j$ and $V^c_i$ represent the delivery location of online order $j$, and the home location of in-store $i$, respectively. Accordingly, $t_{ij}$ denotes the travel times between the locations associated with $i$ and $j$. The placement time of online order $j$ and the arrival time of in-store customer $i$ at the store $d$ are denoted by $\tau^o_j$ and $\tau^c_i$, 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 $\rho^o_j = \tau^o_j + S$, where we have implicitly assumed that no online orders are accepted after $T$. Note that in a more general setting, the service level may be location dependent, by considering $S_v$ to be the service level at location $v \in V^o$. The in-store customer availability, representing the maximum time between the time an in-store customer announces his willingness to deliver an online order and the time this in-store customer needs to be taken up on his offer (otherwise he will leave the store), is denoted by $R$. That is, if in-store customer $i$ is not deployed to deliver an online order by time $\tau^c_i + R$, the opportunity to use this in-store customer passes.

We assume that an in-store customer will deliver at most one online order. This simplifies the problem as it avoids having to consider routing decisions for online orders delivered by in-store customers. However, it may not be unrealistic as it also helps limit the inconvenience for the in-store customer. The inconvenience for an in-store customer is also 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 location in a circular region around their home. More specifically, we set the coverage area of in-store customer $i$ to be a circle centered at $V^c_i$ and with radius $r_i = \gamma t_{di}$ for some $\gamma < 1$ (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 area close to their home as opposed to any location along the way to their home. It is easy to adapt our approach to different definitions of coverage area.

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 single online order in the vicinity of his home location, company drivers can, and likely will, deliver multiple online orders on a single trip (starting and ending at the store). For simplicity and presentational convenience, we assume that it is always cheaper to deliver an online order using an in-store customer than using a company driver. This reflects that we assume that the retailer is considering crowdsourced delivery primarily as a means to reduce cost, and not as a means to better handle demand variations or as a means to offer additional delivery services. Furthermore, the compensation of in-store customers making deliveries of online orders may be in the form of store credits rather than actual payments. As a consequence, our dispatching algorithms always prefer to use in-store customers over company drivers for deliveries, i.e., company drivers are used only when too few in-store customers are available to make deliveries or when it is anticipated that using only in-store customers will be insufficient to promised service to online buyers. More specifically, we assume that using in-store customers has zero cost and using company drivers incurs a per-mile cost. Note that a per-mile compensation to company drivers allows consideration of settings in which an employee delivers orders for part of the day, but performs tasks at the store when not delivering orders (Bhattarai 2017). Such a setting is different from a situation in which a driver is employed (and paid) full time to deliver orders, in which case preference should be given to using such a driver. 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 corporate drivers.

At a decision point $h$, the critical decision is whether to dispatch one of 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 point $h$ to decision point $h + 1$ is done in the hope that between decision point $h$ 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). In the first case, the order may be assigned to that in-store customer (and its delivery has zero cost), while in the second case delivering it by a corporate driver may be possible to reduce the per-delivery cost by visiting several geographically close delivery addresses on the same route. 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, $\rho_{ij}^o$).

The goal is to develop technology that decides at each decision epoch, based on the state of the system, which active online orders to serve using available in-store customers, 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, finally, which active orders to postpone (to be considered again at the next decision epoch). As decision will have to be made in near real-time, the efficiency of the technology is of critical concern.

We develop 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 version of the problem, decisions are made altogether \textit{a priori}. In this variant of the problem, it is assumed that the decision maker has \textit{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 availability period of each in-store customer, the service guarantee of the orders, and finally the coverage area of each in-store customer, and 2) vehicle dispatch times and vehicle routes. In the following sections, we discuss how these two categories of decisions are made in the case of the static variant.

3.1. Order-customer assignment

For each order $j$, we define $\theta_j$ to be the latest dispatch time from the store, based on a direct delivery. The value of $\theta_j$ can simply be obtained by $\theta_j := \rho_o^j - t_{dj}$. Also, for each in-store customer $i \in N^c$, let $I(i)$ be the set of online orders that can potentially be served by in-store customer $i$. For all in-store customer $i \in N^c$, set $I(i)$ contains all order $j \in N^o$ for which the following three conditions are satisfied:

1. $t_{ij} \leq r_i$, where $r_i := \gamma t_{di}$
2. $\tau^o_j \leq \tau^c_i + R$
3. $\tau^c_i \leq \theta_j$
The decisions regarding the assignment of online orders to in-store customers can be made by solving a maximum weighted matching problem. The goal of this matching problem is to maximize the total collected weight through assignment of orders to customers. Binary variable $z_{ij}$ equals 1 if in-store customer $i$ serves online order $j$. The matching problem can be formulated as follows.

$$
\text{maximize} \sum_{i \in \mathcal{N}^c} \sum_{j \in I(i)} w_j z_{ij} \quad (1)
$$

subject to

$$\sum_{j \in I(i)} z_{ij} \leq 1, \quad i \in \mathcal{N}^c, \quad (2)$$

$$\sum_{i \in \mathcal{N}^c} z_{ij} \leq 1, \quad j \in \mathcal{N}^o, \quad (3)$$

$$z_{ij} \in \{0, 1\}, \quad i \in \mathcal{N}^c, j \in I(i). \quad (4)$$

Constraints (2) ensure that each in-store customer serves at most one online order, while constraints (3) guarantee that each online order is at most served by a single in-store customer. The weights in the objective function (1) are obtained using Eq. (5).

$$w_j = 1 + \frac{t_{dj}}{\max_{k \in \mathcal{N}^o\{t_{dk}\}} \theta_j}, \quad j \in \mathcal{N}^o. \quad (5)$$

Since the cost structure is based on the routes performed by corporate drivers, our goal is to serve more costly/risky orders using in-store customers, as much as possible. Accordingly, this weighting scheme prioritizes assignment of online orders that are far away from the store (large $t_{dj}$) and/or those that are urgent (small $\theta_j$) to in-store customers.

### 3.2. Vehicle routing

For those orders that are not assigned to any in-store customer based on the solution to model (1)-(4), a set of vehicle routes are generated. In the static variant of the problem, all order placements are known *a priori*; the routing problem takes the form of a multi-trip vehicle routing problem with release and due times (MTVRP-RD). A vehicle can perform potentially multiple trips per day starting and ending from/at the store. An order can only be loaded on a vehicle if it is already
placed (release date). Accordingly, each trip carries those orders which were placed before vehicle’s departure time. The problem has similarities to the well-known vehicle routing problem with time windows (VRPTW). The main difference between the MTVRP-RD and VRPTW is related to the implication of release time versus the start of time window of an order delivery. While the release time constrains the dispatch time of the vehicle from the store, the time window determines the earliest and latest start of service at a customer location. In that sense, the due time has a similar function as the end of time window.

While efficient algorithms exist to perform move evaluation in neighborhood search for the VRPTW (see Vidal et al. 2012, Nagata and Bräysy 2009), VRPs with release times received very limited attentions. In fact, one can model the MTVRP-RD as a VRPTW following a series of three steps.

1. Let $\Sigma = (\sigma_1, \ldots, \sigma_M)$ represent a route $\Sigma$ with $M$ trips. Associated with each trip $\sigma_m$, we consider a temporal copy $d_m$ of the store, from where the trip departs. Let call the intermediate visits to the store simply as intermediate stores. Accordingly, each route consists of visits to a subset of online customer locations (to deliver orders) as well as visits to intermediate stores $\{d_1, \ldots, d_M\}$. Note that $M$, the number of intermediate stores of a route $\Sigma$, is unknown in advance: intermediate stores are inserted as required while designing the routes.

2. Release and due times can be transformed into time windows associated with orders and intermediate store. For each order $j$, let $[0, \rho_j]$ represent its time window, which essentially guarantees due times. Note that the beginning of an order’s time window can be pushed to $\tau_j$ or $\theta_j$ without making any impact to the results.

3. Release times of orders served on trip $\sigma_k$ are taken into account by adjusting the time window associated with intermediate store $d_k$. More precisely, the time window associated with $d_k$ is set to $[\max_{j \in \sigma_k} \tau_j, T]$. Note that the lower bound of this interval is dynamically updated following the insertion or removal of orders in the corresponding trip.

Based on the above definitions, the MTVRP-RD can be transformed into a VRPTW. Note that in the static variant of the problem, vehicle departures can be chosen from a continues set of times.
(decisions regarding dispatch times of different trips of a route are not tied to any discretized set of time points). Therefore, the choice of dispatch times of different trips of a vehicle as well as the sequence of visits must be designed in such a way that primarily the total transportation cost and secondarily the total lateness (due time violation) are minimized. One of the main differences between our problem and the classical VRPTW is associated with the softness of due times. In fact, in our problem the goal is to respect the due times as much as possible by minimizing the total lateness.

We propose a solution method based on Tabu Search to generate the vehicle routes. Note that the choice of a metaheuristic solution method is related to the fact that in such a dynamic environment, decisions must be made at a near-real-time pace, and therefore efficient heuristic approaches providing “good” solutions in short period of time seem more adequate.

3.2.1. 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 $\phi$ to the best solution in its neighbourhood. 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 $C(\phi) = TD(\phi) + \eta TL(\phi)$, where $TD(\phi)$ and $TL(\phi)$ represent the total travel time (transportation cost) and total lateness, respectively. Moreover, $\eta$ is a penalty term which is updated adaptively based on the performance of neighbourhood search. The parameter $\eta$ is initially set to 1. After each block of $Iter^{adj}$ iterations, we multiply $\eta$ by 2 if the number of solutions with non-zero lateness in the last $Iter^{his}$ iterations is greater than $\delta_{max}$, and we divide it by 2 if the number of such solutions is less than $\delta_{min}$.

With each routing solution $\phi$ is associated an attribute set $g(\phi) = \{(j,k) : \text{order } j \text{ is served by vehicle } k\}$. The neighbourhood $N(\phi)$ of solution $\phi$ 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 \( i \) is removed from route \( k \), a tabu status is assigned to the attribute \((i, k)\). 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 \([\text{minTabu}, \text{maxTabu}]\) where \text{minTabu} and \text{maxTabu} are fixed by the user. As long as attribute \((i, k)\) is considered as tabu, the reinsertion of customer \( i \) 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 \( \kappa \), comprising a sequence of order deliveries and intermediate store visits, let \( D(\kappa) \) be the duration (including waiting times at intermediate stores) of a partial path \( \kappa \), and \( E(\kappa) \) and \( L(\kappa) \) be the earliest and latest visits to the first vertex of \( \kappa \) 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).

\[
D(\kappa \oplus \kappa') = D(\kappa) + D(\kappa') + t_{\kappa_u\kappa'_u} + \Delta_{WT}; \quad (6)
\]
\[
E(\kappa \oplus \kappa') = \max\{E(\kappa') - \Delta, E(\kappa)\} - \Delta_{WT}; \quad (7)
\]
\[
L(\kappa \oplus \kappa') = \min\{L(\kappa') - \Delta, L(\kappa)\} + \Delta_{TW}; \quad (8)
\]

where \( \Delta = D(\kappa) + t_{\kappa_u\kappa'_u} \), \( \Delta_{WT} = \max\{E(\kappa') - \Delta - L(\kappa), 0\} \), and \( \Delta_{TW} = \max\{E(\kappa) + \Delta - L(\kappa'), 0\} \). The definition of \( \Delta \) 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 \( D(\kappa) \)) and minimum delivery lateness for the orders delivered on \( \kappa \). Obviously, if the total lateness of a partial path \( \kappa \) is non zero, \( E(\kappa) \) and \( L(\kappa) \) coincide. Moreover, the time
interval \([E(\kappa), L(\kappa)]\) when partial path \(\kappa\) contains one single node, corresponds to the node’s time window. Notice that \(D(\kappa)\) may be different than \(TD(\kappa)\).

For a given route \(\Sigma\), evaluating a move with respect to potential change in \(TL(\Sigma)\) is done in \(O(n^\Sigma)\), with \(n^\Sigma\) being the number of orders delivered on route \(\Sigma\) and by taking \(E(\Sigma)\) as the initial dispatch time from the store. On the other hand, evaluating \(TD(\Sigma)\) 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) \(\Sigma\). Calculation of \(TD(\Sigma)\) 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 2a 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 \(\kappa^1-\kappa^5\) are concatenated. These 5 partial paths can be described as below:

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

\(\kappa^2\): 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. (9) ([a][b] stands for the partial path from node \(a\) to node \(b\)):

\[
E[d_2][d_2] := \max\{E[d_2][d_2], \tau^\circ\} = \max\{\tau^3, \tau^4, \tau^7, \tau^5\}.
\]  

(9)

\(\kappa^3\): The partial path starting from the first delivered order after the intermediate store of the trip in which the node is inserted (order 3 in Figure 2) to the predecessor of the newly inserted order (order 4 in Figure 2).

\(\kappa^4\): The partial path corresponding to the order being inserted.

\(\kappa^5\): The partial path from the successor of the inserted order (order 5 in Figure 2) to the last visit to the store, \(d_4\).
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 3. 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:

\[
E[d_2][d_2] := \max\{E[d_2][\text{pred}(7)], E[\text{succ}(7)][\text{pred}(d_3)]\} = \max\{\tau_3^o, \tau_4^o, \tau_5^o\}. \tag{10}
\]

The routes generated using the described TS covers the remaining orders that were not assigned to any in-store customer by solving model (1)-(4). It is worth mentioning that the solution of the
static version of the problem provides an upper bound on the total number of in-store customers that can be used over the dynamic solution. Similarly, the transportation cost obtained by the static variant (if solved to optimality) provides a lower bound on the cost of the dynamic version of the problem due to a) generally less orders are served through the vehicle routes, b) the knowledge about the future arrivals allows the decision maker to better design the dispatch times and the sequence of visits on different routes, and finally c) vehicle dispatch times are selected from a continues set of choices while in the dynamic decision making, decisions are made over a discrete set of times and consequently vehicle dispatches are also set following the same discretization.

4. Dynamic variant

In the dynamic version of the problem, decisions are made at various times, referred to as decision epochs \( h \in \mathcal{H} \), during the planning horizon \( T \). Slightly abusing notation, for a given decision epoch \( h \), we use \( h \) as the identifier of the epoch as well as of the time of the decision. The state of the system at each decision epoch \( h \) can be represented by a tuple \(( h, \mathcal{N}_h^o, \mathcal{N}_h^c, \mathcal{K}_h)\) described below:

- \( h \): the time of the decision epoch,
- \( \mathcal{N}_h^o \): the set of active online orders,
- \( \mathcal{N}_h^c \): the set of available in-store customers, and
- \( \mathcal{K}_h \): 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 \( h \), while for en-route company vehicles, the next availability is strictly greater than \( h \).

In our solution approaches, the set \( \mathcal{H} \) comprises two types of decision epochs: 1) fixed times throughout the planning horizon (e.g., every \( \delta \) 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:
\( I N_h^o \subseteq N_h^o \): the subset of active online orders to be assigned to available in-store customers 
\( N_h^c \) (with information identifying the in-store customer making the delivery),
\( R N_h^o \subseteq N_h^o \setminus I N_h^o \): the subset of active online orders to be delivered on routes performed by company drivers (with information on the delivery sequences of the routes);
\( P N_h^o = N_h^o \setminus I N_h^o \cup R N_h^o \): 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 transportation cost of routes performed by the company drivers and the total lateness of deliveries. 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.

### 4.1. Myopic decision making

In the myopic decision making approach, at every decision epoch \( h \in H \), we take into account only system state information to make decisions, i.e., \((h, N_h^o, N_h^c, K_h)\). 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. In the proposed myopic scheme, decisions at a given epoch \( h \) are made following a sequence of steps as described in Algorithm 1.
Algorithm 1: Myopic decision logic at a decision epoch $h$

1. **Given**: $h$, $N^o_h$, $N^c_h$, and $K_h$

2. **STEP 1**: Match as many online orders from $N^o_h$ with in-store customers in $N^c_h$

   // Solve a weighted matching problem

3. **STEP 2**: Design vehicle routes to serve all remaining active online orders, $N^o_h \setminus I^N_o^h$

   // Tabu Search given $K_h$

4. **STEP 3**: Release less urgent active online orders, $P^o N^o_h$, from the routes generated in **STEP 2**

   // Based on thresholds $\alpha_1$ and $\alpha_2$

Decisions regarding the assignment of active orders to in-store customers (**STEP 1**) are made by solving a maximum weighted matching problem, similar to model (1)-(4). Note that in the dynamic setting the matching considers only the set of active orders and the available in-store customers. We want to serve as many online orders as possible by the in-store customers. At a given decision epoch $h$, the weight associated with the assignment of online order $j$ is a function of the following factors:

1. The latest possible departure from the store, $\theta_j$, and

2. The distance to the delivery address from the store, $t_{dj}$.

An online order $j$ with a small $\theta_j$ and a large distance $t_{dj}$ is given a higher priority to be matched to an in-store customer. Delivering farther online orders with in-store customers reduces the transportation cost incurred by company drivers. Let $I^h(i) \subseteq N^o_h$ be the set of orders that can be served by in-store customer $i \in N^o_h$. The maximum weight matching problem solved at each decision epoch $h$ takes the following form:

\[
\begin{align*}
\text{maximize} & \quad \sum_{i \in N^c_h} \sum_{j \in I^h(i)} w^h_{ij} z^h_{ij} \\
\text{subject to} & \quad \sum_{j \in I^h(i)} z^h_{ij} \leq 1, \quad i \in N^c_h
\end{align*}
\]
\[
\sum_{i \in N^c_h, j \in I^h(i)} z^h_{ij} \leq 1, \quad j \in N^o_h, \quad (13)
\]
\[
z^h_{ij} \in \{0, 1\}, \quad i \in N^c_h, j \in I^h(i). \quad (14)
\]

where binary variable \(z^h_{ij}\) equals 1 if order \(j\) is assigned to in-store customer \(i\) at epoch \(h\). The weights in the objective function are set as follows:

\[
w^h_j = 1 + \frac{t_{dj}}{\max_{k \in N^c_h} \{t_{dk}\}} \theta_j, \quad j \in N^o_h. \quad (15)
\]

Similar to the static version, relying solely on in-store customers for delivering online orders may not be sufficient. Therefore, at each decision epoch, a set of routes is generated using the tabu search algorithm described in Section 3.2.1 to deliver all or some of the remaining online orders (STEP 2). Note that the set of available vehicles at the store at decision epoch \(h\), \(K^A_h := \{k \in K|K_h[k] \leq h\}\), is an input of the algorithm, and it defines the maximum number of routes departing from the store at time \(h\).

Since we would like to use in-store customers as much as possible rather than company drivers, it might be beneficial to identify a subset of online orders which are currently assigned to a route of a company driver (generated in STEP 2) and release them from the routes (STEP 3). 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 \(\hat{i}\) 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 \(h + 1 \leq \theta_j\). An example of such a situation occurs when only one vehicle is available at epoch \(h + 1\), while a set of relatively urgent orders from different (distant) geographical regions needs to
be delivered. In such a situation, even if the vehicle’s departure takes place before \( \theta_j \) for all orders \( j \) delivered on the trip, one or more orders may be delivered late due to the time traveled before getting to the order’s delivery location.

As detailed in Algorithm 2, in the myopic approach, decisions regarding which online orders to release from the routes are made based on a set of thresholds. We denote the cost of the detour performed to serve online order \( j \) in the current routing solution \( \phi \) at decision point \( h \) by \( \text{cost}^h_j(\phi) \). The detour cost based on a given routing solution \( \phi \) is obtained as

\[
\text{cost}^h_j(\phi) = t_{\text{pred}(j),j} + t_{j,\text{succ}(j)} - t_{\text{pred}(j),\text{succ}(j)}.
\]  

(16)

Let \( \text{cost}^h_j^+ \) represent an estimate of the cost of serving online order \( j \) at a decision epoch later that \( h \). A simple, most likely pessimistic, estimate of the cost of serving online order \( j \) at a later decision epoch is \( \text{cost}^h_j^+ \approx 2t_{dj} \). Note that \( 2t_{dj} \) represents the cost of delivering order \( j \) on with an out-and-back direct delivery route. Using this future cost estimate, an order \( j \) is released from its current route at epoch \( h \) if the value of \( \Gamma_j = \text{cost}^h_j(\phi)/2t_{dj} \) 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 (\( h + 1 \leq \theta_j < h + 2 \)),
- **Non-urgent** Orders \( j \) that will become urgent later (\( \theta_j \geq h + 2 \)).

Note that an order with \( \theta_j < h + 1 \) (called an urgent order) must be dispatched at decision point \( h \). Let \( \alpha_1 \) and \( \alpha_2 \), 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 \( \alpha_1 \) and \( \alpha_2 \) 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 (\( h + 1 = \min h' \in H | h' > h \) and \( |K^A_{h'}| \geq 1 \)). 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 \( H \) in its proper position (\( H \) is maintained in ascending order).
Algorithm 2: Releasing orders from routes (STEP 3) in myopic approach

1. Algorithm Release($\phi, \alpha_1, \alpha_2$)
2. initialize set released_nodes := $\emptyset$
3. do
4. \hspace{1em} released_nodes := $\emptyset$
5. \hspace{1em} while ($|\phi| > 0$) do
6. \hspace{2em} \begin{comment}
7. \hspace{3em} $|\phi|$: the number of orders served on vehicle routes
8. \hspace{3em} $\sigma_i \in \phi$: all trips $\sigma_i$ in routing solution $\phi$
9. \hspace{3em} $L(\sigma_i) \geq h + 1$: late departure
10. \hspace{3em} $\theta_j \geq h + 2$: non-urgent
11. \hspace{3em} $\Gamma_j \geq \alpha_2$: high priority
12. \hspace{3em} $\Gamma_j \geq \alpha_1$: medium priority
13. \end{comment}
14. \hspace{2em} foreach $\sigma_i \in \phi$ do
15. \hspace{3em} if $L(\sigma_i) \geq h + 1$ then
16. \hspace{4em} foreach $j \in \sigma_i$ do
17. \hspace{5em} release $j$ from its current route
18. \hspace{5em} released_nodes := released_nodes $\cup$ $\hat{j}$
19. \hspace{4em} $\hat{\mathcal{N}}^h_k := \hat{\mathcal{N}}^h_k \cup \{\hat{j}\}$
20. \hspace{3em} end
21. \hspace{2em} end
22. \hspace{2em} foreach $\hat{j} \in \phi$ do
23. \hspace{3em} if $\theta_{\hat{j}} \geq h + 2$ then
24. \hspace{4em} \begin{comment}
25. \hspace{5em} $\hat{j}$: non-urgent
26. \hspace{5em} $\hat{j}$: high priority
27. \hspace{5em} $\hat{j}$: medium priority
28. \end{comment}
29. \hspace{4em} if $\Gamma_{\hat{j}} \geq \alpha_2$ then
30. \hspace{5em} release $\hat{j}$ from its current route
31. \hspace{5em} released_nodes := released_nodes $\cup$ $\hat{j}$
32. \hspace{5em} $\hat{\mathcal{N}}^h_k := \hat{\mathcal{N}}^h_k \cup \{\hat{j}\}$
33. \hspace{4em} end
34. \hspace{3em} end
35. \hspace{2em} end
36. \hspace{2em} foreach $\hat{j} \in \phi$ do
37. \hspace{3em} if $h + 1 \leq \theta_{\hat{j}} < h + 2$ then
38. \hspace{4em} \begin{comment}
39. \hspace{5em} $\hat{j}$: semi-urgent
40. \hspace{5em} $\hat{j}$: high priority
41. \hspace{5em} $\hat{j}$: medium priority
42. \end{comment}
43. \hspace{4em} if $\Gamma_{\hat{j}} \geq \alpha_1$ then
44. \hspace{5em} release $\hat{j}$ from its current route
45. \hspace{5em} released_nodes := released_nodes $\cup$ $\hat{j}$
46. \hspace{5em} $\hat{\mathcal{N}}^h_k := \hat{\mathcal{N}}^h_k \cup \{\hat{j}\}$
47. \hspace{4em} end
48. \hspace{3em} end
49. \hspace{2em} end
50. \hspace{2em} Reoptimize($\phi$) // reoptimize $\phi$ using tabu search
51. \hspace{2em} while (released_nodes ! = $\emptyset$ & TL($\phi$) ! = 0)
52. \hspace{2em} return $\phi$

We use the tabu search algorithm designed for the MTVRP-RD in Section 3.2.1 to generate routes in STEP 2. Thus, the routes may contain more than one trip. If a trip $\sigma_i$ of a route generated at epoch $h$ has a latest departure time $L(\sigma_i) \geq h + 1$, then all orders on trip $\sigma_i$ are released. It is worth mentioning that $L(\sigma_i) \geq h + 1$ implies that the same vehicle to which orders on $\sigma_i$ are currently assigned at epoch $h$ will be available at the store to start trip $\sigma_i$ at time $h + 1$. Releasing such orders increases the chance of serving them by in-store customers arriving in time
interval \((h, h + 1]\) or possibly serving one or more orders placed in time interval \((h, h + 1]\) on trip \(\sigma_i\) 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., \((h, N^o_h, N^c_h, K_h)\), 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 3** are influenced by anticipated future events.

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, \(\text{cost}^{h+}_j\), was available. A more accurate estimate of the cost of serving order \(\hat{j}\) 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.

To estimate \(\text{cost}^{h+}_j\) and \(\mathcal{L}^{h+}_j\), i.e., the cost and lateness that can be attributed to online order \(\hat{j} \in \mathcal{N}^o_h \setminus \mathcal{N}^c_h\) 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 \(\text{cost}^{h+}_j\) and \(\mathcal{L}^{h+}_j\) for order \(\hat{j}\). 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.

The cost of serving order \(\hat{j}\) after \(h\) in scenario \(s \in \mathcal{S}\), denoted by \(\text{cost}^{sth+}_j\), is zero if order \(\hat{j}\) is served by a future in-store customer, and the cost of serving \(\hat{j}\) on a route dispatched after \(h\) in case it is not served by an in-store customer.
We now estimate the cost of serving order \( \hat{j} \) in a decision epoch later than \( h \) with respect to a set of scenarios \( S \) as

\[
\text{cost}^{h+}_{\hat{j}} = \frac{1}{|S|} \sum_{s \in S} \text{cost}^{s+h+}_{\hat{j}}. \tag{17}
\]

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

\[
L^{h+}_{\hat{j}} = \frac{1}{|S|} \sum_{s \in S} L^{s+h+}_{\hat{j}}, \tag{18}
\]

where \( L^{s+h+}_{\hat{j}} \) is the lateness that can be attributed to order \( \hat{j} \) in scenario \( s \). Note that computing \( L^{s+h+}_{\hat{j}} \) requires knowledge of the route on which \( \hat{j} \) 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. (We denote this set of orders by \( N^{s+o}_h \). Similarly, we denote the set of in-store customers arriving in the store after time \( h \), based on scenario \( s \), by \( N^{s+c}_h \).)

Next, we discuss in more depth how \( \text{cost}^{s+h+}_{\hat{j}} \) and \( L^{s+h+}_{\hat{j}} \) are computed.

A matching problem is solved for each scenario \( s \in S \) to determine if \( \hat{j} \) will be served by an in-store customer in that scenario. Order \( \hat{j} \) competes with all other online orders placed after time \( h \) for the arriving in-store customers. The matching problem for scenario \( s \) takes the following form:

\[
\text{maximize } \sum_{i \in N^{s+c}_h} \sum_{j \in I^{s+h}(i)} w_{ij}^{s+h} z_{ij}^{s+h} \tag{19}
\]

subject to

\[
\sum_{j \in I^{s+h}(i)} z_{ij}^{s+h} \leq 1, \quad i \in N^{s+c}_h, \tag{20}
\]

\[
\sum_{i \in N^{s+c}_h | j \in I^{s+h}(i)} z_{ij}^{s+h} \leq 1, \quad j \in N^{s+o}_h, \tag{21}
\]

\[
z_{ij}^{s+h} \in \{0, 1\}, \quad i \in N^{s+c}_h, j \in I^{s+h}(i). \tag{22}
\]

where the weights in the objective function are as in (15). Let \( N^{s+o}_{sh} \) be the set of online orders \( j \in N^{s+o}_h \) which are assigned to in-store customers.

If order \( \hat{j} \) is not assigned to any in-store customer arriving after time \( h \), we estimate both the cost of serving \( \hat{j} \) on a future vehicle route and the lateness that can be attributed to \( \hat{j} \) because \( \hat{j} \).
is served on that route. This is done by making routing decisions for all orders in \( N_h^{so} \) using the tabu search algorithm presented in Section 3.2.1.

The cost of serving order \( \hat{j} \) in scenario \( s \), \( \text{cost}^{sh+}_j \), is taken to be the extra-mileage cost associated with \( \hat{j} \). The lateness that can be attributed to \( \hat{j} \), \( L^{sh+}_j \), is determined as follows. Let \( \Sigma^{sh+}_j \) be the vehicle route delivering order \( \hat{j} \) and let \( \Sigma^{sh+}_{(j)} \) be the vehicle route obtained by removing of \( \hat{j} \) from \( \Sigma^{sh+}_j \). Then the lateness that can be attributed to \( \hat{j} \) is taken to be \( TL(\Sigma^{sh+}_j) - TL(\Sigma^{sh+}_{(j)}) \).

Observe that for small values of \( h \), the set \( N_{so}^{h} \) may be quite large, in which case the time required by the tabu search algorithm may be prohibitively large. However, the orders that are placed much later than \( \hat{j} \) are likely to have little impact on the cost of serving \( \hat{j} \). Therefore, we accelerate the tabu search algorithm by limiting the set of events considered. Specifically, we ignore all events after time \( \beta_h \) and where we set \( \beta_h = \max\{ t_v, h \} \), where \( t_v \) and \( h \) are, respectively, the latest vehicle return and the latest fixed decision epoch before \( \theta_{(j)} \):

\[
t_v = \max\{ t_v \in K_h \mid t_v > h \text{ and } t_v \leq \theta_{(j)} \},
\]

\[
h = \max\{ \bar{h} \mid \bar{h} = \left\lfloor \frac{h}{\delta} \right\rfloor + z\delta, z \in \mathbb{Z}_+ \text{ and } \bar{h} \leq \theta_{(j)} \}.
\]

Accordingly, we consider \( N_{so}^{h}(\beta_h) \) instead of \( N_{so}^{h} \) where \( N_{so}^{h}(\beta_h) := \{ j' \in N_{so}^{h} \mid \tau_{so}^{j'} \leq \beta_h \} \). Similarly, we consider \( K_A^{h}(\beta_h) \) instead of \( K_A^{h} \) as the set of vehicles available to serve orders in \( N_{so}^{h}(\beta_h) \).

Limiting the set of events considered in the tabu search algorithm reduces its computational requirements, but, depending on the number of scenarios, \( |S| \), it may still be (too) time consuming. Therefore, we also consider a simpler alternative: we approximate the detour cost to serve an order based on the idea that an order is likely to be served between the two geographically closest (active) orders. Thus, the cost of serving order \( \hat{j} \) on a vehicle route departing after time \( h \) is estimated as

\[
C^{sh}_j \approx t_{j1} + t_{jj_2} - t_{j1j2},
\]

where

\[
j_1 = \arg \min_{j \in \{d\} \cup N_{so}^{h}(\beta_h)(\{j\})} t_{jj}, \quad j_2 = \arg \min_{j \in \{d\} \cup N_{so}^{h}(\beta_h)(\{j_{j1}\})} t_{jj}.
\]
Unfortunately, since no explicit routes are constructed, in this variant it is no longer possible to compute an estimate of the lateness that can be attributed to the order.

As mentioned above, in sample-scenario planning the decision regarding releasing an order \( \hat{j} \) from its current route at decision epoch \( h \) is based on a comparison of its cost and the lateness that can be attributed to it in the current route and an estimate of its cost and the lateness that can be attributed to it when order \( \hat{j} \) is postponed. In the first variant (SSP-V1), when routes are constructed for each scenario, both decision elements, cost and lateness, are available and used in decision making. In the second, simpler variant (SSP-V2), when no routes are constructed, only cost is available and used in decision making.

At each decision epoch \( h \), a set of tentative routes is generated to deliver orders in \( N_h^{\alpha}(\beta_j^h) \). An iterative process is used to release and postpone orders. In SSP-V1, in each iteration, among all orders with lower future cost and lower lateness, the one with the largest cost savings is released and postponed. This step is repeated until there are no more orders with lower cost and lower lateness. Then, the routes are reoptimized using tabu search. If the resulting routes still exhibit lateness, an attempt is made to release and postpone additional orders. 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 take these effects into account once at the beginning of each iteration, and not after each order release in an iteration. (See Algorithm 3.)

In SSP-V2, the same logic is used, except that only future cost estimates are used, since no lateness estimates are available. (See Algorithm 4.)

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-V1, and SSP-V2)
Algorithm 3: Releasing orders from routes (STEP 3) in SSP-V1

1 Algorithm Release(φ, S)
2 initialize set released_nodes := ∅
3 initialize float max_saving := 0
4 initialize int node_to_release := -1
5 initialize bool update := 1
6 do
7   released_nodes := ∅
8     while (∣φ∣ > 0) do
9       foreach σ_i ∈ φ do
10          if L(σ_i) ≥ h + 1 then
11             foreach j ∈ σ_i do
12                release j from its current route
13                released_nodes := released_nodes ∪ {j}
14       end
15     end
16     node_to_release := -1
17     max_saving := 0
18     foreach j ∈ φ do
19       if update != 0 then
20         Solve matching problem for all scenario s ∈ S
21         Solve routing problem for all scenario s ∈ S in which j is not assigned to an in-store customer
22       end
23       if L^h_j ≥ L^h^+_j & cost^h_j(φ) ≥ cost^h^+_j then
24         if cost^h_j(φ) − cost^h^+_j > max_saving then
25             max_saving := cost^h_j(φ) − cost^h^+_j
26         end
27       node_to_release := j
28     end
29   end
30   if node_to_release != -1 then
31     release node_to_release from its current route
32     released_nodes := released_nodes ∪ {node_to_release}
33     \( P^h N^h_k := P^h N^h_k \cup \{node_to_release\} \)
34   else
35     break
36   end
37 end
38 update = 0
39 Reoptimize(φ) // reoptimize φ using tabu search
40 update = 1
41 while (released_nodes ! = ∅ & TL(φ) ! = 0)
42 return φ

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.
Algorithm 4: Releasing orders from routes (STEP 3) in SSP-V2

1 Algorithm Release(\(\phi, S\))
2 initialize set released_nodes := \(\emptyset\)
3 initialize float max_saving := 0
4 initialize int node_to_release := -1
5 initialize bool update := 1
6 do
7 released_nodes := \(\emptyset\)
8 while (\(|\phi| > 0\)) do
9 // \(|\phi|\): the number of orders served on vehicle routes
10 foreach \(\sigma_i \in \phi\) do
11 // \(\sigma_i \in \phi\): all trips \(\sigma_i\) in routing solution \(\phi\)
12 if \(L(\sigma_i) \geq h + 1\) then
13 foreach \(j \in \sigma_i\) do
14 release \(j\) from its current route
15 released_nodes := released_nodes \(\cup j\)
16 end
17 end
18 node_to_release := -1
19 max_saving := 0
20 foreach \(j \in \phi\) do
21 if update != 0 then
22 Solve matching problem for all scenario \(s \in S\)
23 Estimate detour cost to serve \(j\) for all scenario \(s \in S\) in which \(j\) is not assigned to any in-store customer
24 end
25 if \(cost_H^j(\phi) - cost_H^j > max_saving\) then
26 max_saving := cost_H^j(\phi) - cost_H^j
27 node_to_release := \(j\)
28 end
29 end
30 if node_to_release != -1 then
31 release node_to_release from its current route
32 released_nodes := released_nodes \(\cup \{node_to_release\}\)
33 \(P^T_N^h := P^T_N^h \cup \{node_to_release\}\)
34 else
35 break
36 end
37 end
38 update := 0
39 end
40 Reoptimize(\(\phi\)) // reoptimize \(\phi\) using tabu search
41 update := 1
42 while (released_nodes != \(\emptyset\) \&& TL(\(\phi\)) != 0)
43 return \(\phi\)

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 an instance class, we generate $S + 1$ realizations: $S$ 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 set of online orders with their delivery location and their placement time,
- A set of in-store customers with their home location and their arrival times ($\tau_{i}^{c}$) in the store.

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

- Time horizon length $T = 480$ minutes,
- Service guarantee $S = 60$ minutes,
- In-store customers’ availability $R = 30$ minutes,
- In-store customer coverage area, through parameter $\gamma = 0.5$,
- Company fleet size $|K| = 7$,
- 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 $100 \times 100$ 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 $\exp(-\varphi \tilde{d}_j)$, where $\tilde{d}_j$ 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., \( \lambda^o_j = \lambda^o, \forall j \in V^o \), and \( \lambda^c_i = \lambda^c, \forall i \in V^c \), 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 generate 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 \( S \) 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 $\alpha_1$ and $\alpha_2$ (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 $\alpha_1 := \alpha_2 := 0.5$. Next, $\alpha_1$ is iteratively incremented with a step size of 0.1 up to 1.0. For each value of $\alpha_1$, the value of $\alpha_2$ is incremented in the interval $[0.5, \alpha_1]$ with a step-size 0.1. The performance of the algorithm for each threshold pair $(\alpha_1, \alpha_2)$ is evaluated based on $C(\phi)$, with $\eta = 4$, where $\phi$ represents the routing solution obtained for the instance. The best threshold pair $(\alpha_1^*, \alpha_2^*)$ for each instance class is identified.

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 third category is further broken down into two groups of results representing the SSP-V1 and SSP-V2 approaches. The approaches are compared based on the following metrics:

**cost:** The total cost of the routes performed by company vehicles.
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 cost of a given solution approach compared to the cost of the static approach (as a percentage).

gap2 %: The cost of a given solution approach compared to the minimum cost among the dynamic approaches (as a percentage).

Note that because the routing problem that has to be solved in the static version can be quite large, for some instances the tabu search was unable to find solutions with zero total lateness given the limit on the number of iterations.

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 costs (fewer and/or shorted 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 average total cost for the myopic approach is 1818.0 whereas the average total cost for the two SSP variants is 1791.0 and 1394.2, respectively. Maybe even more important, the two SSP variants reduced the total lateness by 73% and 68%, respectively. One of the reasons, maybe the primary reason, the two SPP variants have lower total costs and smaller total lateness is their ability to take more advantage of the in-store customers by identifying and releasing orders with a high chance to be served by future in-store customers.

A closer look into the performance of SSP-V1 and SSP-V2 shows that despite the more restrictive rule for releasing an order from its route of SSP-V1 (lower predicted lateness and lower predicted cost than current lateness and cost) compared to SSP-V2 (lower predicted cost than current cost),
SSP-V1 tends to release more orders over the planning horizon. This is likely a result of the fact that the basis of cost and lateness predictions in the two approaches is different. SSP-V2 tends to steadily release a moderate number of orders throughout the time horizon. SSP-V1, on the other hand, shows a more fluctuating pattern. More specifically, it starts by releasing many orders at the beginning of the time horizon, which increases the chance of benefiting from the arrival of future in-store customers. However, when the arrival rate of in-store customers is much lower than placement rate of online orders (Rate1), releasing too many orders may result in an accumulation of orders. While these orders are not necessarily urgent, a lack of available vehicle capacity, may result in a large number of orders having to be delivered by the same vehicle on a long route, resulting in a high cost and a high lateness. When this starts to happen, SPP-V1 starts to release fewer orders, since few vehicles will be available in the near future. Once this state of “panic” has passed, SPP-V1 again starts to release orders more aggressively. And the pattern repeats. Interestingly, changing the rule used to release order in SSP-V1 to only account for cost (rather than cost and lateness) does not improve the results.

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 matching problems as well as several routing problems are solved in order to decide on the release of orders. It is also clear that the simple approximation of the detour cost used in SSP-V2, results in a significant reduction in the average time spent per epoch. Interestingly, the simple approximation in detour cost resulted in a lower total cost at the price of a slightly higher total lateness. 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-V2 because of its better quality - solve time ratio.
Table 1: Results for instances in the 12 instance classes using different solution approaches - Fleet size: 7.

<table>
<thead>
<tr>
<th></th>
<th>Static</th>
<th>Myopic</th>
<th>Ignoring future</th>
<th>Anticipating future</th>
<th>Full information</th>
<th>Ignoring future</th>
<th>Anticipating future</th>
<th>Ignoring future</th>
<th>Anticipating future</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>cost lateness</td>
<td>cost lateness</td>
<td>cost lateness</td>
<td>cost lateness</td>
<td>cost lateness</td>
<td>cost lateness</td>
<td>cost lateness</td>
<td>cost lateness</td>
<td>cost lateness</td>
</tr>
<tr>
<td></td>
<td>used in-cust</td>
<td>used in-cust</td>
<td>used in-cust</td>
<td>used in-cust</td>
<td>used in-cust</td>
<td>used in-cust</td>
<td>used in-cust</td>
<td>used in-cust</td>
<td>used in-cust</td>
</tr>
<tr>
<td></td>
<td>time(\sigma)</td>
<td>time/epoch(\sigma)</td>
<td>gap1(%)</td>
<td>gap2(%)</td>
<td>gap1(%)</td>
<td>gap2(%)</td>
<td>gap1(%)</td>
<td>gap2(%)</td>
<td>gap1(%)</td>
</tr>
<tr>
<td>Rate 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Loc 1</td>
<td>1943.2</td>
<td>258.8</td>
<td>162.0</td>
<td>4026.3</td>
<td>258.1</td>
<td>667.6</td>
<td>133.9</td>
<td>567.3</td>
<td>15.3</td>
</tr>
<tr>
<td></td>
<td>2946.8</td>
<td>197.1</td>
<td>148.8</td>
<td>181.1</td>
<td>52.3</td>
<td>26.6</td>
<td>2385.8</td>
<td>193.8</td>
<td>44.5</td>
</tr>
<tr>
<td></td>
<td>1946.0</td>
<td>132.7</td>
<td>157.1</td>
<td>1029.8</td>
<td>2841.2</td>
<td>630.5</td>
<td>190.2</td>
<td>515.3</td>
<td>17.9</td>
</tr>
<tr>
<td>Rate 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Loc 1</td>
<td>2044.6</td>
<td>346.4</td>
<td>161.0</td>
<td>1098.2</td>
<td>2831.7</td>
<td>877.5</td>
<td>137.2</td>
<td>43.3</td>
<td>14.9</td>
</tr>
<tr>
<td></td>
<td>2946.8</td>
<td>197.1</td>
<td>148.8</td>
<td>181.1</td>
<td>52.3</td>
<td>26.6</td>
<td>2385.8</td>
<td>193.8</td>
<td>44.5</td>
</tr>
<tr>
<td></td>
<td>1946.0</td>
<td>132.7</td>
<td>157.1</td>
<td>1029.8</td>
<td>2841.2</td>
<td>630.5</td>
<td>190.2</td>
<td>515.3</td>
<td>17.9</td>
</tr>
<tr>
<td>Rate 3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Loc 1</td>
<td>1108.2</td>
<td>0.0</td>
<td>279.3</td>
<td>787.5</td>
<td>1973.9</td>
<td>107.8</td>
<td>227.8</td>
<td>106.0</td>
<td>55.8</td>
</tr>
<tr>
<td></td>
<td>1981.4</td>
<td>17.6</td>
<td>262.5</td>
<td>87.1</td>
<td>67.6</td>
<td>29.8</td>
<td>1525.7</td>
<td>12.0</td>
<td>45.0</td>
</tr>
<tr>
<td></td>
<td>2030.7</td>
<td>183.4</td>
<td>239.0</td>
<td>118.8</td>
<td>55.3</td>
<td></td>
<td>1312.4</td>
<td>2.1</td>
<td>60.9</td>
</tr>
<tr>
<td>Rate 4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Loc 1</td>
<td>990.2</td>
<td>0.0</td>
<td>279.3</td>
<td>787.5</td>
<td>1973.9</td>
<td>107.8</td>
<td>227.8</td>
<td>106.0</td>
<td>55.8</td>
</tr>
<tr>
<td></td>
<td>1981.4</td>
<td>17.6</td>
<td>262.5</td>
<td>87.1</td>
<td>67.6</td>
<td>29.8</td>
<td>1525.7</td>
<td>12.0</td>
<td>45.0</td>
</tr>
<tr>
<td></td>
<td>2030.7</td>
<td>183.4</td>
<td>239.0</td>
<td>118.8</td>
<td>55.3</td>
<td></td>
<td>1312.4</td>
<td>2.1</td>
<td>60.9</td>
</tr>
<tr>
<td>Rate 5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Loc 1</td>
<td>227.0</td>
<td>0.0</td>
<td>347.1</td>
<td>712.9</td>
<td>572.0</td>
<td>114.8</td>
<td>322.3</td>
<td>172.1</td>
<td>74.4</td>
</tr>
<tr>
<td></td>
<td>381.5</td>
<td>3.3</td>
<td>345.3</td>
<td>28.1</td>
<td>78.7</td>
<td>16.7</td>
<td>341.7</td>
<td>0.0</td>
<td>38.7</td>
</tr>
<tr>
<td></td>
<td>496.2</td>
<td>1.2</td>
<td>346.1</td>
<td>26.5</td>
<td>46.3</td>
<td>14.7</td>
<td>454.7</td>
<td>0.0</td>
<td>35.3</td>
</tr>
<tr>
<td>Rate 6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Loc 1</td>
<td>342.1</td>
<td>0.0</td>
<td>348.1</td>
<td>724.7</td>
<td>686.2</td>
<td>131.0</td>
<td>322.0</td>
<td>159.9</td>
<td>64.4</td>
</tr>
<tr>
<td></td>
<td>196.8</td>
<td>0.0</td>
<td>351.7</td>
<td>662.7</td>
<td>506.3</td>
<td>112.6</td>
<td>331.1</td>
<td>228.9</td>
<td>106.0</td>
</tr>
<tr>
<td></td>
<td>400.7</td>
<td>4.3</td>
<td>350.4</td>
<td>17.2</td>
<td>64.3</td>
<td>10.4</td>
<td>288.3</td>
<td>0.0</td>
<td>44.4</td>
</tr>
<tr>
<td>Rate 7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Loc 1</td>
<td>1316.3</td>
<td>0.0</td>
<td>343.7</td>
<td>711.8</td>
<td>715.3</td>
<td>116.2</td>
<td>315.6</td>
<td>139.4</td>
<td>80.3</td>
</tr>
<tr>
<td></td>
<td>526.5</td>
<td>1.6</td>
<td>341.7</td>
<td>31.7</td>
<td>66.7</td>
<td>26.4</td>
<td>439.6</td>
<td>0.0</td>
<td>312.0</td>
</tr>
<tr>
<td></td>
<td>1970.0</td>
<td>0.8</td>
<td>354.2</td>
<td>229.8</td>
<td>101.4</td>
<td>49.5</td>
<td>1394.2</td>
<td>112.4</td>
<td>45.6</td>
</tr>
<tr>
<td>average</td>
<td>1186.2</td>
<td>83.9</td>
<td>263.4</td>
<td>834.1</td>
<td>1838.0</td>
<td>354.2</td>
<td>229.8</td>
<td>101.4</td>
<td>49.5</td>
</tr>
</tbody>
</table>

For Rate 2, Loc 1 has the highest cost lateness among the instances, while Rate 3, Loc 1 has the lowest cost lateness. Ignoring future solution approaches result in lower lateness compared to anticipating future approaches. The SSP-V1 approach generally outperforms other approaches in terms of cost and lateness, followed by the Full information approach.
5.4. 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. As expected, a larger fleet size improves the service quality by decreasing average total delivery lateness. However, 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). 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.

To be sure that these findings are meaningful, we also solved the more difficult instances (Rate1) with SSP-V1 with different fleet sizes. The results show, again, that a larger fleet size improves the service quality, but in this case, the cost increased as well.

5.5. The impact of decision making frequency

Table 3 reports results when the time between consecutive fixed decision epochs ($\delta$) varies from 5 minutes to 30 minutes, represented by EP5, EP10, EP20, and EP30. The results show that a higher decision making frequency may result in a lower total cost. This is likely due to fact that the decision maker has more opportunities to recover from “unfortunate decisions” made at earlier decision epochs. We also observe that EP5 and EP30 result in the lowest total lateness. A lower total lateness in the case of EP5 can be associated with the higher flexibility. When the system state is checked more frequently, there is a higher chance to identify matches between available in-store customers and online orders. The high service quality in the case of EP30 may be related to the fact that the decision making is more fact-based rather than estimation-based. 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.
Table 2  Results for instances in the 12 instance classes based on different fleet sizes.

<table>
<thead>
<tr>
<th></th>
<th>7 Veh</th>
<th>8 Veh</th>
<th>9 Veh</th>
<th>10 Veh</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>cost</td>
<td>cost</td>
<td>cost</td>
<td>cost</td>
</tr>
<tr>
<td></td>
<td>lateness</td>
<td>lateness</td>
<td>lateness</td>
<td>lateness</td>
</tr>
<tr>
<td>Rate 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Loc 1</td>
<td>2336.8</td>
<td>2342.9</td>
<td>2293.9</td>
<td>2282.3</td>
</tr>
<tr>
<td></td>
<td>193.8</td>
<td>63.2</td>
<td>24.2</td>
<td>0.8</td>
</tr>
<tr>
<td>Loc 2</td>
<td>2467.5</td>
<td>2483.0</td>
<td>2467.8</td>
<td>2443.3</td>
</tr>
<tr>
<td></td>
<td>349.8</td>
<td>75.9</td>
<td>34.5</td>
<td>5.6</td>
</tr>
<tr>
<td>Loc 3</td>
<td>2604.4</td>
<td>2599.9</td>
<td>2550.2</td>
<td>2578.6</td>
</tr>
<tr>
<td></td>
<td>611.3</td>
<td>194.0</td>
<td>56.0</td>
<td>12.9</td>
</tr>
<tr>
<td>Loc 4</td>
<td>2283.3</td>
<td>2299.2</td>
<td>2305.7</td>
<td>2308.3</td>
</tr>
<tr>
<td></td>
<td>178.9</td>
<td>45.4</td>
<td>7.9</td>
<td>0.0</td>
</tr>
<tr>
<td>Rate 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Loc 1</td>
<td>1284.8</td>
<td>1284.0</td>
<td>1289.9</td>
<td>1289.9</td>
</tr>
<tr>
<td></td>
<td>0.3</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>Loc 2</td>
<td>1525.7</td>
<td>1528.0</td>
<td>1530.9</td>
<td>1530.3</td>
</tr>
<tr>
<td></td>
<td>12.0</td>
<td>0.3</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>Loc 3</td>
<td>1312.4</td>
<td>1293.3</td>
<td>1302.2</td>
<td>1288.6</td>
</tr>
<tr>
<td></td>
<td>2.1</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>Loc 4</td>
<td>1400.9</td>
<td>1405.2</td>
<td>1404.8</td>
<td>1402.4</td>
</tr>
<tr>
<td></td>
<td>0.2</td>
<td>0.2</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>Rate 3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Loc 1</td>
<td>341.7</td>
<td>341.7</td>
<td>341.7</td>
<td>341.7</td>
</tr>
<tr>
<td></td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>Loc 2</td>
<td>454.7</td>
<td>454.7</td>
<td>454.3</td>
<td>454.7</td>
</tr>
<tr>
<td></td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>Loc 3</td>
<td>288.3</td>
<td>288.4</td>
<td>288.3</td>
<td>288.4</td>
</tr>
<tr>
<td></td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>Loc 4</td>
<td>429.6</td>
<td>429.6</td>
<td>429.6</td>
<td>439.4</td>
</tr>
<tr>
<td></td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>average</td>
<td>1394.2</td>
<td>1395.8</td>
<td>1388.3</td>
<td>1387.3</td>
</tr>
<tr>
<td></td>
<td>112.4</td>
<td>31.6</td>
<td>10.2</td>
<td>1.6</td>
</tr>
</tbody>
</table>
Table 3  The impact of decision making frequency (fixed epoch length).

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>cost</td>
<td>latency</td>
<td># used in-cust</td>
<td>cost</td>
<td>latency</td>
<td># used in-cust</td>
<td>cost</td>
<td>latency</td>
<td># used in-cust</td>
<td>cost</td>
<td>latency</td>
</tr>
<tr>
<td>Rate 1</td>
<td>Loc 1</td>
<td>2290.4</td>
<td>150.3</td>
<td>145.7</td>
<td>2336.8</td>
<td>193.8</td>
<td>143.5</td>
<td>2380.6</td>
<td>238.7</td>
<td>142.0</td>
<td>2380.5</td>
</tr>
<tr>
<td></td>
<td>Loc 2</td>
<td>2426.1</td>
<td>290.7</td>
<td>146.8</td>
<td>2467.5</td>
<td>349.8</td>
<td>145.7</td>
<td>2488.2</td>
<td>353.5</td>
<td>140.7</td>
<td>2540.9</td>
</tr>
<tr>
<td></td>
<td>Loc 3</td>
<td>2546.1</td>
<td>561.7</td>
<td>150.7</td>
<td>2604.4</td>
<td>611.3</td>
<td>149.5</td>
<td>2614.8</td>
<td>624.6</td>
<td>147.2</td>
<td>2705.6</td>
</tr>
<tr>
<td></td>
<td>Loc 4</td>
<td>2299.0</td>
<td>202.8</td>
<td>142.1</td>
<td>2283.3</td>
<td>178.9</td>
<td>142.2</td>
<td>2384.8</td>
<td>175.6</td>
<td>138.3</td>
<td>2356.8</td>
</tr>
<tr>
<td>Rate 2</td>
<td>Loc 1</td>
<td>1241.7</td>
<td>0.1</td>
<td>255.5</td>
<td>1284.8</td>
<td>0.3</td>
<td>252.0</td>
<td>1386.5</td>
<td>0.0</td>
<td>245.1</td>
<td>1435.6</td>
</tr>
<tr>
<td></td>
<td>Loc 2</td>
<td>1463.9</td>
<td>6.7</td>
<td>258.1</td>
<td>1525.7</td>
<td>12.0</td>
<td>254.2</td>
<td>1630.7</td>
<td>5.5</td>
<td>246.1</td>
<td>1696.0</td>
</tr>
<tr>
<td></td>
<td>Loc 3</td>
<td>1235.9</td>
<td>0.1</td>
<td>266.1</td>
<td>1312.4</td>
<td>2.1</td>
<td>261.2</td>
<td>1468.8</td>
<td>0.8</td>
<td>254.7</td>
<td>1540.4</td>
</tr>
<tr>
<td></td>
<td>Loc 4</td>
<td>1323.2</td>
<td>2.2</td>
<td>254.0</td>
<td>1400.9</td>
<td>0.2</td>
<td>249.0</td>
<td>1493.6</td>
<td>0.9</td>
<td>243.4</td>
<td>1603.7</td>
</tr>
<tr>
<td>Rate 3</td>
<td>Loc 1</td>
<td>288.0</td>
<td>0.0</td>
<td>339.8</td>
<td>341.7</td>
<td>0.0</td>
<td>336.7</td>
<td>446.1</td>
<td>0.0</td>
<td>329.5</td>
<td>578.2</td>
</tr>
<tr>
<td></td>
<td>Loc 2</td>
<td>393.3</td>
<td>0.0</td>
<td>340.8</td>
<td>454.7</td>
<td>0.0</td>
<td>337.1</td>
<td>616.9</td>
<td>0.0</td>
<td>328.6</td>
<td>767.5</td>
</tr>
<tr>
<td></td>
<td>Loc 3</td>
<td>239.8</td>
<td>0.0</td>
<td>346.8</td>
<td>288.3</td>
<td>0.0</td>
<td>344.3</td>
<td>394.1</td>
<td>0.0</td>
<td>337.0</td>
<td>539.3</td>
</tr>
<tr>
<td></td>
<td>Loc 4</td>
<td>389.2</td>
<td>0.0</td>
<td>334.5</td>
<td>429.6</td>
<td>0.0</td>
<td>332.0</td>
<td>603.6</td>
<td>0.0</td>
<td>322.3</td>
<td>708.4</td>
</tr>
<tr>
<td>average</td>
<td></td>
<td>1344.7</td>
<td>101.2</td>
<td>248.4</td>
<td>1394.2</td>
<td>112.4</td>
<td>245.6</td>
<td>1492.4</td>
<td>116.6</td>
<td>239.6</td>
<td>1571.1</td>
</tr>
</tbody>
</table>
5.6. The impact of coverage area of in-store customers

As mentioned in Section 2, we assume that the coverage area of each in-store customer $i$ is defined by a circle, centered at his home location, where the radius of the circle is $r_i = \gamma d_i$, with $\gamma$ being a fixed parameter. In practice, the magnitude of parameter $\gamma$ can be a function of the compensation to the in-store customers. Table 4 reports results for two different values of $\gamma$: $\gamma = 0.5$ and $\gamma = 0.25$. A smaller value of $\gamma$ results in a lower chance of matching an online order with an in-store customer. This can clearly be observed as the average number of total in-store customers used across the different instance classes drops from 245.6 to 191.2. As a result, the average total cost and total lateness deteriorate (shown in the column labeled gap %).

<table>
<thead>
<tr>
<th>Rate</th>
<th>Loc 1</th>
<th>Loc 2</th>
<th>Loc 3</th>
<th>Loc 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cost</td>
<td>2336.8</td>
<td>2467.5</td>
<td>2604.4</td>
<td>2283.3</td>
</tr>
<tr>
<td>Lateness</td>
<td>193.8</td>
<td>349.8</td>
<td>611.3</td>
<td>178.9</td>
</tr>
<tr>
<td># in-cust</td>
<td>143.5</td>
<td>145.7</td>
<td>149.5</td>
<td>142.2</td>
</tr>
<tr>
<td>Cost</td>
<td>2768.6</td>
<td>2898.7</td>
<td>3084.3</td>
<td>2634.4</td>
</tr>
<tr>
<td>Lateness</td>
<td>795.7</td>
<td>910.1</td>
<td>1710.5</td>
<td>411.2</td>
</tr>
<tr>
<td># in-cust</td>
<td>99.0</td>
<td>102.5</td>
<td>103.2</td>
<td>106.5</td>
</tr>
<tr>
<td>Gap (%)</td>
<td>19.1</td>
<td>17.9</td>
<td>19.2</td>
<td>15.8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Rate</th>
<th>Loc 1</th>
<th>Loc 2</th>
<th>Loc 3</th>
<th>Loc 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cost</td>
<td>1284.8</td>
<td>1525.7</td>
<td>1312.4</td>
<td>1400.9</td>
</tr>
<tr>
<td>Lateness</td>
<td>0.3</td>
<td>12.0</td>
<td>2.1</td>
<td>0.2</td>
</tr>
<tr>
<td># in-cust</td>
<td>252.0</td>
<td>254.2</td>
<td>261.2</td>
<td>249.0</td>
</tr>
<tr>
<td>Cost</td>
<td>2069.5</td>
<td>2209.3</td>
<td>2272.5</td>
<td>1828.3</td>
</tr>
<tr>
<td>Lateness</td>
<td>91.7</td>
<td>106.6</td>
<td>172.2</td>
<td>23.4</td>
</tr>
<tr>
<td># in-cust</td>
<td>179.3</td>
<td>190.1</td>
<td>185.7</td>
<td>194.9</td>
</tr>
<tr>
<td>Gap (%)</td>
<td>62.9</td>
<td>45.5</td>
<td>74.9</td>
<td>32.1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Rate</th>
<th>Loc 1</th>
<th>Loc 2</th>
<th>Loc 3</th>
<th>Loc 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cost</td>
<td>341.7</td>
<td>454.7</td>
<td>288.3</td>
<td>429.6</td>
</tr>
<tr>
<td>Lateness</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td># in-cust</td>
<td>336.7</td>
<td>337.1</td>
<td>344.3</td>
<td>332.0</td>
</tr>
<tr>
<td>Cost</td>
<td>1175.4</td>
<td>1112.2</td>
<td>1148.9</td>
<td>863.1</td>
</tr>
<tr>
<td>Lateness</td>
<td>0.4</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td># in-cust</td>
<td>273.7</td>
<td>287.5</td>
<td>285.6</td>
<td>286.8</td>
</tr>
<tr>
<td>Gap (%)</td>
<td>255.8</td>
<td>157.3</td>
<td>357.3</td>
<td>119.2</td>
</tr>
</tbody>
</table>

| Average | 1394.2 | 2005.4 |
| Lateness | 112.4 | 351.8 |
| # in-cust | 245.6 | 191.2 |
| Gap (%) | 98.1 |
5.7. 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 5 provides results for two different service guarantees. The first set of results corresponds to a service guarantee of 1 hour (the same level of service that we considered everywhere else in this paper), while the second set corresponds to a service guarantee of 2 hours. As expected, by increasing the service guarantee the total service level violation tends towards zero, while the transportation cost is decreased by 50%, due to a larger chance to use in-store customers (259.5 vs. 245.6), and more time to make better routing decisions by accumulating orders with delivery addresses that are close and dispatching them together on a vehicle route. Interestingly, in instance Loc4-Rate3-R16 from the instance class Loc4 and Rate3, when solved with a 2h service guarantee, only in-store customers were used to deliver orders.

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 an online order 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

<table>
<thead>
<tr>
<th>Rate</th>
<th>Loc 1</th>
<th>Loc 2</th>
<th>Loc 3</th>
<th>Loc 4</th>
<th>Rate 2</th>
<th>Loc 1</th>
<th>Loc 2</th>
<th>Loc 3</th>
<th>Loc 4</th>
<th>Rate 3</th>
<th>Loc 1</th>
<th>Loc 2</th>
<th>Loc 3</th>
<th>Loc 4</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>cost</td>
<td>lateness</td>
<td># in-cust</td>
<td>cost</td>
<td>lateness</td>
<td># in-cust</td>
<td>cost</td>
<td>lateness</td>
<td># in-cust</td>
<td>cost</td>
<td>lateness</td>
<td># in-cust</td>
<td>cost</td>
<td>lateness</td>
</tr>
<tr>
<td>S = 1h</td>
<td>2336.8</td>
<td>193.8</td>
<td>143.5</td>
<td>1278.8</td>
<td>0.2</td>
<td>154.8</td>
<td>1284.8</td>
<td>0.3</td>
<td>252.0</td>
<td>630.6</td>
<td>0.0</td>
<td>271.2</td>
<td>341.7</td>
<td>0.0</td>
</tr>
<tr>
<td>S = 2h</td>
<td>2467.5</td>
<td>349.8</td>
<td>145.7</td>
<td>1319.0</td>
<td>0.0</td>
<td>156.5</td>
<td>1525.7</td>
<td>12.0</td>
<td>254.2</td>
<td>726.7</td>
<td>0.0</td>
<td>272.5</td>
<td>454.7</td>
<td>0.0</td>
</tr>
<tr>
<td></td>
<td>2604.4</td>
<td>611.3</td>
<td>149.5</td>
<td>1355.9</td>
<td>0.0</td>
<td>160.5</td>
<td>1312.4</td>
<td>2.1</td>
<td>261.2</td>
<td>612.5</td>
<td>0.0</td>
<td>279.6</td>
<td>288.3</td>
<td>0.0</td>
</tr>
<tr>
<td></td>
<td>2283.3</td>
<td>178.9</td>
<td>142.2</td>
<td>1254.7</td>
<td>0.0</td>
<td>152.3</td>
<td>1400.9</td>
<td>0.2</td>
<td>249.0</td>
<td>678.0</td>
<td>0.0</td>
<td>268.2</td>
<td>429.6</td>
<td>0.0</td>
</tr>
<tr>
<td>average</td>
<td>1394.2</td>
<td>112.4</td>
<td>245.6</td>
<td>700.3</td>
<td>0.0</td>
<td>259.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
of the compensation offered for performing the delivery. We have not explicitly addressed the compensation aspect and simply assumed that using in-store customers is always cheaper than using company drivers. As long as this is a valid assumption, then the logical preference is to use in-store customers as much as possible. 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).

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.

References


