Transportation Research Part C 130 (2021) 103320

Contents lists available at ScienceDirect

Transportation Research Part C

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

Operational strategies for on-demand personal shopper services

Alp M. Arslan *, Niels Agatz, Mathias A. Klapp

Technology Policy and Management, Delft University of Technology, The Netherlands
Rotterdam School of Management, Erasmus University, The Netherlands
School of Engineering, Pontificia Universidad Católica de Chile, Chile

ARTICLE INFO

Keywords:
On-demand delivery
Personal shopper
Pickup and delivery
Consolidation
Routing

ABSTRACT

Inspired by recent innovations in the retail industry, we study an on-demand personal shopper service that lets customers shop online for products from brick and mortar stores. A fleet of personal shoppers fulfills customer orders, performing both shopping at the stores and delivery to the customers. The operation of such a service requires to dynamically offer service to incoming shopping and delivery requests, coordinate a fleet of shoppers, schedule shopping operations at stores, and execute deliveries to customers on time. This paper studies this new problem and proposes three different order-consolidation strategies for service operators. Our numerical results show that order consolidation, particularly shopping consolidation, can significantly increase the number of served orders by better using the available shopper capacity. We observe that splitting orders can further enhance the performance of the system. The results also suggest that personal shopper services increase time utilization and execute shopping activities using significantly less human resources than customers shopping for themselves.

1. Introduction

Online retailers continuously seek faster delivery services to satisfy the customers' need for instant gratification. Online shopping services such as Amazon Prime Now offer delivery within one or two-hour deadlines in selected US cities. Moreover, we see the rise of online platforms that allow customers to receive expedited deliveries from local brick and mortar stores. These platforms aggregate the assortment of different affiliated stores to provide a convenient 'one-stop shopping' experience.

These services are increasingly popular in the delivery of groceries and convenience goods, since they integrate the comfort of online shopping with product assortment at brick and mortar stores. The Coronavirus pandemic has boosted the demand for online grocery delivery services due to lock-downs and customers' safety concerns of shopping at the physical stores. For example, online grocery sales in the US in August 2020 increased by 400% as compared to the previous year (Bishop, 2020). It is expected that many of the new customers that moved online during the pandemic will continue to shop online after the pandemic (Gunday et al., 2020).

A lot of the growth in online grocery was captured by personal shopper services (Nesin, 2020). The reason for this is that traditional warehouse-based online grocery retailers could not quickly ramp up their fulfillment capacity (*e.g.*, warehouse space) to handle the demand surge. As the *asset-light* personal shopper services pick directly from existing stores, their fulfillment capacity is more flexible, *e.g.*, Instacart hired 300,000 new drivers in March 2020. Consequently, these platforms saw spectacular growth rates. Grocery delivery platform Instacart is now operating in all major cities in the US and Canada and its valuation has more than doubled in 2020 to 17.7 billion USD. Cornershop (The Americas), GrabMarkt (South East Asia), Getir (Europe), and Meutian (China) are other examples of this online grocery model that have seen strong growth.

* Corresponding author.
E-mail address: A.M.Arslan@tudelft.nl (A.M. Arslan).

https://doi.org/10.1016/j.trc.2021.103320
Received 2 December 2020; Received in revised form 13 June 2021; Accepted 15 July 2021
Available online 29 July 2021

Most personal shopper services offer on-demand delivery within a specific delivery deadline, *e.g.*, 120 min. To guarantee timely delivery, an automated dispatcher dynamically offers service to incoming customer requests and assigns orders (*i.e.*, accepted requests) to *personal shoppers*. The fulfillment of an order involves shopping at specific stores and delivery to the customers' location. These operations are similar to those in meal delivery. However, in personal shopper operations there is typically more time and route flexibility as compared to the delivery of fresh meals. Another difference is that the personal shopper may have to visit multiple retail stores to serve a single customer.

In this paper, we introduce the Personal Shopper Problem (PSP), which models an online shopping service dynamically receiving and serving on-demand shopping and delivery requests. All orders must be assigned to shoppers and be fully served before a known delivery deadline. The objective of the PSP is to maximize the number of requests served subject to a fixed shopper capacity.

To simplify planning, personal shopper service providers may operate a sequential delivery strategy, in which each shopper serves a single customer order at a time. However, it may be possible to reduce shopping and travel by combining the service of multiple customer orders in one trip. These time reductions could add more service capacity, especially if the different orders share the same shopping locations, or if delivery locations are close. Unfortunately, a delivery service with tight delivery deadlines offers limited order consolidation opportunities. Therefore, we explore different operational strategies for personal shopper services. In particular, we study the benefits of splitting a customer order that involves shopping at multiple stores into separate tasks that are served by different shoppers. Also, we study the additional consolidation opportunities that arise from splitting customer orders.

Our main contributions are summarized as follows. (i) We are the first to study a new service model in which personal shoppers shop for lists of goods at local stores for delivery to the customer. It differs from other same-day delivery systems, since it vertically integrates product fulfillment (shopping) and order distribution activities in one person. This integration enables a fast response service without intermediate order transfers. (ii) We introduce the Personal Shopper Problem model and present a rolling horizon framework that dynamically solves a snapshot version of the problem. Moreover, we present three operational strategies, each involving a different setting of order consolidation and service splitting. (iii) We assess our solution approach and the performance of different operational strategies with a set of computational experiments across different instance settings. We observe that the number of served orders increases as order consolidation and service splitting strategies are implemented. (iv) Finally, we also numerically study the effectiveness of the personal shopper service model compared to a setting in which customers travel to the stores and shop for themselves; *i.e.*, a 'Do it Yourself' (DIY) benchmark.

The remainder of this paper is organized as follows. Section 2 provides a review of the related literature. We present the problem setting, a decision framework and operational strategies in Section 3. Section 4 presents solution methods for the snapshot optimization problem. Section 5 outlines the results of our computational study. Finally, Section 6 provides conclusions and future research opportunities.

2. Literature review

Conceptually, the Personal Shopper Problem (PSP) is related to the vast family of Pick-up and Delivery Problems (PDP), that involve designing cost-efficient routes to serve a set of transport requests, each with a specific origin and destination (Savelsbergh and Sol, 1995). The PSP extends the PDP, as one single transportation request may require pickups at multiple locations. Moreover, while the 'pickup time' is often negligible as compared to the travel time in the PDP, it is significant for the PSP and concave increasing on the number of simultaneous pickups per store visit. Also, the PSP vertically integrates the shopping and delivery activities into the same mobile resource (the shopper) and involves planning an operation having two order consolidation levels: pickup at stores and routing.

An important feature of the personal shopper service is that shopping and delivery requests continuously come in over time in parallel to the execution of the delivery operation. This links it to the growing body of literature on dynamic vehicle routing problems (Pillac et al., 2013; Ulmer et al., 2020). Recently, work on dynamic routing problems has focused on same-day and on-demand delivery services, *e.g.*, (Arslan et al., 2019; Arslan, 2019; Klapp et al., 2018a,b, 2020; Voccia et al., 2019; Ulmer et al., 2018). Also, it has to dynamically adapt delivery plans to newly revealed requests considering that the system may benefit from consolidation opportunities with pending visits. For example, assigning an order with pickup and delivery locations close to already committed visits may only lead to a marginal increase in route duration.

Similar consolidation problems have also been studied in the context of order picking operations in warehouses (De Koster et al., 2007; Van Gils et al., 2018). Here, this is often referred to as 'order batching' (Henn et al., 2012; Yu and De Koster, 2009). The main difference is that while order pickers have fixed start and end points and move around in simple and structured warehouse lay-outs, the personal shopper has to navigate between different stores and customer locations. This means that the underlying routing problems are more challenging in our personal shopper context.

The concept of 'zone picking' (Jane and Laih, 2005) for warehouse order pickers is also related to the PSP. In zone picking, each picker is responsible for a dedicated picking zone, while each order may be associated to multiple zones and split between multiple pickers. The PSP has some similarities with zone picking and may be seen as a dynamic version in which zones are assigned in real time.

If the service of orders can be split into two or more partial deliveries, the personal shopper operation also shares some features of the Split Delivery Vehicle Routing Problem in which it is allowed to split the service of each customer into multiple visits (Archetti and Speranza, 2008) and the Vehicle Routing Problem with Divisible Deliveries and Pickups (Nagy et al., 2015) in which customers receive and send goods and pickups and deliveries can be performed by different vehicles. Archetti et al. (2008) shows in a setting with vehicle capacity restrictions, that it is possible to reduce vehicle miles by visiting the same customer with multiple vehicles. For

	Single pickup per request	Multiple pickups per request	Multiple pickups per request with pickup consolidation
Static requests	Yildiz&Savelbergh, 2019	Escudero, 1988 Guerriero & Mancini, 2003 Naccache et al, 2018 Aziez et al, 2020	
Dynamic requests	Pillac et al, 2013 Ulmer et al, 2020 Klapp et al, 2018a,b Voccia et al, 2109 Reyes et al, 2018 Ulmer et al, 2021	Steever et al 2019	**This work**

Fig. 1. Literature classification.

the Vehicle Routing Problem with Divisible Deliveries and Pickups, Jargalsaikhan et al. (2021) show that multiple visits to the same customer can be advantageous even if vehicle capacity is not binding. Similar benefits arise in the context of Pickup and Delivery problems (Nowak et al., 2008, 2009).

While these vehicle routing variants focus on obtaining 'routing' benefits from allowing split deliveries, our work is the first to integrate routing decisions into an order pickup model that also exploits time savings when consolidating pickups of multiple requests per location. This feature gives rise to economies of scale in pickup operations and brings out new trade-offs in terms of which activities to split and which ones to combine that have not been explored in the extend literature.

As the PSP considers a service in which customer requests require products to be picked at multiple locations, it is related to the multi-pickup and delivery problem (MPDP), see Naccache et al. (2018) and Aziez et al. (2020). Conceptually, this also relates the PSP to the sequential ordering problem (SOP), that aims to find a minimum weight Hamiltonian path, subject to precedence relationships among the nodes (Escudero, 1988; Guerriero and Mancini, 2003). The PSP also has some similarities with on-demand meal delivery routing problems (MD) (Reyes et al., 2018; Steever et al., 2019; Ulmer et al., 2021; Yildiz and Savelsbergh, 2019). However, to ensure warm and fresh food upon arrival, meal delivery operations dictate very short in-transit times. The difference in the objective severally limits the length of the delivery routes and thus consolidation opportunities. From a methodological perspective, the tight delivery deadlines result in simpler routing and dispatching decisions as compared to the personal shopper context. Steever et al. (2019) consider a dynamic meal delivery problem with potential splitting of requests from multiple restaurants. Different from our paper, they focus on a setting with soft time windows without in-store shopping related to the request pickups. Their experiments suggest that splitting is not that beneficial in their specific setting.

Fig. 1 structures the most closely related literature across several dimensions, *i.e.*, (i) dynamic versus static requests and (ii) the number and characteristics of the pickup operations. Compared to dynamic (same-day) pickup and delivery problems, the PSP involves multiple pickups per request. In contrast to the multi-pickup and delivery problems, the PSP considers a setting with dynamic requests and pickup time-saving opportunities by consolidating the pickup of multiple tasks per store. In addition, it allows for a single order to be served by multiple shoppers.

3. Personal shopper problem

In this section, we present the personal shopper problem and represent it as a Markov Decision Process model. Moreover, we discuss a rolling horizon framework to compute dynamic policies.

3.1. Problem description

A personal shopper service offers an online platform where customers can shop for products at a set of local stores M. A set of customer requests $R := \{1, \dots, n_R\}$ arrive during an operating period of T time units. Request $r \in R$ arrives at time $e_r \in (0, T)$ and consists of a set O_r of shopping tasks to be executed at one or more stores and delivered to a customer location N_r before time $l_r := e_r + L$. Deadline L is the maximum 'click-to-door' time.

When a customer request arrives, the platform immediately decides whether to offer service or not and if so to accept the corresponding order. If r is offered service, then each task $o \in O_r$ must be purchased at a specific store $m_o \in M$ and dropped off at N_r before l_r. The service area is defined by a directed graph $\mathcal{G} = (\mathcal{N}, \mathcal{A})$, where $\mathcal{N} = \{N_0 \cdots, N_n\} \cup M$ and $\mathcal{A}$ represent the set of all customer locations and arcs, respectively. Also, we assume a known travel time function t_a for each $a \in A$.

The personal shopper service coordinates a set $K = \{1, \dots, n_K\}$ of personal *shoppers*, who are assumed to be homogeneous delivery employees. Each shopper $k \in K$ executes shopping and delivery tasks as instructed by the platform in real time, carries at most Q tasks at a time and starts and ends each shift at a base store $m \in M$.

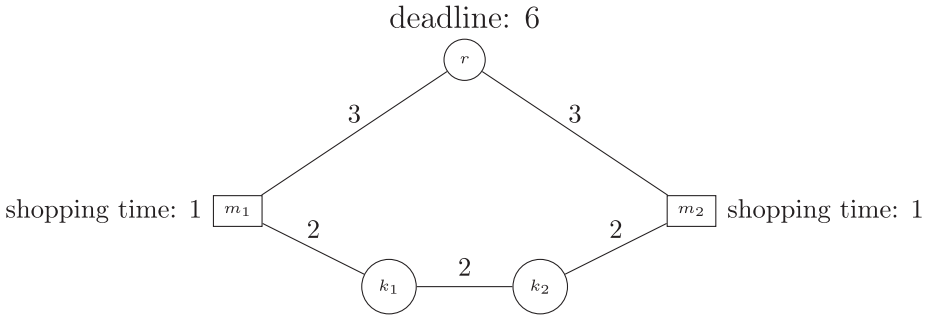

Fig. 2. Example of a personal shopper service instance.

The shopping operations at the store involve picking goods from the store shelves and paying for them at the checkout. The time it takes shoppers to perform these tasks depends on different factors, such as store size and layout, the number of products per order, time spent in parking and waiting for the check-out. To simplify notation, we assume that the total shopping time at the store is solely determined by the number of tasks. There will likely be economies of scale when a shopper can shop multiple tasks of different customer orders in one single store visit. To represent these task consolidation economies at stores, we expect that the total time spent in a store visit to be increasing and concave as a function of the number of tasks.

The shopper service dynamically assigns complete orders or separate tasks to shoppers. An order is said to be 'split' if it is associated with multiple tasks that are assigned to multiple shoppers. Based on the order assignment decisions, the shopper service maintains a *delivery plan* defining the trips of the shoppers, i.e., the sequence of store and customer visits, arrival and departure times to and from those visiting locations.

Splitting orders into tasks served by different shoppers, can have several benefits. It may help to increase the time and capacity utilization, which is particularly relevant when shoppers have limited capacity and tight delivery deadlines. We refer to this as the *packing benefit*. Fig. 2 illustrates an example in which a customer request r demands delivery from two stores m_1 and m_2. Two shoppers are located at k_1 and k_2. Travel and shopping times are displayed above the arcs and store nodes, respectively. Suppose that service must occur within six time units. An on-time service is infeasible when one shopper must serve the order alone. However, if the order is split into two separate store-tasks, then both shoppers can deliver to the customer by $t = 6$.

When splitting is allowed, the platform also has a larger set of feasible routing options, i.e., the *routing benefit*. The example in Fig. 2 shows routing time reductions as a single shopper must travel at least 11 time units to serve r, while two shoppers spend 10 time units. Also, one could save shopping time if multiple tasks originating from a common store are assigned to a single shopper, which we call *shopping consolidation benefit*.

In this work, we assume that the shopper service aims to maximize the total number of orders served at the end of the operating period assuming a fixed fleet of shoppers and provided that it will immediately offer service to any customer request as long as it finds a feasible option of adapting the current route plan to include.

3.2. Markov decision process model

In this section, we formulate the PSP as a sequential decision making problem. We adapt the route-based Markov Decision Problem terminology introduced by Ulmer et al. (2020) to a pure online problem setting without *a priori* knowledge of future customer request information. In the following, we define the model components:

Decision epochs. A decision epoch occurs at the arrival of request $r \in R$ at time e_r.

System states. The system state S_{e_r} at each decision epoch e_r is defined by a delivery plan $\Lambda_{e_r} = \{\lambda_1, \ldots, \lambda_{n_K}\}$ detailing the delivery routes to serve all orders including the planned arrival and departure times at the different locations, and which tasks will be shopped at the store locations. At each decision epoch e_r, a shopper $k \in K$, is either positioned at a store (shopping) or customer location (delivery) in $j \in \mathcal{N}$ or, alternatively, traveling between locations. We assume that shoppers cannot deviate from their route after they have started moving towards the next service location. In particular, shoppers are only available to receive updates to their planned trip after completing the activity that they started, *i.e.*, shopping at a store or delivery at a customer. The moment from which a shopper's plan can be modified is specified within the system state as $e_r^k \geq e_r$; it represents the earliest time after e_r that shopper k is available at the corresponding location.

We consider all tasks that are not yet completed by decision epoch e_r as *active* tasks. That is, we can ignore all tasks that are already completed or that are part of a rejected request. An active task is *committed* to shopper k, if this shopper will complete the shopping activity for this task before e_r^k.

Actions and rewards. An action is an update of the delivery plan that tries to feasibly serve all active orders and new request r. This update can (re)assign uncommitted active tasks across shoppers and (re)sequence all active tasks within shopper trips as long as it feasibly serves all active orders.

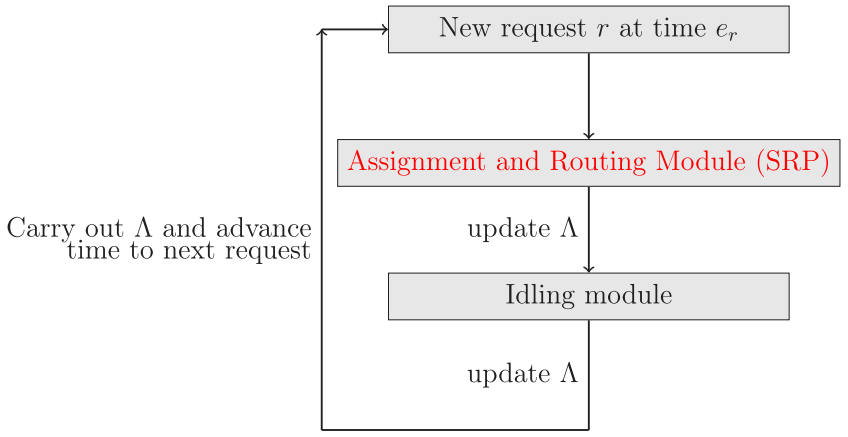

Fig. 3. Rolling horizon framework.

If such a new plan exists, then it updates the delivery plan into $S^x_{e_r}$ (*i.e.*, a post-decision plan); otherwise, service is not offered to request r, and the delivery plan remains unaltered; *i.e.*, $S^x_{e_r} := S_{e_r}$. Let $\mathcal{X}^S_{e_r}$ be the action space at the decision epoch e_r and state S. $\mathcal{X}^S_{e_r}$ is defined by all feasible delivery plan updates.

As long as $\mathcal{X}^S_{e_r} \neq \emptyset$, the platform can receive a unit reward I_{e_r} defined as

$$I_{e_r}(x) = \begin{cases} 1, & \text{if } x \in \mathcal{X}^S_{e_r} \\ 0, & \text{o/w} \end{cases}$$

Transitions. After action x is taken at time e_r, the system jumps to the next decision time at time e_{r+1}. The post-decision delivery plan S^x_r will transition to state S_{r+1}; *i.e.*, all previously executed shopping and delivery tasks are removed from the delivery plan. Also, each shopper earliest available time is updated to e^k_{r+1} according to the new decision time e_{r+1}.

Objective. The objective of the personal shopper platform is to maximize the total number of requests served

$$\max \sum_{r \in R} (I_{e_r}(x(S_{e_r}))). \tag{1}$$

3.3. Rolling horizon framework

Due to the large state and action space of the PSP, enumerating all solutions is computationally prohibitive. Also, an optimal solution to the PSP may not exist in the deterministic sense due to the online nature of the problem. Therefore, we propose a rolling horizon framework that solves a deterministic routing problem each time a new customer request becomes available. Such a model, may be understood as a snapshot solution approach that reacts to the new information disclosed at a decision point. This approach is commonly used in the literature to model similar dynamic delivery operations, see Arslan et al. (2019), Srour et al. (2018).

Fig. 3 presents the outline of our rolling horizon framework. At each decision epoch e_r the rolling horizon framework carries the system's **state** S_{e_r}, which tracks the set of active tasks and the current delivery plan Λ_{e_r}. When a new request r arrives, the delivery plan Λ is revised and updated to include the shopping and delivery tasks within r. To perform this update, we solve a static *Shopper Routing Problem* (SRP) to identify a feasible delivery plan covering both newly disclosed and active tasks; *i.e.*, the SRP acts as task assignment and shopper routing module.

As our overall objective is to maximize the number of served customer orders, the routing subproblem basically serves as a feasibility check. However, to also steer towards 'efficient' solutions, we choose to minimize the total time required to serve all orders in the SRP subproblem. This helps increasing shopper utilization and thus contributes to a higher service level. We discuss the details of the operational strategies and the SRP in Section 4.

Furthermore, the platform dynamically decides if idle shoppers must be relocated calling an *Idling module* after the SRP is solved. In the idling module, we assume that each shopper k leaves a location as soon as possible if there exists subsequent visits scheduled in trip λ_k. Otherwise, the platform employs a *nearest store* strategy, *i.e.*, relocates the shopper to the closest store. In preliminary experiments, we also tested different shopper waiting strategies, but none of them produced time savings. It seems that waiting strategies in this setting are not required. Probably, because the platform can always update a plan, even if a shopper is en-route. The resulting delivery plan is executed and time is advanced to the next decision epoch.

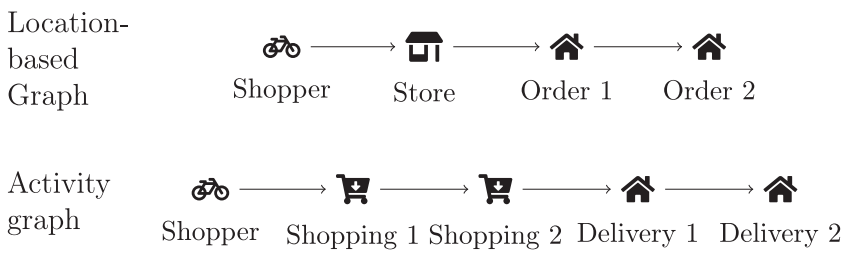

Fig. 4. Activity Representation: A shopper does shopping for Order 1 and Order 2 at the same store. Shopping and delivery activities are represented by unique vertexes in below.

4. Shopper routing problem

In this section, we present the *Shopper Routing Problem* (SRP), which is called as a subroutine within the Personal Shopper Problem to update the delivery plan. We formulate the SRP as a set partition problem and present exact and heuristic solution approaches. Here, we consider the most flexible operational strategy, *i.e.*, 'Consolidation & Split ($C\&S$)'. This allows to split orders into different store tasks that can be served by different shoppers in parallel. In Section 4.4, we present some simpler benchmark strategies.

4.1. Problem formulation: Shopper routing problem

The SRP is defined over an activity graph $G(S_{e_r}) = (V, E)$ depending on the system state S_{e_r}. The set of nodes V includes all active shopping and delivery activities for active tasks by time e_r including the new tasks of request r, denoted by set B_{e_r}. The set E represents the edges between nodes.

A customer order can consist of different tasks, where each task requires a shopping activity at a store node and a delivery activity at its customer node. As the same location may be visited multiple times, we define an *activity* graph in which each node represents a store or customer delivery activity associated with a certain task. An example of an activity graph is presented in Fig. 4, and the formal definition of the activity representation and arc cost (c_{ij}) is given in Appendix.

In the activity graph, shopper k *trip* is modeled as a sequence of activities (represented by a unique node) $\lambda_k = \{v_k, v_1, v_2, \dots, v_h\}$ starting from the current shopper location v_k. A shopper trip is feasible if it meets shopper carrying capacity and satisfies the delivery deadlines of associated task. The total time required for all activities in trip λ is denoted by c_λ.

Let $\Lambda(k)$ be the set of all feasible trips for shopper k and $\Lambda'(o)$ be the set of shopper trips in which task $o \in B_{e_r}$ is served. Furthermore, let binary decision variable w_λ take value 1 if λ is chosen. Then, the set partitioning formulation for the SRP is written as follows:

$$\min \quad z_{SRP} = \sum_{k \in K} \sum_{\lambda \in \Lambda(k)} c_\lambda w_\lambda \tag{2}$$

s.t.

$$\sum_{k \in k} \sum_{\lambda \in \Lambda'(o) \cap \Lambda(k)} w_\lambda = 1, \quad \forall o \in B_{e_r} \tag{3}$$

$$\sum_{\lambda \in \Lambda(k)} w_\lambda \leq 1, \quad \forall k \in K \tag{4}$$

$$w_\lambda \in \{0,1\}, \quad \forall k \in K, \forall \lambda \in \Lambda(k).$$

The objective function (2) minimizes the total trip time required to fulfill all active tasks. Constraints (3) ensure that each task is served by a shopper trip, and constraints (4) guarantee that each shopper is assigned to at most one trip. The chosen operating strategy will determine the set of feasible trips $\Lambda(k)$.

4.2. Exact approach: A smart enumeration

This section describes an exact solution approach for the Shopper Routing Problem capable of solving small instances.. In particular, we enumerate all feasible and promising shopper trips at each decision epoch. To do so, we create all activity-to-shopper partitions and subsequently determine minimum duration trip sequences for each shopper. Let $p = \{P_1, \dots, P_{n_K}\}$ be a partition of the set of nodes (activities) in V in the auxiliary graph $G(S_{e_r})$, where $P_k \subset V$ is the set of tasks assigned to shopper k. Let $\mathbb{P}$ be the collection of all feasible partitions. The partition p is feasible if and only if there exist a feasible trip for each shopper $k \in K$ covering all nodes in P_k. For each shopper k, we solve a Hamiltonian path problem with Precedence Constraints and Deadlines to identify a trip λ_k serving all nodes in P_k.So, the set of optimal trips for all shoppers in partition p, if exists, corresponds to an optimal delivery plan for partition p.

To identify which partition p minimizes the total service time, we explore all partitions sequentially, but prune some unattractive partitions *a priori* via bounding rules similar to a Branch and Bound tree search. At any moment, we store the best partition found and its objective value as an upper-bound to the optimal value, *i.e.*, an incumbent solution. If the partial cost of a subset of already optimized shopper trips within partition p already exceeds the incumbent solution cost, then we discard the partition and jump to the next one. The approach terminates when all partitions are checked.

Each Hamiltonian path problem is solved by a forward labeling algorithm, similar to the one proposed by Tilk and Irnich (2016). In this method, partial shopper trips are constructed and recursively extended via a resource extension function.

Let y_i be a partial trip from a source node; *e.g.*, shopper's k current location at e_r^k, to a node $i \in P_k$ with label (Y, v, i), where $Y \subset P_k$ is the ordered subset of visited nodes in the partial trip, v is the trip's duration and i is the last node in y_i. The initial label for shopper k is $(Y = \{\omega_k\}, 0, \omega_k)$, where ω_k is shopper k starting location. A label extends to node $j \in P_k$ and produces a new label $(Y \cup \{j\}, v + c_{ij}, j)$. An extended (partial) trip can be infeasible, feasible but dominated, or non-dominated. We only keep non-dominated partial trips.

Algorithm 1 describes the forward labeling algorithm for shopper k over tasks P_k. Let $\mathcal{Y}_l$ be a collection of partial trips with $l = |Y| - 1$ arcs. Subroutine $FeasibilityCheck()$ evaluates whether or not an extension of the partial path to node j meets delivery deadlines and the capacity limit of the shopper. It is feasible to extend a partial trip if it is possible to reach node j before its deadline. Routine $BoundingCheck()$ checks if the total duration of the partial plan for Partition P_k exceeds the incumbent's solution, and if so, stops the algorithm and prunes the partition.

Algorithm 1 Forward labeling algorithm

```
Set 𝒴_0 = {Y = {ω_k}, 0, ω_k}
for l = 1 ··· |P_k| − 2 do
    for y ∈ 𝒴_l with label ((Y, v, i)) do
        for j ∈ P_k, j ∉ Y do v' ← v + c_ij
            if FeasibilityCheck(v') then
                if BoundingCheck(v') then
                    Add y_j to 𝒴_{l+1} with label (Y ∪ {j}, v', j)
                end if
            end if
        end for
    end for
    DominanceRule(𝒴_{l+1}),
end for
Find a path y with label (P_k, v, ·) such that c is minimal.
```

We eliminate partial trips according to the following dominance rule:

Definition 1. Let y and y' be two partial trips for the same shopper with labels (Y, v, i) and (Y', v', i). Partial trip y dominates y' if $Y' \subset Y$ and $v < v'$.

We can further improve the performance of the forward label algorithm. Recall that as a result of activity graph $G(S_{e_r})$, a shopper may consecutively visit multiple nodes associated to the same physical location; such a structure leads to multiple node partial sequences representing the same output. We exploit this fact in Observation 1 to reduce the search space.

Observation 1. *Let $\lambda = \{i_0^-, i_1^+, Y, i_1^-, \ldots\}$ and $\lambda' = \{i_0^-, i_1^+, Y', i_1^-, \ldots\}$ be trips such that both contain the same node sequences except partial sub-sequences Y and Y'. Also, let Y and Y' be two different permutations of a given set of nodes in $G(S_{e_r})$ representing activities within the same location. Then, λ and λ' have the same trip duration.*

Observation 1 allows us to eliminate redundant partial paths by forcing a lexicographical order for nodes corresponding to the same location without loss of generality.

4.3. Heuristic approach: ALNS

For larger instances, it is impossible or time-consuming to optimally solve the SRP. Therefore, we propose a fast solution approach that consists of the following steps: (i) Generate multiple initial feasible solutions and (ii) Improve each initial solution by adaptive large neighborhood search (ALNS). From the set of final solutions, we select the solution that minimizes our objective value.

To generate an initial solution, we use a randomized variant of cheapest insertion. That is, we sequentially insert the activities corresponding to the different tasks at the best position of the shopper route plan. At each step, we randomly insert a task from a set of potential candidates with low insertion costs. More precisely, the candidate set consists of tasks with an insertion cost not exceeding $c_{min} + \tau \cdot (c_{max} - c_{min})$, where c_{max} and c_{min} are the maximum and minimum insertion costs, respectively.

Each initial solution is later improved by an adaptive large neighborhood search. The neighborhoods are defined by the following destroy and repair operators.

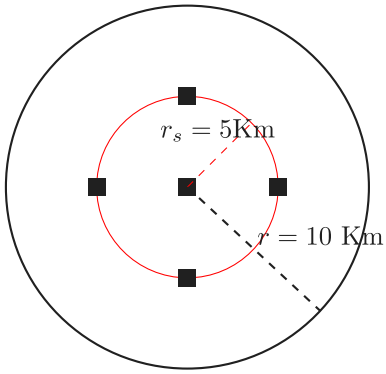

Fig. 5. Layout of the service region in our experiments.

Destroy operators. For a given number of n_s^a uncommitted tasks and n_r^a orders in the incumbent solution, the following five operators removes activity nodes according to criteria that defines. We set a noise rate $\eta \in [0, 1]$ that controls the amount of tasks and orders will be removed from the solution at each iteration; *i.e.*, $q_s = \eta \cdot n_s^a$ tasks or $q_r = \eta \cdot n_r^a$ orders.

- *Random task removal.* Randomly remove the activity nodes of q_s uncommitted tasks.
- *Random order removal.* Randomly remove the activity nodes of q_r orders.
- *Most time-consuming task removal.* Remove the activity nodes of q_s most time-consuming tasks from the current solution. The time-consumption of a task is defined as the time savings generated if the task is ejected from the trip.
- *Most time-consuming order removal.* Remove the activity nodes of q_r most time-consuming orders.
- *Shaw Removal.* Randomly remove associated activities of a task o, and then also remove the $q_s - 1$ closest tasks to s in terms of the euclidean distance.

Repair operators Removed tasks are sequentially inserted back into the delivery plan based on the following operators: (i) Cheapest insertion: tasks are inserted from low to high insertion cost. (ii) Non-split insertion: All removed tasks from the same order are inserted into the trip of one shopper. (iii) Regret insertion. Tasks are inserted in decreasing order of regret, defined as the difference between the cheapest and the second cheapest insertion cost.

Within the ALNS framework, the incumbent solution is destroyed and repaired by a pair of above operators until the stopping criteria is met. This is a common approach in the literature (Ropke and Pisinger, 2006; Pisinger and Ropke, 2010) After each ALNS iteration, we use local search to further improve the different shopper trips, i.e., by applying *2opt* and *1-point* moves.

The destroy and repair operators are selected according to their weights and a *roulette wheel selection principle*. The weights are continuously updated based on their performance. That is, our procedure increases the weight of an operator after an iteration if it finds (i) the overall best solution, or (ii) a solution that has not been visited before. A solution is accepted if it is better than the incumbent solution. Also, we set a 20 s time limit for the iterations to make our method suitable for our dynamic setting.

4.4. Benchmark strategies

In this section, we discuss two simpler benchmark strategies to assess the relative benefits of our flexible $C\&S$ approach.

- One by one ($1B1$): This strategy assumes that each shopper serves one customer order at a time, *i.e.*, a shopper cannot start shopping for a new order before delivering the previous one. It emulates the pick-by-order strategy in warehouse order picking operations.
- Consolidation (C): This strategy allows to consolidate multiple shopping tasks from different orders before delivering these in each shopper trip. Nonetheless, all tasks of a single order must be served in one single delivery visit.

Both the exact approach and heuristic can be easily adapted to simpler operational strategies such as C and $1B1$. These only limit the set of feasible trips and partitions and therefore simplify the search space.

5. Computational study

In this section, we present computational experiments to assess the quality of our solution approach and explore the potential benefits of different operational strategies for the Personal Shopper Service. Next, we describe the experimental setting and instances for the base-case experiments.

Table 1
Base case parameters.

Number of arriving requests	100
Shopper speed	30 km/h
Fixed time per store visit f	9 min
Variable shopping time per task p	1 min
Lead-time	120 min
Shopper capacity	10 tasks
Fleet Size	8 Shoppers

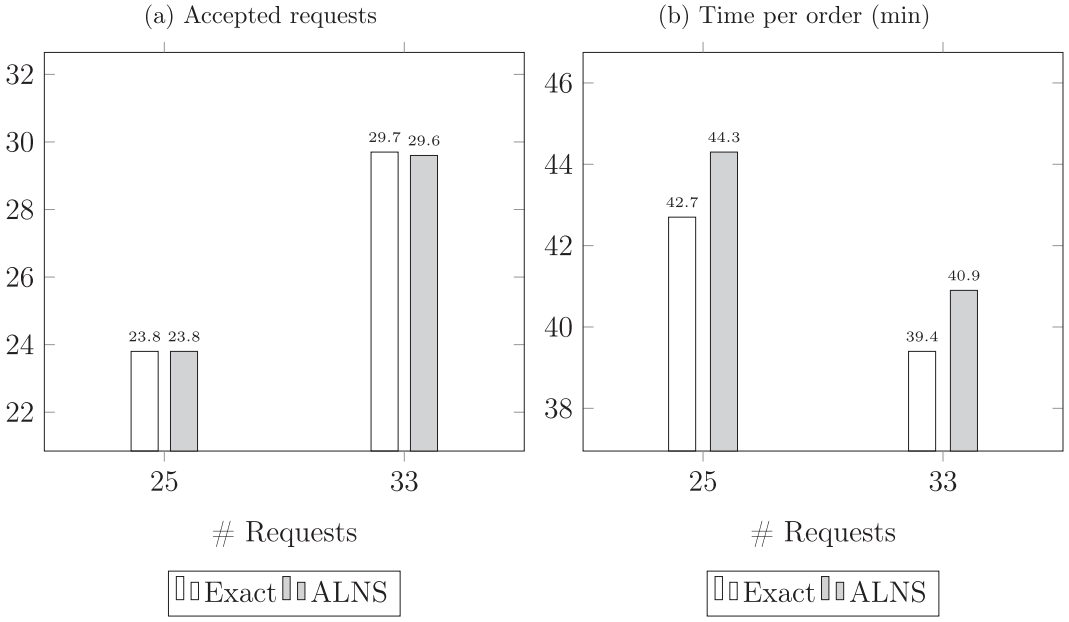

Fig. 6. Heuristic performance **2 shoppers, L:120 min**.

5.1. Experimental setup

For our experiments, we designed a set of instances inspired by the geographical location of grocery retailer stores in the metropolitan area of Amsterdam (the Netherlands). Thai (2016) empirically observes that stores are concentrated towards the city center. The distribution of stores like Amsterdam is very common in many urban areas around the world (Roig-Tierno et al., 2013; Han et al., 2019). Thus, we define a circular service area with a 10 km radius and five stores: one in the center and four located at equal intervals of an inner circle with a 5 km radius (see Fig. 5). The customers are uniformly spread throughout the service region.

The base case experiment considers instances with 100 customer requests that arrive over a ten hour service period. We consider exponential request inter-arrival times and, thus, realizations are uniformly distributed within the service period.

Each accepted request results in an order that requires to be shopped at one or more store locations. In particular, we assume the number of tasks per request to be uniformly distributed between 1 and 4. To simplify the interpretation of the results, we also assume that a shopping task is standard in size and volume, and that each shopper can carry up to $Q = 10$ of them at the same time. Shoppers are assumed to travel at a speed of 30 km/h.

At each store, we consider a shopping time $c_m(n) := f + p \cdot n$, meaning that a store visit spends a fixed time of $f = 9$ min and a variable shopping time of $p = 1$ minute per task. Table 1 summarizes all parameter values used in the base case experiment.

5.2. Heuristic solution quality

In this section, we evaluate the performance of our heuristic solution approach for the Shopper Routing Problem (SRP). In particular, we apply our exact approach (Section 4.2) and our ALNS heuristic approach (Section 4.3) to multiple snapshot problems for the most complex assignment and routing operating strategy: $C\&S$. We ran experiments for ten random streams of 25 requests arrivals (63 tasks on average) and 33 requests arrivals (average of 83 tasks). The requests are randomly chosen from the base instances as described in the previous section. Fig. 6 shows the number of accepted requests and time per order per SRP solution method.

The results suggest that the ALNS heuristic performs well. That is, it leads to the same number of accepted orders as our exact approach for the smaller instances (25) and only slightly less (i.e., 4%) for the larger instances (33). Preliminary experiments show

Table 2
Average base case results, **100 requests, L: 120 min, 8 shoppers**.

	DIY	$1B1$	C	$C\&S$
Accepted requests	100.0	81.4	97.5	100.0
Time per order (min)	71.4	57.1	36.0	31.8
Shopping time per order (min)	25.2	21.7	13.2	8.9
Travel time per order (min)	46.3	35.4	22.8	22.9
Orders per shopper(#)	NA	10.2	12.2	12.5
Click-to-door (min)	71.4	92.2	93.2	92.2
Orders split (%)	NA	NA	NA	54.0
Delivery interval (min)	NA	NA	NA	36.7

that simpler heuristics, i.e., cheapest insertion, perform worse than our ALNS approach. This suggests that good routing is indeed important.

5.3. Base-case results

Table 2 presents the results for the proposed strategies averaged over 100 random streams of 100 request arrivals. The platform has a fleet of eight shoppers. We report the following performance indicators: (i) *Accepted requests*: The number of requests that are served. (ii)*Time per order:* total shopper time over the number of orders. We further break this value down into two: *Shopping time per order* and *Travel time per order*. (iii) *Orders per shopper*: the number of orders over the available number of shoppers. (iv) *Click to door (CtD)*: the average time between an order's arrival and the delivery of its latest task. (v) *Delivery interval*: the average time between the earliest and latest delivery of a split order, measured as the sum of all time intervals divided by the total number of split orders. (vi) *Orders split*: the percentage of split orders over all orders.

The results show that we can consistently serve more requests for a given fleet in the *Consolidation* (C) and *Consolidation and Split* ($C\&S$) strategies as compared to the *One by one* ($1B1$) strategy. While we can serve all request using the $C\&S$ strategy, we can only serve 81.4% of the requests, on average, when using the simple $1B1$ strategy.

The main reason for this better performance is that consolidating shopper trips significantly decreases the time required to serve an order. For strategy C, the time per order is 37% smaller than for $1B1$, due to savings in both shopping and travel time. Splitting allows for additional time savings (11.6%) as it enables more consolidation opportunities by reducing task granularity and increasing the *packing benefits*. Moreover, we also observe a smaller shopping time per order, which indicates *shopping consolidation benefits*. Conversely, there is a slight increase in travel time per order, *i.e.*, from 22.8 to 22.9. This relates to the fact that some customers are now visited by multiple shoppers.

We observe that the average click-to-door time is similar for the different operating strategies, i.e., between 92 and 93 min. Nevertheless, for $C\&S$, we see that on average 54.0% of the orders are split and thus served by multiple shoppers. For these orders, the average time between the first partial delivery and the final partial delivery, *i.e.*, the delivery interval, is 36.7 min.

Besides our operational strategies described in Section 4.4, we generate a setting in which customers carry out shopping activities by themselves, referred to as the Do-it-Yourself (DIY) benchmark. Here, we assume that each customer chooses the fastest route to visit all stores that are part of the shopping request. Also, we assume that each customer leaves her home at the time when she would otherwise place a delivery request online.

The results show that personal shopper services can provide substantial savings as compared to the DIY benchmark. Even for the simple $1B1$ policy, we see savings of 20.0% and 23.6% in the total time and travel time spent, respectively. These savings are possible as personal shoppers can efficiently link the start and end points of consecutive single customer trips. Note that this efficiency gain only leads to slightly higher click-to-door times for $1B1$ as compared to DIY, *i.e.*, , 71.4 versus 92.2 min.

In the following, we evaluate the sensitivity of the three operating strategies: 1b1, C, $C\&S$ to a fleet size number varying between 5 and 19.

Fig. 7 shows the number of accepted requests for the different strategies averaged over 100 random streams of request arrivals. The results show that both the *Consolidation* and *Consolidation and Split* strategies significantly outperform the *One to one* policy. For example, with 7 shoppers and equipped with the *C* and *C&S* strategies we can already serve 94.4% and 99% of the requests, respectively. Instead, only 75.3% is served via *1B1*.

As we see from the previous experiments, each strategy requires a different number of shoppers to serve all requests. In the remaining set of experiments, we are interested in comparing our strategies under similar service levels. So, we will use the smallest fleet size required to serve all customer requests for each instance and operating strategy.

To do so, we run the simulation of our dynamic system and increase the number of shoppers each time that we would have to reject a request. For each instance, we keep track of the total number of required shoppers and report the average number of required shoppers for each strategy.

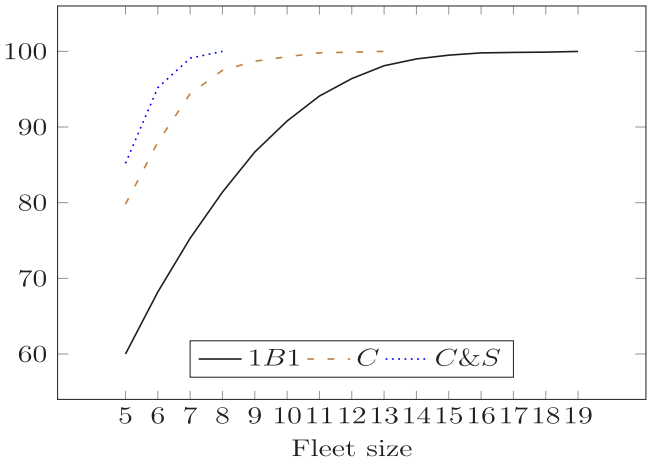

Fig. 7. Accepted requests, **100 requests, L:120 min**.

5.4. Impact of shopping economies of scale α

In this section, we study the impact of the relative weight of the fixed part of the shopping time on the performance of the different proposed strategies. To do so, we set the fixed and variable shopping time parameters equal to $f = 10 \cdot ha$ and $p = 10 \cdot (1 - ha)$, respectively, and vary parameter $ha \in [0.5, 1]$. A value of $\alpha = 0.5$ indicates that the fixed visit duration for a store visit is equivalent to the pickup duration per task. This represents the case with the economies of scale is relatively minimum. Conversely, a value of $\alpha = 1$ represents a situation with the highest possible economies of scale, in which store shopping times are fixed and independent of the number of tasks shopped at the store. Also, the total store shopping time per visit is independent of α if no shopping consolidation occurs.

Fig. 8 presents the performance for the different values of α over all strategies. Fig. 8(a) reports that the average time per order decreases with α for strategies C and $C\&S$. This is intuitive as the value of consolidating, in terms of economies of scale for shopping, increase with α. As expected, α does not affect the performance of the $1B1$ strategy as it does not allow consolidations.

Fig. 8(b) also shows that for strategies with consolidation (C and $C\&S$), the total travel time per order decreases with α as each shopper visits less stores. Furthermore, we see in terms of travel time $C\&S$ is less reactive to α, shopping consolidation than the policy C is.

Both Figs. 8(a) and 8(b) present the major benefit of order splits in shopping time. With consolidation (C and $C\&S$), the average shopping time per order decreases as α increases. By simultaneously shopping multiple tasks in a store, it is possible to reduce the shopping time per task; this reduction is higher when orders can be split.

Similarly, we observe in Fig. 8(c) that for strategies C, and $C\&S$ the required fleet size decreases with α. This is a consequence of the decreasing time per order associated with the consolidation and split strategies. Fig. 8(d) shows that splitting increases with α. This is intuitive as the potential shopping benefits increase with α which makes it more beneficial to split.

5.5. Impact of the number of tasks per request

To study the impact of the number of tasks per request in isolation, we consider instances in which all requests have the same number of tasks. Recall that each task corresponds to a shopping activity at a specific store and its delivery. To allow for comparison, we use the same stream of requests in each of the different scenarios, only varying the number of tasks per request.

Our results are summarized in Fig. 9. Fig. 9(a) shows the time per order for each strategy and a varying number of tasks per order. As expected, the time per order increases with the number of tasks. However, we observe that the increase is less steep for strategies C and $C\&S$ as compared to DIY and $1B1$.

The reason for this is that these strategies allow shoppers to combine multiple tasks that are associated with a specific store. $C\&S$ has more opportunities to do this than C, which leads to additional benefits.

Fig. 9(b) presents the travel time per order for each strategy and a varying number of tasks per order. In line with the time per order, the travel time per order grows with the number of tasks. Here, we see not much difference between C and $C\&S$, which suggests that most benefits of splitting are in terms of shopping time.

Fig. 9(c) reports the average number of shoppers required to serve all orders throughout the day. As expected, the number of shoppers increases with the number of tasks per request for all strategies. However, we observe that the increase is less pronounced for the strategies with consolidation (C and $C\&S$) as compared to the simple $1B1$.

The box plots in Fig. 8(d) insights into the solutions for the $C\&S$ strategy as it shows that the number of orders with at least one split increases with the number of tasks per order. These results are intuitive as the number of splitting options increases with the number of tasks. We see that the majority of the orders are split in the instances with four tasks.

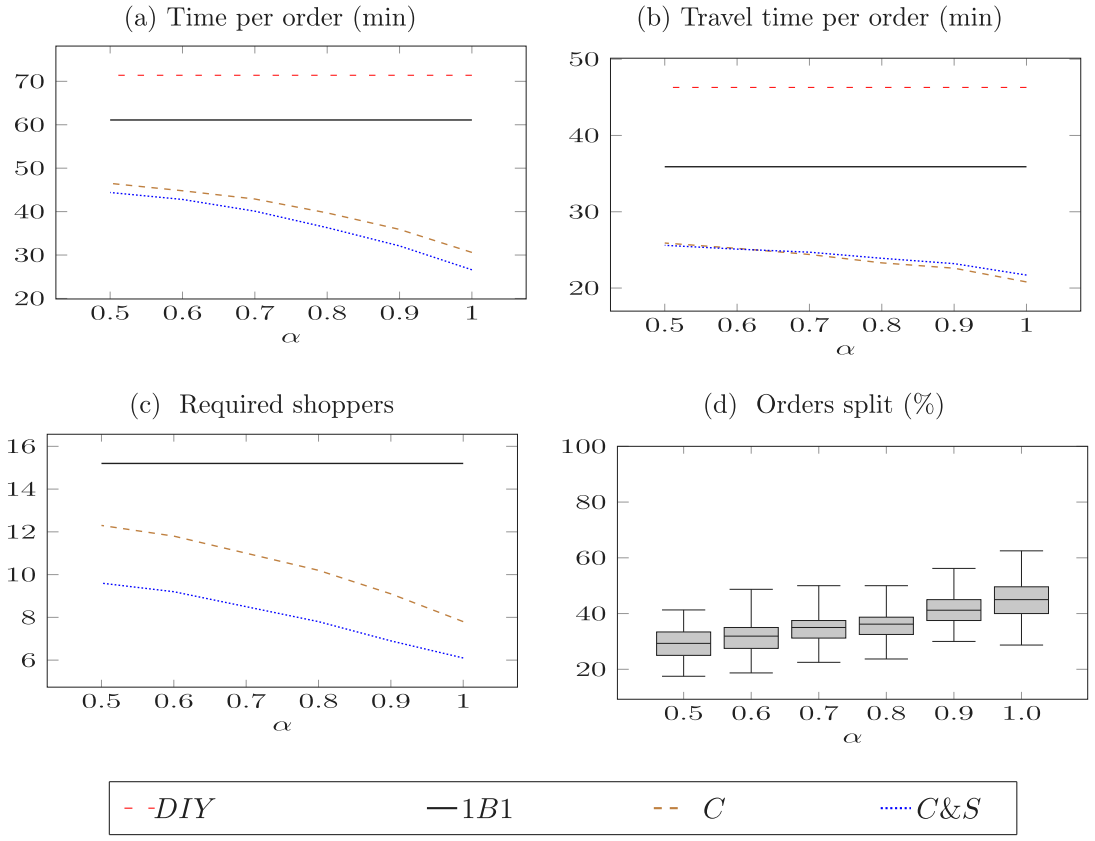

Fig. 8. Impact of shopping consolidation factor α, **100 orders, L: 120 min**.

5.6. Impact of order split preferences

In practice, not all customers may accept split deliveries as it may be perceived as inconvenient. Therefore, the retailer may allow customers to specify their delivery preferences. Amazon, for example, allows customers to specify their preferences to 'receive as few deliveries as possible' or 'I want my items sooner'.

In the next set of experiments, we study the impact of these preferences by generating instances in which we cannot split the deliveries for all customers. To create these instances, we introduce parameter $\rho \in [0, 1]$ that representing the expected fraction of customers that agrees to a split delivery. For example, $\rho = 0.2$ means that the likelihood that any given customer will accept a split is 20%. Or, in other words, that on average across instances 20% of the customers allow splits.

Fig. 10 reports the time per order and the required number of shoppers for different ρ values. Note that $\rho = 0$ corresponds to the consolidation strategy without splitting and $\rho = 1$ corresponds to the earlier $C\&S$ results without split restrictions. As expected, the performance of the system decreases with ρ. That is, we see that we require more shoppers (and more time per order) to serve all customers when some orders cannot be split. This performance impact appears to be proportional in ρ.

In Section 5.3, we saw that, on average, 54% of the orders were split in the base case in which all orders can be split. In this experiment, we see that the number of required shoppers increases even when 60%–80% of the customers allows splitting. This suggest that it is not just about the fraction of customer that allow splitting but also about which specific customer can be split.

5.7. Impact of lead-time, store and customer density

In this section, we study the impact of varying the customer densities, store densities and delivery lead-time. Instead of locating all stores on a circle, in this set of experiments we explore uniformly distributing different numbers of stores across the service region.

Fig. 11 presents the results for 5, 10 and 20 different stores. In particular, we present the time per order and the number of number of shoppers for each operational strategies. We observe that the store density has little impact on the performance of the DIY benchmark and the $1B1$ strategy. However, an increasing number of stores in the system (and store density) deteriorates strategies with consolidation (C and $C\&S$). That is, Fig. 11a shows roughly a 10 min increase in the time per order for both consolidation strategies when comparing the 20 store setting to the 5 store stetting. Similarly, we see that the number of required

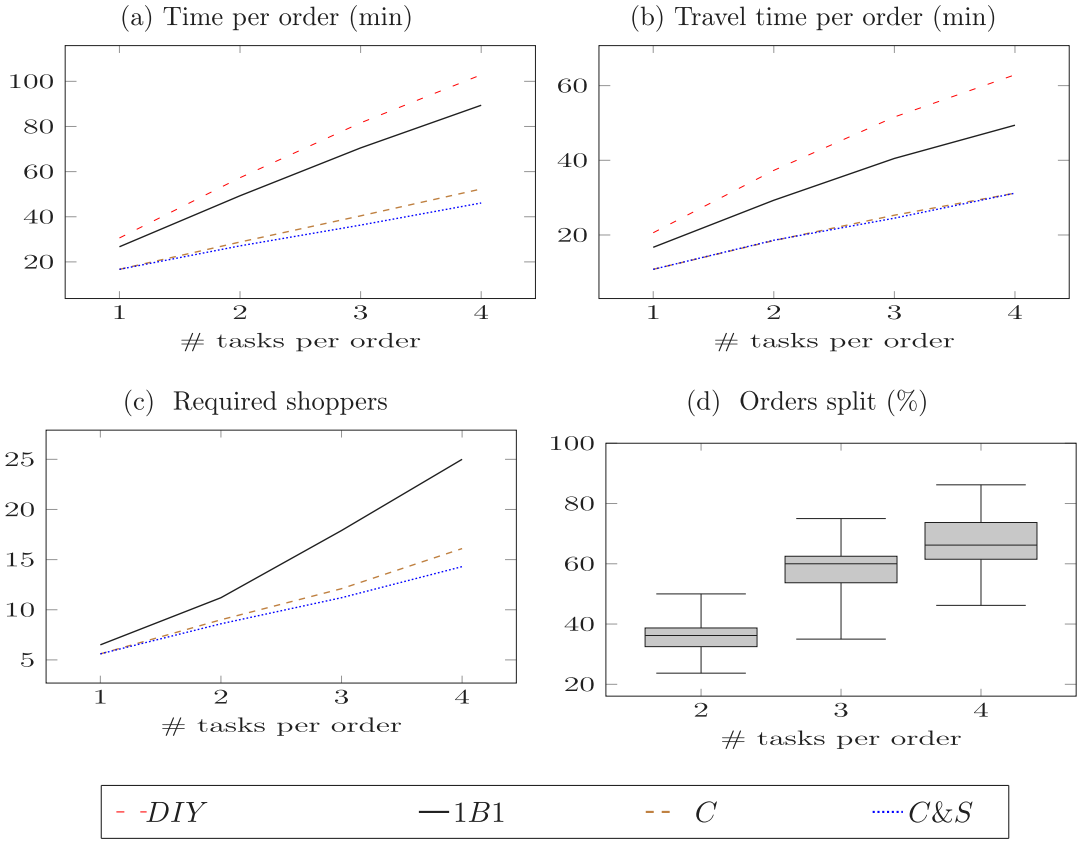

Fig. 9. Impact of request size, **100 orders,** α : 0.9**, L: 120 min**.

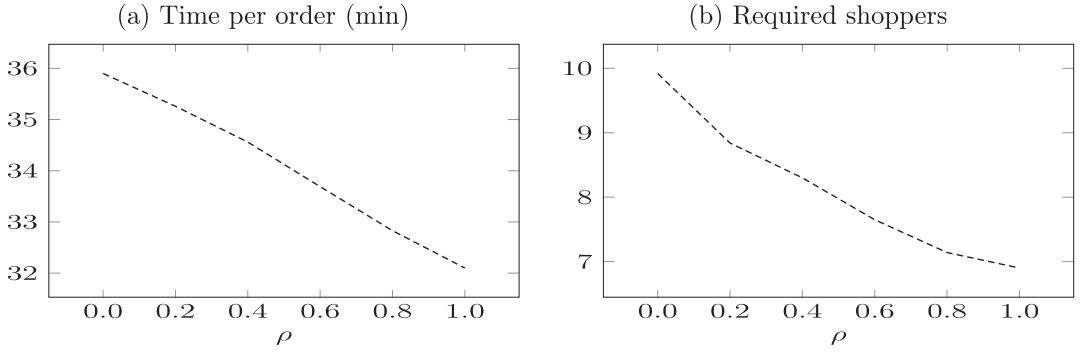

Fig. 10. Impact of delivery preferences ρ, **100 orders, L: 120 min**.

shoppers increases in Fig. 11b. With more stores it is less likely to have two orders sharing the same store and thus less possibilities to consolidate orders.

In the next set of experiments, we extend the lead-time from 120 min (base case) to 150 and 180 min. Fig. 12 reports average service time per order and orders per shoppers throughout the service period. As expected, we observe that the performance of strategies having consolidation options increases as lead-time increases. That is, the time per order decreases for strategies C and $C\&S$ with a higher lead-time possibly because more time is available to consolidate shopping orders at stores and while routing. Interestingly, a longer lead-time also benefits $1B1$, as stated in Fig. 12b. The reason for this is that we can better utilize the existing shopper time over the day with a more flexible routing operation. In the same vein, we see that strategies C and $C\&S$ benefit even

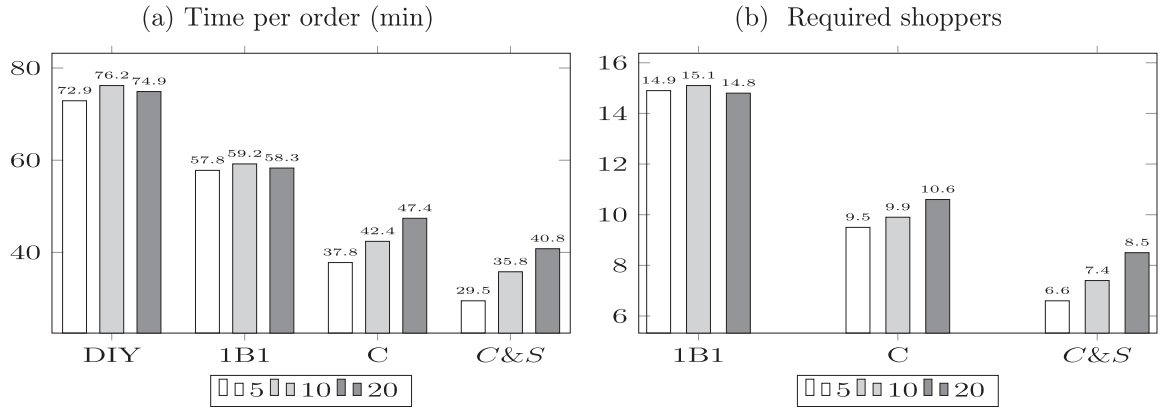

Fig. 11. Impact of store numbers, **100 orders, L:120 min**.

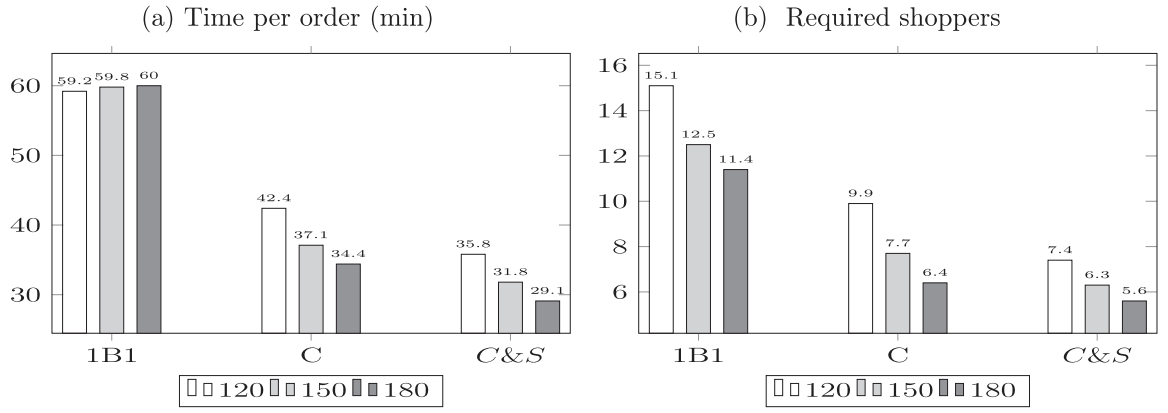

Fig. 12. Impact of lead-time, **100 orders, 10 stores**.

more in terms of orders per shopper from having more time flexibility. This is related to the additional packing and routing benefits that can be achieved.

Fig. 13 shows the performance for 100, 200 and 400 customer requests. We see that the performance in terms of the time per order and the number of required number of shoppers increase with the number of orders. A relatively larger number of requests creates more consolidation opportunities. This means that the strategies with consolidation benefit more from the increased customer density. While we only see minor benefits for $1B1$, strategies C and $C\&S$ do significantly better. For example, doubling the number of requests from 100 to 200 leads to only 1.7% less time per order for $1B1$ but 18.3% less time for $C\&S$. A similar pattern is observed in the required number of shoppers. If no shopping consolidation is allowed, the growth rate of required fleet size is approximately proportional to the number of requests. Nevertheless, the required fleet size grows relatively slower for strategies C and $C\&S$. The reason for this is that these strategies can exploit shopping benefits in addition to the packing and routing benefits.

6. Concluding remarks

The paper focuses on the assignment and routing strategies for the personal shopper service that provides on-demand delivery from local brick and mortar stores. The service provider uses a fleet of shoppers to serve all customer requests while aiming at maximizing customer service. We model the problem as a sequential decision problem and present a rolling horizon approach to solve it. Particularly, we present and evaluate three operational strategies for the personal shopper service with varying levels of order consolidation and service splitting flexibility.

We have conducted an extensive computational study to compare the performance of the different operational strategies and also benchmark against a setting in which customers go to the store to shop themselves. Our results provide the following insights: (i) Personal shoppers can save time and resources over customers shopping by themselves, even when shoppers serve customers one by one. (ii) Shopping and travel times per order are significantly further reduced by consolidating requests in shopper trips. (iii) It is possible to further enhance the performance of the system by splitting requests that require shopping at multiple stores.

Numerically, the benefits of shopping consolidation significantly differ with the instance setting. These time savings increase as a function of the fixed shopping time, delivery lead-time and the number of requests over the day, and decrease with the number of

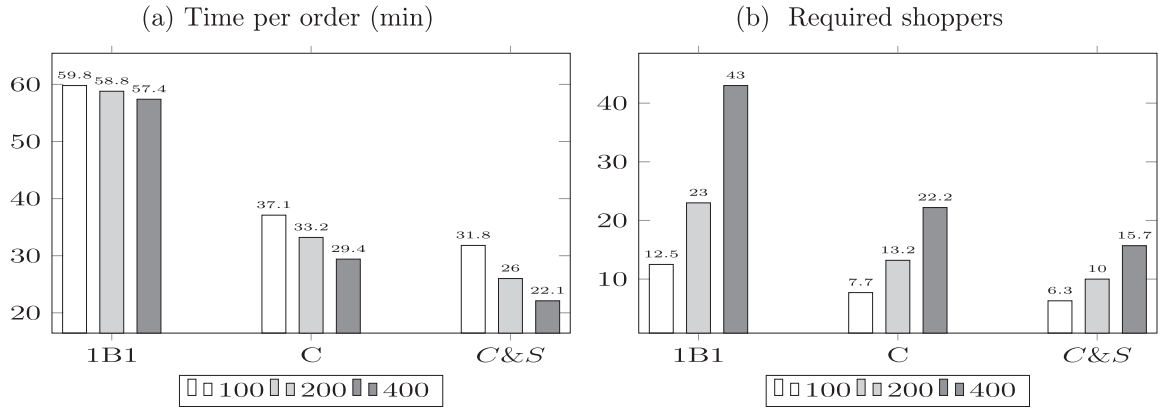

Fig. 13. Impact of customer density, **10 stores, lead-time: 150 mins**.

stores. These consolidation benefits are partially obtained by splitting requests, which is particularly important when fixed shopping times, number of tasks per request or delivery lead-times are relatively higher.

As this is one of the first papers that studies the personal shopper business model, there are many avenues for future research. One natural extension is to consider a setting in which we plan decisions considering probabilistic information on the placement, location and size of future customer requests. Such information could help us to estimate the expected cost of present decisions in future states of the PS service operation. It can be valuable knowledge when the volume of future request realizations per unit of time changes significantly throughout the day with a considerable fraction of predictable stationary. This would be the case of consolidated PS services with well-known demand patterns.

Another interesting research avenue involves incorporating both splits and transfers between personal shoppers. This means allowing transfers of shopped tasks between shoppers to consolidate delivery operations and enhance customer service while utilizing the part of request split benefits. In addition, one could study the concept of multiple pickups and splits in the context of meal delivery.

CRediT authorship contribution statement

Alp M. Arslan: Term, Conceptualization, Methodology, Software, Writing – original draft, Writing – review & editing, Visualization. **Niels Agatz:** Term, Conceptualization, Methodology, Writing – original draft, Writing - review & editing. **Mathias A. Klapp:** Term, Conceptualization, Methodology, Writing - original draft, Writing - review & editing.

Acknowledgment

Mathias Klapp thanks ANID (Agencia Nacional de Investigación y Desarrollo), Chile, for its support through the FONDECYT Iniciación Project number 11190392.

Appendix. Activity graph representation

Fig. 14 illustrates the how we can derive an activity graph from a location-based graph with an example with one shopper, two stores and two orders. Order 1 consists of one task that involves shopping at store m and delivering to customer d_1. Request 2 consists of two tasks, requiring shopping from store m and m' respectively and delivering to customer d_2. The path displayed above the line, physically represents a shopper trip visiting stores (rectangles) and customer locations (pentagon). Below the line, that shopper trip is represented in an activity graph, where each task shopping operation and each task delivery is represented by a node. Here, s_1^{1+} represents shopping task 1 within order 1 from store m, s_1^{2+} represents shopping task 1 within order 2 from store m, and s_2^{2+} represents shopping task 2 within order 2 from store m'. Analogously, s_1^{2-}, s_2^{2-} and s_1^{1-} represent task deliveries to customer locations.

In the activity graph, the cost of an arc consists of multiple components (*e.g.*, parking time, task picking time, and, travel time between different locations.), as this value depends on the source and sink nodes of the arc.

In Eq. (5), we formally define arc costs on the activity graph, defined in Section 3. Let t_{ij} be the average travel time from node i and j. We incur travel time only if node i and j maps to two physical location. The logical operator $i \overset{\text{L}}{\neq} j$ checks whether i and j are two different physical locations. In addition we can integrate the total shopping duration at store m into arc cost as follows:

$$c_{ij} := \mathbb{1}_{\left(i \overset{\text{L}}{\neq} j\right)} t_{ij} + \mathbb{1}_{(j \in S_m^+)} \vartriangle c_m(n), \tag{5}$$

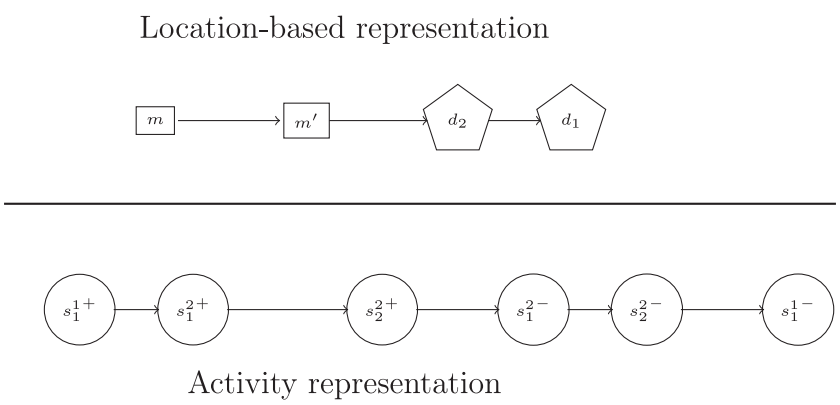

Fig. 14. Location-based and activity graphs for a particular shopper's route. Pentagons and squares indicate customer and store locations, respectively. Circles represent shopping and delivery nodes.

where $\triangle c_m(n) := c_m(n) - c_m(n-1), \forall n \geq 1$, represent the incremental shopping time for additional task picking. Notice that $\triangle c_m(1)$ considers the fixed cost for visiting store m. Also, $\mathbb{1}_{(\cdot)}$ is an indicator function that takes value one if the statement in $(\cdot)$ is true, and S_m^+ denotes set of shopping task at store m.

References

Archetti, C., Savelsbergh, M.W., Speranza, M.G., 2008. To split or not to split: That is the question. Transp. Res. Part E Logist. Transp. Rev. 44 (1), 114–123.

Archetti, C., Speranza, M.G., 2008. The split delivery vehicle routing problem: a survey. In: The Vehicle Routing Problem: Latest Advances and New Challenges. Springer, pp. 103–122.

Arslan, A., 2019. Operational Strategies for on-Demand Delivery Services. EPS-2019-481-LIS.

Arslan, A.M., Agatz, N., Kroon, L., Zuidwijk, R., 2019. Crowdsourced delivery: A dynamic pickup and delivery problem with ad hoc drivers. Transp. Sci. 53 (1), 222–235.

Aziez, I., Côté, J.-F., Coelho, L.C., 2020. Exact algorithms for the multi-pickup and delivery problem with time windows. European J. Oper. Res. 284 (3), 906–919.

Bishop, D., 2020. August 2020 scorecard: Online grocery market rebalancing after covid spike. https://www.brickmeetsclick.com/august-2020-scorecard--online-grocery-market-rebalancing-after-covid-spike.

De Koster, R., Le-Duc, T., Roodbergen, K.J., 2007. Design and control of warehouse order picking: A literature review. European J. Oper. Res. 182 (2), 481–501.

Escudero, L.F., 1988. An inexact algorithm for the sequential ordering problem. European J. Oper. Res. 37 (2), 236–249.

Guerriero, F., Mancini, M., 2003. A cooperative parallel rollout algorithm for the sequential ordering problem. Parallel Comput. 29 (5), 663–677.

Gunday, G., Kooij, S., Moulton, J., Karabon, M., Omeñaca, J., 2020. How European Shoppers Will Buy Groceries in the Next Normal. McKinsey & Company, https://www.mckinsey.com/industries/retail/our-insights/how-european-shoppers-will-buy-groceries-in-the-next-normal.

Han, Z., Cui, C., Miao, C., Wang, H., Chen, X., 2019. Identifying spatial patterns of retail stores in road network structure. Sustainability 11 (17), 4539.

Henn, S., Koch, S., Wäscher, G., 2012. Order batching in order picking warehouses: a survey of solution approaches. In: Warehousing in the Global Supply Chain. Springer, pp. 105–137.

Jane, C.-C., Laih, Y.-W., 2005. A clustering algorithm for item assignment in a synchronized zone order picking system. European J. Oper. Res. 166 (2), 489–496.

Jargalsaikhan, B., Romeijnders, W., Roodbergen, K.J., 2021. A compact arc-based ILP formulation for the pickup and delivery problem with divisible pickups and deliveries. Transp. Sci..

Klapp, M.A., Erera, A.L., Toriello, A., 2018a. The dynamic dispatch waves problem for same-day delivery. European J. Oper. Res. 271 (2), 519–534.

Klapp, M.A., Erera, A.L., Toriello, A., 2018b. The one-dimensional dynamic dispatch waves problem. Transp. Sci. 52 (2), 402–415.

Klapp, M.A., Erera, A.L., Toriello, A., 2020. Request acceptance in same-day delivery. Transp. Res. E Logist. Transp. Rev. 143, 102083. http://dx.doi.org/10.1016/j.tre.2020.102083.

Naccache, S., Côté, J.-F., Coelho, L.C., 2018. The multi-pickup and delivery problem with time windows. European J. Oper. Res. 269 (1), 353–362.

Nagy, G., Wassan, N.A., Speranza, M.G., Archetti, C., 2015. The vehicle routing problem with divisible deliveries and pickups. Transp. Sci. 49 (2), 271–294.

Nesin, B., 2020. Will customer stick with online grocery the post-covid world of US online grocery growth. Rabobank https://research.rabobank.com/far/en/sectors/beverages/will-consumers-stick-with-online-grocery.html.

Nowak, M., Ergun, Ö., White III, C.C., 2008. Pickup and delivery with split loads. Transp. Sci. 42 (1), 32–43.

Nowak, M., Ergun, O., White III, C.C., 2009. An empirical study on the benefit of split loads with the pickup and delivery problem. European J. Oper. Res. 198 (3), 734–740.

Pillac, V., Gendreau, M., Guéret, C., Medaglia, A.L., 2013. A review of dynamic vehicle routing problems. European J. Oper. Res. 225 (1), 1–11.

Pisinger, D., Ropke, S., 2010. Large neighborhood search. In: Handbook of Metaheuristics. Springer, pp. 399–419.

Reyes, D., Erera, A., Savelsbergh, M., Sahasrabudhe, S., ONeil, R., 2018. The meal delivery routing problem. Optimization Online.

Roig-Tierno, N., Baviera-Puig, A., Buitrago-Vera, J., Mas-Verdu, F., 2013. The retail site location decision process using GIS and the analytical hierarchy process. Appl. Geogr. 40, 191–198.

Ropke, S., Pisinger, D., 2006. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. Transp. Sci. 40 (4), 455–472.

Savelsbergh, M.W., Sol, M., 1995. The general pickup and delivery problem. Transp. Sci. 29 (1), 17–29.

Srour, F.J., Agatz, N., Oppen, J., 2018. Strategies for handling temporal uncertainty in pickup and delivery problems with time windows. Transp. Sci. 52 (1), 3–19.

Steever, Z., Karwan, M., Murray, C., 2019. Dynamic courier routing for a food delivery service. Comput. Oper. Res. 107, 173–188.

Thai, N.V.Q., 2016. Spatial Distribution of Food Retailers in Amsterdam. University of Amsterdam, https://scripties.uba.uva.nl/download?fid=643908.

Tilk, C., Irnich, S., 2016. Dynamic programming for the minimum tour duration problem. Transp. Sci. 51 (2), 549–565.

Ulmer, M.W., Goodson, J.C., Mattfeld, D.C., Thomas, B.W., 2020. On modeling stochastic dynamic vehicle routing problems. EURO J. Transp. Logist. 100008.
Ulmer, M.W., Mattfeld, D.C., Köster, F., 2018. Budgeting time for dynamic vehicle routing with stochastic customer requests. Transp. Sci. 52 (1), 20–37.
Ulmer, M.W., Thomas, B.W., Campbell, A.M., Woyak, N., 2021. The restaurant meal delivery problem: Dynamic pickup and delivery with deadlines and random ready times. Transp. Sci. 55 (1), 75–100.
Van Gils, T., Ramaekers, K., Caris, A., de Koster, R.B., 2018. Designing efficient order picking systems by combining planning problems: State-of-the-art classification and review. European J. Oper. Res. 267 (1), 1–15.
Voccia, S.A., Campbell, A.M., Thomas, B.W., 2019. The same-day delivery problem for online purchases. Transp. Sci. 53 (1), 167–184.
Yildiz, B., Savelsbergh, M., 2019. Provably high-quality solutions for the meal delivery routing problem. Transp. Sci. 53 (5), 1372–1388.
Yu, M., De Koster, R.B., 2009. The impact of order batching and picking area zoning on order picking system performance. European J. Oper. Res. 198 (2), 480–490.