European Journal of Operational Research 316 (2024) 295–309

Contents lists available at ScienceDirect

European Journal of Operational Research

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

Innovative Applications of O.R.

Multi-period time window assignment for attended home delivery

Jean-François Côté [a,*], Renata Mansini [b], Alice Raffaele [c]

[a] CIRRELT, Université Laval, 2325 Rue de la Terrasse, Québec, G1V 0A6, Canada
[b] Department of Information Engineering, University of Brescia, Via Branze 38, Brescia, 25123, Italy
[c] Department of Mechanical, Energy and Management Engineering, University of Calabria, Via Pietro Bucci – Cubo 46/C, Arcavacata di Rende, 87036, Italy

ARTICLE INFO

Keywords:
Transportation
Time window assignment
Vehicle routing problem
Stochastic programming
Sample average approximation method

ABSTRACT

We study a multi-period stochastic variant of the Time Window Assignment Vehicle Routing Problem, where customers' demands, locations, and service times are uncertain. Customers are partitioned into geographical zones, each of which has to be visited a predetermined number of times over a planning period of several days. Whenever a zone is visited, a time window is assigned. Time windows are decided before knowing customers and their demands. A fleet of homogeneous vehicles is available to serve customers each day. At a tactical level, the problem looks for a static time window assignment that minimizes the expected traveling costs plus the expected penalty costs for unserved customers. We propose a two-stage formulation and a solution approach, which relies on the Sample Average Approximation Method, while encompassing a perturbation method to assign time windows in the first stage and an Adaptive Large Neighborhood Search to optimize routes in the second stage. We experimentally evaluate three instance sets, including real ones from a Canadian company, comparing our results to lower bounds from the exact solution of a deterministic equivalent formulation over a finite number of scenarios. Our method outperforms the manual approach used by the company.

1. Introduction

Time is critical for last-mile delivery. Companies optimize costs while ensuring on-time delivery to meet customer demands. An effective routing schedule is a priority, though optimal schedules are complex to build without complete information on customers' location and demand. Unfortunately, in many real applications, such information is not available in advance, but companies may have historical data to produce useful statistics Thus, finding an optimal schedule is complex, and the goal turns into looking for a schedule that works quite well on every occasion.

In some applications, customers explicitly select preferred delivery windows and are willing to pay more to get them, while others like furniture buyers mainly care about the service day and are less sensitive to precise time windows. In this paper, we analyze the problem encountered by a Canadian retailer that sells and delivers furniture and appliances in the regions centered around Edmonton and Calgary (about 460 customers served per week in each region). The company finds it reasonable and convenient to visit customers living in the same neighborhood on the same day, possibly avoiding going back to the same place at another moment. For this reason, the company has classified customers based on their residential zone (zip code). Currently, the company provides its customers with weekly schedules that specify the days of the week when the service can be accomplished in each zone. Once the purchase is concluded, customers have the flexibility to select a delivery date from the available options. Two days before the chosen delivery date, customers receive an automated phone call, informing them about the designated delivery time window, typically spanning three hours on the selected day.

The company would like to improve customer satisfaction by partially modifying the service. One of the most unsatisfactory aspects related to service is that customers cannot select their delivery time window. Thus, the company aims to define a new service policy where a predefined number of time windows are assigned to each zone in advance so that customers can express their preferences. Improving the service policy is complicated by a demand that, according to historical data, undergoes strong variations with customers varying positively or negatively by over 100 units week to week. The new service will be organized over a defined *time horizon* (typically a week). To meet a high service level, the company aims to have multiple visits to the same zone throughout the planning period, with the condition that each zone is visited no more than once per day. Striking the right balance is crucial. The company seeks to have a sufficient number of visits to enhance customer satisfaction while avoiding an excessively high frequency of visits that could fragment customer requests and drive up travel expenses. To determine this ideal number, the company intends to leverage historical data about prior purchases and deliveries

* Corresponding author.
E-mail address: jean-francois.cote@fsa.ulaval.ca (J.-F. Côté).

https://doi.org/10.1016/j.ejor.2024.01.021
Received 18 January 2023; Accepted 18 January 2024
Available online 20 January 2024

in each zone. Under the revised service policy, unlike before, customers will decide their delivery day by evaluating Under the revised service policy, unlike before, customers will decide their delivery day by evaluating the available combinations of delivery days and time windows offered for their specific zone when they place an order. In more detail, customers will rank the available delivery days based on their preferences. Once the time windows are allocated to each zone, customers will be served on their most preferred delivery day, assuming that a time window was assigned on that day.

Currently, the company manually decides delivery schedules by using maps and push pins with a process requiring the work of several employees for many working days. For this reason, the created schedule is usually kept for months. This causes serious inconveniences, making the schedule insensitive to changes in demand: a strong demand growth usually causes a delay in schedule planning, thus reducing customer satisfaction, whereas a demand decrease could make the current planning inefficient, which, in turn, could increase costs. The company wants to take a step forward by replacing the manual planning process with a computer-based one focused on advanced algorithms. This tool will help the company quickly modify the current schedule or even change it every week by overcoming the main drawbacks of the old procedure.

To tackle all these requirements, we define a stochastic multi-period variant of the *Time Window Assignment Vehicle Routing Problem* (TWAVRP) introduced by Spliet and Gabor (2015). Our problem differs from the latter in many aspects. Firstly, in our case, the geographical area of interest is partitioned into zones defined by zip codes. Moreover, time windows are assigned to zones before knowing not only customers' requests but also their locations and service times. The assignment decides the distribution of time windows in the working days of the planning period. If a customer cannot be served by the company's fleet of vehicles, a penalty corresponding to the cost of a backup delivery done in outsourcing is paid. Given the zones into which the geographical area is divided, the *Stochastic Multi-period Time Window Assignment Problem* (SMTWAP) decides how to assign a predefined number of time windows to each zone while minimizing the expected traveling costs required to visit the customers within the allocated time windows, plus expected penalty costs associated with unserved customers. Regarding constraints, the decision maker decides the set of possible time windows for each zone by first establishing how many times each zone can be visited during the planning period using historical data. Past information is provided as probability distributions on the number of customers, their demands, and service times for each zone. These allow the creation of a nearly infinite number of scenarios representing realizations of the described random variables. In each scenario, customers' placing requests are known, along with their preferred days based on the time windows available for their specific zones on those days. A fleet of vehicles is available to make deliveries every day. No cancellation or rescheduling is allowed.

This paper provides the following contributions:

1. *New problem.* The stochastic aspects of this problem pertain to the number of customers, their locations, demands, and service times, all uncertain. Although previous research has examined each of these aspects individually or, at most, in pairs, there has not been a publication that incorporates all of them together while also considering the geographical feature of dealing with multiple zones and the multi-period nature of the time window assignment problem, and the presence of a penalty cost in the case customers are not served by the fleet. To tackle the problem, we introduce a two-stage stochastic formulation with general recourse.
2. *First-stage control of the zone visit frequency*. The delivery company wants to manage the frequency of visits to each zone throughout the planning period. In our approach, the number of visits per zone is thus a known value established by the company and influenced not only by immediate economic considerations but also by other factors that may affect the company's long-term relationship with its customers. In the competitive furniture industry, the company acknowledges that customer satisfaction and loyalty are critical for success. These can be improved by favoring some zones more than others controlling the number of visits per zone.
3. *Centrality of customers.* Although the time windows are assigned to residential zones, the routing problem is based on customer locations rather than just zones, which sets this problem apart from the existing literature.
4. *Second-stage rank-based assignment of customers to days*. When customers make their purchases, they express their preferences by ranking the delivery days in a specific order. Then, when time windows are allocated to the zones, the routing process of the second stage assigns to each customer the time window in his/her highest-ranked day among those available. This aspect sets the problem apart from other time window assignment problems and introduces the concept of prioritizing ranked time windows (as discussed, e.g., by Farias, Jagabathula, & Shah, 2013).
5. *Effective solution methodology.* Given the complexity of the problem, we develop a *Sample Average Approximation Method* (SAAM) to approximate our stochastic program. It works by solving a series of SMTWAP problems, where each problem, called sample average approximating (SAA) problem, uses a small random sample of scenarios and is solved using a meta-heuristic. This enables us to provide solutions that are efficient and effective when compared with the deterministic equivalent formulation solved using CPLEX on small-size instances and substantially improve the ones obtained by the manual approach of the company.

The paper is organized as follows. In Section 2, we discuss the main literature on attended home delivery problems and known variants of TWAVRP by also providing a summarizing table highlighting contributions and research gaps. Section 3 provides the formal description of SMTWAP, its two-stage definition, and its deterministic equivalent formulation. In Section 4, we describe the SAAM-based methodology. Section 5 provides details on the perturbation procedure used to solve the first stage and on an *Adaptive Large Neighborhood Search* (ALNS) framework used to tackle the routing problem in the second stage. In Section 6, first, we outline the design of the computational study, showcase extensive results, and derive some managerial insights, including an analysis of the solution stochastic value and quality of the service. Then, we delve into the results obtained on the real instances provided by the Canadian company. Finally, Section 7 draws the main conclusions.

2. Literature review

The primary challenge in *Attended Home Delivery* (AHD) problems revolves around determining the optimal timing for customer service. To the best of our understanding, one of the earliest studies in this area is due to Madsen, Tosti, and Vælds (1995), where a company offers a dynamic daily repair service. Customers become known only when they call for assistance and the company has to specify a suitable time window to perform the service. Lin and Mahmassani (2002) focus on the influence of customer demand and their spatial distribution on the number of depots and vehicles used, as well as the management of time windows on both tactical and operational levels.

In recent literature, many contributions have been published on both *static* management of time windows (roughly estimating routing efforts, e.g., by imposing a maximum number of requests for each time window) and *dynamic* one (adjusting partial schedules according to time-dependent and stochastic travel time information). The static *Time Slot Management* (TSMP) in AHD, tackled by Agatz, Campbell, Fleischmann, and Savelsbergh (2011), deals with time windows offered in the zip codes (zones) of a service region. Differently from our problem, an expected number of customers is associated with each geographical zone, and only a single time window for each zip code is allocated. Demand is measured in terms of customer orders in a planning period of a shift (a morning or an afternoon), therefore the expected amount for each zip code is known. Time windows cannot overlap among different zones. To minimize the expected delivery costs, the authors use two different approaches a continuous approximation model, to estimate the delivery cost of a given time window schedule and an integer programming model with approximated delivery costs by grouping customers of the same zip code. In both cases, vehicle routes to serve customers over the zones are not explicitly considered.

The TSMP has also been formulated as a *Periodic* VRP (PVRP) in Hernandez, Gendreau, and Potvin (2017), where each zone has a service frequency and demand is known. The goal is to define a schedule by assigning zones to vehicles on any given day of the planning period minimizing total travel costs. The authors propose a Tabu Search by using simple moves, such as the removing a zone from a route in a certain period and the reinsertion in another route.

The *Time Window Assignment Vehicle Routing Problem* (TWAVRP) was defined by Spliet and Gabor (2015), and successively studied by Spliet and Desaulniers (2015), Spliet, Dabia, and Van Woensel (2018), and Dalmeijer and Desaulniers (2021). In these papers, the planning period is a single day, customer locations are known, and only their demand is stochastic. In Spliet and Gabor (2015), the authors establish that time windows can start at any time within a predefined exogenous time window decided by the customer. The TWAVRP is formulated as a two-stage stochastic optimization: in the first stage, a time window is assigned to every customer from the set of possible time windows; in the second stage, after demand is known, vehicle routes are computed. The authors propose a Column Generation algorithm to find lower bounds by solving one pricing problem for every scenario and a branch-and-cut-and-price algorithm based on the separation of different valid inequalities. In the *Discrete TWAVRP* (Spliet & Desaulniers, 2015), for each customer, there is instead a discrete set of candidate time windows from which just one has to be selected. In Dalmeijer and Spliet (2018), the authors present a compact formulation to speed up the previous branch-and-cut (Spliet & Gabor, 2015) and solve larger instances. A novel class of valid inequalities, the *precedence inequalities*, are also discussed. Time-dependent travel times are considered in the variant of TWAVRP presented by Spliet et al. (2018). Finally, in Dalmeijer and Desaulniers (2021), the authors focus on the TWAVRP and propose a branch-and-price-and-cut to eliminate orientation symmetry from the search tree.

Subramanyam, Wang, and Gounaris (2018) propose a scenario decomposition approach to include both continuous and discrete time windows, adapting an algorithm developed for the *Consistent VRP* (see Groër, Golden, & Wasil, 2009) to gain modularity and scalability through a parallel implementation. Hoogeboom, Adulyasak, Dullaert, and Jaillet (2021) formulate a robust version of the TWAVRP, where routes and time window assignments are simultaneously computed to minimize the expected travel time and the risk of violating time windows. Jalilvand, Bashiri, and Nikzad (2021) take into account stochastic service times too but still deterministic locations, whereas Yu, Shen, Badri-Koohi, and Seada (2023) deal with stochastic locations, demands, and service times by considering one delivery area, continuous time windows of varying duration, and cancellations. They both provide a static and a dynamic version of the problem, formulating the former as a two-stage program.

Time window preferences can also be influenced by the application of *incentives*, as proposed originally by Campbell and Savelsbergh (2005), to steer customers to select time windows so as to maximize the overall profit (see, e.g., Campbell & Savelsbergh, 2006). Incentives and differentiated rates are also used to cope with higher demand in the late part of the day, making customers pay extra to get the service as described by Ehmke and Campbell (2014). Furthermore, incentives can be exploited to increase the duration flexibility of time windows without impacting the company's profit. Köhler, Ehmke, and Campbell (2020) consider a mixed set of short and long time windows. Customers arrive in real-time, and, when they reveal their locations, a set of time windows is computed and offered to them. The goal is to maximize the number of accepted customers. In addition to the strategy of just rejecting customers, one can model an outsourced backup service by introducing suitable penalties to measure the corresponding cost (see, e.g., Stenger, Vigo, Enz, & Schwind, 2013). About time window width, Manerba, Mansini, and Zanotti (2018) analyze the negative impact of restrictive time windows on the environment. Agatz, Fan, and Stam (2021) investigate the effect of using green labels to encourage customers to select time windows.

Table 1 presents an overview of the relevant attributes of the SMTWAP alongside the principal aspects of pertinent works in the literature. This allows for a comparative analysis of their main features including the main focus (*Time Window Assignment* (TWA) or AHD); the objective function (e.g., minimize total distance or maximize profit); the problem type (static and/or dynamic), the planning period (one or multiple periods); the geographical extension (one or multiple areas); known or uncertain customer location, demand, service time, and travel times, presence of customer preferences (Yes/No), time windows information (continuous and/or discrete, their duration, hard and/or soft), the use of scenarios, and the method developed (exact and/or heuristic). From this comparison, we observe that no other work in the literature addresses all the features of the SMTWAP. Indeed, we innovate concerning these works since customers' locations, demands, and service times are unknown and since we consider multiple areas. In particular, as in Spliet and Gabor (2015) and Spliet and Desaulniers (2015), we also consider static management of time windows with predefined widths. Moreover, we assume that a discrete set of candidate time windows is available for each zone, but, differently from Spliet and Desaulniers (2015), time windows are not associated with customers but with zones. For this, the SMTWAP may seem similar to the problem in Hernandez et al. (2017) but, in our case, customers' orders are supplied only once during the planning period and thus are not periodic. Also, in Hernandez et al. (2017), the authors do not address the construction of delivery routes based on customers' orders but these are seen as sequences of zones. Regarding the methodology, we take into account multiple service areas and a planning period composed of multiple periods. Finally, we integrate a complete recourse mechanism to get always feasible solutions.

3. Problem formulation

Let $D = \{1, \ldots, \tau\}$ be a discretized planning period of τ working days (each one consisting of a predefined number of working hours) over which the company organizes the delivery service. Customers, who are not known in advance, are spread over a geographical area partitioned into a set $Z = \{1, \ldots, h\}$ of h zones (e.g., zip codes or small areas). Each zone $z \in Z$ has a set of candidate time windows W_z. Each time window w is defined as a tuple (l_w, u_w, d_w, z_w) where l_w and u_w are the starting and ending times, whereas $d_w \in D$ and z_w are the day of the planning period in which the time window is available and its zone, respectively. Based on past experience, the decision maker sets a number n_z of visits for each zone z, corresponding to the selection of an equivalent number of time windows to be scheduled over the planning period. We indicate as $W_z(d) \subseteq W_z$ the subset of candidate time windows of W_z offered by zone z on the day d. At most one window can be selected per zone per day (every zone is visited at most once each day). We call *zone schedule* the assignment of the time windows of a given zone over the planning period, and *global schedule* the set of all zone schedules. A global schedule is *feasible* if it has exactly n_z time windows per each zone z scheduled over the planning period, with at most a time window per day per zone.

Table 1
Comparison between the stochastic time window assignment problem and the main related works.

Authors	Year	Main focus	Objective	Type	Period	Area	Customer location	Customer demand	Service times	Travel times	Time windows	Customer preferences	Scenarios	Method
Madsen et al. (1995)	1995	AHD	Min distance	Dynamic	One	One	Unknown	Unknown	Random (30 or 60 min)	Unknown	Continuous (2 h) Hard	No	No	Heuristic
Lin and Mahmassani (2002)	2002	AHD	Min costs	Static	One	One	Unknown	Unknown	Known	Known	Discrete and continuous Hard	No	Yes	Heuristic
Campbell and Savelsbergh (2005)	2005	AHD	Max profits	Dynamic	One	One	Known	Unknown	N/A	Known	Discrete (1 h) Hard	Yes	Yes	Heuristic
Campbell and Savelsbergh (2006)	2006	AHD	Max profits	Dynamic	One	One	Known	Known	Known	Known	Discrete (1 h) Hard	Yes	No	Heuristic
Agatz et al. (2011)	2011	AHD	Min costs	Static	One	Multiple	Known	Unknown	N/A	Known	Discrete Hard	No	Yes	Heuristic
Ehmke and Campbell (2014)	2014	AHD	Max number of accepted customers	Dynamic	One	Two	Known	Unknown	Known	Unknown	Discrete Hard	Yes	No	Heuristic
Spliet and Gabor (2015)	2015	TWA	Min costs	Static	One	One	Known	Unknown	Known	Known	Continuous Hard	No	Yes	Exact
Spliet and Desaulniers (2015)	2015	TWA	Min costs	Static	One	One	Known	Unknown	Known	Known	Discrete Hard	No	Yes	Exact
Hernandez et al. (2017)	2017	TWA	Min costs	Static	Multiple	Multiple	Known	Known	Known	Known	Discrete Hard	No	N/A	Heuristic
Manerba et al. (2018)	2018	AHD	Min distance	Static	One	One	Known	Known	Known	Known	Discrete Hard	No	No	Exact
Spliet et al. (2018)	2018	TWA	Min costs	Static	One	One	Known	Unknown	Known	Known	Continuous Hard	No	Yes	Exact
Subramanyam et al. (2018)	2018	TWA	Min costs	Static	One	One	Known	Known and unknown	Known	Known	Discrete and continuous Hard	No	Yes	Exact
Köhler et al. (2020)	2020	TWA	Max number of accepted customers	Dynamic	One	One	Known	Unknown	N/A	Known	Discrete (30' or 4 h) Hard	Yes	Yes	Heuristic
Agatz et al. (2021)	2021	TWA	Min distance	Dynamic	One	One	Unknown	Unknown	Known	Unknown	Discrete Hard	Yes	Yes	Heuristic
Dalmeijer and Desaulniers (2021)	2021	TWA	Min costs	Static	One	One	Known	Unknown	Known	Known	Continuous Hard	No	Yes	Exact
Hoogeboom et al. (2021)	2021	TWA	Min costs	Static	One	One	Known	Unknown	Known	Known	Continuous Hard	No	Yes	Exact
Jalilvand et al. (2021)	2021	TWA	Min costs	Static	One	One	Known	Unknown	Unknown	Known	Continuous Hard and soft	Yes	Yes	Exact
Yu et al. (2023)	2023	TWA	Min costs and duration	Static and dynamic	One	One	Unknown	Unknown	Unknown	Unknown	Continuous Hard	No	Yes	Exact + heuristic
This manuscript	2023	**TWA**	**Min costs**	**Static**	**Multiple**	**Multiple**	**Unknown**	**Unknown**	**Unknown**	**Known**	**Discrete Hard**	**Yes**	**Yes**	**Heuristic**

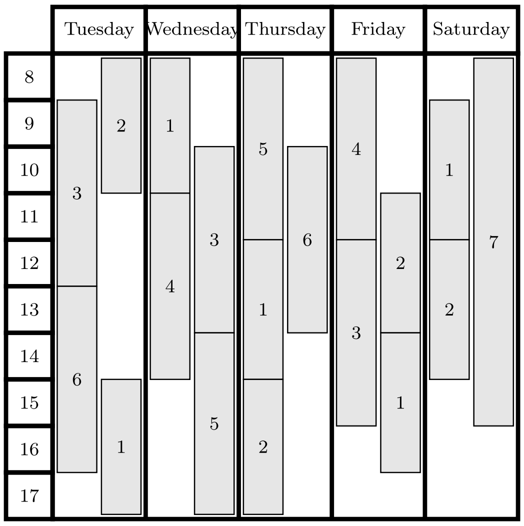

Fig. 1. Example of a global schedule.

Fig. 1 shows an example of a global schedule with 7 zones, with time windows spanning from 8:00 a.m. to 5:00 p.m., over a planning period of 5 days (from Tuesday to Saturday). Zone 1 is visited 5 times within a 3-h time window positioned differently during each day. Zone 2 is visited 4 times and has time windows of 3 h each, whereas Zone 3 is visited on Tuesday, Wednesday, and Friday with time windows of 4 h each. Zones 4, 5, and 6 are visited twice and have time windows of 4 h each. Finally, Zone 7 is visited only on Saturday, within an 8-h time window.

The SMTWAP can be formally defined as follows. Let ξ be a vector of random variables corresponding to customers' locations, demands, and service times. Each realization of ξ is called a *scenario*. We assume that the support of ξ is finite, and we indicate as S the set of all possible scenarios. Each scenario $s \in S$ has a probability p_s to occur. Let $G_s = (V_s, A_s)$ be the complete graph associated with scenario s, where the node set $V_s = N_s \cup \{0, n_s + 1\}$ consists of the set of customers $N_s = \{1, \dots, n_s\}$ making a request under scenario s, plus the starting depot 0 and the ending one n_s+1. In particular, the set N_s is partitioned into h subsets, each one including the customers making part of the same zone. We denote by $N_s^z \subseteq N_s$ the subset of customers belonging to zone $z \in Z$ under scenario s. The transportation cost c_{ij} is incurred if a vehicle traverses the arc $(i,j) \in A_s$. Let also t_{ij} be the travel time to reach customer j from customer i. Each customer $i \in N_s$ asks for q_i units of demand to be delivered on a selected day d, and their service time is b_i. When customers decide to make a purchase, they express their preferences by establishing an order of the available days. More precisely, each customer i sorts the set of available days by assigning an integer o_{id} to every day d in the planning period, with 1 (minimum value) as the first preference. On each day d of the planning period, a number k_d of identical vehicles is available, each having capacity Q, departing at time l_0 and going back to the depot by time u_0. Customers who cannot be accommodated by the company's fleet of vehicles (*unserved customers*) can receive a backup service through a third-party logistic carrier (outsourcing). We denote by β the cost (penalty) incurred when a customer is serviced by a third-party provider. We assume that this cost is substantially greater than the routing expenses, to maximize the use of the company fleet. The SMTWAP aims to define a global schedule such that the expected transportation costs to visit customers and the penalty cost paid for unserved ones are minimized.

The SMTWAP can be formulated as a two-stage stochastic program. Let y_{zw} be a first-stage binary variable indicating if time window $w \in W_z$ is assigned to zone $z \in Z$. The following variables are in the second-stage. In particular, the binary variable z^s_{id} takes value 1 if customer $i \in N_s$ is visited on the day d under scenario s, whereas the binary variable z^s_i is 1 if customer i in the same scenario is served by a third party paying a penalty β. Let x^{ts}_{ij} be a binary variable taking value 1 if arc $(i,j) \in A_s$ is traversed on day d under scenario s. Finally, the continuous variable a^s_i indicates the arrival time at customer i under scenario s, and thus the starting time of the service. A first-stage solution corresponds to a feasible global schedule. A second-stage solution not only offers the routing plan based on an established feasible global schedule but also considers customer preferences, as previously explained, and a list, if any, of unserved customers. The mathematical formulation of the SMTWAP is as follows:

$$\min \sum_{s\in S} p_s \left[\sum_{d\in D} \sum_{(i,j)\in A_s} c_{ij} x^{ts}_{ij} + \beta \sum_{i\in N_s} z^s_i \right] \tag{1}$$

$$\text{s.t.} \quad \sum_{w\in W_z} y_{zw} = n_z \qquad z \in Z \tag{2}$$

$$\sum_{w\in W_z(d)} y_{zw} \le 1 \qquad z \in Z, d \in D \tag{3}$$

$$\sum_{d\in D} z^s_{id} + z^s_i = 1 \qquad s \in S, i \in N_s \tag{4}$$

$$\sum_{j\in V_s} x^{ts}_{ji} = z^s_{id} \qquad d \in D, s \in S, i \in N_s \tag{5}$$

$$\sum_{j\in V_s} x^{ts}_{ij} = \sum_{j\in V_s} x^{ts}_{ji} \qquad d \in D, s \in S, i \in N_s \tag{6}$$

$$\sum_{j\in V_s} x^{ts}_{0j} \le k_d \qquad d \in D, s \in S \tag{7}$$

$$\sum_{i\in H} \sum_{j\in H} x^{ts}_{ij} \le |H| - \left\lceil \frac{\sum_{i\in H} q_i}{Q} \right\rceil \qquad d \in D, s \in S, H \subseteq V_s, |H| \ge 2 \tag{8}$$

$$a^s_j \ge a^s_i + (b_i + t_{ij}) \sum_{d\in D} x^{ts}_{ij} - u_0 \left(1 - \sum_{d\in D} x^{ts}_{ij} \right) \qquad (i,j) \in A_s, s \in S \tag{9}$$

$$a^s_i \le \sum_{w\in W_z(d)} u_w y_{zw} + u_0 (1 - z^s_{id}) \qquad d \in D, s \in S, z \in Z, i \in N^z_s \tag{10}$$

$$a^s_i \ge \sum_{w\in W_z(d)} l_w y_{zw} - u_0 (1 - z^s_{id}) \qquad d \in D, s \in S, z \in Z, i \in N^z_s \tag{11}$$

$$z^s_{id} + z^s_i \ge \sum_{w\in W_z(d)} y_{zw} - \sum_{\substack{\bar{d}\in D \\ o_{i\bar{d}} < o_{id}}} \sum_{w\in W_z(\bar{d})} y_{zw} \qquad d \in D, s \in S, z \in Z, i \in N^z_s \tag{12}$$

$$z^s_i, z^s_{id} \in \{0,1\} \qquad d \in D, s \in S, i \in N_s \tag{13}$$

$$x^{ts}_{ij} \in \{0,1\} \qquad d \in D, s \in S, (i,j) \in A_s \tag{14}$$

$$a^s_i \ge 0 \qquad s \in S, i \in N_s \tag{15}$$

$$y_{zw} \in \{0,1\} \qquad z \in Z, w \in W_z \tag{16}$$

The objective (1) minimizes the expected routing costs over the planning period and the expected total penalty cost for unserved customers. Allowing customers not to be served means we have a *complete recourse*, i.e., a feasible second-stage solution for any first-stage solution. Constraints (2) ensure that, for each zone z, exactly n_z time windows are selected. Constraints (3) assign to each zone z on each day d at most a single time window in $W_z(d)$. Constraints (4) impose that each customer i in scenario s is either served by the fleet or by an external carrier ($z^s_i = 1$). Constraints (5) state that, if node i is visited under scenario s on the day d, an arc has to enter the node. Constraints (6) guarantee flow conservation. Constraints (7) set at most k_d routes on each day d under scenario s. Constraints (8) are subtour-elimination and rounded-capacity constraints. Constraints (9)–(11) guarantee the schedule feasibility w.r.t. the chosen time windows. Constraints (12) ensure that customer i is served on his/her most preferred day d if there is a time window available in his/her zone on such a day and no time window is available on another day $\bar{d}$ with a higher preference ($o_{i\bar{d}} > o_{id}$). Constraints (13)–(16) define the variables domain.

4. Solution methodology

The challenge in addressing the SMTWAP primarily stems from the large number of scenarios involved. The presence of binary variables makes this two-stage stochastic problem impractical even for small instances and nearly impossible for real-life applications (e.g., 25 zones and 30 customers per zone lead to at least 25^{30} scenarios). To manage this complexity, we employed the *Sample Average Approximation Method* (SAAM), a Monte Carlo simulation-based approach proposed by Kleywegt, Shapiro, and Homem-de Mello (2002) and particularly suitable to solve stochastic discrete optimization problems involving a very large scenario set. In our case, SAAM addresses a sequence of SMTWAP problems (the so-called *SAA problems*), where the scenario set of each problem consists of an independent and identically distributed (i.i.d.) random sample $\bar{S}$ consisting of $\xi_1, \xi_2, \dots \xi_{|\bar{S}|}$ realizations of the random vector ξ (cfr. Section 3), over which the expected value of the objective function of the problem is approximated by the sample average function. The random sample has size $|\bar{S}|$ much smaller than that of the original scenario set S, i.e., $|\bar{S}| \ll |S|$. The larger the sample size $\bar{S}$, the better, on expectation, the value of an optimal solution to an SAA problem, and the tighter the bounds on the optimality gap. However, the larger the sample size, the higher the computational complexity. This trade-off makes critical the selection of the size of the sample. Thus, one might choose a small sample $\bar{S}$ to optimally solve the SAA problems efficiently but then estimate more accurately the value of the optimal solution of each SAA problem over a sample $\hat{S}$, with $|\hat{S}| \gg |\bar{S}|$. The goal of the method is twofold: estimating the optimality gap by a statistical lower bound (i.e., the optimal values obtained by solving the SAA problems) and providing an upper bound on the optimal value (by evaluating each solution of the SAA problems on a scenario set $\hat{S}$).

4.1. SAAM application to SMTWAP

To apply SAAM, we refer to model (1)–(16) with the following notation:

$$v^* = \min_{y\in \mathbf{Y}} \mathbb{E}_{s\in S}[C(y,s)], \tag{17}$$

where $\mathbf{Y}$ is the set of first-stage solutions, $C(y,s)$ is the cost of the second stage with the first-stage solution y fixed for scenario s, and v^* is the value of the optimal solution. The evaluation of $\mathbb{E}_{s\in S}[C(y,s)]$ for a given value of y requires the solution of numerous second-stage optimization problems. Since S contains a finite number of scenarios with associated probabilities, the expectation can be evaluated as the following finite sum:

$$\mathbb{E}_{s\in S}[C(y,s)] = \sum_{s\in S} p_s C(y,s) \tag{18}$$

where p_s is the probability for scenario s to occur. Since the number of scenarios grows exponentially fast with the data dimension, applying Eq. (18) directly can be impractical. For this reason, we generate a scenario sample $\bar{S}$ of reasonable size and then solve the following deterministic optimization problem (SAA) specified by the generated sample as follows:

$$v_{\bar{S}} = \min_{y\in \mathbf{Y}} \frac{1}{|\bar{S}|} \sum_{s\in \bar{S}} C(y,s), \tag{19}$$

where the expected value function is approximated by the sample average function. The optimal value $v_{\bar{S}}$ to the SAA problem and the corresponding optimal solution provide estimates of their actual counterparts in the stochastic program.

This procedure can be repeated by generating several samples and solving related optimization problems until a stopping rule is met (e.g., time limit, optimality gap, number of SAA problems solved). Let us assume that the sample generation is repeated m times and that the corresponding SAA problems is solved using a deterministic optimization algorithm. We indicate as $\bar{S}_i$ the i-sample, $v_{\bar{S}_i}$ its optimal value, and y^i its optimal solution. Following Mak, Morton, and Wood (1999), a statistical lower bound on v^* can be computed as a sample average of the m optimal values as follows:

$$\bar{v}^m = \frac{1}{m}\sum_{i=1}^{m} v_{\bar{S}_i}, \tag{20}$$

with the following variance:

$$\sigma^2_{\bar{v}^m} = \frac{1}{m(m-1)}\sum_{i=1}^{m}(\bar{v}^m - v_{\bar{S}_i})^2. \tag{21}$$

The estimator $\bar{v}^m$, depending on the sample, is a random variable for which $\mathbb{E}[\bar{v}^m] \le v^*$, with a negative bias. By the Law of Large Numbers, $\bar{v}^m$ converges to the optimal value v^* with probability one as $m \to \infty$ (Mak et al., 1999). A way to evaluate the quality of a solution y^i is to bound its optimality gap, computed as $\mathbb{E}_{s\in\bar{S}_i}\left[C(y^i,s)\right] - v^*$, by using the lower-bound estimator $\bar{v}^m$ instead of v^*. Moreover, we can obtain a more accurate estimate of each solution y^i by using a larger sample $\hat{S}$ as follows:

$$\hat{v}(y^i) = \frac{1}{|\hat{S}|}\sum_{s\in\hat{S}} C(y^i,s), \tag{22}$$

with variance:

$$\sigma^2(y^i) = \frac{1}{|\hat{S}|(|\hat{S}|-1)}\sum_{s\in\hat{S}}(\hat{v}(y^i) - C(y^i,s))^2. \tag{23}$$

Then, $\hat{v}(y^i) - \bar{v}^m$ is an estimator of the optimality gap for solution y^i and its variance is $\sigma^2(y^i) + \sigma^2_{\bar{v}^m}$. This estimator has a negative bias $\mathbb{E}[\bar{v}^m]$ that monotonically decreases with a larger sample size.

In our methodology, we introduce a heuristic approach to solving SAA problems. This has the drawback that the value $\bar{v}^m$ is very likely not to be a valid lower bound, as it overestimates the optimal value of each SAA problem. Values $\hat{v}(y^i)$ are also quite likely overestimated. Still, these provide a rough estimate of the optimal value and enable us to evaluate the quality of found solutions. As a stopping criterion, we use the maximum number of iterations M (generated samples) instead of a rule based on the optimality gap, which might make the heuristic end too soon or perform too many iterations. About the trade-off between solution quality and computational effort, with a larger sample size, $\bar{v}^m$ will provide a more accurate estimate, and y^i, having the smallest $\hat{v}(y^i)$ value, will tend to be a better solution but the computational effort will increase at least linearly. In Section 6, we perform some computational experiments to find the best setting of these parameters.

The pseudo-code of our approach is described in Algorithm 1. The sample dimension and the number of iterations M are provided as input. For each of the M iterations, an i.i.d. scenario set $\bar{S}_i$ of size $\bar{r}$ is generated, and the corresponding SAA problem solved by using the heuristic methods described in Section 5.1. Then, $\hat{v}(y^i)$ is computed (Line 5) by using the scenario set $\hat{S}$ of size $\hat{r} > \bar{r}$ generated in Line 1. When the stopping rule is met, the algorithm computes the lower-bound estimate $\bar{v}$ and returns the best first-stage solution y^* generated across the M replications.

Algorithm 1 SAAM

```
Input: Number of iterations $M$, sample sizes $\bar{r}$ and $\hat{r}$, with $\hat{r} > \bar{r}$.
Output: Solution $y^*$, lower bound $\bar{v}$.
1: Generate a large sample set $\hat{S}$ of size $|\hat{S}| = \hat{r}$.
2: for $m = 1$ to $M$ do
3:     Generate sample set $S_i$ of size $|S_i| = \bar{r}$.
4:     Solve the SAA problem to obtain solution $y^i$ of value $v_{S_i}$.
5:     Compute $\hat{v}(y^i)$ (estimate $y^i$ over sample $\hat{S}$).
6: $\bar{v} = \frac{1}{M}\sum_{i=1}^{M} v_{S_i}$.
7: Compute $y^* = \arg\min_{i=1}^{M}\{\hat{v}(y^i)\}$.
8: return $y^*$, $\bar{v}$.
```

5. Solving SAA problems

In this section, we describe the algorithms used to solve the SAA problems. The presentation is made separately for the two stages.

5.1. First-stage solution

In what follows, H denotes a first-stage solution, i.e., a global schedule such that each zone z is visited exactly n_z times and at most once per day.

Local search

Since computing the second-stage solution is computationally expensive, first-stage algorithms are based on a small-size neighborhood to avoid calculating the cost of the second stage too many times. The neighborhood move exchanges time windows in the same zone while maintaining the feasibility of the first-stage solution. The quality of a move is assessed through the second-stage cost. Our solution framework first creates an initial solution H, then improves it by using the heuristic Perturbation based on LocalSearch, which works by relocating time windows over the planning period. The pseudo-code of LocalSearch is shown in Algorithm 2. Given a global schedule H as input, at each step, an attempt is made to decrease the second-stage cost by replacing a time window w in H with another one w' belonging to the same zone.

Algorithm 2 LocalSearch

```
Input: a global schedule $H$, a scenario sample $\bar{S}$.
Output: a (possibly improved) global schedule $H^*$.
1: $find \leftarrow$ true
2: $H^* \leftarrow H$
3: $f^* \leftarrow \infty$
4: while find do
5:     $find \leftarrow$ false
6:     for each time window $w \in H$ do
7:         Let $z$ be the zone associated with $w$.
8:         for each time period $d \in D$ do
9:             for each time window $w' \in W_z(d)$ do
10:                $H' \leftarrow$ UpdateSchedule($H, w, w'$)
11:                $f(H') \leftarrow$ OptimizeRoutes($H', \bar{S}$)
12:                if $H'$ is feasible and $f(H') < f^*$ then
13:                    $f^* \leftarrow f(H')$, $w^* \leftarrow w'$
14:                    $find \leftarrow$ true
15:        $H^* \leftarrow$ UpdateSchedule($H, w, w^*$)
16:        $f(H^*) \leftarrow f^*$
17:    $H \leftarrow H^*$
18: return $H^*$
```

For each move, we ensure that the generated solution has no more than one time window for each day and zone. The algorithm is run until no further improvement can be obtained. The function UpdateSchedule(H, w, w') substitutes the time window w with w' in the global schedule H, whereas the function OptimizeRoutes($H, \bar{S}$) solves the second stage by computing the routes according to the time windows assigned in H and by considering the scenario set $\bar{S}$.

We are to save computing time by employing the following two strategies. The first one tries to avoid unnecessary computation. In particular, when solving the second stage for a given scenario and period in the procedure OptimizeRoutes, this checks whether customers or their time windows have changed. If at least one change is observed, the second stage of that period is solved. Otherwise, the second-stage cost of that period is returned. Since moves involve changing a single time window, only one or two periods are involved. Thus, significant savings can be obtained when solving problems with a large planning period. We observed that computing times are close to $\frac{1}{\tau}$ (with τ being the planning horizon) of the time needed when not using this strategy.

The second strategy tries to explore the neighborhood with the aim of making the fewest changes to the second stage. For example, if two consecutive moves relocate a first time window w_1 from period 2 to 3 and a second time window w_2 from period 1 to 4, then the second stage has to be solved only for periods 1 to 4. Instead of going through the time windows one by one (Lines 6, 8–9), all possible moves are generated before attempting them and stored in an array sorted in such a way that all moves in the same periods are next to each other. Computing time savings of about 20% are observed when using this strategy.

Initial solution

Algorithm 3 constructs a first-stage initial solution by generating a global feasible schedule H, assigning to each zone $z \in Z$ exactly n_z time windows, each occurring in a day randomly selected and not already associated with another time window of the same zone z. Each time window is represented by a tuple (l, u, d, z) where start and end times l and u are initialized to l_0 and u_0, d is the day of the planning period to which the window belongs, and z is the zone. In Lines 2–6, to prevent more than one of the n_z time windows in zone z are assigned to the same day, the procedure Shuffle randomly perturbates an integer array D with values from 1 to τ. The days associated with the n_z time windows will correspond to the first n_z entries in the array D. Each time the function is called, a new set of days is assigned to the time windows of a selected zone. The corresponding time windows are then added to H by the function Add. Then, the procedure tries to improve the day-zone assignment of the current schedule H (Lines 7–16) such that the time windows of neighboring zones are in the same period. For each time window w, the values f^* and d^* are the cost and period of the best move. At each iteration, w is moved to a temporary period d', and values f^* and d^* are updated if there is no other time window of the same zone on period d' and if the second-stage cost f' is lower than f^*. Then, w is moved to the best period found d^* (Line 16). Notice that, in Lines 7–25, each time window is evaluated on all possible periods and starting times to obtain the largest decrease in the cost of the second stage. Initially, a copy of the current solution is stored in H^*. Then, the method tries to substitute each time window w in the current first-stage solution with a feasible time window $w' \in W_z$ by reducing its duration. In Line 20, the algorithm updates H by substituting an all-day time window w with a restricted one w', thus narrowing it to a proper duration. The algorithm updates the values of f^* and w^* when improving the cost over w^*. Finally, w is replaced by the best w^* found, and H^* is returned.

Algorithm 3 InitialSolution

```
Input: the zone set $Z$ and a scenario sample $\bar{S}$
Output: a global schedule $H^*$
1: $H \leftarrow \emptyset$; $D \leftarrow [1, ..., \tau]$;
2: for each zone $z \in Z$ do
3:     $D \leftarrow$ Shuffle($D$)
4:     for $i = 1$ to $n_z$ do
5:         $w \leftarrow (l_0, u_0, D[i], z)$
6:         $H \leftarrow$ Add($H, w$)
7: $H^* \leftarrow H$
8: for each time window $w = (l, u, d, z) \in H$ do
9:     $f^* \leftarrow \infty$, $d^* \leftarrow d$
10:    for each period $d' \in D$ do
11:        $w' \leftarrow (l, u, d', z)$
12:        $H' \leftarrow$ UpdateSchedule($H, w, w'$)
13:        $f(H') \leftarrow$ OptimizeRoutes($H', \bar{S}$)
14:        if $H'$ is feasible and $f(H') < f^*$ then
15:            $f^* \leftarrow f(H')$, $d^* \leftarrow d'$
16:    $w \leftarrow (l, u, d^*, z)$
17: for each time window $w = (l, u, d, z) \in H$ do
18:    $w^* \leftarrow w$, $f^* \leftarrow \infty$
19:    for each time window $w' \in W_z$ do
20:        $H' \leftarrow$ UpdateSchedule($H, w, w'$)
21:        $f(H') \leftarrow$ OptimizeRoutes($H'$)
22:        if $H'$ is feasible and $f(H') < f^*$ then
23:            $f^* \leftarrow f(H')$, $w^* \leftarrow w'$
24:    $H^* \leftarrow$ UpdateSchedule($H^*, w, w^*$)
25:    $f(H^*) \leftarrow f^*$
26: return $H^*$
```

Perturbation *meta-heuristic*

We now describe the meta-heuristic Perturbation that allows us to escape from local optima by moving a number α_1 of time windows, randomly selected, to the most convenient days (in terms of cost) different from their current ones. Thus, the parameter α_1 controls the perturbation *intensity*. Procedure LocalSearch is then run on each perturbed solution.

Perturbation, as shown in Algorithm 4, runs over a maximum number $itMax$ of iterations. At the beginning of each iteration, the best first-stage solution H^* found until that point is copied into the current solution H (Line 3). The solution is perturbed using two inner loops (Lines 4–14) controlled by the two parameters α_1 and α_2, initially set to a low value and then adjusted dynamically. While α_1 works similarly to the radius of a Variable Neighborhood Search (deciding the number of time windows to move and increasing in value when no better solution can be found), α_2 controls the number of times the current solution must be perturbed. In Line 6, the function MoveTWs randomly moves α_1 time windows to their cheapest periods, and then LocalSearch is applied to find a local minimum (Line 10). If the new solution found has improved over the best one, α_1 is set back to its initial value, and the perturbation continues. Otherwise, the value of α_1 is increased. When it reaches its maximum value γ_1, the search is restarted, and the best solution is copied again into the current one. The incumbent solution is updated every time a new best solution is found (Lines 9 and 13), and eventually, the best global schedule found H^* is returned (Line 15). Best values for parameters $itMax$, γ_1, and γ_2 are investigated in Section 6.

Algorithm 4 Perturbation

```
Input: a global schedule $H$, a scenario sample $\bar{S}$
Output: a global schedule $H^*$
1: $H^* \leftarrow$ LocalSearch($H, \bar{S}$)
2: for $iter = 1$ to $itMax$ do
3:     $H \leftarrow H^*$
4:     for $\alpha_1 = 1$ to $\gamma_1$ do
5:         for $\alpha_2 = 1$ to $\gamma_2$ do
6:             $H \leftarrow$ MoveTWs($H, \alpha_1$)
7:         $f(H) \leftarrow$ OptimizeRoutes($H, \bar{S}$)
8:         if $f(H) < f(H^*)$ then
9:             $H^* \leftarrow H$
10:        $H \leftarrow$ LocalSearch($H^*, \bar{S}$)
11:        $f(H) \leftarrow$ OptimizeRoutes($H, \bar{S}$)
12:        if $f(H) < f(H^*)$ then
13:            $H^* \leftarrow H$
14:            $\alpha_1 \leftarrow 1$
15: return $H^*$
```

5.2. Second-stage solution

In the second stage, the global schedule H computed in the first stage is used by OptimizeRoutes to determine the routes to visit the customers of the different zones. Since a zone might be visited more than once over the planning period, and possibly in different time windows, we need to establish when a customer of a given zone has to be served. All customers have an ordered list of days on when they prefer to be served. We thus assume that they will select the best combination of day provided for their zone when placing the order. Also, each customer will be served in the time window of the current schedule H corresponding to the day having the highest order in their preference list. This allows us to guarantee, on average, good customer satisfaction. To compute the cost associated with the routing of a fleet of vehicles, given a global schedule H, we solve a modified VRPTW for each scenario of the sample and each period $d \in D$ by minimizing the total travel distance and the number of unserved customers. To this aim, we run some classical local search procedures and an Adaptive Large Neighborhood Search (ALNS) framework following the general scheme proposed by Ropke and Pisinger (2006).

OptimizeRoutes receives as input a global schedule H and a set of scenarios $\bar{S}$. The cost of routing f for each scenario s and period d is initialized to 0 (Line 1). After assigning customers to time windows (Line 3), the procedure builds an initial solution for each period $d \in D$ by applying the ConstructRoutes procedure (Line 5). This is a construction heuristic where customers are sequentially inserted in an available route by the *Regret-k* insertion heuristic described in Potvin and Rousseau (1993). In particular, at each iteration, the heuristic computes the minimal insertion cost into each route and selects the

Algorithm 5 OptimizeRoutes

```
Input: a global schedule $H$, a scenario sample $\bar{S}$
Output: average routing cost of the global schedule $H$
1: $f \leftarrow 0$
2: for $s = 1$ to $|\bar{S}|$ do
3:     Assign customers to days/time windows
4:     for $t = 1$ to $|T|$ do
5:         $R^s_d \leftarrow$ ConstructRoutes($H, N_s, d, k_d$)
6:         $R^s_d \leftarrow$ ALNS($R^s_d$)
7:         $R^s_d \leftarrow$ Improvement($R^s_d$)
8:         $f \leftarrow f + C(R^s_d)$
9: return $\frac{f}{|\bar{S}|}$
```

customer who maximizes the sum of differences between the cost of inserting in their best route minus the best insertion costs in the other routes. This is a lookahead value indicating the loss that could be incurred if the customer is not inserted. If some customers cannot be feasibly inserted into a route, they are left in a customer bank to be inserted later. R^s_d and $C(R^s_d)$ indicate the set of routes serving customers of period d in scenario s and their cost, respectively. Then, the ALNS procedure is called (Line 6). At each iteration, removal and insertion heuristics are selected among the ones available in the framework. The purpose of the removal heuristic is to remove some customers from the solution and put them into the customer bank. Then, the insertion heuristic selects some other customers in the customer bank to insert them back into the solution, possibly improving it. We have selected and implemented a subset of the destroy/repair operators described in Ropke and Pisinger (2006). In particular, after extensive preliminary tests, we have selected the *Random* removal, the *Shaw* removal, and the *Regret-k* insertion, because they offer the best trade-off in terms of time and quality. All operators, as well as the roulette-wheel mechanism, are implemented as in Ropke and Pisinger (2006). We introduced only two minor changes corresponding to the cooling rate (set to 0.8) and the number of iterations decided after some tests, as described in Section 6. After the removal/insertion process, the procedure Improvement (Line 7) sequentially applies two local search operators until no further improvement can be found: 2-Opt*, described in Potvin and Rousseau (1995), and Relocate, proposed by Savelsbergh (1992), which tries to move a customer from one route into another. In the end, the average routing cost $f/|\bar{S}|$ is returned.

5.3. Speed-up techniques

We implement two techniques to improve our algorithms speed. The former is an ALNS *hot-starting*. In many cases, global schedules, provided as input to the second stage, do not drastically change. Typically, they differ for only a time window or its starting time. Instead of constructing a solution from scratch, the current second-stage solution is kept in memory for each scenario. The time window of each customer is updated each time the second-stage problem is about to be solved. Also, the feasibility of every route is checked before executing ALNS. Customers causing infeasibility are removed and put into the customer bank. Once all routes are made feasible, customers in the bank are reinserted into the solution using the regret heuristic, and then ALNS is started.

The second technique uses a *hash table* to store and retrieve the cost of the second stage of every first-stage solution already visited. Each global schedule is hashed to a 64-bit integer as follows. First, time windows are sorted by non-decreasing zone and breaking ties by day. Each time window is transformed into a 32-bit integer, where the first 10 bits correspond to the day, the next 11 to the starting time, and the last 11 to the ending time (both times are represented in the number of minutes since midnight). Then, the integer identifier of each time window is added to a list that is hashed into a 64-bit integer key by using the *one-at-a-time* function by Jenkins.[1] The key and cost of the solution are added to a traditional hash table. Each time a call to OptimizeRoutes($H, \bar{S}$) is made, a check is performed in the hash table whenever this solution has been previously found. If an entry exists, the retrieved cost is returned; otherwise, the cost of the second stage is computed and stored in the hash table. A speed-up of 5x can be achieved by using this technique.

Table 2
Instance classes used in the different experiments of our evaluation.

Experiments	Instances
Perturbation parameter tuning and performance	Class A, Class B
Second-stage parameter tuning and performance	Class A
Value of the stochastic solution	Class A
Quality of service	Class A
Application to real-world data	Class C

6. Experimental evaluation

This section provides an overview of the computational experiments conducted to assess the performance of our solution approaches. It is partitioned into six parts: (i) *Instances* offers information about the instances used in our experiments; (ii) Perturbation and (iii) *Second-stage parameter setting and performance* respectively delve into fine-tuning components and parameters, and assessing performance of the metaheuristic and OptimizeRoutes in the second stage; (iv) *Value of the stochastic solution* analyzes the solution obtained by SAAM; (v) *Quality of service* investigates the variations in the quality of service in response to modifications, such as changes in time window characteristics, daily working hours, and the availability of drivers; and (vi) *Application to real-world data* tests the applicability of our approach to real-world scenarios that served as the initial motivation for this study, yielding managerial insights. All algorithms have been coded in C++ and run on an Intel 2.667 GHz Westmere EP X5650 processor under Scientific Linux 6.3.

6.1. Instances

Since no benchmark instances exist for this problem, we create three new data sets, namely Class A, B, and C. For each aspect previously identified, Table 2 indicates which classes of instances are used to perform the related experiments.

Class A instances. The delivery region is partitioned into a grid, where each square corresponds to a zone of 500 × 500. For each zone, we assume to know the probability distributions for the locations, demands, and service times of the customers. In particular, the total number of customers follows a Poisson distribution. Locations are uniformly distributed, whereas demands and service times follow two normal distributions, with a standard deviation of 5. Demands have an average value of 20, whereas service times average values are shown in Table 3. If we obtain a negative value when generating the random demands and service times, we keep generating a new number until it is non-negative. Distances are Euclidean and the vehicle capacity is set to 800. The operating hours of the depot start at 8 AM or 9 AM and end between 2 PM and 5 PM. The β cost associated with unserved customers is set to 10^6 to favor the number of served customers over the distance. All time windows have a width of 3 h. The required number of time windows per zone is estimated by approximating each zone's work time, including the total travel and service times. The total travel time is computed by using the approximation of the optimal routing cost proposed by Figliozzi (2008):

$$L^* = \frac{1}{speed} \cdot \left(1.45 \cdot \frac{n_\alpha - k}{n_\alpha} \cdot \sqrt{E\bar{n}} + 2rk\right), \tag{24}$$

[1] http://burtleburtle.net/bob/hash/doobs.html

Table 3
Structure of `Class A` instances.

Group	Zones (#)	TWs (#)	Days (#)	Daily schedule	Average customers (#)	Average service time (s)	Drivers per day (#)	Work time per zone (min)
1	6	1	2	8 AM–2 PM	32	20	1	166.3
2	8	1	2	8 AM–4 PM	48	18	1	162.5
3	8	1	2	8 AM–4 PM	64	12	1	148.1
4	12	1	2	9 AM–4 PM	72	16	2	146.6
5	12	1	2	9 AM–4 PM	96	14	2	166.2
6	16	1	3	9 AM–4 PM	132	14	2	170.4
7	16	2	3	8 AM–5 PM	132	30	3	327.1
8	20	2	4	8 AM–5 PM	160	32	3	336.5
9	16	3	5	8 AM–4 PM	192	30	3	464.2
10	20	3	5	8 AM–4 PM	200	30	3	390.5

Table 4
Structure of `Class B` instances.

Group	Zones (#)	TWs (#)	Days (#)	Daily schedule	Average customers (#)	Average demand	Average service time (s)	Drivers per day (#)	Work time per zone (min)
1	4	1	1	8 AM–5 PM	8	20	32	1	119.7
2	4	1	2	8 AM–3 PM	12	52	26	1	139.8
3	4	1	1	8 AM–4 PM	16	22	15	1	108.7
4	4	1	2	8 AM–3 PM	16	54	22	1	151.5
5	4	1	1	8 AM–4 PM	24	15	10	1	108.2
6	4	1	2	8 AM–3 PM	24	30	18	1	175.8
7	6	1	3	8 AM–2 PM	24	52	20	1	135.5
8	6	2	2	8 AM–3 PM	24	55	30	2	191.7
9	6	2	2	8 AM–4 PM	30	45	30	2	228.8
10	8	1	4	8 AM–1 PM	32	52	22	1	143.4

where L^* is the total travel time (in hours), n_α is the number of customers, E is the total area of the delivery region, k is the number of drivers, r is the average distance customer/depot computed as half the diagonal of a square of area equal to E, and *speed* is the traveling speed of the vehicles. The *speed* is set to $\frac{100}{60} \cdot 50$ km/h to represent time as a function of the travel distance. The formula does a regression over these parameters to approximate the total travel time of the fleet. The total work time in each zone is approximated as $L_z^* = \frac{L^*}{|Z|} + b_z \frac{n_\alpha}{|Z|}$. The approximated work time L_z^* in zone z is composed of the total travel time L^* divided by the number of zones, plus the expected service time b_z of zone z times the number of customers in the zone $\left(\frac{n_\alpha}{|Z|}\right)$. Since the instances in each configuration are homogeneous, all L_z^* values are the same. Then, the number of time windows (all with a width equal to 3 h) in zone z is as follows:

$$\#\text{ of time windows in zone } z = \left\lceil \frac{L_z^*}{\text{TW Width}} \right\rceil . \tag{25}$$

To ensure an adequate number of time windows, we do not rely solely on the expected number of customers, but we align it with the required number of time windows. The value n_α is such that $P(X \leq n_\alpha) = \alpha$, where X is a Poisson random variable with an expected value equal to n. A value of $\alpha = 0.95$ increases L^* and L_z^* by ensuring a relatively low number of unserved customers.

For each instance, we derive the corresponding scenarios as follows. Firstly, the total number of customers is randomly generated according to a Poisson distribution. Secondly, a zone is randomly selected with equal probability for each customer. Thirdly, each customer is randomly located inside a zone with demand and service time randomly generated according to a normal distribution. Each customer's preferences on the delivery date are given by a random permutation of the array $D = [1, 2, \dots, \tau]$. We assume that each scenario is equally likely to occur. Finally, an expected scenario is also generated for comparing what would produce a deterministic approach (like in Hernandez et al., 2017) versus a stochastic one. This scenario is characterized by having the number of customers, their service times, and demands exactly equal to the expected values.

Table 3 describes the different *groups* of instances of `Class A`. Each group consists of 5 instances. In each instance, the depot is randomly located inside the geographical grid. Moreover, each group is characterized by a number of zones, a number of time windows per zone, a number of days in the planning period, an average number of customers, an average service time, a number of available drivers for each day, and an estimation of the required work time spent in each zone.

Class B instances. Table 4 details Class B instances. These were generated in the same way as those of `Class A`. To make them more easily solvable by CPLEX for solving model (1)–(16), the average number of customers is smaller. Moreover, the depot operating hours, demands, and service times are fine-tuned to have all customers served. The vehicle capacity is set to 600.

Class C instances. This set encompasses data from a real-world application of a Canadian furniture company. The company operates two warehouses located in the cities of Edmonton and Calgary. Both warehouses deliver products to customers located in their neighborhood area.

The company's historical database consists of 80 000 deliveries over several months. The warehouse of Edmonton offers deliveries in 49 postal codes with an average of 463 customers per week and a fleet

Table 5
Tuning of Perturbation parameters.

(a) Distance deviation with different parameters γ_1 and γ_2.

	γ_2			
γ_1	1	2	3	4
1	4.19%	1.92%	1.17%	0.93%
2	1.51%	1.49%	0.98%	1.33%
4	1.49%	1.13%	1.59%	1.28%
6	1.83%	1.19%	1.25%	1.63%
8	1.37%	1.49%	1.66%	1.75%

(b) Runtime versus number of iterations.

$itMax$	Unserved customers (Δ)	Distance (Δ)	Time (s)
0	0.22	10.39%	1.9
1	0.20	9.41%	2.9
10	0.06	6.65%	13.5
20	0.06	5.16%	24.6
50	0.02	3.45%	58.1
100	0.04	1.92%	111.1
200	0.04	1.06%	220.1
400	0.04	0.00%	437.1

Table 6
Comparison of heuristics on `Class A` instances.

Algorithm	Unserved customers (Δ)	Distance (Δ)	Time (s)	Best (#)
ConstructRoutes	0.48	12.7%	0.7	38/50
Improvement	0.18	8.2%	1.7	42/50
Perturbation	0.00	0.1%	111.4	50/50

of 5 vehicles per day. For Calgary, the company delivers in 44 postal codes with an average of 469 customers per week and a fleet of 6 vehicles per day. The vehicle capacity is 1100. A data analysis shows that the number of customers per week follows a Poisson distribution. However, instead of randomly generating customers, we randomly pick them from the bank of 80 000 deliveries. A set of 200 scenarios is created to be used in the optimization, and another set of 400 scenarios is generated to calculate the expected number of unserved customers and distance deviation. Postal codes are highly heterogeneous: some zones require more than 12 h of work per week, others less than 2 h. All vehicles are available from 8 AM to 6 PM.

6.2. Perturbation *parameter setting and performance*

First, we test different values for Perturbation parameters to tune them. Then, we evaluate its performance by comparing it with the other heuristics in Section 5 and with the CPLEX solver.

Perturbation *parameter tuning*

Table 5(a) presents the results of different combinations of values of γ_1 and γ_2 in the meta-heuristic Perturbation. We recall that γ_1 and γ_2 correspond to the maximum values that the two parameters α_1 and α_2, controlling the two inner loops of the meta-heuristic, can reach, respectively. We run the algorithm for exactly 10 min on Groups 1 and 2, 20 min on Groups 3 and 4, and 40 min on the remaining Groups 5–10 in `Class A`. Each table entry contains the distance percentage deviation to the best solution value found. Despite being a little counter-intuitive, it is evident from the findings that it is more convenient, in terms of the smallest percentage deviation, to move only a single time window more times rather than many time windows at the same time. The best combination of the two parameters controlling the two inner loops of Perturbation is thus given by $\gamma_1 = 1$ and $\gamma_2 = 4$.

Next, we conduct a second group of tests to evaluate the evolution of the objective function value when increasing the number of iterations $itMax$ in the Perturbation heuristic without setting any time limit. Table 5(b) reports the results, where the first column provides the number of iterations (value 0 means the algorithm consists only of the constructive method and the local search heuristic), the second and the third columns show the deviation in terms of the number of unserved customers and distance from the best solution, respectively. The last column provides the average runtime (in seconds). Performing a limited number of iterations has a minimal impact on enhancing the solution value. However, when $itMax = 100$ iterations, the results significantly improve while maintaining a reasonable computational time. The total runtime strongly depends on the number of scenarios and is equal to the time reported in the fourth column multiplied by the number of scenarios. Thus, we decide to set $itMax$ to 100 iterations.

Perturbation *performance*

To assess performance in terms of solution quality and computational times, we first compare Perturbation against the heuristic algorithms discussed in Section 5 on `Class A` instances. Then, on the smaller `Class B` instances, we use the CPLEX solver as a benchmark.

Comparison with ConstructRoutes *and* Improvement

In Table 6, we evaluate how the Perturbation meta-heuristic of the first stage can positively impact second-stage solutions, rather than using only ConstructRoutes heuristic or the local-search Improvement. The first column indicates the algorithm, whereas the second and third ones are the deviation with the best number of unserved customers and the percentage distance deviation. The last two columns indicate the running time and the number of times the algorithm finds the best solution. We observe that using the Perturbation meta-heuristic significantly improves the second-stage solutions, always finding the best solutions. ConstructRoutes and Improvement obtain an average distance deviation of 12.7% and 8.2% worse than Perturbation, respectively.

Comparison with CPLEX

Here, we test Perturbation on `Class B` instances having up to 5 scenarios. We set the time limit for CPLEX to 1 h. Table 7(a) shows the results for each group. Column "Δ_P" reports the percentage gap $(UB_P - BKS)/BKS$ between the solution found by Perturbation (UB_P) and the best-known solution (BKS) found by either CPLEX or Perturbation itself. The column "Time (s)" is the average time out of the instances on each group. Regarding CPLEX, "Opt (#)" indicates the number of instances solved to optimality, whereas "Gap" is the percentage difference between the best-known solution and the lower bound computed by the solver (LB), i.e., $(BKS - LB)/LB$. Perturbation performs very well on all groups of instances with an average Δ of 0.03%, and a time two orders of magnitude lower than CPLEX. Indeed, CPLEX already struggles even with a very small number of scenarios, as highlighted in Table 7(b). To tackle complex problems such as the SMTWAP, the adoption of meta-heuristics like Perturbation becomes a crucial requirement.

6.3. *Second-stage parameter setting and performance*

This section analyzes the impact of the different components of OptimizeRoutes. Results are shown in Table 8, where the first line corresponds to the initial routes obtained with ConstructRoutes. We consider the performance of the Improvement local-search methods (2-Opt* and Relocate), first tested separately and then together. Instead, since the number of iterations has a major impact on the quality of the

Table 7
Perturbation performance against CPLEX on Class B instances.

(a) Performance by groups.

	Perturbation		CPLEX		
Group	Δ_P	Time (s)	Opt (#)	Gap	Time (s)
1	0.00%	5.4	25	0.00%	0.9
2	0.00%	8.6	25	0.00%	14.2
3	0.12%	10.4	21	0.78%	595.3
4	0.00%	12.5	20	1.77%	753.9
5	0.02%	15.4	14	4.68%	1722.9
6	0.00%	29.5	9	7.28%	2421.8
7	0.00%	25.4	7	7.20%	2765.7
8	0.17%	31.8	22	0.90%	610.1
9	0.03%	21.7	13	1.54%	1720.8
10	0.00%	40.0	2	19.82%	3514.8
Total/Avg.	0.03%	20.1	158	3.15%	1412.0

(b) Performance by number of scenarios.

Scenarios	Perturbation		CPLEX		
(#)	Δ_P	Time (s)	Opt (#)	Gap	Time (s)
1	0.06%	5.8	46	13.33%	447.4
2	0.01%	13.1	36	8.10%	1078.8
3	0.02%	20.2	29	8.71%	1513.7
4	0.09%	27.2	26	8.81%	1857.7
5	0.01%	34.1	21	7.78%	2162.6

Table 8
Testing different components of OptimizeRoutes and their combination on Class A instances.

Configuration	Unserved customers (Δ)	Distance (Δ)	Time (s)
ConstructRoutes	1.22	0.00%	8.1
2-Opt*	0.58	−8.86%	8.5
Relocate	0.66	−10.59%	9.1
Improvement (Relocate & 2-Opt*)	0.52	−12.56%	9.7
ALNS 25	0.06	−18.84%	33.0
ALNS 50	0.04	−19.51%	55.5
ALNS 100	0.02	−20.30%	103.3
ALNS 200	0.04	−20.68%	198.1
ALNS 400	0.02	−21.63%	377.0
ALNS 25 & Improvement	0.02	−20.79%	36.5
ALNS 50 & Improvement	0.02	−21.11%	62.7
ALNS 100 & Improvement	0.00	−21.56%	112.5
ALNS 200 & Improvement	0.00	−21.46%	213.4
ALNS 400 & Improvement	0.04	−22.11%	416.7

solution and on the running time, we test ALNS with 25, 50, 100, 200, and 400 iterations, with and without the Improvement methods.

In terms of the number of unserved customers (second column), the best configurations are ALNS 100 or 200 & Improvement. Each other entry reports the absolute difference with respect to these best solutions. Column "Distance (Δ)" contains the percentage deviation of each configuration with respect to the initial solution value obtained with ConstructRoutes. The last column indicates the average computational times.

Employing ALNS enhances solution quality at the cost of increased runtime. Improvement plays a valuable role in qualifying the initial routes, yet ALNS remains indispensable for achieving performing outcomes. Superior results in terms of distance are achieved by combining both components and running ALNS for 400 iterations. However, this configuration is impractical when dealing with 200 scenarios as it would lead to a computation time of 9000 s. Moreover, it provides a limited improvement with respect to 100 or 200 iterations. As observed above, ALNS 100 or 200 & Improvement are equivalent in terms of number of unserved customers. However, doubling the number of iterations to 200 increases the runtime for a marginal increase in solution quality. Thus, 100 iterations strike a reasonable balance between quality and computational time.

6.4. Value of the stochastic solution

Hereafter, we evaluate the relevance of the stochastic solution by assessing the performance of SAAM over the Class A instances. To this aim, we generate a large set of 1000 scenarios, partitioned into sample sets of size $|\bar{S}| = 5, 25, 50, 100, 200$. We set the number of replications M to $1000/|\bar{S}|$ and the size of the estimator $\hat{S}$ to 500. The number of iterations $itMax$ of Perturbation was set to 20, instead of 100, only for large-size Groups 6–10, so to conclude all tests in five days of computation. Additionally, we examine the potential advantages of approaching the problem through a stochastic programming method compared to a deterministic one. To this aim, we run Perturbation on the expected scenario $\bar{s}$, obtain the solution $\bar{y}$, and assess its cost with an estimator set of 500 scenarios.

Table 9(a) summarizes the results over all instances for different values of $|\bar{S}|$ and M, and entries represent the number of unserved customers. Columns contain the values of the lower bound $\bar{v}^m$, the best solution found $\hat{v}(y^*)$, and the average estimated solution cost. The best values are underlined.

We remark that $\bar{v}^m$ does not necessarily represent a valid lower bound as it may overestimate the optimal value of each replication. Thus, some values of the best solutions are lower than the corresponding lower bounds. We note that the lower-bound quality is low for smaller values of $|\bar{S}|$ and improves for higher values, reaching its peak with $|\bar{S}| = 200$. The difference between the best solution and the lower bound is small for Groups 1–6 and samples with $|\bar{S}| \geq 50$. However, for larger instances, this gap widens due to the increased difficulty. Next, we observe that for Groups 1–6, best solutions are obtained with a relatively small value of $|\bar{S}|$, whereas, for Groups 7–10, these are obtained with $|\bar{S}| = 200$. This can be attributed to the generation of a higher number of solutions, increasing the likelihood of finding a good one among, for instance, 200 or 400 solutions. Since all tests are executed within a similar duration, it appears more effective on smaller instances to conduct as many replications as possible. However, for large instances, the best quality can be achieved with a higher number

Table 9
Evaluation of the SAAM on `Class A` instances.

(a) Results for several values of $|\bar{S}|$ and M.

	LB $\bar{v}^m$					Best $\hat{v}(y^*)$					Avg.				
$\lvert\bar{S}\rvert$	5	25	50	100	200	5	25	50	100	200	5	25	50	100	200
Group \ M	200	40	20	10	5	200	40	20	10	5	200	40	20	10	5
1	2.49	2.93	3.06	3.11	3.14	3.22	3.23	3.23	3.24	3.26	3.53	3.39	3.37	3.33	3.32
2	3.18	3.88	4.01	4.08	4.09	4.07	4.05	4.06	4.04	4.08	4.69	4.38	4.29	4.20	4.17
3	4.25	4.98	5.18	5.31	5.40	5.26	5.35	5.38	5.35	5.39	5.73	5.56	5.56	5.51	5.49
4	0.03	0.10	0.17	0.23	0.28	0.33	0.35	0.37	0.36	0.37	0.96	0.48	0.47	0.45	0.43
5	0.97	1.53	1.70	1.83	1.89	1.94	1.95	1.92	1.98	1.96	2.63	2.33	2.25	2.21	2.11
6	0.28	1.08	1.38	1.56	1.68	1.73	1.82	1.80	1.84	1.78	2.74	2.36	2.24	2.15	2.01
7	1.22	2.68	3.16	3.49	3.68	4.53	4.40	4.38	4.34	4.31	5.63	4.91	4.77	4.66	4.55
8	0.80	2.64	3.24	3.74	3.94	5.10	4.72	4.64	4.55	4.51	6.63	5.42	5.13	4.94	4.87
9	1.12	3.49	4.36	4.91	5.34	6.56	6.43	6.33	6.22	6.11	8.10	7.03	6.80	6.64	6.42
10	4.09	7.05	8.03	8.67	8.97	11.21	10.62	10.62	10.46	10.34	12.81	11.43	11.08	10.82	10.67
Avg.	1.84	3.04	3.43	3.69	3.84	4.40	4.29	4.27	4.24	4.21	5.35	4.73	4.60	4.49	4.40

(b) Stochastic approach versus deterministic approach.

	SAAM					Expected Scenario			Gap
Group	LB $\bar{v}^m$	Best $\hat{v}(y^*)$	Avg.	STD	Time (hrs)	$C(\bar{y},\bar{s})$	$\hat{v}(\bar{y})$	Time (hrs)	
1	3.14	3.22	3.32	0.05	1.4	0.00	3.41	0.00	0.19
2	4.09	4.04	4.17	0.09	5.4	2.00	4.41	0.01	0.37
3	5.40	5.26	5.49	0.09	8.0	4.00	5.81	0.01	0.55
4	0.28	0.33	0.43	0.06	5.5	0.00	1.78	0.02	1.46
5	1.89	1.92	2.11	0.13	9.6	0.00	2.82	0.02	0.91
6	1.68	1.73	2.01	0.21	6.6	0.00	3.38	0.02	1.65
7	3.68	4.31	4.55	0.20	11.0	0.00	6.95	0.04	2.64
8	3.94	4.51	4.87	0.29	19.9	0.00	9.20	0.09	4.69
9	5.34	6.11	6.42	0.24	23.0	0.00	12.06	0.10	5.95
10	8.97	10.34	10.67	0.27	30.5	0.20	17.24	0.11	6.90
Avg.	3.84	4.18	4.40	0.16	12.1	0.62	6.71	0.04	2.53

(c) Avg runtime per SAA iteration.

	Time (hrs) per iteration				
Group\\$\lvert\bar{S}\rvert$	5	25	50	100	200
1	0.01	0.04	0.07	0.14	0.28
2	0.03	0.14	0.27	0.55	1.08
3	0.04	0.20	0.41	0.81	1.59
4	0.03	0.14	0.27	0.53	1.03
5	0.05	0.24	0.48	0.96	1.89
6	0.03	0.16	0.34	0.69	1.38
7	0.05	0.27	0.56	1.14	2.34
8	0.09	0.49	1.03	2.10	4.21
9	0.10	0.58	1.18	2.43	4.76
10	0.14	0.75	1.54	3.20	6.45

of scenarios. The average solution quality improves with higher values of $|\bar{S}|$. Indeed, average best results for all instance groups are obtained with $|\bar{S}| = 200$.

Table 9(b) compares the best values obtained in Table 9(a) (cf. underlined entries for "LB $\bar{v}^m$", "Best $\hat{v}(y^*)$", and "Avg".) using SAAM versus a deterministic approach. Column "STD" reports the standard deviation of the $|\bar{S}|$ and M having the best average solution, whereas column "Time (hrs)" contains the computation time in hours. Column "$C(\bar{y},\bar{s})$" shows the number of unserved customers in the solution $\bar{y}$ when evaluated on the expected scenario. Column "$\hat{v}(\bar{y})$" shows the expected optimal cost. The last column, "Gap", measures the difference between $\hat{v}(\bar{y})$ and the values in the column "Best $\hat{v}(y^*)$" (i.e., the difference between the cost of the deterministic solution and the stochastic one). For the deterministic approach, we notice that the number of unserved customers of the expected scenario, $C(\bar{y},\bar{s})$, is close to zero for almost all instance groups, whereas the estimated number of unserved customers, $\hat{v}(y^*)$, is significantly higher. As for the gap, in the smallest instances (Groups 1 to 5), it is relatively small, meaning that, in these cases, a stochastic approach does not add much value. Instead, for larger instance groups (Groups 6 to 10), the gap grows very large, and the stochastic approach becomes significant. However, this improvement comes with a remarkably higher computation effort (up to 30.5 h). By looking at the "Avg". columns of Table 9(a), good results can still be achieved by solving a few SAA problems with $|\bar{S}| = 25$ or 50, with a lower computational effort. Finally, in Table 9(c), we report on the running time in hours per iteration for different values of $|\bar{S}|$. As expected, for each group, the time increases as $|\bar{S}|$ grows.

6.5. Quality of the service

In this section, we evaluate the quality of service of the solutions obtained by our approach by investigating the following aspects: the number and duration of time windows, the number of drivers per day, and their impact on transportation costs and the number of unserved customers.

Impact evaluation of time windows variation

In Section 6.4, we employed 3-h time windows. Nevertheless, companies may be interested in enhancing customer service by using narrower time windows. We assume that the quality of service deteriorates as the width of the time windows increases. Indeed, wider time windows imply higher waiting times and are usually less preferred by customers who do not like to wait at home for several hours. In contrast, with shorter time windows (e.g., 1 h), customers might be more satisfied, but this level of satisfaction may come at an increased transportation cost for the company. Each new offered time window is a possible additional trip to the zone that might increase transportation costs. Special care must be given when selecting the time window width since changing size might imply a reduction of the overall work time in a zone. Table 10(a) shows this phenomenon. Each entry indicates the expected free time in minutes available in each zone, computed as follows:

$$\text{Free Time} = (\#\ \text{Time Windows per Zone} \times \text{TW Width}) - L^*_z. \qquad (26)$$

For example, consider Group 3 with a work time of 148.1 min. The free time with 4-h time windows is equal to 91.9 min. Then, it goes down to 31.9 min with 3-h time windows and back up to 91.9 min for 2-h time windows. In practice, this means we might be able to serve more customers with shorter time windows. Another phenomenon might also happen when offering two time windows of width v versus a single one of size $2v$. Since the two shorter time windows must be on different days, having twice the number of drivers available can impact the number of unserved customers. For these reasons, tuning the time window width should be carefully done to achieve the best results in terms of transportation costs and number of unserved customers. Table 10(b) indicates the number of time windows in each scenario for each possible time window width (i.e., from spanning an entire day up to 1-h width).

In Table 10(c), we analyze the impact of different time window widths. We execute our meta-heuristic using the same parameter values as before. First, we generate a single set of 200 scenarios for which

Table 10
Quality of service in solutions to `Class A` instances.

(a) Free time per different time window width.

Group	Work time (min)	All-day	5 h	4 h	3 h	2 h	1.5 h	1 h
1	166.3	433.7	133.7	73.7	13.7	73.7	13.7	13.7
2	162.5	437.5	137.5	77.5	17.5	77.5	17.5	17.5
3	148.1	451.9	151.9	91.9	31.9	91.9	31.9	31.9
4	146.6	453.4	153.4	93.4	33.4	93.4	33.4	33.4
5	166.2	433.8	133.8	73.8	13.8	73.8	13.8	13.8
6	170.4	429.6	129.6	69.6	9.6	69.6	9.6	9.6
7	327.1	272.9	272.9	152.9	32.9	32.9	32.9	32.9
8	336.5	263.5	263.5	143.5	23.5	23.5	23.5	23.5
9	464.2	135.8	135.8	15.8	75.8	15.8	75.8	15.8
10	390.5	209.5	209.5	89.5	149.5	89.5	59.5	29.5

(b) Number of time windows per different width.

Group	Days	Work	TWs per Width (#)						
	(#)	Time (min)	All-day	5 h	4 h	3 h	2 h	1.5 h	1 h
1	2	166.3	1	1	1	1	2	2	3
2	2	162.5	1	1	1	1	2	2	3
3	2	148.1	1	1	1	1	2	2	3
4	2	146.6	1	1	1	1	2	2	3
5	2	166.2	1	1	1	1	2	2	3
6	3	170.4	1	1	1	1	2	2	3
7	3	327.1	1	2	2	2	3	4	6
8	4	336.5	1	2	2	2	3	4	6
9	5	464.2	1	2	2	3	4	6	8
10	5	390.5	1	2	2	3	4	5	7

(c) Cost of time windows.

TW width	Unserved customers (#)	Distance (Δ)	Time (hrs)
All Day	3.26	0.0%	0.2
5 h	3.69	12.8%	1.9
4 h	3.79	14.3%	2.2
3 h	4.28	19.0%	2.5
2 h	5.82	37.3%	2.3
1.5 h	8.06	43.2%	1.3
1 h	12.76	48.6%	1.4

Table 11
Results when varying the number of working hours or drivers available daily.

(a) Impact evaluation when changing the number of daily working hours.

Hours variation	Unserved customers (#)	Distance (Δ)	Time (hrs)
−2 h	21.47	−16.8%	2.2
−1 h	10.44	−6.8%	2.6
Base case	4.28	0.0%	2.5
+1 h	1.71	3.2%	3.2
+2 h	0.97	2.4%	3.4

(b) Impact evaluation when changing the daily number of available drivers.

Drivers variation	Unserved customers (#)	Distance (Δ)	Time (hrs)
−2	23.45	−34.8%	4.3
−1	14.44	−23.8%	4.3
Base case	4.28	0.0%	2.5
+1	1.33	9.9%	3.4
+2	0.57	5.2%	5.1

we find the best solution. Then, we estimate the expected transportation costs on a set of 400 scenarios. The first column provides the width of the time windows. Columns "Unserved Customers (#)" and "Distance (Δ)" provide the average number of unserved customers and the average percentage deviation in terms of traveled distance, respectively. We notice that the distance deviation increases as the time window width decreases. The number of unserved customers also slightly increases when moving from the All-Day time windows to 3-h time windows. However, when shifting from 3-h to 1-h time windows, the number of unserved customers significantly rises. This phenomenon occurs because there are more time windows available, each with a shorter duration, and, in most cases, no additional time windows are introduced. With shorter time windows, drivers tend to allocate a significant portion of their time to traveling between zones rather than serving customers. Additionally, the presence of unserved customers in the solutions mainly depends on the use of Eq. (25) to define the number of time windows per zone rather than being strictly correlated with customers' demand or their distance from the depot. Also, it is possible that, in some zones for certain scenarios, a higher number of customers may appear with service times that exceed the average values. In such a case, the probability of not serving all customers increases because we allow at most a time window per day per zone, i.e., time constraint is more binding. In terms of computing time, we note that this does not increase as the time window width decreases. When very few time windows are offered, increasing their number can indeed affect the computing time because the number of global schedules increases as well. Beyond a certain point, adding more time windows reduces the computing time, as there are fewer global schedules.

To conclude, the insight we can gain from these observations is that offering shorter time windows (e.g., 1.5-h width) may increase customer satisfaction while reducing drivers' free time. However, it would also increase the number of unserved customers and the distance traveled.

Impact evaluation of daily working hours and driver availability

The quality of customer service is not only influenced by the quantity and the duration of the time windows, it can also be impacted by unforeseen events, such as the abrupt unavailability of a driver or the impossibility of completely exploiting all available daily working hours. Conversely, a company may also be interested in assessing the potential improvements that could be achieved by hiring additional drivers or increasing their daily working time, always complying with existing regulations. The purpose of these experiments, summarized in Table 11, is to show what would happen by varying the number of working hours or drivers available daily.

In Table 11(a), the first column indicates the change in the number of daily working hours whereas the remaining ones have the usual

Table 12
Results on real-world data corresponding to `Class C` instances.

	Edmonton (49 postal codes)				Calgary (44 postal codes)			
TW configuration	TWs (#)	Unserved customers (#)	Distance	Dist. (Δ)	TWs (#)	Unserved customers (#)	Distance	Dist. (Δ)
No TWs	–	0.003	3654.0	−23.9%	–	0.000	3602.5	−19.7%
3 h	81	0.590	4172.6	−13.1%	81	0.048	4008.3	−10.7%
3 h+1	129	0.642	4818.9	0.4%	125	0.050	4341.6	−3.2%
3 h+2	175	0.503	5141.1	7.1%	166	0.040	4536.6	1.1%
3 h+3	216	0.463	5169.8	7.7%	201	0.030	4645.2	3.5%
3 h+4	245	0.433	5180.8	8.0%	220	0.022	4756.4	6.0%
4 h	65	0.545	4379.8	−8.7%	69	0.035	3862.7	−13.9%
4 h+1	114	0.742	4474.5	−6.8%	113	0.005	4246.5	−5.4%
4 h+2	162	0.612	4759.8	−0.8%	157	0.062	4494.3	0.2%
4 h+3	208	0.363	5067.8	5.6%	197	0.022	4624.2	3.1%
4 h+4	245	0.350	5044.4	5.1%	220	0.013	4709.4	5.0%
6 h	58	0.797	4218.1	−12.1%	53	0.098	3737.5	−16.7%
6 h+1	107	0.537	4547.7	−5.2%	97	0.082	4137.6	−7.8%
6 h+2	156	0.510	4866.6	1.4%	141	0.050	4400.6	−1.9%
6 h+3	204	0.468	4908.7	2.3%	185	0.007	4560.0	1.6%
6 h+4	245	0.313	4982.2	3.8%	220	0.013	4613.2	2.8%
1 All-Day	50	0.405	3848.4	−19.8%	44	0.075	3754.8	−16.3%
2 All-Day	99	0.595	4299.5	−10.4%	88	0.052	4093.6	−8.8%
3 All-Day	148	0.552	4620.8	−3.7%	132	0.093	4252.1	−5.2%
4 All-Day	197	0.415	4712.5	−1.8%	176	0.005	4410.4	−1.7%
5 All-Day (Company)	**245**	**0.348**	**4798.9**	**0.0%**	**220**	**0.015**	**4486.9**	**0.0%**

meaning. We can observe a trade-off between the number of unserved customers and the distance deviation. As expected, a reduction in the number of daily working hours leads to a higher average number of unserved customers as the traveled distance decreases. Conversely, the addition of extra time yields intriguing findings. Even with a two-hour daily extension, there is still one unserved customer on average. Anyway, the distance deviation is lower when adding two hours compared with the value observed when adding just one hour. Table 11(b) has a similar structure. The only difference is in the first column, which contains the change in the number of drivers available daily. The observations we can make are similar to those from Table 11(a). By considering both analyses together, we derive the insight that it is preferable to hire more drivers than make the current ones work one or two hours more each day.

6.6. Application to real-world data

The focus of this last part is to apply the Perturbation meta-heuristic and evaluate its performance on the real-world data of the application motivating our study.

For each possible time window width (3, 4, 6 h, and All Day), we compute the number of time windows by applying Eq. (25). By using these values, we notice that several postal codes requiring a few working hours obtain only a single time window. Thus, to give more choices to customers, we test the case with 1, 2, 3, and 4 additional time windows per each zone. For example, "3 h+2" refers to the number of time windows obtained using Eq. (25) with a width of 3 h plus two additional time windows. A *No Time Windows* test is made to know the minimum possible cost. All tests are run for 36 h, and all distances are computed by using the Haversine formula.

Results are presented in Table 12. The first column is the time window configuration. For each delivery region, the table shows the total number of time windows, the expected number of unserved customers, the expected distances, and their percentage deviation compared with the company's approach (last line). The following insights can be derived. (i) Fewer and wider time windows can lead to more efficient routes and reduce travel distances, which can result in cost savings and improve the operational efficiency of the company. With wider time windows, certainly drivers would need to make fewer trips within postal code areas. Consequently, this reduction in the number of trips would lead to cost savings. Furthermore, as the time window size increases, there is greater flexibility and possibility for route optimization. (ii) Limiting the number of delivery dates could be beneficial for the company. Instead of offering any delivery date, the company may choose to propose deliveries on four out of five days. This adjustment could lead to reducing distances by 1.8% in the Edmonton region and 1.7% in the Calgary one. Reducing the number of delivery days would yield more significant decreases. (iii) Reductions in distance traveled can be obtained using a 3, 4, or 6-h time window configuration compared with the company's policy of five delivery days, because fewer trips are made to each postal code. These configurations with one extra time window can also be advantageous. (iv) Choosing the time window setting remains a managerial decision because of a trade-off between transportation costs and customer satisfaction. If the company adopts a different policy by allowing the choice of delivery time windows, it could use the 4h+2 configuration at no additional cost. In this way, customers in the Edmonton and Calgary regions would get $162/49 = 3.3$ and $157/44 = 3.6$, respectively, time windows per week on average.

7. Conclusions

In this paper, we introduce a new problem called the Stochastic Multi-period Time Window Assignment Vehicle Routing Problem. Customers are distributed into a grid of zones, and their locations, demands, and service times are stochastic and evaluated through a set of possible scenarios based on historical data. The aim is to build a schedule that assigns a set of time windows to each zone during a planning period of a week by minimizing the total expected transportation costs plus a penalty cost for each unserved customer. Formulating the problem as a two-stage stochastic program, we propose a solution approach based on a Sample Approximation Average Method, exploiting a perturbation meta-heuristic in the first stage and an Adaptive Large Neighborhood Search framework in the second stage. To evaluate the effectiveness and the efficiency of the proposed solution method, we generate three sets of benchmark instances (two randomly and one from real-world data) with a different number of zones, customers, and scenarios. Results are promising: the approach represents a valuable tool for the industrial case motivating the study.

In future research, the model could consider the number of visits per zone as a variable rather than a parameter. Indeed, the frequency of visits to each zone undoubtedly determines a trade-off between customer satisfaction and traveling costs: larger values would increase customer satisfaction but also traveling costs because of requests disaggregation.

Acknowledgments

This work was partly supported by the Canadian Natural Sciences and Engineering Research Council (NSERC) under grants 2021-00028. We thank the Digital Research Alliance of Canada for providing high-performance computing facilities. We also thank the anonymous referees for their helpful comments that improved the quality of the manuscript.

References

Agatz, N., Campbell, A., Fleischmann, M., & Savelsbergh, M. (2011). Time slot management in attended home delivery. *Transportation Science*, *45*(3), 435–449.

Agatz, N., Fan, Y., & Stam, D. (2021). The impact of green labels on time slot choice and operational sustainability. *Production and Operations Management*, *30*(7), 2285–2303.

Campbell, A. M., & Savelsbergh, M. (2005). Decision support for consumer direct grocery initiatives. *Transportation Science*, *39*(3), 313–327.

Campbell, A. M., & Savelsbergh, M. (2006). Incentive schemes for attended home delivery services. *Transportation Science*, *40*(3), 327–341.

Dalmeijer, K., & Desaulniers, G. (2021). Addressing orientation symmetry in the time window assignment vehicle routing problem. *INFORMS Journal on Computing*, *33*(2), 495–510.

Dalmeijer, K., & Spliet, R. (2018). A branch-and-cut algorithm for the time window assignment vehicle routing problem. *Computers & Operations Research*, *89*, 140–152.

Ehmke, J. F., & Campbell, A. M. (2014). Customer acceptance mechanisms for home deliveries in metropolitan areas. *European Journal of Operational Research*, *233*, 193–207.

Farias, V. F., Jagabathula, S., & Shah, D. (2013). A nonparametric approach to modeling choice with limited data. *Management Science*, *59*(2), 305–322.

Figliozzi, M. A. (2008). Planning approximations to the average length of vehicle routing problems with varying customer demands and routing constraints. *Transportation Research Record*, *2089*(1), 1–8.

Groër, C., Golden, B., & Wasil, E. (2009). The consistent vehicle routing problem. *Manufacturing & Service Operations Management*, *11*(4), 630–643.

Hernandez, F., Gendreau, M., & Potvin, J.-Y. (2017). Heuristics for tactical time slot management: A periodic vehicle routing problem view. *International Transactions in Operational Research*, *24*(6), 1233–1252.

Hoogeboom, M., Adulyasak, Y., Dullaert, W., & Jaillet, P. (2021). The robust vehicle routing problem with time window assignments. *Transportation Science*, *55*(2), 395–413.

Jalilvand, M., Bashiri, M., & Nikzad, E. (2021). An effective progressive hedging algorithm for the two-layers time window assignment vehicle routing problem in a stochastic environment. *Expert Systems with Applications*, *165*, Article 113877.

Kleywegt, A. J., Shapiro, A., & Homem-de Mello, T. (2002). The sample average approximation method for stochastic discrete optimization. *SIAM Journal on Optimization*, *12*(2), 479–502.

Köhler, C., Ehmke, J. F., & Campbell, A. M. (2020). Flexible time window management for attended home deliveries. *Omega*, *91*, Article 102023.

Lin, I. I., & Mahmassani, H. S. (2002). Can online grocers deliver?: Some logistics considerations. *Transportation Research Record*, *1817*(1), 17–24.

Madsen, O. B., Tosti, K., & Vælds, J. (1995). A heuristic method for dispatching repair men. *Annals of Operations Research*, *61*(1), 213–226.

Mak, W.-K., Morton, D. P., & Wood, R. K. (1999). Monte Carlo bounding techniques for determining solution quality in stochastic programs. *Operations Research Letters*, *24*(1–2), 47–56.

Manerba, D., Mansini, R., & Zanotti, R. (2018). Attended home delivery: Reducing last-mile environmental impact by changing customer habits. *IFAC-PapersOnLine*, *51*(5), 55–60.

Potvin, J.-Y., & Rousseau, J.-M. (1993). A parallel route building algorithm for the vehicle routing and scheduling problem with time windows. *European Journal of Operational Research*, *66*(3), 331–340.

Potvin, J.-Y., & Rousseau, J.-M. (1995). An exchange heuristic for routeing problems with time windows. *Journal of the Operational Research Society*, *46*(12), 1433–1446.

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

Savelsbergh, M. (1992). The vehicle routing problem with time windows: Minimizing route duration. *INFORMS Journal on Computing*, *4*(2), 146–154.

Spliet, R., Dabia, S., & Van Woensel, T. (2018). The time window assignment vehicle routing problem with time-dependent travel times. *Transportation Science*, *52*(2), 261–276.

Spliet, R., & Desaulniers, G. (2015). The discrete time window assignment vehicle routing problem. *European Journal of Operational Research*, *244*(2), 379–391.

Spliet, R., & Gabor, A. F. (2015). The time window assignment vehicle routing problem. *Transportation Science*, *49*(4), 721–731.

Stenger, A., Vigo, D., Enz, S., & Schwind, M. (2013). An adaptive variable neighborhood search algorithm for a vehicle routing problem arising in small package shipping. *Transportation Science*, *47*(1), 64–80.

Subramanyam, A., Wang, A., & Gounaris, C. E. (2018). A scenario decomposition algorithm for strategic time window assignment vehicle routing problems. *Transportation Research, Part B (Methodological)*, *117*, 296–317.

Yu, X., Shen, S., Badri-Koohi, B., & Seada, H. (2023). Time window optimization for attended home service delivery under multiple sources of uncertainties. *Computers & Operations Research*, *150*.