CIRRELT

Centre interuniversitaire de recherche
sur les réseaux d'entreprise, la logistique et le transport

Interuniversity Research Centre
on Enterprise Networks, Logistics and Transportation

Tactical Time Window Management in Attended Home Delivery

Jean-François Côté
Renata Mansini
Alice Raffaele

March 2019

CIRRELT-2019-09

Document de travail également publié par la Faculté des sciences de l'administration de l'Université Laval, sous le numéro FSA-2019-005.

Bureaux de Montréal :
Université de Montréal
Pavillon André-Aisenstadt
C.P. 6128, succursale Centre-ville
Montréal (Québec)
Canada H3C 3J7
Téléphone : 514 343-7575
Télécopie : 514 343-7121

Bureaux de Québec :
Université Laval
Pavillon Palasis-Prince
2325, de la Terrasse, bureau 2642
Québec (Québec)
Canada G1V 0A6
Téléphone : 418 656-2073
Télécopie : 418 656-2624

www.cirrelt.ca

Tactical Time Window Management in Attended Home Delivery

Jean-François Côté[1,*], Renata Mansini[2], Alice Raffaele[3]

1. Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT) and Department of Operations and Decision Systems, 2325 de la Terrasse, Université Laval, Québec, Canada G1V 0A6
2. Department of Information Engineering, University of Brescia, Italy
3. Department of Mathematics, University of Trento,Italy

Abstract. We study a multi-period stochastic variant of the Time Window Assignment Vehicle Routing Problem where customer demands, their locations, and service times are all non-deterministic. Customers are geographically distributed into zones, and each zone has to be visited a given number of times within specified time windows in a defined time horizon ({e.g.,} five working days). A fleet of homogeneous vehicles is available to serve customers at their homes. At a tactical level, the problem looks for the time windows to assign to zones over a time horizon to minimize the expected traveling costs required by vehicles to visit the customers plus expected penalty costs for unserved ones. We call this problem the Stochastic Multi-period Time Window Assignment Vehicle Routing Problem (SMTWAVRP). We propose a two-stage formulation, and a solution approach encompassing a perturbation method to manage time windows assignment to zones in the first stage, and an Adaptive Large Neighborhood Search (ALNS) framework to optimize routes in the second one. Computational results conducted over a large set of instances (including some real ones provided by a Canadian company) show that the solution method is effective at obtaining good schedules and outperforms the manual solution obtained by the company.

Keywords. Time window assignment, vehicle routing problem, multi-period, stochastic programming.

Acknowledgements. This work was partly supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) under grants 2015-04893. We thank Calcul Québec for providing high performance parallel computing facilities.

---

* Corresponding author: Jean-Francois.Cote@cirrelt.ca

Dépôt légal – Bibliothèque et Archives nationales du Québec
Bibliothèque et Archives Canada, 2019

1 Introduction

In industrial distribution, one of the main goals is to manage every phase of the logistic process, trying to efficiently handle critical resources like time. Beyond production, efficiently time management is essential above all in the last-mile delivery, when goods or services are provided to end users. Usually, a company tries to accomplish customer requests guaranteeing a high level of satisfaction and, simultaneously, to optimize the shipping in terms of costs. The availability of an effective plan becomes therefore a priority, to optimally settle the delivery service in a daily or weekly horizon. To build an optimal schedule, the company must know exactly how many customers have to be served, where they are located and the amount of their orders. Unfortunately, in many real applications as the home delivery following an e-commerce purchase, such information is not known with certainty in advance, and only statistics can be used (e.g. probability distributions on past demands and their locations). In all such cases, finding the optimal schedule is a too complex task, and the goal turns into looking for a schedule which has a good behavior in every eventuality.

In this paper, we consider a multi-period stochastic variant of the Time Window Assignment Vehicle Routing Problem introduced by Spliet and Gabor [17] and where time windows have to be assigned to customers before demand is known. Our problem differs from the latter in many aspects. First of all, it is reasonable for a company to deliver goods to customers who are in the same streets at the same time, avoiding if possible to come back to the same place in another moment. For this reason, we assume that the geographical area of interest is divided into *zones* defined by zip codes, and consequently customers are classified, according to their locations, as belonging to a defined zone. In some cases, customers explicitly ask for deliveries in some particular hours of the day or in a specific day of the week, and may choose among delivery alternatives offered by the company, paying more or less according to the option selected. In some other cases, as in the one we are analyzing, customers do not much care about the delivery time, and may accept to be served in a time window decided by the supplier. In practice, the company defines some intervals of time that are assigned to each zone, during which customers belonging to that area can be supplied. Zones can have a different number of time windows, depending on historical customer demands. Moreover, according to the type of goods, the delivery service can be performed over a *time horizon* of a single day, of a week or a month. In our case, we consider an horizon of five working days, and the customers typically change week after week. Time windows are assigned to each zone before knowing not only their requests (demand) but also their location. The assignment decides time windows distribution over the working days of the time horizon. If a customer cannot be served during the schedule of its zone, a penalty has to be paid, corresponding to the cost of a backup delivery accomplished in outsourcing by another company.

Given the zones into which the geographical area is divided, the Stochastic Multi-period Time Window Assignment Vehicle Routing Problem (SMTWAVRP) decides how to assign a predefined number of time windows (selected from a candidate set) to each zone, while minimizing the expected traveling costs required to visit the customers within the allocated time windows plus the expected penalty costs associated with non-served customers. As far as constraints are concerned, the decision

maker, once he has decided the set of possible time windows for each zone, establishes how many times a zone has to be visited during the planning horizon (to provide a guaranteed service level), and imposes that a zone can be visited at most once per day. Favoring more visits to a zone may cause a disaggregation of customers which, in turn, increases company costs, but improves customer satisfaction by allowing more choices. Inversely, setting a smaller number of visits forces aggregation which decreases costs, but also reduces customer satisfaction. This trade-off is decided by the decision maker in advance on the basis of available historical data concerning previous purchases and deliveries. Past information is provided in the form of probability distributions on the number of customers, their demands and service times for each zone. These distributions allow the creation of a finite number of scenarios representing realizations of the described random variables. At the moment of time windows assignment, the supplier knows the probability with which each scenario occurs.

On the contrary, the supplier does not have any information on the preference of his/her customers on when they want their goods to be delivered. Considering customer preferences may lead to an over-complicated approach, as the probability of selecting a specific time window would also depend on the ones that are offered. To deal with this lack of information on preferences, we assume that customers select their time window randomly among the available ones. No cancellation and rescheduling can happen. Finally, a fixed fleet of vehicles, partitioned over the periods, is available to make the deliveries.

Motivation

The problem has been motivated by a real case study of a Canadian retailer that sells and delivers large and heavy items such as furniture and appliances. One of the most unsatisfactory aspects of home delivery service is frequently related to the way it is planned. There are some companies that ask their customers to be present at home for an entire day in order to receive required services, thus causing frustration and disappointment. To increase customers satisfaction some companies try to provide shorter time windows. Nowadays, several companies (including the one of our case study) offer weekly schedules to their customers specifying day of the week and time window in which the service can be accomplished. In our real case, customers can choose any delivery date successive to the day in which requested furniture is made available at the company depot. Since that moment, customers will receive their goods in the next 3-5 days. Two days prior to the delivery date, customers receive a call from an automatic calling machine informing them that the their delivery will occur in a specific 3 hour time window. According to the retailer, cancellation and rescheduling occur rarely.

Another reason that motivated our work was that, at present, delivery schedules are made manually by using maps and push pins and the process requires the employment of several people, and takes many working days to be completed. Moreover, once obtained, the schedule is usually reused by the company for several months. The drawbacks of this procedure are evident. The most important one concerns the sensitivity lack of this handmade schedule to demand changes. A

strong growth in demand might increase the lateness of this schedule, and thus lowering customer satisfaction. On the contrary, a demand decrease might render the current schedule inefficient which in turn will increase costs. The main fallout of this work will be the replacement of the manual planning process used by the company with a computer-based one focused on advanced algorithms. Being an automatic approach, the company will be able to rapidly modify the current schedule or even change it every week.

Contributions

The paper provides several contributions. The vehicle routing problem of serving a set of customers, after assigning time windows and in presence of stochastic information, has been already treated in the literature, but never jointly considering the geographical aspect of dealing with zones, the multi-period nature of the time window assignment problem and the possible loss of earnings (measured as penalty cost) in case customers are not serviced. This is the first relevant contribution. It is worth underlying that the problem considers a double level of optimization, since time windows are assigned to zones to which customers belong, but then the routes have to be built on customer locations, not on zones. Moreover, whereas in the literature only demand is stochastic in our case customers location and service times are also non-deterministic. Finally, to tackle the problem, we introduce a two-stage stochastic formulation with general recourse. The first-stage deals with the assignment of time windows to zones, whereas the second-stage, when demands and their locations are already revealed, solves a variant of the VRP with Time Windows (VRPTW) where customers can be served in different days, and within the time window assigned to their zone in the day of the service. We also propose effective metaheuristics for the solution of each of the two stages. Provided solutions largely improve the ones obtained by the manual approach presently used in the company.

The paper is organized as follows. In Section 2, we summarize main literature on attended home delivery problems and known variants of the Time Window Assignment Vehicle Routing Problem (TWAVRP). In Section 3, we provide the formal description of the problem, its two-stage definition and its deterministic equivalent formulation. Section 4 is devoted to the methodology used to tackle the problem, whereas full details on the algorithms can be found in Section 5, where we examine in depth the solution method developed for each stage. In particular, a perturbation procedure is developed for the first stage problem, whereas an ALNS with several destroy and repair methods is designed for the routing problem in the second stage. In Section 6, we describe the design of the computational study, present extensive results, and derive some managerial insights. A specific focus is posed on the analysis of the stochastic value of the solution, and of the quality of the solution service. The section terminates with the discussion of the results obtained on real instances. Finally, Section 7 draws main conclusions, and identifies possible future developments.

2 Literature review

Different lines of research can be associated with last-mile delivery. In the past, a main focus was put on quantifying factors that may influence on-line shipping with respect to traditional one (Lin and Mahmassani [9]) or on analyzing accept/refuse policies for attended home delivery customers to decide if a delivery request can be accommodated or not (Bent and van Hentenryck [2]; Ehmke and Campbell [5]). Another relevant issue was the introduction of incentive schemes to encourage customers to select time slots that could lead to lower transportation costs for the supplier (Campbell and Savelsbergh [4]) or the setting of different delivery prices to influence customer behavior about the offered time slots, so to maximize overall profit (Klein *et al.* [10]). Recently, Manerba et al. [11] have analyzed how the selection of restrictive time windows for home delivery affect last-mile routing operations and negatively impact on environment.

A relevant number of contributions have been published on static management of time slots. The *Time Slot Management in Attended Home Delivery* (TSMP), tackled by Agatz et al. [1], deals with *time slots* offered in the zip codes of a service region (zones). An expected number of customers is associated with each geographical zone. The TSMP allocates one time slot for each zip code. Demand is measured in terms of customer orders over a time horizon of a week, therefore the expected amount for each zip code is known. Split delivery is allowed and time slots cannot overlap among different zones. To minimize the expected delivery costs, the authors use two different approaches. The first one is a continuous approximation model used to estimate the delivery cost of a given time slot schedule assigned to a set of zip codes. The second approach solves 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 studied, by developing heuristics and meta-heuristics, in Hernandez *et al.* [7], where the problem is formulated as a *Periodic* VRP (PVRP) and each zone has a service frequency. More precisely, each customer is associated with a geographical zone and every zone is represented with a square, whose side and center depend on the coordinates of customers' location. Demand is known and the goal is to select a particular schedule for each zone, assigning zones to vehicles on any given day of the time horizon, trying to minimize the total travel cost. The classical formulation of the PVRP considers several days where customers must be visited more times; in [7], the number of visits is associated with zones and not with customers, as in our case: in the SMTWAVRP, orders of the customers are supplied only once during the time horizon and thus are not periodic. Customers are grouped into geographical zones, the problem is defined over a discrete time horizon and a time slot must be associated with each zone, which can be visited more than once according to the decisions taken by the planner. However, information is deterministic, the demand and time service depend on the zone and do not change over the time horizon (demand uniformity) for each zone. Finally, the authors do not address the construction of delivery routes based on real customer orders but they are seen as sequences of zones. The authors solve the problem by means of a Tabu Search, where neighborhood structures are defined using simple moves, such as the removal of a zone from a route in a certain time period and the reinsertion in another route.

They designed two heuristics to solve the problem directly or with a decomposition approach. To compare performance results, they generated a new set of benchmark instances.

Two important contributions on Time Window Assignment Vehicle Routing Problem (TWAVRP) are due to Spliet and Gabor [17] and to Spliet and Desaulniers [18]. In both these papers, the customers are known and only their demand is stochastic, modeling all practical situations where supplier knows his/her customers, and the day in which each of them has to be periodically served, but demands may fluctuate. In [17], the authors establish that the time windows can start at any time within a predefined exogenous time window decided by the customer. Since customers are preassigned to each day, computing the expected total transportation cost is equivalent to minimize the cost for each single day. The problem 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 also propose a mixed integer linear programming model and solve its relaxation (allowing non elementary routes) with a Column Generation algorithm, finding lower bounds by solving one pricing problem for every scenario. Finally, a branch-and-cut-and-price algorithm is introduced by separating some valid inequalities. In the DTWAVRP [18], for each customer there is instead a discrete set of candidate time windows, from which just one has to be selected, and the time horizon is composed of several days.

We innovate with respect to these contributions by assuming that both customers (with their demands) and their locations are unknown, as it typically happens in an e-commerce context. As in [17] and [18], also in our case the time windows have a predefined width, and following [18] we assume that a discrete set of candidate windows is available for each zone. However, differently than in [18], time windows are not associated with customers but zones. Finally, given the multi-period nature of the problem we cannot optimize the expected distance day by day as in the existing literature.

3 Problem formulation

Let us consider a time horizon $T = \{1, \ldots, \tau\}$ of τ periods (working days), within which customers demand has to be satisfied. 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, t_w, z_w) where l_w and u_w are the starting and ending times, whereas $t_w \in T$ and z_w are the period (day) of the time horizon in which the time window is available and its zone, respectively. Every zone z requires n_z visits, corresponding to the selection of n_z time windows to be scheduled over the time horizon. We indicate as $W_z(t)$ the subset of candidate windows of W_z offered by zone z in period t. At most one time window can be selected per zone per period, i.e. every zone is visited at most once per day. We call *zone schedule* the assignment for a given zone of the time windows over the time horizon, and *global schedule* the set of all zone schedules. We say that a global schedule is *feasible* if it has at most one time window per day per zone. The selection of zone

schedules is guided by the cost of the corresponding delivery plan as set of routes accomplishing the global schedule. Once the zone schedules are decided, delivery routes of minimum cost are constructed for each day of the horizon by using estimates of the demand and service time in each zone. The objective is to plan the global schedule while minimizing the expected transportation cost and the penalty cost paid for unserved customers.

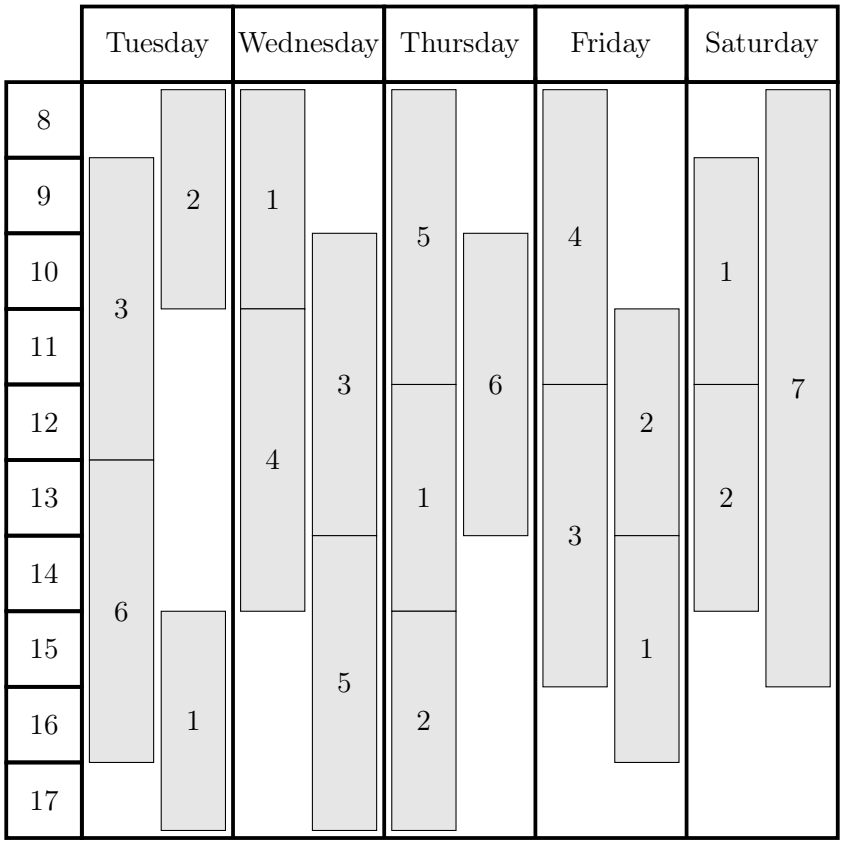

Figure 1: Example of a schedule

Figure 1 shows an example of a global schedule characterized by 7 zones having time windows that span from 8:00 a.m. to 5:00 p.m. over a time horizon of 5 days (from Tuesday to Saturday). Zone 1 is visited every day inside time windows of 3 hours each and that are differently positioned during each day; zone 2 is visited 4 times and has time windows of 3 hours each, zone 3 is visited on Tuesday, Wednesday and Friday with time windows of 4 hours each, whereas zones 4, 5 and 6 are visited 2 times and have time windows of 4 hours. Finally, zone 7 can represent a foreign region and is visited only on Saturday with a 8 hours time window.

The SMTWAVRP can be formally defined as follows. Let ξ be a vector of random variables corresponding to customer locations, their demands and service times. Each realization of ξ is called a *state of the world* or *scenario*. We assume that the support of ξ is finite and we indicate as S the set of all possible scenarios, where each scenario s represents a realization of future demands intended as customers with their locations, quantities to be delivered, service times and the zones they belong to. Each scenario s has a probability p_s to occur. Consider the complete graph $G_s = (V_s, A_s)$

associated with scenario s. $V_s = N_s \cup \{0, n_s + 1\}$ is the set of customers $N_s = \{1, \ldots, n_s\}$ making a request under scenario s, plus the starting depot 0 and the ending one $n_s + 1$. A 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 d_i units of demand to be delivered, and its service time is s_i. A customer can only be served in one of the time windows associated with his/her zone. A fixed number of identical vehicles are available. Each vehicle has a capacity Q and must depart at time l_0, and come back to the depot before time u_0. Let K be the set of vehicles. Since their use has to be planned in each day and assigned to subcontractors, we indicate as $K(t) \subseteq K$ the subset of vehicles available in period $t \in T$.

The SMTWAVRP can be formulated as a two-stage stochastic program. Let y_{zw} be a first-stage binary variable indicating if the time window $w \in W_z$ is selected for zone $z \in Z$. We also denote as $\mathbb{E}[C(\mathbf{y}, \xi)]$ the expected cost value of the second-stage problem $C(\mathbf{y}, \xi)$, which is a nested optimization problem aimed at deciding the routing to serve customers, depending on the time windows assignment decisions $\mathbf{y}$ and on the realization of the random variable ξ. The formulation of the SMTWAVRP is as follows:

$$\min \mathbb{E}[C(\mathbf{y}, \xi)] \tag{1}$$

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

$$\sum_{w \in W_z(t)} y_{zw} \leq 1 \qquad z \in Z, t \in T \tag{3}$$

$$y_{zw} \in \{0, 1\} \qquad z \in Z, w \in W_z \tag{4}$$

The objective is to minimize the expected cost of the recourse. Constraints (2) ensure that each zone has its required number of time windows, and constraints (3) make sure that at most one time window is selected for every zone in each period. The evaluation of the recourse $\mathbb{E}[C(\mathbf{y}, \xi)]$ for a given solution $\hat{\mathbf{y}}$ can be written as:

$$\mathbb{E}[C(\hat{\mathbf{y}}, \xi)] = \sum_{s \in S} p_s C(\hat{\mathbf{y}}, s) \tag{5}$$

This requires the solution of the following variant of the VRPTW problem for each scenario $s \in S$:

$$C(\hat{\mathbf{y}}, s) = \min \sum_{k \in K} \sum_{(i,j) \in A_s} c_{ij} x_{ij}^k + \beta \sum_{i \in N_s} z_i \tag{6}$$

$$\text{s.t.} \sum_{k \in K(t_{\hat{w}_i})} z_{ik} + z_i = 1 \qquad i \in N_s \tag{7}$$

$$\sum_{j \in V_s} x_{ij}^k = z_{ik} \qquad i \in N_s,\ k \in K(t_{\hat{w}_i}) \tag{8}$$

$$\sum_{j \in V_s} x_{ij}^k = \sum_{j \in V_s} x_{ji}^k \qquad i \in N_s,\ k \in K(t_{\hat{w}_i}) \tag{9}$$

$$\sum_{i \in V_s} x_{0i}^k \leq 1 \qquad k \in K \tag{10}$$

$$\sum_{i \in V_s} \sum_{j \in V_s} d_j x_{ij}^k \leq Q \qquad k \in K \tag{11}$$

$$a_j^k \geq a_i^k + (s_i + t_{ij}) x_{ij}^k - M(1 - x_{ij}^k) \qquad k \in K,\ (i,j) \in A_s \tag{12}$$

$$a_i^k \leq u_{\hat{w}_i} + M(1 - z_{ik}) \qquad i \in N_s,\ k \in K(t_{\hat{w}_i}) \tag{13}$$

$$a_i^k \geq l_{\hat{w}_i} - M(1 - z_{ik}) \qquad i \in N_s,\ k \in K(t_{\hat{w}_i}) \tag{14}$$

$$z_i, z_{ik} \in \{0, 1\} \qquad i \in N_s,\ k \in K \tag{15}$$

$$x_{ij}^k \in \{0, 1\} \qquad (i,j) \in A_s,\ k \in K \tag{16}$$

$$a_i^k \geq 0 \qquad i \in N_s,\ k \in K \tag{17}$$

In the model, $\hat{w}_i$ is the time window that was chosen by customer i among the ones defined by assignment $\hat{\mathbf{y}}$, and $K(t_{\hat{w}_i})$ is the set of the available vehicles on delivery day $t_{\hat{w}_i}$. Binary variable z_{ik} takes value 1 if customer i is visited by driver k, whereas binary variable z_i is equal to 1 if customer i is not visited, and a penalty β has to be paid. Let also x_{ij}^k be a binary variable taking value one if vehicle k traverses the arc $(i,j) \in A_s$. Finally, continuous variable a_i^k indicates the arrival time of vehicle k in node i. The objective (6) of the recourse problem is to minimize the costs of routing and of unserved customers. Allowing customers to not be served enables us to have a *complete recourse*, that is, having a feasible second-stage solution for any first-stage solution. The higher the value of β, the higher the importance of serving customers with respect to the one of minimizing routing costs. Constraints (7) impose that each customer i in scenario s is served by one of the vehicles working in day $t_{\hat{w}_i}$ or the penalty β has to be paid ($z_i = 1$). Constraints (8) state that if node i is visited by vehicle k the latter has to enter the node. Constraints (9) are the flow conservation equalities. Constraints (10) ensure that at most one route is used by vehicle k. Constraints (11) impose that capacity of each vehicle is not violated. Constraints (12)-(14) guarantee schedule feasibility with respect to time windows. In particular, the inequalities (12) state that if a vehicle k is traveling from i to j, it cannot arrive at j at a time a_j^k lower than the time of arrival in node i (a_i^k) plus the traveling time from i to j and the service time in i. Here, M is a large constant. Finally, constraints (15)–(17) are binary and nonnegative conditions on variables. The problem is NP-hard.

4 Methodology

The complexity of solving the SMTWAVRP is mainly related to formulation (1)–(15) and the number of scenarios involved. The solution of this two-stage stochastic problem with binary variables is impractical for small instances and nearly impossible for real-life applications. For example, an instance with 25 delivery zones and 30 possible customers for each zone will lead to 25^{30} scenarios.

To tackle this difficulty, we decided to use the *Sample Average Approximation Method* (SAAM) proposed by Werweij et al. [19]. This method solves stochastic optimization problems by using Monte Carlo simulation. A sample of the scenarios is generated to approximate the recourse function. Let $S_p = \{s^1, \ldots, s^p\}$ be a sample consisting of p scenarios. At each step of the resolution process, for a given first-stage solution $\hat{\mathbf{y}}$, the recourse problem is solved for each sample scenario by using a deterministic solution procedure. Afterwards, the real recourse solution value is approximated by calculating the average out of all the deterministic solution values obtained for scenarios in the sample:

$$C(\hat{\mathbf{y}}, S_p) = \frac{1}{p} \sum_{i=1}^{p} C(\hat{\mathbf{y}}, s^i). \tag{18}$$

We also define the *Sample Average Approximation Problem* as follows:

$$C^*(S_p) = \min_{\mathbf{y} \in Y} \{f(\mathbf{y}, S_p)\} \tag{19}$$

where Y is the set of first-stage solutions. It is well known that the approximation converges to the true optimal value as the size of the sample S_p grows to infinity. Unfortunately, there is a trade-off between solution quality and computational time to find it. One can expect that a high number of scenarios will provide a solution of high quality at the cost of a very long resolution time. On the contrary, a low number of scenarios gives a poor solution, but very quickly. This trade-off is analyzed in Section 6.4. To our knowledge, very few papers have used SAAM strategy in the context of multi-period delivery problems. Most of them are related to some *Dynamic and Stochastic Vehicle Routing Problems*, where the time horizon is usually of a single day and every intra day event requires to make new decisions.

In our case, the quality of the final solution is hard to evaluate because of the significant complexity of the SMTWAVRP. Instead of relying on exact algorithms, we propose several ways to heuristically solve the first-stage and then approximate the expected cost providing practical rules of thumbs for the decision maker. A way to approximate the real expected cost is to take the final solution $\mathbf{y}^*$ of the first-stage problem, and calculate the expected value of the second-stage on a set S_q of q scenarios. An approximation is thus obtained by calculating $C(\mathbf{y}^*, S_q)$. This idea is very similar to the concept of *value of the stochastic solution* where one wants to evaluate the potential benefit of using a stochastic programming approach (Birge and Louveaux [3]). By changing the parameter settings in the solution algorithms of the first stage, we obtain different first-stage solutions that can be compared in terms of approximation and may provide to the decision maker a valuable insight on the estimate of future transportation costs.

Our overall methodology is the following. First, a set of scenarios is generated by using some distribution functions differentiated according to the zones. An initial first-stage solution $\mathbf{y}$ is created from this sample by using a construction heuristic. Then an improvement phase is applied possibly terminating with a new solution $\bar{\mathbf{y}}$. Each time a new first-stage solution is visited, we calculate the cost of the second-stage by solving all scenarios of the sample with a heuristic. The best solution found $\mathbf{y}^*$ of this phase is finally returned with the computation of an estimate of the traveling costs.

5 Solution algorithms

In the following, we analyze the implemented two-stage algorithms in detail. First, we describe how the initial solution is built. Then, we present various approaches (a basic local search and a perturbation metaheuristic with its simple multi-start variant) for improving the first-stage initial solution. Finally, the ALNS approach used for solving the second-stage is analyzed. We recall that a solution of the first-stage corresponds to a feasible global schedule. A solution of the second-stage provides the optimal routing over an assigned global schedule.

5.1 First-stage

This section describes the algorithms that are used to create and improve the first-stage solutions. Our overall objective is to develop algorithms around a small and simple neighborhood to avoid calculating the cost of second-stage too many times. A vast and complex neighborhood is likely to be too expensive to compute. All our proposed algorithms use the neighborhood characterized by a move that change the assignment of a time window from the current time moment to another one over the time horizon. The quality of a move is assessed through the cost of second-stage. This neighborhood is small enough, and it is not too cumbersome to explore.

Our solution framework first creates an initial global schedule. Then, this first-stage solution is improved by using a Perturbation heuristic (Perturbation), and its Multi-Start variant (MsPerturbation). Both Perturbation and MsPerturbation make use of the same local search as a building block. Such a local search (procedure LocalSearch) works by relocating time windows over the time horizon. The pseudo-code of the procedure is presented in Algorithm 1. The method receives as input a global schedule H. At each step, an attempt is made for replacing a current time window w in H with another one in W_{w_z} (i.e. one belonging to the zone of the time window) that decreases the cost of the second-stage. For each move, we always ensure that the visited solution has no more than one time window per period per zone (feasible global schedule). The algorithm is run until no further improvement can be found. Function UpdateSchedule(H, w, w') substitutes current time window w with w' inside global schedule H, whereas function OptimizeRoutes(H) solves the second-stage problem by computing the routes visiting the customers according to the time windows assigned in the global schedule H.

Algorithm 1 LocalSearch

Require: a global schedule H

$find \leftarrow true$
$H^* \leftarrow H$
while find **do**
  $find \leftarrow false$
  for each time window $w \in H$ **do**
    $w^* \leftarrow w,\ f^* \leftarrow \infty$
    for each time period $t \in T$ **do**
      for each time window $w' \in W_{z_w}(t)$ **do**
        $H^{'} \leftarrow$ UpdateSchedule(H, w, w')
        $f(H') \leftarrow$ OptimizeRoutes(H')
        if H' is feasible **and** $f(H') < f^*$ **then**
          $f^* \leftarrow f(H')$, $w^* \leftarrow w'$
          $find \leftarrow true$
        end if
      end for
    end for
    $H^* \leftarrow$ UpdateSchedule(H, w, w^*)
    $f(H^*) \leftarrow$ OptimizeRoutes(H^*)
  end for
  $H \leftarrow H^*$
end while
Return H^*

5.1.1 Initial solution

The constructive method used to obtain a first-stage initial solution is presented in Algorithm 2.

A global feasible schedule H is created by assigning to each zone $z \in Z$, exactly n_z time windows, each of which occurs in a random period not already assigned to another time window of the same zone z (Lines 2-8, Algorithm 2). Each time window is represented by a tuple (l, u, t, z) where initial and ending times l and u are initialized to l_0 and u_0, whereas t is the period to which the window belongs, and z indicates its zone. To make sure none of the assigned n_z time windows is on the same period, an array D with numbers from 1 to τ is created and then shuffled (function SHUFFLE in Line 3, Algorithm 2). The period of the first n_z time windows will be equal to the first n_z entries in array D. The corresponding time windows are added to the global schedule H (function ADD) up to the point in which exactly n_z time windows have been assigned to each zone.

At this point, the procedure tries to improve the period-zone assignment of the current schedule H (Lines 9-20, Algorithm 2) in such a way that time windows of nearby zones are in the same period. First, a loop iterates through all the time windows. For each time window w, the values f^* and t^* are equal to the cost and period of the best move. At each iteration, each time window w is moved to a temporary period t', and values f^* and t^* are updated if there is no other time window of the same zone in period t' and if the cost of the second-stage f' is lower than f^*. Time window w is then moved to the best period found t^* (Line 19, Algorithm 2).

Finally, in Lines 9–33, each time window is evaluated on all possible periods and starting times, in order to obtain the largest decrease in costs of the second-stage. At the beginning, a copy of the current solution is stored in H^* and the method tries to substitute each time window w in the current first-stage solution with a feasible time window $w' \in W_z$, i.e. reducing its duration (in Line 25, the algorithm updates the schedule by substituting an all-day time window w with a restricted one w', thus narrowing it to a proper duration), and updating the values of f^* and w^* when improving the cost over w^*. At the end, w is replaced by the best w^* found and the final global schedule H^* is returned as a solution.

Algorithm 2 InitialSolution

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: **end for**
8: **end for**
9: **for each** time window $w = (l, u, t, z) \in H$ **do**
10: $f^* \leftarrow \infty$, $t^* \leftarrow t$
11: **for each** period $t' \in T$ **do**
12: $w' \leftarrow (l, u, t', z)$
13: $H' \leftarrow$ UpdateSchedule(H, w, w')
14: $f(H') \leftarrow$ OptimizeRoutes(H')
15: **if** H' is feasible **and** $f(H') < f^*$ **then**
16: $f^* \leftarrow f(H')$, $t^* \leftarrow t'$
17: **end if**
18: **end for**
19: $w \leftarrow (l, u, t^*, z)$
20: **end for**
21: $H^* \leftarrow H$
22: **for each** time window $w = (l, u, t, z) \in H$ **do**
23: $w^* \leftarrow w$, $f^* \leftarrow \infty$
24: **for each** time window $w' \in W_z$ **do**
25: $H' \leftarrow$ UpdateSchedule(H, w, w')
26: $f(H') \leftarrow$ OptimizeRoutes(H')
27: **if** H' is feasible **and** $f(H') < f^*$ **then**
28: $f^* \leftarrow f(H')$, $w^* \leftarrow w'$
29: **end if**
30: **end for**
31: $H^* \leftarrow$ UpdateSchedule(H^*, w, w^*)
32: $f(H^*) \leftarrow$ OptimizeRoutes(H^*)
33: **end for**
34: **Return** H^*

5.2 Metaheuristics

This section describes the algorithm Perturbation, and its variant MsPerturbation. They both escape local optima by using a perturbation that consists in moving a number of time windows randomly selected to the cheapest periods different from the current ones. The *intensity* of perturbation is controlled by a parameter α_1 that establishes the number of time windows to move. Procedure LocalSearch is then run on each perturbed solution.

Metaheuristic Perturbation (pseudo-code shown in Algorithm 3) runs over a maximum number of $itMax$ iterations. At the beginning of each iteration, the best first-stage solution H^* found so far is copied into the current solution H (Line 3, Algorithm 3). Perturbation intensity is initialized

to a low value ($\alpha_1 = 1$), and then adjusted dynamically. More precisely, the solution is perturbed by means of two inner loops (Lines 4-18, Algorithm 3) by using the two parameters α_1 and α_2. While α_1 works similarly to the radius of a Variable Neighborhood Search deciding the number of time windows to move and increasing when no better solution can be found, on the contrary parameter 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. It is possible that *MoveTWs* finds an improving solution. Afterwards, the local search is applied to find a local minimum (Line 12, Algorithm 3). If the new solution found by the local search has improved over the best solution, the strength level is set back to its initial value and the perturbation continues. If not, the intensity is increased. When α_1 reaches its maximum level γ_1, then the search is restarted and the best solution is copied again into the current solution.

The incumbent solution is updated every time a new best solution is found (Lines 9 and 15, Algorithm 3), the best global schedule found H^* is finally returned in Line 20, Algorithm 3.

Algorithm 3 PERTURBATION

Require: a global schedule H
1: $H^* \leftarrow$ LOCALSEARCH(H)
2: **for** $iter = 1$ to $itMax$ **do**
3: $H \leftarrow H^*$
4: **for** $\alpha_1 = 1$ to γ_1 **do**
5: **for** $\alpha_2 = 1$ to γ_2 **do**
6: $H \leftarrow$ MOVETWS(H, α_1)
7: $f(H) \leftarrow$ OPTIMIZEROUTES(H)
8: **if** $f(H) < f(H^*)$ **then**
9: $H^* \leftarrow H$
10: **end if**
11: **end for**
12: $H \leftarrow$ LOCALSEARCH(H)
13: $f(H) \leftarrow$ OPTIMIZEROUTES(H)
14: **if** $f(H) < f(H^*)$ **then**
15: $H^* \leftarrow H$
16: $\alpha_1 \leftarrow 1$
17: **end if**
18: **end for**
19: **end for**
20: **Return** H^*

We also developed a simple multi-start variant, whose pseudo-code is presented in Algorithm 4. One possible drawback of PERTURBATION algorithm is that it always restarts from the best solution. Over the long run, it might get stuck on the same solution. To overcome this potential problem, the simple idea of algorithm MSPERTURBATION is to restart the search at each iteration of the outer loop from a newly created solution as can be seen in Line 3 of Algorithm 4. Different solutions are generated at each call of INITIALSOLUTION() thanks to the SHUFFLE() routine. Also, to allow more restarts, the value of α_1 is not reset to 1 every time a new improving solution

is found. Instead, the perturbation continues until α_1 reaches the value γ_3.

Algorithm 4 MsPerturbation

Require: a global schedule H
1: $H^* \leftarrow$ LocalSearch(H)
2: **for** $iter = 1$ to $itMax2$ **do**
3: $H \leftarrow$ InitialSolution()
4: $H \leftarrow$ LocalSearch(H)
5: $f(H) \leftarrow$ OptimizeRoutes(H)
6: **if** $f(H) < f(H^*)$ **then**
7: $H^* = H$
8: **end if**
9: **for** $\alpha_1 = 1$ to γ_3 **do**
10: **for** $\alpha_2 = 1$ to γ_4 **do**
11: $H \leftarrow MoveTWs(H, \alpha_1)$
12: **end for**
13: $H \leftarrow$ LocalSearch(H)
14: $f(H) \leftarrow$ OptimizeRoutes(H)
15: **if** $f(H) < f(H^*)$ **then**
16: $H^* \leftarrow H$
17: **end if**
18: **end for**
19: **end for**
20: **Return** H^*

The best values for $itMax$ and $itMax2$, γ_1, γ_2 and γ_3, γ_4 for both algorithms are investigated in Section 6.

5.3 Second-stage

The first-stage provides a global schedule H corresponding to the assignment of at most one time window to each day and zone. In the second-stage, such a schedule is used to determine the routes visiting the customers of the different zones (procedure OptimizeRoutes).

Since a zone might be visited more than once over the time horizon and possibly in different time windows, we need to establish when a customer of a given zone has to be served (the day and the corresponding time window). In common practice, each customer selects a time window from his/her list of preferences. We thus assume that all customers have an ordered list of days in which they prefer to be served. Moreover, each customer will receive the service in the time window of the current schedule H having the highest order in his/her personal list. This allows the company to guarantee, on average, a good level of customer satisfaction.

To compute the cost associated with a given global schedule H, in the second-stage we look for the routing of a fleet of vehicles with the objective of first maximizing the number of served customers, and then minimizing the distances. To this aim, we solve a VRPTW for each scenario of the sample and for each period $t \in T$. The set R indicates the set of routes serving customers of period t in scenario s and $C(R)$ its cost. The value f is updated with the cost of routing for each

Algorithm 5 OptimizeRoutes

Require: a global schedule H and a set of scenarios S

$f \leftarrow 0$
for $s = 1$ to $|S|$ **do**
  Assign customers to days/time windows
  for $t = 1$ to $|T|$ **do**
    $R \leftarrow$ ConstructRoutes$(H, N_s, t, K(t))$
    $R \leftarrow$ ALNS(R)
    $R \leftarrow$ Improvement(R)
    $f \leftarrow f + C(R)$
  end for
end for
Return $f/|S|$

scenario and period. At the end, the average routing cost $f/|S|$ is returned. To solve the VRPTW, we run some classical local search procedures, and an Adaptive Large Neighborhood Search (ALNS) framework. Our ALNS implementation follows the general scheme proposed by Pisinger and Ropke [12].

The second-stage algorithm (OptimizeRoutes) receives a global schedule as an input, and after assigning the customers to the time windows construct an initial solution for each period $t \in T$. Algorithm ConstructRoutes is a construction heuristic where customers are inserted sequentially in an available route by the *Regret-k* insertion heuristic of [13]. At each iteration, the heuristic calculates the minimal insertion cost into each route and it selects the customer that maximizes the sum of differences between the cost of inserting in its best route minus the best insertion costs in the other routes. This value is a kind of look ahead that indicates the lost that can be incurred if the customer is not inserted now. If some customers could not be feasibly inserted into a route, they are left in a customer bank to be inserted later. Then the ALNS procedure is called. At each iteration of ALNS, a removal and an insertion heuristic 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. Once some customers are removed, the insertion heuristic selects customers in the customer bank to insert them back in the solution hoping to improve it. We decided to select and implement a subset of the destroy/repair operators described in Pisinger and Ropke [12]. In particular, after extensive preliminary tests, we 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 selected destroy and repair operators, the roulette-wheel mechanism used to select the operators according to a probability that depends on their past performance, and all the parameters are implemented as in [12]. We introduced only two minor changes corresponding to the cooling rate, that is set to 0.8, and the number of iterations that has been decided after some tests as described in Section 6.

After the removal/insertion process, two local search operators are finally applied to the incumbent solution. We call this phase of the OptimizeRoutes algorithm, the Improvement

procedure. The 2-Opt* local search, described in Potvin and Rousseau [14], is first applied. It consists in selecting two arcs from two different routes. The removal of both arcs create four parts: two beginnings and two endings. Then, the beginning of one route is merged with the ending of the other one. This method is shown to be particularly powerful for problems with time windows. Finally, the Relocate local search proposed by Savelsbergh [16] is executed. It tries to insert a customer from one route into another. Both methods are run until no further improvement can be found.

5.4 Speed-up Techniques

Two techniques were implemented to improve the speed of our algorithms. The first one is the *hot-starting* of ALNS. In many cases, global schedules given as input to the second-stage problem do not change drastically. Typically, they differ for only one time window, and often just for its starting time. Instead of constructing a solution from scratch, a 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. Then, 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 customer bank are reinserted into the solution using the regret heuristic and then ALNS is started. In this way, better solutions could be found.

The second technique relies on a *hash table* used to store the cost of the second-stage corresponding to every first-stage solution that is visited. Each global schedule is hashed to a 64 bits integer in the following way. First, all time windows are sorted by zone and period. Each time window is transformed into a 32 bits integer, where the first 10 bits correspond to the period, the next 11 to the starting time and the last 11 to the ending time. Both times are represented in number of minutes since midnight. Then, the integer identifier of each time window is added to a list. The list of integers is hashed into a 64 bits integer key using the *one at a time* function of Jenkins [8]. The key and the cost of the solution are added in a traditional hash table. A speed-up of 5 could be achieved by using this technique.

6 Experimental Analysis

This section is devoted to the description of the testing environment including the instance generation, the choice of parameters for the implemented algorithms, and the tests performed. Each algorithm has been implemented in C++, and run on an Intel 2.667 GHz Westmere EP X5650 processor under Scientific Linux 6.3.

6.1 Generation of Instances

Since no benchmark instances exist for the problem, we created two sets of new instances. The first set contains homogeneous instances to test the different parameters and assess the performance of

Group	# Zones	# TWs	# Days	Average # Customers	Average Service Time	# Drivers per Day	Work Time per Zone
1	8	1	3	48	12	1	120.6
2	12	1	3	64	20	2	163.6
3	12	1	4	96	12	2	156.4
4	16	1	4	96	20	2	178.5
5	16	1	5	144	12	3	179.9
6	24	1	5	192	14	3	175.5
7	16	2	4	132	20	3	238.7
8	24	2	4	192	24	4	266.6
9	16	3	5	192	24	4	393.5
10	24	3	5	240	30	5	401.4

Table 1: Structure of Class A instances

our algorithms (Class A instances). The second set of instances is built upon real data of a delivery company (Class B instances).

6.2 Experiments on generated instances

Class A instances consist of delivery regions shaped in the form of a grid with rows and columns. Each square of the grid corresponds to a zone of extension 500×500.

We assume to know the probability distributions for locations, demands and service times of customers in a given zone: the total number of customers follows a Poisson distribution, the locations of customers inside each zone are uniformly distributed, the demands and service times follow a normal distribution. To ease our experiments, we made sure that the vehicle capacity constraint is not restrictive (the capacity of all vehicles is set to 800). With a single tight constraint, corresponding to the time windows, it is much easier to generate instances with the right number of customers. Each demand has a normal distribution with mean equal to 20, and standard deviation equal to 5. The operating hours of the depot are all set between 8AM and 6PM. All offered time windows have a width of 180 minutes. Also, to favor the number of served customers over distance, we set the parameter β, the cost associated with non-served customers, in our model to 1000000.

Table 1 describes our set of different configurations. Each configuration is characterized by a number of zones, a number of time windows per zone, a number of days in the time horizon, 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. Each configuration consists of 5 instances, and for each one the depot is located randomly inside the grid.

The required number of time windows per zone was estimated by using an approximation of the work time for each zone. This value includes the total travel time and the total service time. The total travel time was computed by using an approximation of the optimal routing cost proposed by

Figliozzi [6], as follows:

$$L^* = \frac{1}{speed} \times \left(1.45 \times \frac{n_\alpha - m}{n_\alpha} \times \sqrt{Bn_\alpha} + 2rm\right) \tag{20}$$

where L^* is the total travel time, n_α is the number of customers, B is the total area of the delivery region, m is the number of drivers, r is the average distance customer/depot, and *speed* is the traveling speed of the vehicles. The average distance to the depot r is equal to half the diagonal of a square of area B. The *speed* is set to $\frac{100}{60}\times$ 50km per hour to represent the required 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 was approximated in the following way:

$$L_z^* = \frac{L^*}{|Z|} + s_z \frac{n_\alpha}{|Z|} \tag{21}$$

where L_z^* is an approximation of the work time in zone z. It is composed of the total travel time L^* divided by the number of zones plus the expected service time s_z of zone z times the number of customers in the zone ($\frac{n_\alpha}{|Z|}$). Because these instances are homogeneous, all the L_z^* values are the same. Then, the number of time windows (all with width equal to 3 hours) in zone z is calculated as follows:

$$\#\text{ of time windows in zone } z = \left\lceil \frac{L_z^*}{3\text{h}} \right\rceil \tag{22}$$

The number of customers was adjusted to ensure the time windows are sufficient in the majority of the cases. If time windows are planned using the expected number of customers, then they might be insufficient because the number of customers follows a Poisson law and it is highly probable that 50% of the scenarios will contain more than the expectation. Instead of using the expected value, the value n_α was increased to have a sufficient plan for α percent of the cases. The value n_α is such that $P(X \leq n_\alpha) = \alpha$, where X is a random variable following a Poisson distribution with expected mean equal to n. A value of $\alpha = 0.95$ increases L^* and L_z^*, and it ensures that there is a relatively low number of unserved customers in the solution.

Each instance has a set of scenarios and each one is generated in the following way. First, the total number of customers n' is generated randomly following a Poisson distribution. Secondly, a zone is randomly selected for each of the n' customers. Each zone is selected with equal probability. Thirdly, the customer is then randomly located inside the zone and its demand and service time are randomly generated according to a normal distribution. Finally, The preferences of each customer on the delivery date are given by a random permutation of the array $T = [1, 2, ..., h]$. We also assume that each scenario is equally likely to occur. Finally, the expected scenario is also generated. It is characterized by having a number of customers equals to the expected number of customers. Customers are uniformly scattered inside their zone. Their service time and demand

	Perturbation γ_2					MsPerturbation γ_4			
γ_1	1	2	3	4	γ_3	1	2	3	4
1	1.64%	1.32%	1.03%	1.02%	1	1.92%	1.79%	1.99%	1.89%
2	1.29%	**0.86%**	1.22%	1.37%	2	1.63%	1.57%	1.68%	1.60%
4	1.18%	1.48%	1.31%	1.57%	4	1.16%	1.32%	1.61%	1.26%
6	1.22%	1.40%	1.56%	1.93%	6	1.16%	1.59%	1.66%	1.47%
8	1.11%	1.61%	1.71%	1.90%	8	**1.09%**	1.45%	1.64%	1.53%

Table 2: Parameters tuning: Values of γ_1, γ_2 and γ_3, γ_4.

are exactly equal to the expected values. This scenario is useful for comparing what would produce a deterministic approach, like in [7], versus a stochastic approach as done in this work.

6.3 Parameters tuning

We tested several different parameters in order to assess the performance of our algorithms for solving the SMTWAVRP. We solved all the instances changing one parameter at a time, and computing the average gap of that particular configuration with respect to the best result obtained for each instance. All tests have been made by using the expected scenario described in the previous section.

The first test concerns the possible values of γ_1 and γ_2 in metaheuristic Perturbation and of γ_3 and γ_4 in the corresponding MsPerturbation variant. In this preliminary test, we run both algorithms for 20 minutes on scenarios 1 and 2, 40 minutes on scenarios 3 and 4, and 1 hour for the other scenarios. Table 2 presents the results of different values of γ_1 and γ_2. Each column contains the percentage deviation to the best solution found for both algorithms separately. Results indicate that doing a too high or too low number of perturbations can generally lower the quality of the solutions. We have chosen the two best configurations corresponding to the values $\gamma_1 = \gamma_2 = 2$ for Perturbation method, and $\gamma_3 = 8$ and $\gamma_4 = 1$ for MsPerturbation, and used them in the remaining computational experiments.

As far as ALNS is concerned, we have initialized all the parameters, but one, to the same values reported in Ropke and Pisinger [15]. The number of iterations of ALNS has a major impact on the quality of the solution and on the running time. This analysis is reported in Table 3. First, we tested a setup where only ConstructRoutes is used to build an initial solution. Secondly, we also tested the pure application of the Relocate and 2-Opt* local searches (procedure Improvement). Finally, we run ALNS for 25, 50, 100, 200 and 400 iterations with and without the local searches. The test was done using the Perturbation heuristic for 100 iterations.

Table 3 presents the results. The first column contains the type of tested configuration, the second represents the cost percentage deviation from the best solution among all methods, whereas column time indicates the average runtime. Based on these results, we observe that using ALNS improves the solution quality but increases the runtime. Both local searches are useful at improving the initial solution, but ALNS remains essential to obtain good results. Better results can be obtained by combining ALNS with the local searches. We believe that running ALNS for 100

Configuration	Cost	Time
ConstructRoutes	35.68%	6.8
Relocate	18.52%	25.9
2-Opt*	23.25%	15.7
Improvement	14.71%	33.5
ALNS 25	3.90%	273.0
ALNS 50	3.17%	593.1
ALNS 100	1.87%	1235.9
ALNS 200	1.97%	2660.5
ALNS 400	1.78%	5332.9
ALNS 25 + Improvement	2.58%	322.6
ALNS 50 + Improvement	1.87%	653.6
ALNS 100 + Improvement	1.34%	1337.7
ALNS 200 + Improvement	1.31%	2701.7
ALNS 400 + Improvement	1.23%	5435.0

Table 3: Testing different algorithm configurations.

Algorithm	Cost	Time	# Best
ConstructRoutes	15.55%	6.9	2
Improvement	10.75%	24.0	2
Perturbation	0.09%	3242.4	46
MsPerturbation	1.44%	3243.3	16

Table 4: Comparison of the algorithms

iterations, and using both local search routines represents a very good trade off between quality and runtime. Using 200 iterations doubles the runtime compared to 100, but the quality improvement is too scant.

The third test reports the solution quality of our proposed algorithms. We have compared the behavior of the two metaheuristics when running for the same amount of time. Their results are also compared to the solutions found by the constructive heuristic (ConstructRoutes) and by the local search phase (Improvement algorithm). Table 4 presents the results: the first column shows the algorithm; the second column indicates the percentage deviation with the best found solution; third column contains the average running time; the last column indicates the number of times the algorithm found the best solution. The results indicate that the Perturbation heuristic performs much better than its MsPerturbation variant. Over our test bench, Perturbation heuristic is able to find the best solution in 46 out of 50 instances. Solutions found by the MsPerturbation algorithm are, on average, 1.35% worse. However, both algorithms are able to significantly improve the solutions produced by ConstructRoutes and Improvement.

The purpose of the fourth test is to compare the evolution of the objective function value when increasing the number of iterations $itMax$ in the Perturbation heuristic. We have run the algorithm for 0, 1, 10, 20, 50, 100, 200 and 400 iterations without setting any time limit. The test with no iterations means that the algorithm consists only of the constructive method and of

$itMax$	Cost	Time
0	9.65%	29.2
1	8.65%	40.8
10	5.12%	165.4
20	4.14%	301.1
50	2.38%	706.2
100	1.09%	1344.3
200	0.60%	2582.4
400	0.00%	5046.7

Table 5: PERTURBATION performance: runtime versus number of iterations.

the local search heuristic. Table 5 reports the results where the first column ($itMax$) provides the number of iterations, the second column shows the percentage deviation to the best solution value, and the last column indicates the average runtime (in seconds). It is worth noticing that doing very few iterations provides a small contribution to improve the solution value. However, when the number increases over 100 iterations, the results become much better. The runtime of our algorithms strongly depend on the number of used scenarios. This means that the overall runtime is equal to the time reported in the third column multiplied by the number of scenarios. To have a reasonable overall computational time, we have decided to run the PERTURBATION method for 100 iterations.

6.4 The Value of the Stochastic Solution

In this section, we analyze the potential benefit of using a stochastic programming approach instead of a deterministic one. As pointed out earlier, the high number of scenarios will likely increase the computational times, but it might also provide better first-stage solutions. On the opposite side, a low number of scenarios will reduce the computational times and probably result in poorer first-stage solutions.

As a first step towards this analysis, we evaluate the information provided by a larger number of scenarios. To this aim, we set proportionally the same amount of time to find the best first-stage solution when considering scenario sets of different sizes. The PERTURBATION heuristic has been run on 6, 15, 25, 50, 100, 200 scenarios and also on the expected scenario. The runtime for the case with 200 scenarios has been set to 36 hours. Proportionally, we reduced the time for 100 scenarios to $36/(200/100) = 18$ hours, for 50 scenarios to 9 hours and so on. This ensures that the PERTURBATION heuristic does about the same number of iterations.

Once the computation of the first-stage solutions have been completed, their expected costs are calculated on a set of 400 scenarios. Table 6 presents the results obtained for each set of scenarios. Each size is compared with the expected scenario (last line). Column 2 indicates the average number of unserved customers, whereas column 3 shows the average traveling cost deviation from the expected scenario. Column 4 reports the average runtime. Results indicate that, for approximately the same number of iterations, the more scenarios we have the better the first-stage

# Scenarios	Avg. # unserved Customers	Avg. Distances	Avg. Runtime
6	0.289	-2.82%	1.1 hours
15	0.099	-2.83%	2.7 hours
25	0.055	-2.64%	4.5 hours
50	0.030	-2.41%	9.0 hours
100	0.015	-2.00%	18 hours
200	0.009	-1.64%	36 hours
Expected	0.791	0.00%	0.18 hours

Table 6: Value of the Stochastic Solution

# Scenarios	Avg. # unserved Customers	Avg. Distances
6	0.416	-2.66%
15	0.097	-1.66%
25	0.058	-1.07%
50	0.027	-0.40%
100	0.016	0.65%
200	0.009	2.27%
Expected	0.848	0.00%

Table 7: Value of the Scenarios: runtime equal to 6 hours.

solution will be. As expected, the number of unserved customers decreases substantially with the number of scenarios. Even with very few scenarios, like 6 or 15, we can improve over the deterministic approach represented by the expected scenario. The average distance behaves differently because a decrease in the number of unserved customers increases the vehicle distances.

As a second test, we assess the trade-off between solution quality and time. With a fixed reasonable amount of time, the algorithm searches for the best first-stage solution on sets of 6, 15, 25, 50, 100, 200 scenarios and also the expected scenario. Then, the expected cost of the first-stage solutions are approximated on a set of 400 scenarios. The total runtime is set to 6 hours. The entries in Table 7 have the same meaning of the corresponding columns in Table 6. Findings indicate that the best results, in terms of unserved customers and distances, can be achieved by using larger sets of scenarios if the runtime is a fixed amount.

Some additional considerations can be made by comparing the findings in Tables 6 and 7. Results obtained by using very few scenarios should be taken with a grain of salt. For example, let us consider the results of using the expected scenario and 6 and 25 scenarios. The results in the two tables are not consistent. The algorithm was ran for a much longer time in the second test (Table 7) than in the first one (6), and the obtained solution are worse. For these scenarios, the first-stage solutions found was very good but their expected values on the 400 scenarios were rather random. The tables show that having more than 100 scenarios give more consistent results.

6.5 Quality of Service

The last test compares the quality of the service in terms of number of time windows and transportation costs. Previously, all our tests used time windows of 3 hours. However, a company might be interested in providing a better customer service with time windows of smaller width. We assume the quality of the service is inversely proportional to the width of the proposed time windows. Larger time windows imply higher waiting times and are usually less preferred by customers who do not like stay at home for long hours. Inversely, with short time windows, for example of 1 hour, customers might be more satisfied. However, this level of satisfaction comes at a cost for the transportation company. Each new offered time window is a possible additional trip to zone which might increase transportation costs (for an environmental impact of the choice of different time windows width see Manerba et al. [11]).

For this test, we divided the estimated work time of a zone by the time window width as given in equation (23).

$$\#\text{ Time Windows per Zone} = \left\lceil \frac{\text{Work Time per Zone}}{\text{TW Width}} \right\rceil \tag{23}$$

$$\text{Free Time} = (\#\text{ Time Windows per Zone} \times \text{TW Width}) - \text{Work Time per Zone} \tag{24}$$

A special care must be given when selecting time window width, since reducing their size might imply a reduction of the overall work time in a zone. Table 9 shows this phenomenon. Each cell indicates the expected free time in minutes that is available in each zone. Such a free time is calculated as in equation (24). Consider, for instance, configuration 3 with a work time of 156.4 minutes. The free time with 3h time windows is equal to 23.6 minutes, whereas it grows to 83.6 minutes for 2h time windows. This means we might be able to serve more customers with time windows of width equal to 2h than with those of width equal to 3h. A second problem might also happen when offering two time windows of width v versus a single time window of size $2v$. It is possible that we serve more customers with two time windows than with the single one. This is mainly due to the fact that the fleet of vehicles is fixed on each day. As a matter of fact, the two time windows must be on different days, so we have twice the number of drivers available. This might happens with configurations 9 and 10 for an all-day time window versus 2 time windows of 5h. For these reasons, the fine tuning of time windows should be done carefully to achieve the best results in terms of transportation costs and number of served customers.

Table 8 indicates the number of time windows in each scenario for each time window width. We tested scenarios with a time window spanning an entire day (10h) up to a single hour. It is worth noticing that in scenarios 9 and 10, there are 5 time windows instead of 7 because of the 5 days time horizon. This might increase the number of unserved customers.

This test was done over 200 scenarios for a maximum runtime of 12 hours. Results are presented in Tables 10 and 11. In Table 10, the first column provides the width of the time windows. 10 hours corresponds to the whole day. Columns (Customers) and (Distances) provide the average number of

Configuration	# Days	Work Time	# TWs per Width						
			10h	5h	4h	3h	2h	1.5h	1h
1	3	120.6	1	1	1	1	2	2	3
2	3	163.6	1	1	1	1	2	2	3
3	4	156.4	1	1	1	1	2	2	3
4	4	178.5	1	1	1	1	2	2	3
5	5	179.9	1	1	1	1	2	2	3
6	5	175.5	1	1	1	1	2	2	3
7	4	238.7	1	1	1	2	2	3	4
8	4	266.6	1	1	2	2	3	3	5
9	5	393.5	1	2	2	3	4	5	7
10	5	401.4	1	2	2	3	4	5	7

Table 8: Number of Time Windows

Configuration	Worktime	10h	5h	4h	3h	2h	1.5h	1h
1	120.6	479.4	179.4	119.4	59.4	119.4	59.4	59.4
2	163.6	436.4	136.4	76.4	16.4	76.4	16.4	16.4
3	156.4	443.6	143.6	83.6	23.6	83.6	23.6	23.6
4	178.5	421.5	121.5	61.5	1.5	61.5	1.5	1.5
5	179.9	420.1	120.1	60.1	0.1	60.1	0.1	0.1
6	175.5	424.5	124.5	64.5	4.5	64.5	4.5	4.5
7	238.7	361.3	61.3	1.3	121.3	1.3	31.3	1.3
8	266.6	333.4	33.4	213.4	93.4	93.4	3.4	33.4
9	393.5	206.5	206.5	86.5	146.5	86.5	56.5	26.5
10	401.4	198.6	198.6	78.6	138.6	78.6	48.6	18.6

Table 9: Free time

	Scenario Costs		Expected Costs	
TW Width	Customers	Distances	Customers	Distances
10 hours	0.000	0.0%	0.010	0.0%
5 hours	0.000	8.9%	0.004	9.6%
4 hours	0.000	13.5%	0.008	13.7%
3 hours	0.001	23.8%	0.011	23.7%
2 hours	0.000	53.5%	0.014	52.9%
1.5 hours	0.014	68.5%	0.044	67.6%
1 hours	0.179	118.6%	0.398	117.0%

Table 10: Cost of Time Windows

Configuration	10h	5h	4h	3h	2h	1.5h	1h
1	0.009	0.022	0.023	0.037	0.059	0.254	0.605
2	0.005	0.004	0.012	0.014	0.008	0.008	0.482
3	0.013	0.003	0.011	0.012	0.002	0.001	0.023
4	0.006	0.004	0.015	0.013	0.016	0.041	1.238
5	0.011	0.000	0.002	0.000	0.000	0.001	0.007
6	0.003	0.006	0.011	0.000	0.008	0.013	0.042
7	0.001	0.000	0.001	0.001	0.003	0.013	0.149
8	0.014	0.002	0.001	0.008	0.013	0.066	0.679
9	0.020	0.000	0.000	0.001	0.001	0.009	0.058
10	0.021	0.003	0.005	0.025	0.033	0.035	0.702

Table 11: Expected number of unserved customers

unserved customers and average percentage deviation in terms of traveled distance when considering 200 scenarios (Scenarios Costs) and when calculating the expected costs on 400 scenarios by using the first-stage solutions (Expected Costs). In Table 11, we have the overall expected number of unserved customers for each configuration and for each time window size.

As shown in table 10, distances and number of unserved customers increase as the time window width decreases. One can expect to double its distances when passing from an entire day time window to 1 hour time windows. The used vehicle capacity will be reduced as well as the number of unserved customers increases. Results also indicate that computing a first-stage solution by using 200 scenarios gives costs that are in line with 400 scenarios.

The problems identified earlier are evident in our results. The increase in free time due to a time window reduction can be seen in configurations 2 and 3 with time windows of 3h and 2h, and also in configuration 8 with time windows of 5h and 4h. The second problem occurs also in configurations 9 and 10 with an entire day time window. For these two cases, the algorithm could find schedules that are very tightly packed. There are even days without any assignment of time windows. This result can be explained by the fact that in all cases we use the same number of drivers. The fewer the number of vehicles used when optimizing, the more time windows will be evenly spread across the time horizon.

	Edmonton	Calgary
Postal codes	49	44
Average # customers	463	469
# vehicles	5	6

Table 12: Details on the real-world instances

6.6 Experiments on Real-World Instances

In this section, we consider real-world instances (Class B) obtained by a Canadian company owning stores selling furniture such as beds, sofas, tables, and white appliances. The company operates two warehouses located in the cities of Calgary and Edmonton, both in the province of Alberta. Both warehouses deliver products to customers located in their nearby neighborhood.

The company handed us its historical database of previous deliveries totalizing 80000 deliveries in a time horizon of several months. Their current delivery policy is the following. Customers can choose any delivery date after a minimal date that depends on the availability of the required stocks at the depot. In the normal case, as soon as the stock becomes available, the company can deliver it in the next four days (it might be few days more if the stock is not available). Then, three days prior to a delivery date, the transportation department receives the list of all deliveries. A variant of the vehicle routing problem is solved and then, according to the obtained routes, it is possible to calculate an estimated time of arrival at each customer. Finally, a 3 hour time window around the arrival time is given by an automated calling machine.

The company wants to compare its current delivery policy with a time window assignment based approach on a typical week. Table 12 presents the details of the two delivery regions. The warehouse of Edmonton offers deliveries in 49 postal codes with an average of 463 customers per week and uses a fleet of 5 vehicles. For Calgary, the company delivers in 44 postal codes with an average of 469 customers served per week and has a fleet of 6 vehicles per day. An analysis of the data shows that distribution of the number of customers per week follows a Poisson law. The same method was used to generate the scenarios. However, instead of randomly generating customers, they are randomly picked in the bank of 80000 deliveries. A set of 200 scenarios was created to be used in the optimization and a set of 400 was created to calculate the expected number of unserved customers and distances. Postal codes are highly heterogeneous: some zones require more than 12 hours of work per week, some others require less than 2 hours. All vehicles are available from 8 AM to 6 PM. All distances have been calculated using the haversine distance.

The study aims at comparing the company's approach of letting customers choose their delivery date against different sizes and number of time windows. Given the size of time windows, we obtain the number of time windows by applying formula (23). The comparison is made by using widths of 3, 4, 6 hours and entire day. By using these values, we noticed that several postal codes, requiring few working hours, obtain only one time window. To give more choices to the customers, we tested the case with 1, 2, 3 and 4 more time windows. The final number of time windows per postal code is the minimum between the number obtained from formula (23) and 5, which is the length of the

time horizon. A *No Time Windows* test was made to know the minimum possible cost. All tests were run for 36 hours.

Results are presented in Table 13. The first column is the time window configuration. For each city, the table shows the total number of time windows (#TWs), the expected number of unserved customers (Cust.), the expected distances (Distance) and its percentage deviation (Dist%) compared with the company's approach corresponding to line "5 Days" (the company's approach is to let the customers choose the day they want over 5 days).

A special care must be taken when observing the results for two reasons. First, free time has a strong impact on the quality of the service because of the high heterogeneity of the postal codes. It can be noticed that some zones receive more free time when the size of time windows is of 3 hours than when it is of 4 hours, but for some others the reverse is true. Secondly, optimizing a case with several short time windows is significantly harder (requires higher computational time) than with few large ones. Thus the comparison is not exactly fair and the results might be not perfectly consistent.

Computational experiments provide some evident managerial insights. Table 13 indicates that, on average, the company could reduce its traveled distances by using a few very large time windows. The drivers would have to do a lower number of trips in the postal codes, and in turn this would induce reduction in costs. Also, when time windows are large, there is more space to globally optimize the routes. Controlling the delivery dates could also be an effective tool. Results show that instead of offering any delivery date, the company could offer 4 days out of 5, and achieve in this way a reduction of 1.8% and 1.7% in distances. Offering fewer days provides bigger reductions. Gains are always obtained when using formula (23) for 3, 4 and 6 hours time windows. In these cases, even an additional time window is beneficial.

Also, it is worth noticing that the offer of an additional time window does not behave linearly. In all cases, moving from +0 to +1 increases the number of available time windows by 1 in almost all postal codes. This increases the possible number of trips in each postal code, and as a consequence this justifies the important increase in distances. In some other cases, moving from $+k$ to $+k+1$ has a marginal impact on distances because there is already a high number of time windows.

If the company wants to offer the choice of a delivery time and day at no additional cost, they could offer 4h+2 time windows. In this setting, the customers of Edmonton would get 3.3=162/49 time windows per week on average, and those of Calgary would get 3.6=157/44 time windows. The busiest postal codes get a time window for every day, whereas some other gets only one. The choice of the setting remains a managerial decision because there is a trade-off between transportation costs and customer satisfaction. A high number of short time windows is highly preferred by customers than a lower number of large ones. However, transportation costs will be higher when there are several short time windows. Also, more tests could be done on the offer of heterogeneous time windows. For example, a region with few customers could have few larger time windows, whereas busy postal codes could get several small ones.

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

Table 13: Results on the real-world instances

7 Conclusions

In this paper, we analyze a variant of the Time Window Assignment Vehicle Routing Problem, where customers are distributed into a grid of zones, 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 over a predefined time horizon that assigns to each zone a set of time windows during a time horizon of a week , by minimizing the total expected transportation costs plus a penalty cost for excluding some customers. We called this problem the Stochastic Multi-period Time Window Assignment Vehicle Routing Problem.

Benchmark instances have been generated, with a different number of zones, customers and scenarios to evaluate the effectiveness and the efficiency of the proposed solution approach. Results are extremely promising: the method represents a valuable tool for the industrial case motivating the study.

Acknowledgements

This work was partly supported by the Canadian Natural Sciences and Engineering Research Council (NSERC) under grants 2015-04893. We thank Calcul Québec for providing high performance parallel computing facilities.

References

[1] N. Agatz, A. Campbell, M. Fleischmann, and M. Savelsbergh, Time Slot Management in Attended Home Delivery, *Transportation Science*, 45(3):435-449, 2011.

[2] R. Bent and P. Van Hentenryck, Scenario-based planning for partially dynamic vehicle routing with stochastic customers, *Operations Research*, 52(6):977-987, 2004.

[3] J.R. Birge and F. Louveaux, Introduction to Stochastic Programming, Second Edition, *Springer Series in Operations Research and Financial Engineering*, Springer, 2011.

[4] A.M. Campbell and M. Savelsbergh, Incentive schemes for attended home delivery services, *Transportation science*, 40(3):327-341, 2006.

[5] J.F. Ehmke and A.M. Campbell, Customer acceptance mechanisms for home deliveries in metropolitan areas, *European Journal of Operational Research*, 233(1):193-207, 2014.

[6] M. A. Figliozzi, Planning Approximations to the Average Length of Vehicle Routing Problems with Varying Customer Demands and Routing Constraints, *Transportation Research Record: Journal of the Transportation Research Board*, 46(4):496-506, 2008.

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

[8] B. Jenkins, *The Hash*, `http://burtleburtle.net/bob/hash/doobs.html`.

[9] I. Lin and H. Mahmassani, Can online grocers deliver?: Some logistics considerations, *Transportation Research Record: Journal of the Transportation Research Board*, 1817:17-24, 2002.

[10] R. Klein, M. Neugebauer, D. Ratkovitch, and C. Steinhardt, Differentiated Time Slot Pricing Under Routing Considerations in Attended Home Delivery, *Transportation Science*, 53(1):236-255, 2017.

[11] D. Manerba, R. Mansini, and R. Zanotti, Attended Home Delivery: reducing last-mile environmental impact by changing customer habits, *IFAC-PapersOnLine*, 51(5):55-60, 2018.

[12] D. Pisinger and S. Ropke, A general heuristic for vehicle routing problems, *Computers & Operations Research*, 34(8):2403-2435, 2007.

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

[14] J-Y. Potvin and J-M. Rousseau, An exchange heuristic for routing problems with time windows, *Journal of the Operational Research Society*, 46(12):1433-1446, 1995.

[15] S. Ropke and D. Pisinger, An Adaptive Large Neighborhood Search Heuristic for the Pickup and Delivery Problem with Time Windows, *Transportation Science*, 40(4):455–472, 2006.

[16] M.W.P. Savelsbergh, The vehicle routing problem with time windows: minimizing route duration, *INFORMS Journal on Computing*, 4:146–54, 1992.

[17] R. Spliet and A.F. Gabor, The Time Window Assignment Vehicle Routing Problem, *Transportation Science*, 49(4):721-731, 2014.

[18] R. Spliet and G. Desaulniers, The Discrete Time Window Assignment Vehicle Routing Problem, *European Journal of Operational Research*, 244(2):379-391, 2015.

[19] B. Werweij, S. Ahmed, A.J. Kleywegt, and G. Nemhauser, A. Shapiro, The sample average approximation method applied to stochastic routing problems: a computational study, *Computational Optimization and Applications*, 24(2-3):289-333, 2003.