A new consistent vehicle routing problem for the transportation of people with disabilities

Dominique Feillet, Thierry Garaix, Fabien Lehuédé, Olivier Péton, Dominique Quadri

A new Consistent Vehicle Routing Problem for the transportation of people with disabilities

D. Feillet[1], T. Garaix[1], F. Lehuédé[2], O. Péton[2], D. Quadri[3]

[1] Ecole des Mines de Saint-Etienne, CMP Georges Charpak, F-13541 Gardanne, France

{feillet,garaix@emse.fr}

[2] IRCCyN, Ecole des Mines de Nantes, 4 rue Alfred Kastler, F-44307 Nantes, France

{fabien.lehuede, olivier.peton@emn.fr}

[3] Université d'Avignon et des Pays du Vaucluse, Laboratoire Informatique d'Avignon,

F-84911 Avignon, France

{dominique.quadri@univ-avignon.fr}

May 23, 2013

Abstract

In this paper we address a problem of the transportation of people with disabilities where customers are served on an almost daily basis and expect some consistency in the service. We introduce an original model for the time-consistency of the service, based on so-called time-classes. We then define a new multi-day Vehicle Routing Problem (VRP) that we call the Time-Consistent VRP. We address the solution of this new problem with a Large Neighborhood Search heuristic. Each iteration of the heuristic requires solving a complex vehicle routing problem with multiple time windows and no waiting time which we tackle with a heuristic branch-and-price method. Computational tests are conducted on benchmark sets and modified real-life instances. Results demonstrate the efficiency of the method and highlight the impact of time-consistency on travel costs.

Keywords: consistent vehicle routing problem, transportation of people with disabilities, multiple time windows, heuristic branch-and-price.

1 Introduction

Among the diverse categories of problems that have been investigated in operations research, Vehicle Routing Problems probably constitute one of the most emblematic class.

In practice, their purpose is generally to propose optimized distribution plans while delivering goods to customers (be they individuals, firms or retailers, for example). In most of the many models developed for the routing of vehicles, a single period of time is considered (say, a day). This modeling however corresponds to two very different practical situations: either the routes are intended to be performed once and re-computed with new data the next day, or the routes are repeated identically every single day over a long time horizon. However, there is an intermediate situation in which demands are almost identical from day to day. In addition to the usual cost minimization objective, a new criterion might then make sense: limiting the variability of the routes between the different periods of the time horizon. The problem then becomes a multi-day VRP where routes are expected to show some regularity from day to day.

An important illustration of this problem was given by Groër *et al.* [14] and concerns fast parcel delivery (involving big companies such as UPS). The ability to assure consistent schedules and to limit the number of drivers that a customer has to deal with can significantly improve the satisfaction of these customers: facilitating receipt operations, improving trust between the delivery companies and their customers, etc. A major advantage can thus be provided in comparison with competing companies.

Another interesting case, which motivated this paper, concerns the transportation of people with mental disabilities from and to day care centers [19]. Because of their lack of autonomy, most of the persons concerned have no vehicle and are not capable of traveling by public transport. Consequently, medico-social centers usually resort to specialized transportation companies to organize and perform the daily trips. Customers need to be transported regularly but sometimes with some small variations during the week. For example, a person may not ask for transportation or may not have the same address every day. In addition, these users are particularly sensitive to changes. The issue of keeping consistent time schedules is therefore fundamental.

In this paper, we investigate how consistency can be considered in vehicle routing problems. Because the expectations of stability are context-dependent, as will be detailed below, we essentially address the transportation of people with disabilities and the stability of the time of services. Since there is a paucity of works on this subject, we first discuss the notion and the modeling of consistency, before introducing a new model (Section 2). We then introduce the Time-Consistent Vehicle Routing Problem and propose a solution approach based on a Large Neighborhood Search (Section 3). Computational experiments evaluating the behavior of this algorithm are then described (Section 4), before the paper is concluded (Section 5).

2 Modeling the time-consistency of one customer

The issue of constructing consistent vehicle routes is related to many important lines of research in operations research such as robustness or stochasticity [3, 13]. However,

consistency is generally sought in relation to unexpected events that disrupt the transport.

In contrast, we consider here the consistency of services with respect to deterministic data with slight variations from day to day, which has rarely been investigated in a routing context. Our study mainly compares with the recent paper of Groër *et al.* [14] which introduces the consistent vehicle routing problem. Since its first online publication, this problem has been investigated in [18, 22, 27]. The consistent vehicle problem clearly shares some similarities with the Courier Delivery Problem (CDP) [26] which, in addition, considers time windows, uncertain service times, uncapacitated vehicles and that frequent customers should not necessarily be served by a single vehicle. A few other studies show some connections with this subject [1, 2, 28]. Finally, the consistent VRP has some relationship with multi-period VRPs [12], where the assignment of visit days is also considered.

From the customer point of view, consistency improves the quality of service. Having consistent schedules helps the customer to get prepared for the service, to manage material flows, to schedule upstream and downstream tasks, etc. Assigning the same driver to the different visits of a customer facilitates the delivery process and strengthens the relationship between suppliers and customers [6, 16, 24]. Such criteria can play an essential role in the choice of a supplier [5].

For drivers, having consistent sequences of visits also presents many advantages. It helps develop a good knowledge of the usual itineraries and traffic conditions, thus enabling drivers to estimate distribution times and potential delays better, reducing transportation times [6, 15, 24, 25]. Driver familiarity with the customer also helps in applying security and administrative procedures [16]. With all these improvements, the transportation process is generally considered more comfortable by the drivers [25].

In the remainder of the paper, we essentially restrict the study to the consistency of schedules. Our assumption is that once a time-consistent solution is computed, finding a reasonably consistent assignment of drivers should be possible. This assumption is checked experimentally in Section 4. The issue of simultaneously optimizing time- and driver-consistency is left for future research.

In order to discuss the modeling of time-consistency, let us consider a customer i, a time horizon $D = \{1, \ldots, |D|\}$ and times of service (or schedule) $W = (w_1, \ldots, w_{|D|})$ for customer i over this horizon: the service of customer i starts at time w_d on day d. For the sake of simplicity, note that we assume here that customer i is visited every day of the time horizon. Note also that W is assumed to be known. The method to optimize W will be the subject of Section 3.

We first review the model proposed by Groër *et al.* [14] and then describe our model which generalizes the former.

Note that we do not consider explicitly customers with several addresses. Actually, time-consistency is important for a given address. A person with two distinct addresses is modeled as two people (one per address).

2.1 Modeling time-consistency through time slacks

The time-consistency for customer i is ensured in [14] by limiting the difference between the latest and earliest service dates. More formally, a time of service variation is measured as:

$$TC_{slack}(W) = \max_{1 \leq d \leq |D|} w_d - \min_{1 \leq d \leq |D|} w_d \tag{1}$$

This definition is relevant in the context of the small package shipping industry. Figure 1 illustrates for five different schedules $W_1, \ldots, W_5$, the measure of time-consistency following Groër *et al.* for a horizon ranging from Monday to Friday.

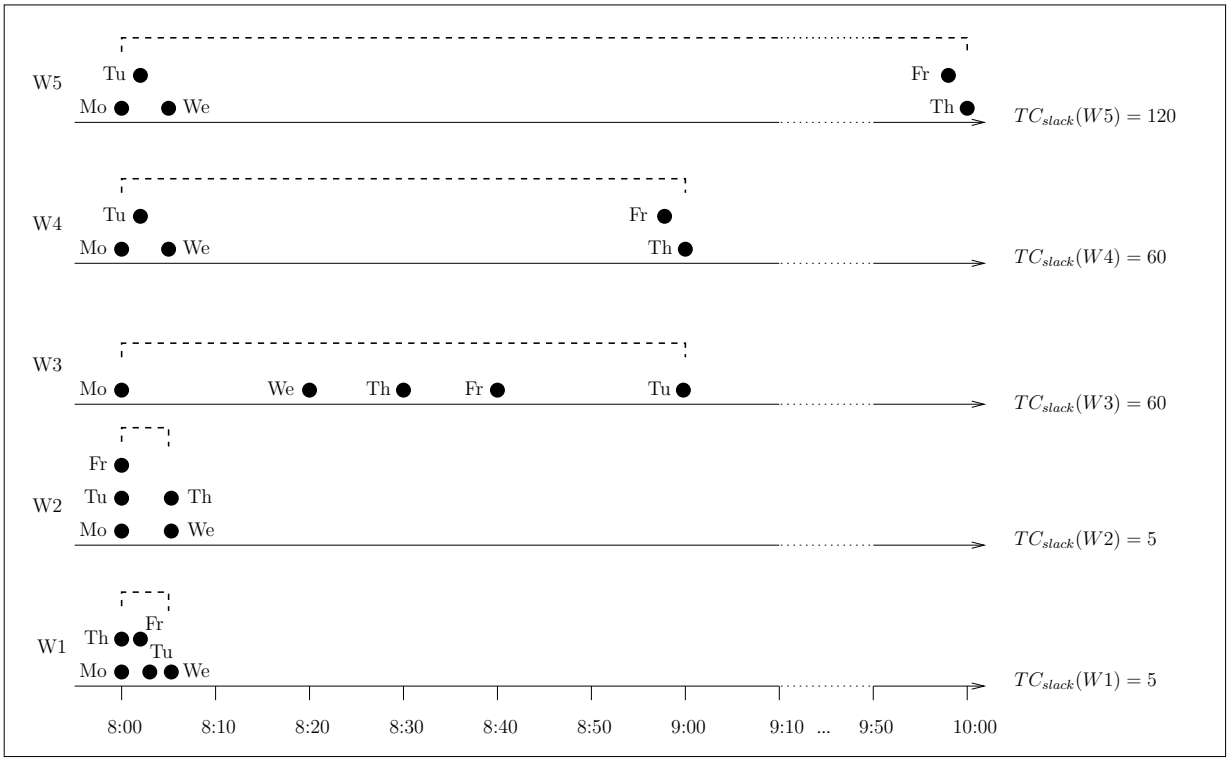

Figure 1: Time-consistency measure of Groër *et al.*

However, considering other application contexts, the preceding modeling needs to be questioned. A first remark is that it is particularly suitable for small variations in service times. Thus, schedules W_1 = (8:00, 8:03, 8:05, 8:00, 8:02) and W_2 = (8:00, 8:00, 8:05, 8:05, 8:00), such that $TC_{slack}(W_1) = TC_{slack}(W_2) = 5$, can be considered equivalent in terms of time-consistency. However, it is less clear for W_3 = (8:00, 9:00, 8:20, 8:30, 8:40) and W_4 = (8:00, 8:02, 8:05, 8:58, 9:00). Although $TC_{slack}(W_3) = TC_{slack}(W_4) = 60$, W_4 shows much more regularity, with two sets of almost identical times of service around 8:00 and 9:00.

Following this observation, a second important remark is that while W_1 or W_2 undoubtedly define more consistent schedules than W_4, should the two schedules W_4 and W_5 = (8:00, 8:02, 8:05, 9:58, 10:00), with $TC_{slack}(W_4) = 60$ and $TC_{slack}(W_5) = 120$ respectively, be considered so different? In the following section and in the remainder of this paper, we introduce a new modeling framework for time-consistency which assumes that such schedules are equivalent.

As could be expected, Groër *et al.* [14] attain a very high level of consistency in their solutions, thus justifying their modeling. Two factors can be considered determinant in this behavior. First, vehicles deliver 100 to 150 customers everyday. Secondly, many of these customers are visited only once during the time horizon, thus giving many opportunities for substitutions in the routes. As a consequence, consistent schedules can be preserved for the relevant customers, giving rise to a strong time-consistency and mostly avoiding being concerned by the above remarks. From an algorithmic perspective, the chosen approach was to define template routes that respect the so-called precedence principle: "If customers i and j are both served by the same vehicle on a specific day and i is serviced before j, then customer i must receive service before customer j from the same vehicle on all days where they both require service". Note that this principle also deals with driver-consistency.

For the transportation of people with disabilities, the situation is very different. Vehicles only visit a very small number of people, who make transportation requests almost everyday, thus rendering substitution between customers much more complicated and increasing the differences between the service times. This explains why we introduced the modeling developed below.

2.2 Modeling consistency through time-classes

From the above discussion, we introduce a new consistency measure that follows these principles: a small gap between service times is not significant; a large gap induces a disturbance, whatever the gap.

Let us consider the parameter $\delta > 0$ reflecting the sensitivity of the customers to schedule variations. In order to evaluate the time-consistency for customer i, we define the notion of time-classes as follows: the times of service w_d and $w_{d'}$ are allowed to be assigned to the same time-class if $|w_d - w_{d'}| \leq \delta$. Hence, the times of service $\{w_1, \ldots, w_{|D|}\}$ can be divided into time-classes.

It can easily be seen from the above definition that there is no unique way of assigning a set of times of service to time-classes. Let us go back to the example of schedule W_3 = (8:00, 9:00, 8:20, 8:30, 8:40). With parameter $\delta = 10$ one can, for example, construct time-classes {{Mo},{We,Th},{Fr},{Tu}} or {{Mo},{We},{Th,Fr},{Tu}}.

We define the time-consistency $TC(W)$ of a schedule $W = (w_1, \ldots, w_{|D|})$ as the minimal

number of time-classes into which $w_1, \ldots, w_{|D|}$ can be ranked:

$$TC(W) = \min C \tag{2}$$

subject to

$$|w_d - w_{d'}| > \delta \Rightarrow C_d \neq C_{d'} \qquad (d, d' \in D), \tag{3}$$
$$C_d \leq C \qquad (d \in D), \tag{4}$$
$$C_d \in \{1, \ldots, |D|\} \qquad (d \in D), \tag{5}$$
$$C \in \{1, \ldots, |D|\}. \tag{6}$$

Constraints (3) impose the assignment of different classes for service times whose difference is greater than δ. Constraints (4) with objective function (2) compute the minimal number of classes required C.

Figure 2 outlines the proposed consistency through time-classes considering schedules W_1 to W_5 introduced in sub-section 2.1, with parameter $\delta = 10$. Note that with this new measure, schedules W_1 and W_2 are still the most consistent, schedules W_4 and W_5 are considered equivalent and schedule W_3 is strongly penalized. The method to compute values $TC(W)$ and to assign visits to time-classes is presented in the following subsection.

2.3 Computation of time-classes

It can easily be seen that model (2)-(6) defines a graph coloring problem. Thus, undirected graph $G = (D, E)$ could be introduced, where edge $(d, d') \in E$ if and only if $|w_d - w_{d'}| > \delta$. Computing the minimal number of time-classes for schedule W then amounts to coloring the vertices in G, such that two vertices connected with an edge are colored with a different color.

While graph coloring problems are generally NP-hard, graph G exhibits a very special structure which implies that the problem is polynomial. This is highlighted by the subsequent equivalent definition of set E: edge $(d, d') \in E$ if and only if $[w_d - \delta/2, w_d + \delta/2] \cap [w_{d'} - \delta/2, w_{d'} + \delta/2] = \emptyset$. As such, the coloring problem can be solved using a greedy algorithm (Algorithm 1). In Algorithm 1, vertices are colored in the increasing order of times of service w_d, with the first available color. Let d represent the first vertex colored with the current color. A sufficient condition for a vertex to be colorable using the current color is to be compatible with vertex d.

The time-classes that would be constructed for schedules W_1 to W_5 with $\delta = 10$, applying Algorithm 1, can be observed in Figure 2.

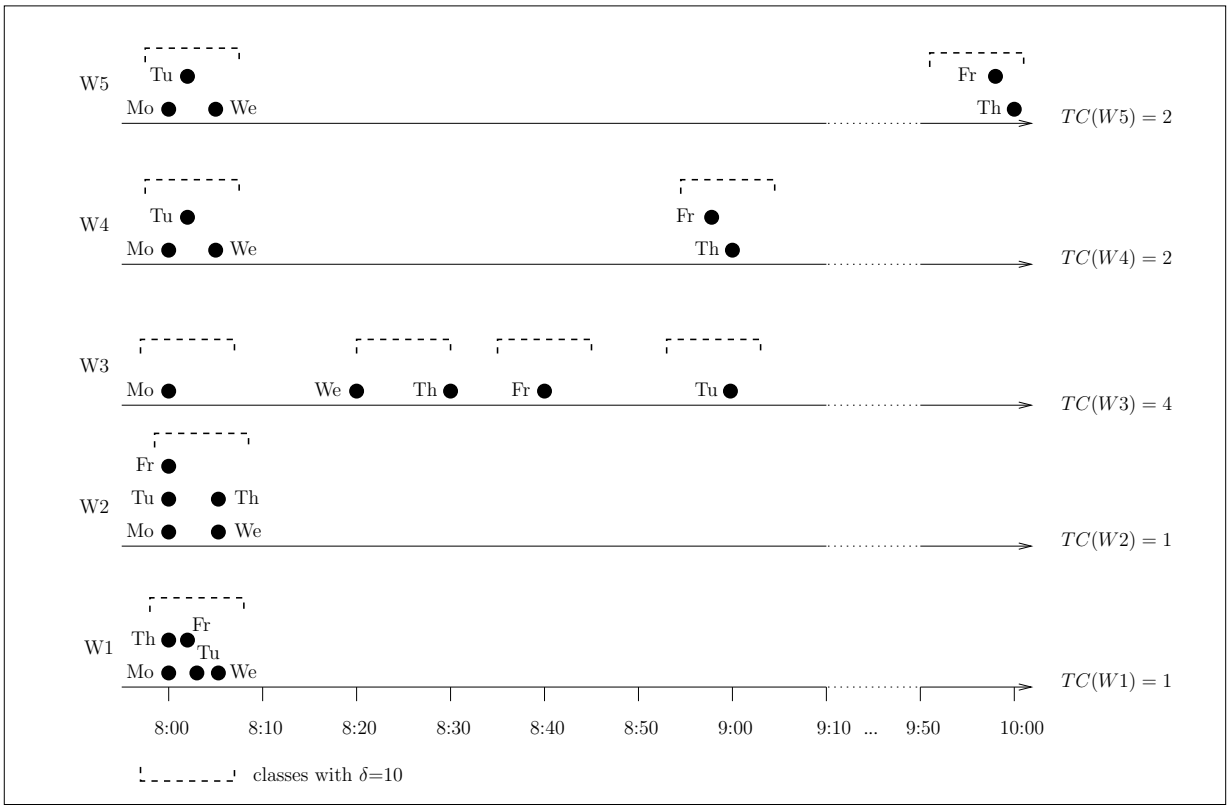

Figure 2: Time-class consistency

Algorithm 1 Computation of time-classes

$\{d_1, \ldots, d_{|D|}\} \leftarrow$ vertices $\{1, \ldots, |D|\}$ sorted in increasing order of times of service
color vertex d_1 with color 1
$d \leftarrow d_1$
for $i = 2$ to $|D|$ **do**
  if edge $(d, d_i) \notin E$ **then**
    color vertex d_i with the same color as d
  else
    color vertex d_i with the next color
    $d \leftarrow d_i$
  end if
end for

3 The Time-Consistent VRP

In the remainder of the paper, we first assess how the proposed definition of time-consistency can be handled in a solution approach. Secondly, we evaluate the impact of considering time-consistency on vehicle routes.

The original application includes various specific constraints and criteria that can complicate the readability of the subsequent results. We thus introduce and address a simplified problem. This is called the Time-Consistent Vehicle Routing Problem (TCVRP) and is basically a multi-day VRP with the additional constraint of optimizing time-consistency over days.

An important difference to be noted compared to previous sections is that customers are initially located at the depot and request to be transported back home with regular arrival times of services, with a set of vehicles all departing at the same time (time 0) from the depot. In the real application that motivates this paper, the people with mental disabilities all had to arrive at a given time and expected to be picked up at regular times. From a computational point of view however, the two situations are completely symmetric and equivalent.

We first define the TCVRP in Section 3.1. Then, the proposed solution algorithm is outlined in Section 3.2. Finally, sections 3.3, 3.4 and 3.5 provide details on the different steps of the algorithm.

3.1 Definition of the TCVRP

Let us denote D the set of days in the time horizon and consider a complete graph $G = (V, A)$, where $V = \{0, \ldots, n\}$ represents a depot (vertex 0) and the destinations of n customers (vertices 1 to n). Customers need to be transported from the depot to their destination on some days of the time horizon, defined by $p_i^d = 1$ if customer i has to be transported on day d, $p_i^d = 0$ otherwise. Set K represents a fleet of identical vehicles available at the depot every day of the time horizon. The capacity of a vehicle is limited to Q customers. A travel time $t_{ij} > 0$ is defined for every arc $(i, j) \in A$. This travel time corresponds equivalently, without loss of generality, to a travel cost. A parameter δ is given to represent the maximal difference of times of service that can be ranked in the same time-class.

Two objectives are pursued. The first is to minimize travel costs, *i.e.* the sum of the route costs thoughout the horizon. The second is to maximize time-consistency, *i.e.* to minimize the maximal number of time-classes over the n customers. The number of time-classes for a customer is defined in equations (2)-(6), where times of service are related to routing decisions.

Note that no time windows are defined for deliveries. In addition, waiting times are not allowed. Without this constraint, waiting could improve time-consistency.

The TCVRP then consists in optimizing these two objectives by finding a set of at most $|K|$ routes every day of the time horizon D such that (i) all the requested deliveries are performed, (ii) all routes satisfy vehicle capacities. Seeing that D is assumed to remain quite small (typically a week), the problem can rather be stated as the problem of solving $|D|$ mono-objective subproblems, where the maximal number of time-classes is limited and successively defined as $C = 1, \ldots, |D|$. The model (7)-(25) presented below shows a mathematical formulation for one of these mono-objective subproblems, with a limited number of time-classes, namely at most C time-classes per customer. Note however that the solution method and experiments presented later deal with the complete (multiobjective) TCVRP. The model is as follows:

$$\text{Min } z = \sum_{d \in D} \sum_{k \in K} \sum_{(i,j) \in A} t_{ij} x_{ijk}^{d} \tag{7}$$

s.t.

$$\sum_{k\in K}\sum_{j\in V} x_{ijk}^d = p_i^d \qquad (i \in V\setminus\{0\}, d \in D), \tag{8}$$

$$\sum_{j\in V} x_{jik}^d - \sum_{j\in V} x_{ijk}^d = 0 \qquad (i \in V\setminus\{0\}, k \in K, d \in D), \tag{9}$$

$$\sum_{i\in V\setminus\{0\}} x_{0ik}^d = 1 \qquad (k \in K, d \in D), \tag{10}$$

$$\sum_{i\in V\setminus\{0\}} x_{i0k}^d = 1 \qquad (k \in K, d \in D), \tag{11}$$

$$\sum_{i\in V\setminus\{0\}} \left(\sum_{j\in V} x_{ijk}^d\right) \le Q \qquad (k \in K, d \in D) \tag{12}$$

$$w_{jk}^d \le w_{ik}^d + t_{ij} + (1 - x_{ijk}^d)M \qquad ((i,j) \in A, k \in K, d \in D), \tag{13}$$

$$w_{jk}^d \ge w_{ik}^d + t_{ij} - (1 - x_{ijk}^d)M \qquad ((i,j) \in A, k \in K, d \in D), \tag{14}$$

$$w_i^d \le w_{ik}^d + (1 - \sum_{j\in V} x_{ijk}^d)M \qquad (i \in V\setminus\{0\}, k \in K, d \in D), \tag{15}$$

$$w_i^d \ge w_{ik}^d - (1 - \sum_{j\in V} x_{ijk}^d)M \qquad (i \in V\setminus\{0\}, k \in K, d \in D), \tag{16}$$

$$w_{0k}^d = 0 \qquad (k \in K, d \in D), \tag{17}$$

$$w_i^{d_2} - w_i^{d_1} \le \delta + M y_i^{d_1 d_2} \qquad (i \in V\setminus\{0\}, d_1, d_2 \in D), \tag{18}$$

$$c_i^{d_1} + 1 \le c_i^{d_2} + M(1 - y_i^{d_1 d_2}) \qquad (i \in V\setminus\{0\}, d_1, d_2 \in D), \tag{19}$$

$$y_i^{d_1 d_2} \in \{0,1\} \qquad (i \in V\setminus\{0\}, d_1, d_2 \in D), \tag{20}$$

$$c_i^d \le C \qquad (i \in V, d \in D), \tag{21}$$

$$x_{ijk}^d \in \{0,1\} \qquad ((i,j) \in A, k \in K, d \in D), \tag{22}$$

$$w_{ik}^d \ge 0 \qquad (i \in V\setminus\{0\}, k \in K, d \in D), \tag{23}$$

$$w_i^d \ge 0 \qquad (i \in V\setminus\{0\}, d \in D), \tag{24}$$

$$c_i^d \in \{1, \ldots, C\} \qquad (i \in V, d \in D). \tag{25}$$

The decision variables have the following interpretation:

x_{ijk}^d: 1 if arc (i,j) is used by vehicle k on day d, 0 otherwise,
w_i^d: time of service for customer i on day d,
w_{ik}^d: time of service for customer i on day d if it is carried out by vehicle k,
c_i^d: number of the time-class assigned to the delivery of customer i on day d.

The intermediate binary variables $y_i^{d_1 d_2}$ are set to 1 if the times of service of customer i on days d_1 and d_2 are increasing in this order and they are not allowed to belong to the same time-class ($w_i^{d_2} - w_i^{d_1} > \delta$), and 0 otherwise.

Constraints (8) state that customer i is serviced on day d if and only if he or she needs

to be transported on that day. Constraints (9) are classical flow conservation constraints. Constraints (10) and (11) enable vehicles to leave the depot and return to the depot once a day at most, respectively. Constraints (12) present vehicle capacities. Constraints (13) and (14) define time variables. They linearize the logical expression

$$\forall (i,j) \in A, \forall k \in K, \forall d \in D, x_{ijk}^d = 1 \Rightarrow w_{jk}^d = w_{ik}^d + t_{ij}.$$

Note that vehicles are not allowed to wait in order to improve time-consistency. Constraints (15) and (16) define times of service for customers, independently of the vehicle used. They linearize the logical expression

$$\forall i \in V \setminus \{0\}, \forall k \in K, \forall d \in D, \sum_{j \in V} x_{ijk}^d = 1 \Rightarrow w_i^d = w_{ik}^d.$$

Constraints (17) set vehicle starting times at the depot. For every customer, constraints (18)–(20) enforce the assignment of distinct time-classes to service times that differ from more than δ. They linearize the logical expression

$$\forall i \in V \setminus \{0\}, \forall d_1, d_2 \in D, w_i^{d_2} - w_i^{d_1} > \delta \Rightarrow c_i^{d_1} + 1 \leq c_i^{d_2}.$$

Constraints (21) limit the maximum number of time-classes allowed for every customer.

Finally, we note that in the case of $C \geq |D|$, constraints (18)–(20) (21) and (25) can be removed since every day can be solved independently. Furthermore, the case $C = 1$ can also be simplified by replacing those constraints by

$$|w_i^{d_2} - w_i^{d_1}| \leq \delta \quad (i \in V \setminus \{0\}, d_1, d_2 \in D). \tag{26}$$

3.2 Solution of the Time-Consistent VRP with a Large Neighborhood Search approach

In this section, we describe the framework of the algorithm proposed for the solution of the TCVRP. Subsequent sections explain the different procedures introduced here.

Time-consistency constraints significantly complicate the problem as they introduce temporal dependences between routes. Very few vehicle routing problems in the literature exhibit similar features. The most closely related problems we were able to find are those with synchronization constraints [4, 17]. As pointed out by Rousseau *et al.* [21], synchronization constraints considerably limit the applicability of traditional local search techniques. Assessing the feasibility or impact of a very small move can be a large and complex problem. From this observation, we decided to address the solution of the TCVRP with a Large Neighborhood Search (LNS) approach.

The LNS heuristic, first introduced by Shaw [23] has been successfully applied to vehicle routing problems in recent years [8, 20]. Although our implementation differs from the

initial algorithm of Shaw, the proposed algorithm follows the fundamental principle of LNS: the heuristic starts from a feasible solution and alternates between the destruction of a large part of the solution and the reconstruction of the resulting partial solution.

The main principle of the proposed heuristic is to start from a solution that is good from the cost point of view and to reduce the number of time-classes progressively, recording the best found solution for each level of time-consistency (thus solving the bi-objective version of the time-consistent VRP). In order to diversify the search, a perturbation mechanism and a restart procedure are performed periodically.

The initial solution is computed by considering each day independently and solving (heuristically) the corresponding VRPs. The solution obtained tends to minimize the total distance traveled during the time horizon. As time-consistency is completely ignored, it is likely that some customers will have a large number of time-classes.

At every iteration, the destruction phase selects one day for which all routes are destroyed. The reconstruction phase relies on the definition of time constraints that enable the maximal number of time-classes for the customers to be controlled. Our approach is to select a customer whom we attempt to reduce the number of time-classes and to limit the number of time-classes for other customers, to varying degrees depending on their distance from the current maximal number of time-classes. These constraints can be expressed in the form of multiple time windows associated with customers. The problem to solve is then a variant of the VRP that we call the Vehicle Routing Problem with multiple Time Windows and no-wait constraints (VRPmTW-nw). This problem is tackled with a heuristic branch-and-price solution method.

Algorithm 2 outlines the main steps of the LNS heuristic.

Algorithm 2 General scheme of the LNS algorithm

1: $sol \leftarrow$ initial feasible solution
2: $D' \leftarrow D$
3: **repeat**
4: every p_1 *useless* iterations without improvement: diversify or restart
5: select a day d at random in D'
6: select a customer i with a maximum number of time-classes among those served on day d
7: define an instance of the VRPmTW-nw for day d according to sol and i
8: $s \leftarrow$ solution of the VRPmTW-nw or **null** if infeasible
9: **if** $s \neq$ **null then**
10: $sol \leftarrow$ replace routes of day d in sol by the routes of s
11: **end if**
12: **if** $D' \setminus \{d\} \neq \emptyset$ **then** $D' \leftarrow D' \setminus \{d\}$ **else** $D' \leftarrow D$
13: **until** a computing time limit is met

The LNS algorithm is initialized (Line 1) by solving the VRP defined every day. For this purpose, we apply the heuristic branch-and-price algorithm that is used for reconstruction (see Section 3.4), initialized with routes obtained by applying the Clarke and Wright

algorithm. In this case, no time windows are defined for the customers. The main loop (Lines 3 to 13) alternates between the selection of a day to destroy and the reconstruction phase.

Day selection is done at random (Line 5), ensuring however that all days have been selected before one is destroyed again (Line 12). In Line 6, a customer is selected to focus time-class reduction on this customer. The one chosen has a maximal number of time-classes among customers served on that day. From the customers who satisfy this criterion, we select one for whom the time-class involved on that day has the lowest cardinality. The purpose is to give more chances of deleting that time-class in future iterations.

Following the selection of a day and a customer, an instance of the VRPmTW-nw is defined (Line 7). The way this instance is constructed is presented in Section 3.3. The heuristic branch-and-price method used for the solution of the VRPmTW-nw (Line 8) is detailed in Section 3.4. When possible, the current solution *sol* is then updated (Line 10) and the LNS iterations are continued until the computing time limit is reached.

When p_1 *useless* iterations of the LNS operators have not led to any improvement of one of the best known solutions (one per level of time-consistency), diversification or restart components are called for (Line 4). These two components and the selection process between them are presented in Section 3.5. Iterations are counted as *useless* when they do not succeed in reducing the number of customers with a maximal number of time-classes.

3.3 Time-window definition for the VRPmTW-nw

The definition of the instance of the VRPmTW-nw to be solved at every iteration of the algorithm is based on the current solution *sol*, the day d that was deconstructed and the selected customer i. As explained in Section 3.2, the objective of the deconstruction and reconstruction phases is to reduce the number of time-classes for i and to limit the number of time-classes for other customers according to a set of rules that are described in this section.

In order to describe our scheme, we introduce the following notation. The number of time-classes for $j \in V \setminus \{0\}$ in *sol* is denoted $TC_j^{sol}(D)$. The number of time-classes for j after the deconstruction of day d is denoted $TC_j^{sol}(D \setminus \{d\})$. $TC^{sol}(D)$ is the maximal number of time-classes of the customers during horizon D, that is the time-consistency value for *sol*. Finally, the construction of the VRPmTW-nw instance relies on a parameter denoted $classMax_j$. This represents the maximal number of time-classes allowed for $j \in V \setminus \{0\}$ in the new solution to be constructed.

According to this notation, $classMax_j = TC_j^{sol}(D) + 1$ means that any time of service is possible for customer j on day d (no time constraint). On the contrary, $classMax_j = TC_j^{sol}(D)$ generally means that the time of service for customer j on day d has to be *compatible* with the times of service on other days for this customer. The sole exception is met when $TC_j^{sol}(D \setminus \{d\}) < TC_j^{sol}(D)$, that is when the time of service of customer j on

day d in *sol* was defining a time-class on its own.

In the above paragraph, *compatible* means that the time of service for customer j on day d can be integrated into one of the existing time-classes for this customer. Given a time-class whose lowest time of service is t_{inf} and highest time of service is t_{sup}, the condition for the time of service w_j^d to be compatible with this time-class is $t_{sup} - \delta \leq w_j^d \leq t_{inf} + \delta$. This condition can be expressed in the form of a time window condition: $w_j^d \in [t_{sup} - \delta, t_{inf} + \delta]$. Being compatible with the different time-classes then comes down to satisfying multiple time windows.

The number and the size of time windows (can be none) assigned to a customer in the travel cost minimization problem on day d depends on how the $classMax$ parameter has been defined. Let us now describe the proposed definition of $classMax$. We recall that i is the customer selected at the current iteration of the LNS algorithm, for whom a decrease in the number of time-classes is expected.

1. Four cases are considered for customers $j \in V \setminus \{0\}, j \neq i$:
   (a) if $TC_j^{sol}(D) = TC^{sol}(D)$:
      i. if this condition held in the past p_2 iterations, $classMax_j = TC_j^{sol}(D) + 1$,
      ii. else, if $TC_j^{sol}(D) = TC^{sol}(D)$ for no more than p_2 iterations, $classMax_j = TC_j^{sol}(D)$,
   (b) if $TC_j^{sol}(D) < TC^{sol}(D)$:
      i. $classMax_j = TC_j^{sol}(D)+1$ with probability $\min(p_3(TC^{sol}(D) - TC_j^{sol}(D)), 1)$,
      ii. otherwise $classMax_j = TC_j^{sol}(D)$.
2. For customer i, if $TC^{sol}(D) > 1$, $classMax_i = TC^{sol}(D) - 1$, otherwise $classMax_i = 2$.

Rules 1(a)i and 1(b)i state that the number of time-classes for customer j can be increased, whereas it remains the same in rules 1(a)ii and 1(b)ii. According to $classMax_j$, time windows are then defined for j. Two situations are possible:

- $classMax_j = TC_j^{sol}(D \setminus \{d\}) \Rightarrow$ [multiple time windows],
- $classMax_j > TC_j^{sol}(D \setminus \{d\}) \Rightarrow$ [no time window].

The rationale behind using probability $\min(p_3(TC^{sol}(D) - TC_j^{sol}(D)), 1)$ in rules 1(b)i and 1(b)ii is that customers with a large number of time-classes are given less opportunity to increase this number. Parameter p_3 is set to 0 at the beginning of the algorithm and increased by 0.1 at every restart (see Section 3.5) with a maximum value of 0.5.

D	1	**d=2**	3	4	5	TC_j^{sol}	TC_j^{sol}	Rule	$classMax$	Time windows
$V \setminus \{0\}$						(D)	$(D \setminus \{d\})$			for day 2
1	54.6	**54.6**	91.8	91.1	91.1	2	2	1(b)ii	2	[44.6;64.6] [81.8;101.1]
2	12.7				29.4	2	2			
3	23	**69**			19.1	2	1	1(b)ii	2	–
4				55.2	55.2	1	1			
5	53.9			7.7		2	2			
6	71.8	**15.6**	23.8	21.8	21.8	2	2	1(b)i	3	–
i=7	37.3	**94.3**	118.3	32.2	117.6	3	2		2	[27.3;42.2] [108.3;127.6]
8	40.8	**40.8**	78	77.3	77.3	2	2	1(b)i	3	–
9	2.4	**2.4**	2.4	44.4	2.4	2	2	1(b)ii	2	[0;12.4] [34.4;54.4]
10	84.6		10.6	8.6	8.6	2	2			
11		**84.8**				1	0	1(b)ii	1	–
12		**29.8**	38		51.8	2	2	1(b)ii	2	[28;48] [41.8;61.8]
13	26.6	**26.6**	63.8	63.1	63.1	2	2	1(b)ii	2	[16.6;36.6] [53.8;73.1]
14		**108.6**		17.9		2	1	1(b)ii	2	–
15			8.1			1	1			

Table 1: Definition of multiple time windows for a solution with 15 customers, 5 days and $\delta = 5$

The only possibility that the number of time-classes in the new solution exceeds $TC^{sol}(D)$ comes from rules 1(a)i and 2 (second case), where the purpose is to inject some diversification.

Note finally that the definition of time windows may not actually reduce the number of time-classes of i in the first case of rule 2. In fact, when $TC_i^{sol}(D \setminus \{d\}) = TC_i^{sol}(D) = TC^{sol}(D)$, no solution with $TC^{sol}(D) - 1$ time-classes exists (remember that i is selected from the customers with a maximal number of time-classes on day d). In this case, $classMax_i$ will not exactly represent the maximal number of time-classes for customer i. Definition $classMax_i = TC^{sol}(D) - 1$ will rather be handled by defining one time window per time-class of customer i except for the time-class in which the time of service of customer i on day d was ranked. The purpose is to decrease the size of this time-class and eventually to empty it in further iterations.

Example The preceding principles are illustrated in the following example: we consider a solution for 15 customers on 5 days with $\delta = 5$, which results in a time-consistency $TC^{sol}(D) = 3$. The selected customer i is customer 7 and day 2 is deconstructed. Columns 2 to 6 of Table 1 present the time schedule of the solution for the 15 customers. Columns $TC_j^{sol}(D)$ and $TC_j^{sol}(D \setminus \{d\})$ present the number of time-classes of each customer in the current and the deconstructed solutions, respectively. The following columns then present the rule applied to each customer (with an increased number of time-classes allowed for customers 6 and 8) and the time windows defined according to these values ("–" stands for no time window).

3.4 Solution of the VRPmTW-nw

The problem to solve at each iteration of the reconstruction phase of the LNS consists in finding a set of routes with minimal cost that serve all customers involved on that day. One additional condition, compared to the standard VRP is that deliveries are restricted by multiple time windows. Also, no waiting time is allowed. We call this problem VRPmTW-nw and describe here the solution approach developed for it. While the VRPmTW-nw is new (to the best of our knowledge), a heuristic branch-and-price solution scheme could relatively easily be adapted from existing VRPTW solution schemes.

The principle of the branch-and-price algorithm for the VRPTW [7, 9] is: a set-covering formulation is defined where path variables represent the set of feasible routes. Due to the very large number of possible routes, the formulation is first restricted to a limited subset of variables (restricted master problem) and then extended using a pricing subproblem. At each iteration of the so-called column generation scheme, the linear relaxation of the restricted master problem is solved and the pricing problem is called for, given a vector of optimal dual variables. The aim of the pricing problem is then to determine one or several feasible routes of negative reduced cost. By subtracting the values of dual variables from arc costs, the pricing problem can be expressed as an elementary shortest path problem with resource constraints and solved with dynamic programming. This whole scheme is embedded in a branch-and-price algorithm so as to obtain optimal integer solutions. Branching rules consist in imposing or forbidding arcs in the solution.

Regarding the VRPmTW-nw, most of the previous approaches can be applied in a similar way. The only difference lies in the pricing problem and concerns the feasibility of routes. While extending paths in the dynamic programming algorithm, one has to address multiple time windows and forbidden waiting times. Feasibility is assessed by simply checking that the destination node is reached within one of its time windows. As waiting times are not allowed, the condition that two paths arrive at the destination at exactly the same time is added to the dominance rule. In fact, earlier arrival, which is usually preferred since it gives more opportunity for further extensions, could here prevent some customers from being reached. The consequence is a significantly weakened dominance.

In the process of maintaining elementary paths and reinforcing the dominance rule, label definition includes the computation of the set of reachable nodes [10]. Here, reachable nodes are those that were not visited in the partial path represented by the label and whose last time window ends after the earliest possible arrival.

Actually, in view of the way multiple time windows are defined (see Section 3.3), it is possible that a large proportion of customers are not subject to time windows. So as to counterbalance the negative effect of the no-waiting-time condition on the dominance rule, we implement the following mechanism. While computing the set of reachable nodes for a label, we determine that we do not fall into the situation where all the remaining nodes have no time windows. For these labels, the traditional dominance rule is applied.

Through these adaptations, the VRPmTW-nw can theoretically be solved exactly. As the branch-and-price algorithm has to be called many times using relatively similar data, a single pool of columns is kept and enriched throughout the successive calls of the algorithm. A filtering algorithm is performed at each new call in order to consider only feasible columns (regarding the current data) in the model. However, the need to solve a VRPmTW-nw instance at every iteration of Algorithm 2 also implies very limited computing times. Hence, a heuristic variant of the previous branch-and-price scheme is developed.

When solving the pricing problem with dynamic programming, the set of labels conserved with every node of the graph is limited to the $p_4 = 20$ best non-dominated labels, in order of increasing time. Exploration in the dynamic programming algorithm is also limited with a Limited Discrepancy Search policy [11]. Basically, extensions are limited to the $p_5 = 2$ nearest neighbors (sorted according to the reduced cost). However, extensions to other neighbors are permitted with the limit of at most $p_6 = 2$ extensions for a complete route.

It is well-known that column generation is subject to slow convergence. In order to avoid long series of column generation iterations without a clear improvement in the objective value, we stop after a given number of iterations with improvements less than $p_7 = 1\%$. This number of iterations is set to $p_8 = 5$ when a feasible solution has already been found, $p_9 = 10$ otherwise.

Finally, a simple diving strategy is implemented for the branching phase. When the current optimal solution of the relaxed restricted master problem is fractional, instead of splitting the search into two descendant nodes, we only explore the descendant node where an arc is imposed.

These three heuristic mechanisms avoid, respectively, long pricing problem solution times, a large number of iterations of column generation at each node of the search tree, and a large number of nodes in the search tree, thus ensuring that many instances of the VRPmTW-nw can be solved in a reasonable time. A drawback of these mechanisms is that they involve a relatively large set of parameters (p_4 to p_9). The value of the different parameters defined above was set through preliminary experiments, which demonstrated, however, that the method was not very sensitive to these values.

3.5 Restart and diversification

The diversification operator works as follows. The first step is to select a customer with a maximal number of time-classes for a maximal number of iterations. This customer is then removed from the solution and reinserted every requested day as the first customer of a route; more precisely, the route that yields the minimal insertion cost among the existing routes on each day. Since vehicles always start from the depot at time 0, the time of service for this customer will always be the same, leading to only one time-class for this customer. This simple operator thus enforces consistency for a customer who was stuck with a large

number of time-classes for a long time and starts the search in a completely new direction.

The restart operator replaces diversification when a solution with a time-consistency equal to 1 has been found since the last diversification or restart. It consists in restoring one of the best known solutions as the current solution. Our choice is to restore solutions with any potential number of time-classes. Thus the first call to the restoration operator will generate a solution with $|D|$ time-classes, then $|D|-1$, etc. When value 1 is reached, the process starts again from $|D|$. As the LNS algorithm tends to search for solutions with a small time-consistency, this restart operator aims to spend some time improving best known solutions with larger time-consistencies.

4 Computational experiments

The LNS heuristic was coded in C++ and run on a machine with a 2.5 GHz processor and 3 Go of RAM and evaluated on two types of data instances. First, we considered the so-called small instances of [14], which comprise 10 instances with 10 or 12 customers over a 3-day horizon. The average level of presence of the customers is around 70%. The instances are named *conVRPx-y*, with $x = 10$ or 12 for the number of customers and $y = 1, \ldots, 5$, for the instance index. In these instances, a route duration limit is included, which can easily be handled in the model and the algorithm.

Then, we considered 14 real cases arising from day care centers for mentally disabled children or adults in the area of Nantes (France). The centers have to transport between 15 and 65 people everyday (inbound trips in the morning and outbound trips in the evening). The planning horizon is 5 days. For each original instance, the demand of the transported persons is represented by vectors of 5 binary values: 1 if the person has to be transported on that day and 0 otherwise. We modified the original demand vector so that the level of presence varied from $\rho = 0.5$ to $\rho = 0.9$ in steps of 0.1. In order to obtain a level of presence ρ with realistic data, we made the following assumptions: a proportion ρ^2 of the people make a demand everyday; a proportion $(1-\rho^2)$ of the people have a level of presence of $\frac{\rho}{1+\rho}$; for the latter category, the level of presence on Tuesdays, Wednesdays and Thursdays is twice that of Mondays and Fridays. The resulting 70 instances are named dataz-x, where $z = 10 \times \rho$ and x is the number of customers.

The original instances [19] include additional constraints such as individual time windows or maximum trip duration, several possible addresses during the week, incompatibilities between people, wheelchair constraints, a heterogeneous fleet, etc. We ignored these complicating constraints in order to focus on the consistency aspect.

As explained in the introductory section of the paper, a post-processing phase is carried out to assign drivers to routes in such a way that driver-consistency is optimized. The objective is to minimize the average number of drivers per customer. This is introduced and discussed in [24], where the authors explain that it is representative of customer satisfaction in terms of driver-consistency. They also propose several integer programming

formulations. In our computations, we used their (DC) model which minimizes the average number of drivers transporting each customer.

4.1 Small instances

The aim of this subsection is threefold: (i) to show that the problem is difficult to solve in practice (*i.e.*, the limits of the integer linear model); (ii) to validate the proposed heuristic in comparison with the best solutions found; (iii) to compare the proposed model with Groër *et al.* [14].

Comparison with the integer linear program

Table 2 first evaluates the ability of the integer linear programming formulation introduced in Section 3.1 to solve the problem. As it seems almost incapable of solving any of the instances, we temporarily relax vehicle capacity constraints, maximum route length constraints and assume unitary demands. These instances, derived from the original ones, are named "RconVRPx-y". The sensitivity parameter δ is set to 5. Three runs of 10 hours using ILOG-CPLEX 12.0 are performed on each instance, setting the maximal number of time-classes C – in inequality (21) – to 3, 2 and 1, respectively. The optimal solutions found are presented in the "ILP" section of the table. Optimal solutions are compared to the solutions provided by our heuristic in 5 minutes of computing time (each run provides a solution for every number of time-classes): the "Avg LNS" part presents the average results over ten runs of the heuristic.

	ILP			Avg LNS		
Time-classes:	3	2	1	3	2	1
RconVRP10-1	92.91	92.91	92.91	92.91	92.91	92.91
RconVRP10-2	80.42	80.42	-	80.42	80.42	80.99
RconVRP10-3	94.12	94.12	94.37	94.12	94.12	94.37
RconVRP10-4	93.71	93.71	-	93.71	93.71	94.09
RconVRP10-5	83.84	83.84	-	83.84	84.54	96.08
RconVRP12-1	103.65	103.65	-	103.65	103.65	104.40
RconVRP12-2	73.89	73.89	-	73.89	73.89	81.27
RconVRP12-3	82.77	82.77	-	82.77	82.77	83.12
RconVRP12-4	97.57	97.57	-	97.57	97.57	101.91
RconVRP12-5	83.63	83.63	-	83.63	83.63	89.25

Table 2: *Optimal (ILP) and heuristic (LNS) solutions on modified small instances ($\delta = 5$)*

The ILP formulation was, in most cases, unable to solve the 1-time-class problems to optimality within 10 hours. Moreover, ILOG-CPLEX was not able to improve the

	Avg LNS	Best LNS	ILP	Best
RconVRP10-1	**92.91**	**92.91**	**92.91**	92.91
RconVRP10-2	**80.99**	**80.99**	82.83	80.99
RconVRP10-3	**94.37**	**94.37**	**94.37**	94.37
RconVRP10-4	**94.09**	**94.09**	94.53	94.09
RconVRP10-5	96.08	**96.01**	**96.01**	96.01
RconVRP12-1	**104.40**	**104.40**	106.87	104.40
RconVRP12-2	81.27	**81.25**	**81.25**	81.25
RconVRP12-3	**83.12**	**83.12**	**83.12**	83.12
RconVRP12-4	**101.91**	**101.91**	**101.91**	101.91
RconVRP12-5	**89.25**	**89.25**	**89.25**	89.25

Table 3: *1-time-class solutions on modified small instances ($\delta = 5$)*

lower bounds given by the 2-time-classes problems. All the known optimal solutions were reached at each run of the LNS algorithm, except for instance RconVRP10-5; for this one, our algorithm was not able to find the optimal solution with 2 time-classes; instead solutions of costs 84.5 and 84.69 were found, 8 and 2 times respectively, for an average value of 84.54.

In order to evaluate the results obtained with 1 time-class, Table 3 provides the average and best solutions found by LNS (after the ten runs), the best feasible solution built by the ILP and the best known solution among these solutions ('Best'). Values that match the best known solutions are written in bold. Our heuristic failed to find the best known solution for only two cases. For the RconVRP10-5 instance, it found a sub-optimal solution of cost 96.7 once out of ten runs. For the RconVRP12-2 instance, our heuristic returned a 1-time-class solution with cost 81.34 instead of 81.25 twice out of ten runs.

For three instances out of ten, our heuristic found better results than the ILP formulation. Moreover, since the gap between the best-LNS solutions and the lower bounds – taken from the 2-time-classes optimal solutions of Table 2 – is 3.8% on average and 14.5% at maximum, we can be confident in the quality of the solutions computed by our heuristic on the modified small instances.

Comparison with the approach of Groër *et al.*

Table 4 presents a comparison between our approach and that proposed by Groër *et al.* [14]. For this set of experiments the original instances *conVRPx-y* introduced in [14] are considered.

The purpose here is not to compare the solution methods but rather to discuss the modeling differences. Indeed, Groër *et al.* enforce in their model that a single driver is used for every customer, while we do not consider this issue implicitly but rather optimize

the number of drivers *a posteriori*. Also, Groër *et al.* do not introduce the concept of time-classes but they consider the maximal difference between the service times. Their heuristic minimizes (implicitly) that difference while it is bounded in their ILP formulation. The chosen limit is 5. This situation corresponds to the set of 1-time-class solutions provided with our heuristic, with parameter δ set to 5.

The optimal costs obtained in [14] are reported in the column "Opt." of Table 4 [1]. The costs of the 1-time-class solutions obtained for these instances with our heuristic are reported in the next column, followed by the observed gap with "Opt.". The difference in cost is not significant on average (3.4%), showing improvements of up to 7.5% when relaxing driver consistency. The impact on driver-consistency can be evaluated by the last three columns of the table, where we report the number of customers with one, two and three drivers, obtained after optimally assigning drivers to the routes in the heuristic solutions. Here, perfect driver-consistency is only reached for instance 'conVRP10-4'. However, the proportion of customers with two drivers never exceeds 30% and no customer has three drivers on these instances. The average losses in driver-consistency (counted in percentage of customers served by more than one driver) equal to 1.8% and 2.6% on instances with 10 and 12 customers, respectively, enable average cost savings of 2.1% and 4.6%. Hence, given the cost reduction obtained, allowing some flexibility in driver-consistency can be worthwhile when possible.

Instance	Opt.(*) Cost	LNS Cost	Gap	Number of customers with 1 driver	2 drivers	3 drivers
conVRP10-1	122.03	118.70	2.8%	7	3	0
conVRP10-2	99.07	96.57	2.6%	8	2	0
conVRP10-3	123.41	123.16	0.2%	9	1	0
conVRP10-4	126.89	126.89	0.0%	10	0	0
conVRP10-5	109.31	104.04	5.1%	7	3	0
conVRP12-1	144.02	143.23	0.6%	10	2	0
conVRP12-2	89.54	83.27	7.5%	9	3	0
conVRP12-3	119.69	113.74	5.2%	9	3	0
conVRP12-4	141.37	137.26	3.0%	10	2	0
conVRP12-5	117.42	110.17	6.6%	9	3	0

Table 4: *Heuristic solutions and driver-consistency on small instances ($\delta = 5$)*

4.2 Real instances

In this subsection, we investigate the efficiency of our solution approach on the 70 real instances. Remember that in these instances no route duration limit is considered. We

[1] Optimal values given in [14] include service durations. They are discarded in the costs of Table 4.

first present a set of results (benchmark results) obtained with a computing time of one hour per instance. The objective here is to give some reference results for further research. Reference results are also provided to evaluate the efficiency of our heuristic with more limited computing times and assess the impact of time-consistency on routing costs.

Benchmark results

Table 5 and 6 details the results obtained by a single run of our heuristic, limited to one hour, on real instances with a time-consistency parameter δ set to 10.

Note that the values given in the first column of this table are the best solutions found with exactly the number of time-classes indicated in the label of the column. In practice, it is clear that a result with fewer time-classes and smaller routing costs would be preferred. For example, on instance data5-15, solutions with 5 and 4 time-classes are actually dominated by the solution with 3 time-classes. Dominated solutions are generally generated by the diversification procedure. When improving cost also decreases the number of time-classes, there may be a large cost difference between the five classes and the four classes solutions. Some dominated solutions may also result from the heuristic solving of the problem or a particular combination of time windows. We decided to present the results as such considering that "non-dominated" solutions can easily be deduced. Statistics on these results and also on "non-dominated" solutions are proposed in Table 7. The first part of the table (*Rough results*) summarizes the results given in Table 5 and provides the average value for the instances for which a solution was found. In the second part of the table (*Non-dominated solutions*), values of "dominated" solutions are replaced by values of solutions that "dominate" them (also, the best solution with fewer time-classes is considered when no solution was found). In the last line of the table, gaps indicate the increase in travel costs observed compared to solutions where 5 time-classes are allowed ($C = 5$).

Several comments can be drawn from Tables 5 to 7. The first rather surprising result is that feasible solutions with a small number of time-classes could be found for all 70 instances. Only some solutions with exactly 5 time-classes were not obtained on the whole set of instances. This latter point is not really surprising since having 5 different classes can sometimes counter-optimize the solution. For example, it is possible that the solution obtained by solving the VRP each day independently (which is optimal in terms of routing) does not introduce any customer with 5 time-classes. In this case, solutions with 5 time-classes are naturally dominated and not explored during the search.

In addition, we remark that imposing time-consistency does not penalize routing costs so much. As long as 2 time-classes are accepted, the impact on routing costs is less than 1%. When a limit of 1 time-class per customer is strictly enforced, routing costs are only increased by 5.90%, which is not negligible but might be offset by the increase in consistency for the customers.

C_{max}	5	4	3	2	1
data5-15	700.8	664.7	663.2	670.5	705.7
data5-21	827.1	775.8	775.9	776.9	823.7
data5-25	724.5	617.8	617.8	618	655.9
data5-26	845.2	777.3	771.1	778.5	829.2
data5-27	1010	946.5	940.8	947.2	1066.8
data5-32		1002.1	987.8	988.4	1023.2
data5-41	1465.5	1439.5	1467.8	1467.1	1615.1
data5-44	1176.5	1158.6	1150.4	1162	1219.2
data5-46	1482	1474.3	1474.2	1497.3	1562.1
data5-48		1460.1	1457.7	1462	1590.6
data5-55		1622.7	1605.2	1606.8	1855.6
data5-59	2804.4	2787.8	2778.6	2789.9	3059.8
data5-64	2141.1	2140.1	2172.3	2163	2321.2
data5-65	1795.5	1797.4	1798.1	1802.3	2052.7
data6-15	736.4	689.7	688.3	691.4	731.3
data6-21	849.3	794.7	793.6	795.1	812.9
data6-25	788.9	695.4	683.1	682.8	736.1
data6-26	853.3	843.8	839.4	841.1	888.1
data6-27	1040.6	951.1	952.9	954.9	1030.9
data6-32	1004.2	1002	992.4	992.4	1013.5
data6-41	1883	1521.2	1540.2	1565.8	1841
data6-44	1351.5	1251.3	1246.6	1251.1	1340.9
data6-46	1618.7	1506.1	1503.1	1515.7	1608.1
data6-48	1529.7	1533.2	1536	1529.6	1663.7
data6-55	1895	1890.1	1858.8	1861.4	2011.1
data6-59	3026.4	3023	3028.4	3017.7	3337.3
data6-64	2338	2327.8	2325.4	2333.7	2558.1
data6-65	2038.4	2018.4	2003.3	2012.8	2104.7
data7-15	766	749.6	746.9	751.3	773.6
data7-21	846.5	832.3	831.6	833.6	880.7
data7-25	776.3	726.4	724.5	727.4	754.9
data7-26	900.5	885.4	882.6	890.2	918.9
data7-27	1122.8	1059.4	1057	1059.9	1108.6
data7-32	1249.5	1103	1094.6	1101.8	1146.8
data7-41	1746.5	1724.5	1669.4	1664.1	1800.1
data7-44		1306.2	1301.4	1301.8	1369.2
data7-46	1697.8	1679	1659.5	1664.1	1717.1
data7-48	1729	1723.3	1713.7	1716.1	1854
data7-55	2003.8	1945.8	1931.8	1933.8	2038.3
data7-59	3388.7	3364.2	3340.2	3348.1	3667.7
data7-64		2608.6	2630.7	2612	2817.7
data7-65		2235.4	2261.6	2280.4	2380.2

Table 5: *Solutions of the heuristic method on real instances ($\delta = 10$)*

C_{max}	5	4	3	2	1
data8-15	817.5	776.5	775.3	777.2	792.8
data8-21	910.4	899.2	899.2	902.9	916.9
data8-25		860.7	853.9	853.3	857
data8-26	1110.6	974.6	965	968.5	1000.8
data8-27	1341.1	1198.8	1186.2	1186.2	1247.3
data8-32	1172.2	1159.8	1152.4	1154.6	1175.4
data8-41		1917.5	1906.9	1910.5	1989.7
data8-44		1433.5	1410.8	1412.9	1462.7
data8-46	1806.2	1799	1785	1782.7	1856.7
data8-48	1862.1	1843.8	1848.4	1848	1942.4
data8-55	2122.5	2085.1	2054.5	2065.3	2197.2
data8-59	3657.5	3656.7	3640.7	3645.5	3848
data8-64	2823	2792.8	2774.4	2809.6	2914.9
data8-65	2498.8	2504.2	2469	2461.6	2570.3
data9-15	861.5	796.1	797.4	799.1	816.1
data9-21	1044.8	1003.6	998.8	999.9	1008.9
data9-25		917	895.4	894.6	908.1
data9-26	1066.3	1025.4	1024.6	1024.6	1025.6
data9-27	1327.8	1221.9	1211.7	1213.7	1257.7
data9-32	1380.9	1210.5	1189.6	1193.1	1202.8
data9-41		2065.3	2052.4	2038.9	2040.4
data9-44	1622.6	1617.7	1562.4	1571.3	1671.2
data9-46		1855.2	1844.5	1857.4	1899.9
data9-48	2089.7	1993.2	1982.2	1982.2	2064.2
data9-55		2183.6	2184.5	2197	2484.2
data9-59	4033.1	4021	3936.9	3932.9	4084
data9-64	3030.3	3023.1	3004.4	2999.4	3037.5
data9-65		2656.7	2625.9	2626.8	2686.3

Table 6: *Solutions of the heuristic method on real instances ($\delta = 10$) continued*

	Number of time-classes				
	5	4	3	2	1
Rough results					
Number of solutions	56	70	70	70	70
Average value	1584.51	1573.54	1564.72	1568.14	1660.65
Non-dominated solutions					
Number of solutions	70	70	70	70	70
Average value	1561.93	1561.95	1563.50	1568.14	1660.65
Average cost of consistency	-	0.00%	0.10%	0.30%	5.90%

Table 7: *Statistics on real instances*

Regarding the solution method, it is interesting to add that an average of 6000 iterations could be performed in one hour, thus allowing a large exploration of the solution space (remember that in one iteration, one fifth (one day) of the solution can be completely redrawn).

Efficiency of the algorithm with shorter computing times

In Tables 8 and 9, we compare the number of solutions found and the values of the solutions, for computing times varying between 60 seconds, 600 seconds and one hour respectively. Table 8 is based on rough results and Table 9 considers non-dominated solutions. Note that when computing a gap, only instances for which a solution was found for both computing time limits are considered.

	Number of time-classes				
Computing time	5	4	3	2	1
60 s	29	68	67	61	44
600 s	43	70	70	70	65
1 h	56	70	70	70	70

Table 8: *Number of solutions found*

	Number of time-classes				
Computing time	5	4	3	2	1
60 s	1.38%	1.45%	2.22 %	3.30 %	2.25 %
600 s	0.56%	0.58%	0.86%	1.19 %	0.56%

Table 9: *Gap compared to (non-dominated) solutions found in one hour of computing time*

One can see from Tables 8 and 9 that "good" solutions can be obtained relatively quickly using the heuristic (less than ten minutes). After these few minutes, the results are only marginally improved. The only exception concerns solutions with 1 time-class, which are the most difficult to compute and sometimes require many iterations to obtain. The shortest computing times are not assessed as, due to the complexity of the problem to be solved at each iteration of the algorithm, the method is not designed to be efficient in a few seconds.

Impact of parameter δ

Tables 10 and 11 aim to show how solutions are impacted when a tighter gap is fixed. We compare the number of solutions found and the values of the solutions obtained when defining $\delta = 5$ with the previous results ($\delta = 10$). Table 10 is based on rough results, Table 11 considers non-dominated solutions. When computing a gap, only instances for which a solution was found for both values δ are considered.

Number of time-classes	5	4	3	2	1
$\delta = 10$	56	70	70	70	70
$\delta = 5$	68	70	70	70	63

Table 10: *Number of solutions found*

Number of time-classes	5	4	3	2	1
$\delta = 5$	0.57%	0.66%	0.69%	1.29%	7.09%

Table 11: *Gap compared to (non-dominated) solutions found with* $\delta = 10$

The main observation that can be drawn from Table 10 is that solutions with one time-class can still be found most of the time (63 times out of 70). However, we cannot know whether the missing solutions could not be obtained because of our algorithm or they do not exist. Table 11 highlights that when more than 1 time-class is allowed, decreasing δ only slightly impacts the costs. Regarding solutions with 1 time-class, one can observe a clear increase in the costs.

Driver-consistency

Table 12 indicates the average number of drivers assigned to customers. Let us recall that driver-consistency is not explicitly addressed in our model and approach, but is rather handled in post-processing so that the assignment of drivers to the routes computed with the heuristic optimizes the driver-consistency. Table 13 gives the percentage of customers

having a given number of drivers out of all the customers of all the solutions (from 5 time-classes to 1) of the 70 instances. Note that these results are compiled from the solutions presented in Table 5, *i.e.* the set of rough results.

Number of time-classes	5	4	3	2	1
Average number of drivers	1.43	1.40	1.35	1.30	1.24

Table 12: *Average number of drivers per customer*

Number of drivers	1	2	3	4	5
Percentage of customers	69.06%	25.9%	4.56%	0.46%	0.00%

Table 13: *Percentage of customers having a given number of drivers*

The average number of drivers remains quite limited though not explicitly handled in the algorithm. In addition, we remark that the majority of customers have 1 or 2 drivers.

5 Conclusion

The content of this paper was initially motivated by the optimization of vehicle routes when transporting people with disabilities to day care centers. So as to capture the specific consistency expectations in this context, a new time-consistency measure, based on time-classes, was defined. From this measure, we introduced the Time-Consistent VRP (TCVRP) and proposed a solution method based on Large Neighborhood Search.

Experiments tend to demonstrate the great difficulty of this new problem and the efficiency of the heuristic proposed. An important result is that, at least for the (realistic) instances used for the computations, very consistent solutions could always be found with a relatively small impact on travel costs.

Though driver-consistency was not addressed in the model nor in the solution approach, a fairly good consistency could also be maintained for drivers.

A clear development of this work is to provide efficient exact algorithms to solve the TCVRP. The integer programming formulation proposed in this paper is not solved efficiently by the solver so it would be worth investigating more evolved modeling, valid inequalities and branch-and-cut approaches. From the application perspective, introducing the diverse constraints that were not considered here would also be important. In addition, other possible models of time-consistency could be investigated: for example, it might be interesting to consider the number of time-class changes over successive periods, instead of just counting the number of classes.

Finally, we introduced another new routing problem in the paper, the so-called VRPmTW-nw. Given the inherent interest and difficulty of this problem, further research regarding its solution certainly deserves to be conducted.

Acknowledgements

This research was funded by the GdR RO (Operations Research Group of the Scientific Research National Center).

References

[1] Frédérique Baniel. *Prise en compte d'objectifs de stabilité pour l'optimisation de collecte de déchets*. Ph.D thesis, Université de Toulouse, France, 2009.

[2] W. C. Benton and Manuel D. Rossetti. The vehicle scheduling problem with intermittent customer demands. *Computers & Operations Research*, 19(6):521–531, 1992.

[3] D.J. Bertsimas. A vehicle routing problem with stochastic demand. *Operations Research*, 40(3):574–585, 1992.

[4] D. Bredström and M. Rönnqvist. Combined vehicle routing and scheduling with temporal precedence and synchronization constraints. *European Journal of Operational Research*, 191(1):19 – 31, 2008.

[5] Ty Choi and J.L. Hartley. An exploration of supplier selection practices across the supply chain. *Journal of Operations Management*, 14(4):333–43, 1996.

[6] J.M. Day, P.D. Wright, T. Schoenherr, M. Venkataramanan, and K. Gaudette. Improving routing and scheduling decisions at a distributor of industrial gases. *Omega*, 37(1):227–237, February 2009.

[7] Guy Desaulniers, Jacques Desrosiers, and Marius M. Solomon, editors. *Column generation*. GERAD 25th Anniversary Series. Springer, 2005.

[8] L.-M. Rousseau E. Prescott-Gagnon, G. Desaulniers. A branch-and-price-based large neighborhood search algorithm for the vehicle routing problem with time windows. *Networks*, 54(4):190–204, 2009.

[9] Dominique Feillet. A tutorial on column generation and branch-and-price for vehicle routing problems. *4OR, A Quarterly Journal of Operations Research*, 8(4):407–424, 2010.

[10] Dominique Feillet, Pierre Dejax, Michel Gendreau, and Cyrille Gueguen. An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems. *Networks*, 44(3):216–229, 2004.

[11] Dominique Feillet, Michel Gendreau, and Louis-Martin Rousseau. New refinements for the solution of vehicle routing problems with branch and price. *INFOR*, 45:239–256, 2007.

[12] Peter M. Francis, Karen R. Smilowitz, and Michal Tzur. The period vehicle routing problem and its extensions. In Bruce Golden, S. Raghavan, Edward Wasil, Ramesh Sharda, and Stefan Voß, editors, *The Vehicle Routing Problem: Latest Advances and New Challenges*, volume 43 of *Operations Research/Computer Science Interfaces Series*, pages 73–102. Springer US, 2008.

[13] Michel Gendreau, Gilbert Laporte, and René Séguin. A tabu search heuristic for the vehicle routing problem with stochastic demands and customers. *Operations Research*, 44(3):469–477, may–june 1996.

[14] Chris Groër, Bruce Golden, and Edward Wasil. The consistent vehicle routing problem. *Manufacturing & Service Operations Management*, 11(4):630–643, 2009.

[15] M.A. Haughton. Measuring and managing the learning requirements of route reoptimization on delivery vehicle drivers. *Journal of Business Logistics*, 23(2):45–66, 2001.

[16] Michael A. Haughton. Assigning delivery routes to drivers under variable customer demands. *Transportation Research Part E: Logistics and Transportation Review*, 43(2):157–172, 2007.

[17] I. Ioachim, J. Desrosiers, F. Soumis, and N. Bélanger. Fleet assignment and routing with schedule synchronization constraints. *European Journal of Operational Research*, 119(1):75 – 90, 1999.

[18] A.A. Kovacs, S.N. Parragh, and R.F. Hartl. A template based adaptive large neighborhood search for the consistent vehicle routing problem. Technical report, Chair of Production and Operations Management, University of Vienna, 2011.

[19] Fabien Lehuédé, Claire Pavageau, and Olivier Péton. Un système d'aide à la décision pour planifier les transports vers les établissements médico-sociaux. In N. Vigouroux and Ph. Gorce, editors, *Proceedings of the Handicap 08 Conference, Paris*, pages 168–173, June 2008.

[20] S. Ropke and D. Pisinger. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science*, 40:455–472, 2006.

[21] L.-M. Rousseau, M. Gendreau, and G. Pesant. The synchronized vehicle dispatching problem. Technical report, Centre de Recherche sur les Transports, Université de Montréal, Canada, 2003.

[22] H. Schneider, A. Reinholz, and H.-W. Graf. Integrated modeling and optimization of the consistent vehicle routing problem with the super node concept. In *Proceedings of the* 9^{th} *Metaheuristics International Conference (MIC 2011)*, July 25-28, 2011.

[23] P. Shaw. Using constraint programming and local search methods to solve vehicle routing problems. In Michael Maher and Jean-Francois Puget, editors, *Principles and Practice of Constraint Programming – CP98*, volume 1520 of *Lecture Notes in Computer Science*, pages 417–431. Springer Berlin / Heidelberg, 1998.

[24] K. Smilowitz, M. Nowak, and T. Jiang. Workforce management in periodic delivery operations. *Transportation Science*, 2012.

[25] Ingmar Steinzen, Leena Suhl, and Natalia Kliewer. Branching strategies to improve regularity of crew schedules in ex-urban public transit. In Christian Liebchen, Ravindra K. Ahuja, and Juan A. Mesa, editors, *ATMOS 2007 - 7th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems*, Dagstuhl, Germany, 2007. Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany.

[26] I. Sungur, Y. Ren, F. Ordóñez, M. Dessouky, and H. Zhong. A model and algorithm for the courier delivery problem with uncertainty. *Transportation Science*, 44(2):193–205, 2010.

[27] C. Tarantilis, F. Stavropoulou, and P. P. Repoussis. Template-based tabu search algorithm for the consistent vehicle routing problem. *Expert Systems with Applications*, 39(4):4233–4239, 2012.

[28] H. Zhong, R.W. Hall, and M. Dessouky. Territory planning and vehicle dispatching with driver learning. *Transportation Science*, 41(1):74–89, 2007.