A Large Neighborhood Search approach to the Vehicle Routing Problem with Delivery Options

Dorian Dumez, Fabien Lehuédé, Olivier Péton

A Large Neighborhood Search approach to the Vehicle Routing Problem with Delivery Options

Dorian Dumez, Fabien Lehuédé, Olivier Péton
IMT Atlantique, Laboratoire des Sciences du Numérique de Nantes (LS2N, UMR CNRS 6004), Nantes, France
{dorian.dumez;fabien.lehuede;olivier.peton}@imt-atlantique.fr

January 23, 2020

Abstract

To reduce delivery failures in last mile delivery, several types of delivery options have been proposed in the past twenty years. Still, customer satisfaction is a challenge because a single location is chosen independently of the time at which customers will be delivered. In addition, delivery in shared locations such as lockers and shops also experience failure due to capacity or opening-time issues at the moment of delivery. To address this issue and foster consolidation in shared delivery points, we investigate the case where a customer can specify several delivery options together with preference levels and time windows. We define, in this article, the Vehicle Routing Problem with Delivery Options, which integrates several types of delivery locations. It consists of designing a set of routes for a fleet of vehicles that deliver to each customer at one of his/her options during the corresponding time window. These routes should respect capacities at shared locations such as lockers and minimum service level requirements, while minimizing the total routing costs. To solve this problem, we have designed a large neighborhood search in which a set partitioning problem is solved to regularly reassemble routes. Specific ruin and recreate operators are proposed and combined with numerous operators from the literature. A thorough experimental study was carried out to determine a subset of efficient and complementary operators. The proposed method outperforms existing algorithms from the literature on particular cases of the problem under consideration, such as the vehicle routing problem with roaming delivery locations and the vehicle routing problem with home and roaming delivery locations. New instances are generated and used both to serve as a benchmark and to propose some managerial insight into the vehicle routing problem with alternative delivery options.

Keywords: city logistics, vehicle routing, matheuristic, large neighborhood search

1 Introduction

The revenue of online stores in France was 92.6 billion euros in 2018, after an average growth of 13% per year during the last four years [Moyou, 2019]. With this growth of e-commerce, an increasing number of parcels have to be delivered each day. Consequently, several possibilities have been developed so that distribution can be faster and cheaper. According to Morganti et al. [2014], in 2012, 90% of the French and German population were within 10 minutes of a locker or a pick-up point (often shops or post offices), and the number of such locations grew by 33% between 2008 and 2012. Furthermore, trunk deliveries were recently tested on an industrial scale [McFarland, 2018].

The customer is generally required to choose one delivery location. Nevertheless, people move around during the day, for example, to go to work or take children to school. Consequently,

it is very likely that a purchaser would like to choose between several delivery options depending on the time of delivery. For attended delivery, the fact that people move around is a problem, because of "no show". According to Allen et al. [2016], in the UK, 14% of all deliveries fail. In 2014, in the UK, the cost of these failed deliveries has been estimated at £771 million.

The objective of carriers is to deliver all their parcels at a minimum cost. They also want to maintain a good quality of service, but the time window width has a great impact on delivery costs. Nevertheless, they are essential because 80% of the parcels delivered in the UK do not fit into the letterbox [Allen et al., 2016] and nobody likes to wait all afternoon at home for a parcel [Agatz et al., 2008]. Accordingly, new strategies are being developed to increase the number of successful deliveries [Florio et al., 2018].

This paper discusses the Vehicle Routing Problem with Delivery Options (VRPDO). It is an extension of both the Vehicle Routing Problem with Time Windows (VRPTW) [Savelsbergh, 1985] and the Generalized Vehicle Routing Problem (GVRP) [Ghiani and Improta, 2000]. In the VRPDO, each customer can choose to have a package delivered through several delivery options. For example, customers can receive their parcel at the office during work hours, at home in the evening or in a locker at any time.

From the standpoint of the carrier, these delivery options can be of two natures. They can take place in individual delivery locations, like a home or a car trunk. This means that only one customer can be served in that location. Otherwise, deliveries can take place in shared delivery locations, like lockers or pick-up points. This means that several parcels can be left at the same location. Some delivery locations can be limited in capacity.

The contribution of this paper is two-fold. First, from the managerial point of view, we propose a new variant of the VRP, the VRPDO. It combines delivery options to reduce delivery costs while guaranteeing a higher quality of service. From the methodological point of view, we propose a Large Neighborhood Search (LNS) metaheuristics that embeds many recent and new ideas. In particular, this LNS includes many operators from the literature and compares them rigorously.

The paper is structured as follows: In Section 2, the VRPDO is described and a mathematical model is proposed. Section 3 exposes the literature on related problems. Section 4 details the LNS developed to solve the VRPDO. In addition, our methodology to configure the algorithm is explained. Finally, Section 5 develops experiments, first to validate the method and second to draw managerial insights from randomly generated instances of the VRPDO.

2 The Vehicle Routing Problem with Delivery Options

In this section, the Vehicle Routing Problem with Delivery Options (VRPDO) is laid out. All the components of the problem are explained in Section 2.1. A mathematical model is described in Section 2.2.

2.1 Problem settings

The VRPDO is an operational optimization problem defined on a short time horizon, typically one day. In this problem, each customer can choose several *delivery options*. In the following, for the sake of conciseness, we will refer to them only as *options*.

Let us define an option as a tuple composed of a location, a preference level and a service duration, as follows:

- A *location* has a geographical address and a time window during which the service can be performed. For each location, a fixed preparation time represents the time needed

to park a vehicle and access the delivery point. We consider two types of locations: individual delivery locations and shared delivery locations. An individual delivery location is typically a personal address. Only one option can be associated with this location. It generally has a tight time window. A shared delivery location (SDL) is typically a pickup point or a locker. Several options can be associated with this location. A shared delivery location generally has a wide time window. It can be given a capacity, which is defined as the maximum number of packages which can be delivered to this location during the time horizon. For the sake of conciseness, shared delivery locations and individual delivery locations may be called "shared locations" and "individual locations", respectively.

- The *preference level* of an option is an integer in $\{1, ..., P\}$, with 1 being the value associated with the most preferred options by a customer and P, the least preferred.
- The *service duration* of an option represents the time needed to deliver the customer's package after accessing the location.

An option is associated with a unique customer.

The quality of service is measured by P numbers, P being the number of preference levels. For each level $p \in \{1, ..., P-1\}$, the service level SL_p of a solution is defined as the percentage of customers served with an option of maximum level of p. A minimal service level β_p has to be achieved for each level $p \in \{1, ..., P-1\}$.

Let us consider a homogeneous fleet of vehicles starting from a given depot and returning to this depot within the time period. Travel time and routing cost between each pair of locations is assumed to be known.

The VRPDO consists in designing a route for each vehicle, serving each customer with one of his/her options, and satisfying the shared delivery locations' capacities and the minimal service level constraints, such that the number of vehicles and the sum of routing cost are minimized in lexicographical order.

Note that each customer's order has to be delivered by a single truck (i.e. split deliveries are not allowed).

To illustrate this definition, Figure 1 shows the options from the customer perspective. Figure 1 shows the example of one customer with 3 delivery options. Delivery at home between 7pm and 9pm is the preferred option, then delivery at a shop between 8am and 11am, then a delivery in the trunk of his/her car between 9am and 6pm.

When a customer orders a parcel, he/she can choose multiple options, sorted by preference level. If the option takes place at an individual location, he/she must specify the time window. If it takes place in a shared location he/she will be able to pick up the package at any time while the location is opened. The time window for the carrier will already have been negotiated, for example shops do not want to receive parcels during their rush hours.

Figure 2 presents the options from the carrier perspective. Each customer is represented by a color/number. The individual locations are the same color (with the number) as the associated customer. The carrier has to serve all customers via exactly one option. This amounts selecting one option inside each group.

Like in the Generalized Vehicle Routing Problem (GVRP) [Ghiani and Improta, 2000], the set of options is composed of subsets, for each customer. Vehicles must visit exactly one option in each subset. In the GVRP, subsets of options form a partition of the set of locations. In the VRPDO, the subsets of options form a cover of the set of locations. This is because options associated with distinct customers can take place at the same shared location. An instance of

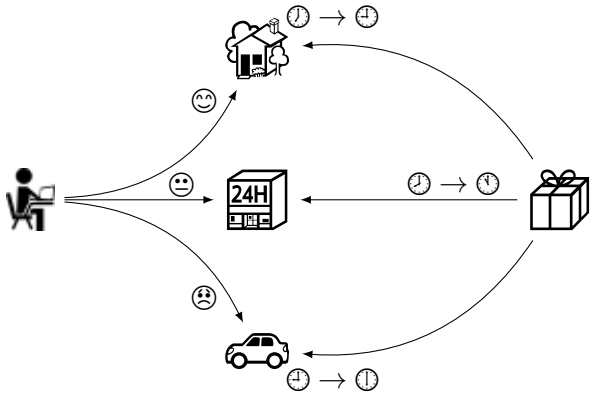

Figure 1: Example of a customer in the VRPDO

the Generalized Vehicle Routing Problem with Time Windows (GVRPTW) [Moccia et al., 2012] is an instance of the VRPDO without shared locations and a single preference level. Hence, the VRPDO can be seen as a generalization of the GVRPTW.

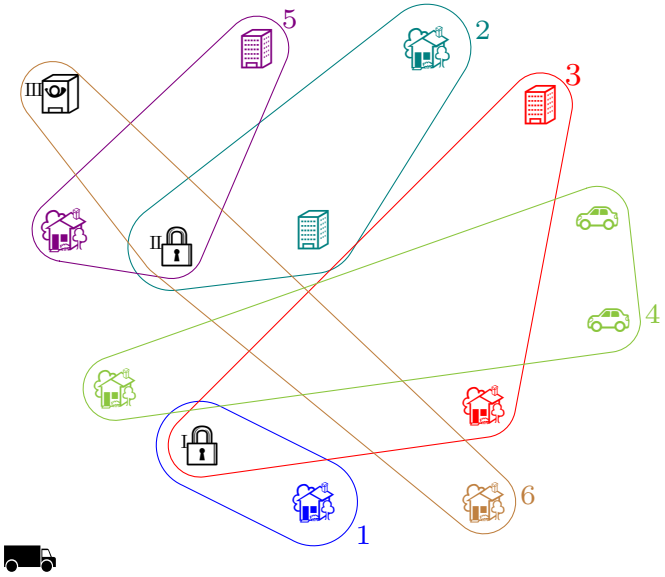

Figure 2: An instance of the VRPDO from the carrier perspective

Figure 3 represents a solution to the VRPDO instance introduced in Figure 2. Figure 4 represents a solution to the VRPTW for the same instance, using home delivery only. For the sake of conciseness, only one route has been drawn. An option that takes place in an individual location is depicted by a circle including the customer's identity and the location. A shared location is depicted by a rectangle including the location and the identity of the customers served.

The time spent by a vehicle at a location is equal to the preparation time of the location plus the service duration of the visited options. The preparation time of a location corresponds to the time to find a parking place. Hence, it is counted only once, regardless of the number of parcels delivered. The service duration of an option corresponds to the time to deliver the parcel or to put it in a box.

Customer 2 has a remote home address. Thus delivering to this customer in an alternative

delivery location avoids a big detour. The parcels for customers 1 and 3 are delivered in a shared location. The time spent at this location is equal to the preparation time of the location plus the service duration of the two options.

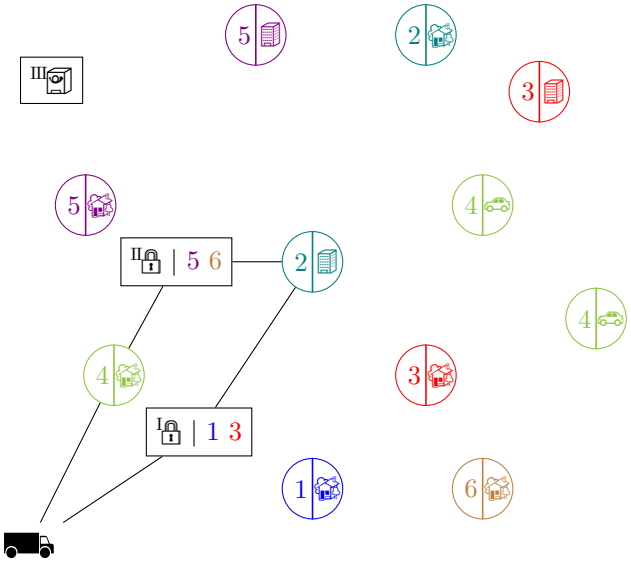

Figure 3: Example of a route in the VRPDO

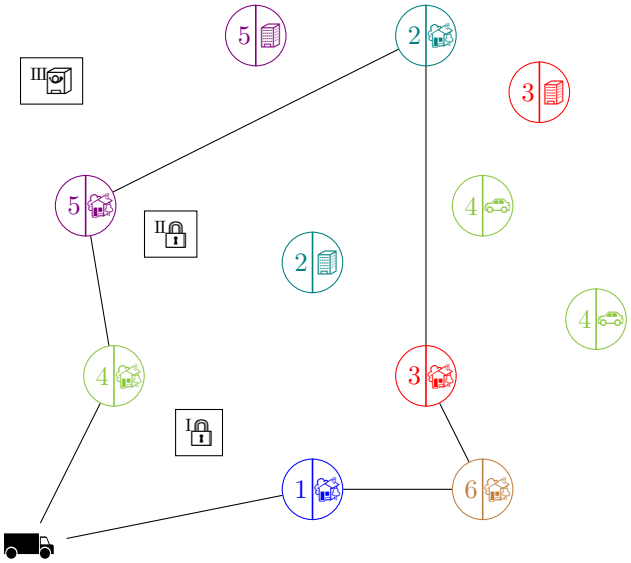

Figure 4: Example of a route in the VRPTW

We assume that a parcel always occupies one box in a locker, no matter its size. Indeed, a locker is a rack of automatic boxes so there is at most one parcel per box, as long as it fits. Hence, the capacity of the shared locations is expressed as a number of parcels. Due to uncertainty on when customers will pick up their parcels, we suppose that each box can be used only once per day. On the contrary, it is realistic to consider that some shared locations (e.g. post offices) have unbinding capacity.

The assignment of parcels to boxes is not taken into account in the VRPDO, but it would

be possible to consider different sizes of boxes by duplicating options.

Figure 5 illustrates the capacity constraints induced by shared locations for the example described in Figure 3. Not all shared locations must be visited. On the contrary, they can be visited by different vehicles as long as the number of parcels is not too large.

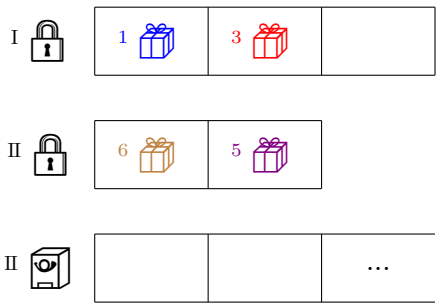

Figure 5: Shared delivery locations' capacity

Considering the quality of the service, Figure 6 represents an example with a service level constraint requiring that at least 50% of the customers are served with their preferred options (β_1) and that at least 75% of the customers are served with options of level 1 or 2 (β_2). Since 50% of the customers are served via their preferred option (SL_1) and 83% of the customers are served by an option of level 1 or 2 (SL_2), the service level constraints are satisfied.

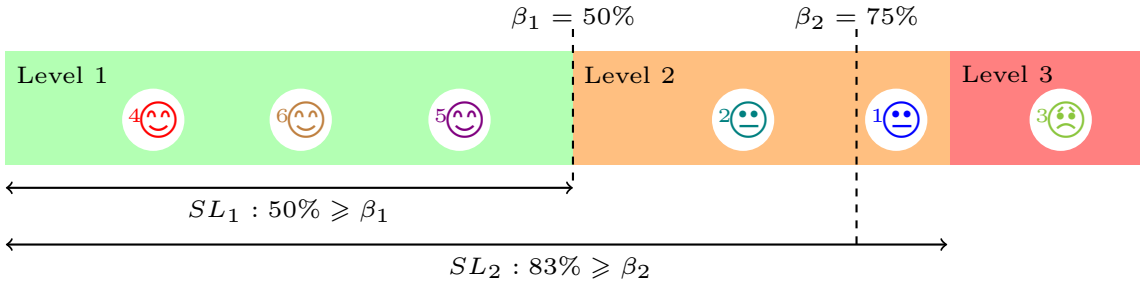

Figure 6: Service level measured with respect to a set of selected options

The last two groups of constraints, namely shared delivery location capacity and service level constraints, can be seen as capacity constraints involving all the vehicles. They fall under the category of "synchronized resources" defined by Drexl [2012] as: "At any point in time, the total utilization or consumption of a specified resource by all vehicles must be less than or equal to a specified limit".

2.2 Mathematical model

Let N be the set of customers and O the set of options. The options of customer $c \in N$ are denoted by $O_c \subset O$. Since an option belongs to a single customer, we have $\biguplus_{c \in N} O_c = O$. Let L denote the set of locations and $O_l \subset O$ be the subset of options that take place at location $l \in L$.

We model the VRPDO on an option-based complete graph $G = (V, A)$. The set of vertices, denoted $V = O \cup \{0, 0'\}$, contains one vertex for each option plus the starting depot 0 and the ending depot $0'$. The routes are defined on the option-based graph G. A route is a sequence of options belonging to distinct customers. Consequently, when multiple deliveries are made in a

shared location, it is represented by several vertices of V. Figure 7 is the representation of the route from of Figure 3.

Figure 7: Representation of the route from Figure 3 on Graph G

The travel time on arc $(i, j) \in A$ is denoted $t_{i,j}$. It includes the preparation time at the location of option j if the two locations are distinct. The cost of traveling on arc (i, j) is denoted by $c_{i,j}$. The service duration s_i at vertex $i \in V$ represents the time necessary to visit the option associated with this vertex. It is assumed to be null at depot vertices 0 and $0'$. Each option $i \in O$ is associated with the time window $[a_i, b_i]$ of its location. We assume that $a_0 = a_{0'} = 0$ and that b_0 and $b_{0'}$ correspond to the end of the working day.

Let P be the number of preference levels. For a preference level $p \in \{1, ..., P\}$, β_p is the minimal percentage of customers that can be served via an option of level p or lower. By definition $\beta_P = 100\%$ and $\forall p \in \{1, ..., P-1\} : \beta_p \leqslant \beta_{p+1}$.

Let C_l be the capacity of location $l \in L$, expressed as a number of parcels. For each option $o \in O$, we define its preference level p_o, and its demand q_o which is equal to the demand of its associated customer. We assume a homogeneous fleet of K vehicles with capacity Q.

Model 1 describes the VRPDO for a fleet of K vehicles. $x^k_{i,j}$ is a binary variable that indicates whether the k^{th} vehicle uses arc $(i, j) \in A$. y_o is a binary variable that states whether option $o \in O$ is visited. h^k_i is the service date of vehicle k at vertex $i \in V$.

The objective function (1.1) minimizes the transportation costs. Constraint (1.2) states that exactly one option must be chosen for each customer. Constraint (1.3) represent vehicle capacity. Constraint (1.4) models the capacity of the shared locations. Constraint (1.5) model the general satisfaction level. Constraint (1.6) states that each selected option should be visited by one route. The remaining constraints are the classical VRPTW constraints [Cordeau et al., 2002] : (1.7) and (1.8) ensure that the routes start and end at the depot, (1.9) ensures the continuity of the routes, (1.10) computes the service time and (1.11) ensures the respect of the time windows.

Model 1: option-based graph model

$$min \sum_{k\in\{1,...,K\}} \sum_{(i,j)\in A} c_{i,j}x^k_{i,j} \tag{1.1}$$

$$s.c \sum_{o\in O_c} y_o = 1 \qquad \forall c \in N \tag{1.2}$$

$$\sum_{o\in O} \sum_{(i,o)\in A} x^k_{i,o} q_o \leqslant Q \qquad \forall k \in \{1,...,K\} \tag{1.3}$$

$$\sum_{o\in O_l} y_o \leqslant C_l \qquad \forall l \in L \tag{1.4}$$

$$\sum_{o\in O|p_o\leqslant p} y_o \geqslant \beta_p \times |N| \qquad \forall p \in \{1,...,P\} \tag{1.5}$$

$$\sum_{(i,o)\in A} \sum_{k\in\{1,...,K\}} x^k_{i,o} \geqslant y_o \qquad \forall o \in O \tag{1.6}$$

$$\sum_{j\in V} x^k_{0,j} = 1 \qquad \forall k \in \{1,...,K\} \tag{1.7}$$

$$\sum_{i\in V} x^k_{i,0'} = 1 \qquad \forall k \in \{1,...,K\} \tag{1.8}$$

$$\sum_{(i,j)\in A} x^k_{i,j} - \sum_{(j,i)\in A} x^k_{j,i} = 0 \qquad \forall k \in \{1,...,K\};\ \forall j \in O \tag{1.9}$$

$$x^k_{i,j}(h^k_i - h^k_j + t_{i,j} + s_i) \leqslant 0 \qquad \forall k \in \{1,...,K\};\ \forall (i,j) \in A \tag{1.10}$$

$$a_i \leqslant h^k_i \leqslant b_i \qquad \forall k \in \{1,...,K\};\ \forall i \in V \tag{1.11}$$

$$x^k_{i,j} \in \{0,1\} \qquad \forall k \in \{1,...,K\};\ \forall (i,j) \in A$$

$$y_o \in \{0,1\} \qquad \forall o \in O$$

$$h^k_i \geqslant 0 \qquad \forall k \in \{1,...,K\};\ \forall i \in V$$

3 Relevant literature

The VRPDO is a VRPTW with multiple delivery options and synchronized resources. This section is organized according to these three aspects: VRPTW, delivery options and resource synchronization. In addition, some closely related problems are laid out.

3.1 VRPTW

Since the introduction of the VRPTW by Savelsbergh [1985], a plethora of papers have been published on this subject. We refer to Bräysy and Gendreau [2005a,b] for a review of the applicable literature and to Vidal et al. [2013] for the latest advances. Many LNS heuristics have been used to solve problems related to the VRPTW. They will be briefly discussed in Section 4.

3.2 Delivery options

In the VRPDO, customer requests can be satisfied at various locations. This characteristic is shared by the Generalized Vehicle Routing Problem with Time Windows (GVRPTW) [Moccia

et al., 2012]. The GVRPTW is a specific case of the VRPDO without shared locations and with a single preference level. The literature on the GVRPTW is very scarce. We refer to Afsar et al. [2014] for details on the GVRP, and to Moccia et al. [2012] for details of the GVRPTW. Yuan et al. [2018] develop a B&C for the GTSPTW (a single vehicle GVRPTW) that can solve instances with 30 clusters. Moccia et al. [2012] propose a tabu search that is able to tackle instances with 120 clusters within few minutes.

The VRP with multiple time windows (VRPMTW) [Favaretto et al., 2007] can be seen as a special case of the VRPDO and of the GVRPTW, where all options of a customer take place at the same location and their time windows are disjointed. For more detail we refer to Tricoire et al. [2010], Belhaiza et al. [2014] and Hoogeboom and Dullaert [2019].

In addition to the GVRP, some classes of vehicle routing problems include choices in the locations to visit. The goal of these problems is to select locations to visit and find routes between these locations, such that requirements are met at minimum cost or such that profit is maximized. This category of problem includes the Team Orienteering Problem (TOP), the VRP with profit [Archetti et al., 2014, Vansteenwegen et al., 2011], the Traveling Purchaser Problem (TPP) [Bernardino and Paias, 2018], and the Covering Tour Problem (CTP) [Kammoun et al., 2017].

Among the heuristics that have been proposed to solve VRP with choices in locations or time windows, most of the authors include a perturbation component in their method. For variants of the TOP, Souffriau et al. [2013] combine GRASP with ILS and Tricoire et al. [2010] use VNS. For variants of the CTP, Allahyari et al. [2015] combine GRASP with ILS, Takada et al. [2015] use ILS, Vargas et al. [2015] work with ALNS and Kammoun et al. [2017] propose a VNS.

Besides, some papers propose using dynamic programming. In the context of a multi-period TOP with multiple time windows, Tricoire et al. [2010] use dynamic programming to choose which time window should be used in a given route. Moccia et al. [2012] handle the GVRPTW, when a customer is inserted in a route, the options for the other customers of this route can be changed through dynamic programming. Vargas et al. [2015] solve the CTP with an ALNS algorithm that uses dynamic programming to decompose a giant tour into routes.

3.3 Resource synchronization

Drexl [2012] underlined that Hempsch and Irnich [2008] was almost the only paper considering synchronized resources. Few other heuristics have been proposed since then. In Hempsch and Irnich [2008], Grangier et al. [2017b], Froger et al. [2017], the resources are only temporarily used during a limited period of time, before becoming available again. For example, in Grangier et al. [2017b], a truck uses a dock at a satellite facility only during the loading and unloading operations.

In the VRPDO, the synchronized resources are permanently used during the whole time horizon. Thus, it is only necessary to count the number of parcels in each locker and the number of visited options of each preference level to check the satisfaction of the synchronized resource constraints.

Souffriau et al. [2013] defines a variant of the TOP applied to tour planning for tourists. It is called the Multiconstraint Team Orienteering Problem with Multiple Time Windows (MC-TOP-MTW). The routes represent the different days of the trip. The synchronized resource constraints represent the budget and the maximal number of monuments of each type that can be visited. They consider these resources in a local search with a label on the locations. These

labels are updated at each modification of the schedule. To get good solutions even with these resources, their algorithm relies on perturbations through a combination of GRASP with ILS. Besides, local search moves are chosen with a score. It is a function of the increase in the objective function, time shift, and resource consumption.

3.4 Closely related problems

Now that the three components of the VRPDO have been described, let us focus on very close problems.

Reyes et al. [2017] define the VRP with Roaming Delivery Locations (VRPRDL). This problem involves delivering parcels into the trunks of cars, which move according to known schedules. In the VRPRDL, there are no synchronized resources, the time windows of a customer are disjointed and cars do not move faster than the delivery truck. This problem is solved with an LNS heuristic. Classical operators are adapted and dynamic programming is used to re-optimize routes by selecting better options without changing the customer sequence in the route. The combination with home delivery is considered in a variant called VRP with Home and Roaming Delivery Locations (VRPHRDL). It is defined by Ozbaygin et al. [2017] and solved to optimality by a Branch-and-Price (B&P) algorithm for instances with up to 60 customers.

Sitek and Wikarek [2017] define the Capacitated VRP with Pick-up and Alternative Delivery (CVRPPAD). This paper considers multiple options for each parcel to be delivered. Lockers and post offices are modeled with limited capacities. Customer preferences are modeled with a penalty on the objective function if a non-desired option is used. No time windows are considered. Their method relies on a pre-processing phase done by constraint programming. A heuristic then groups parcels together before assigning them to a route.

Zhou et al. [2018] define the Multi-Depot Two-Echelon Vehicle Routing Problem with Delivery Options (MD-2EVRP-DO). In this paper, a parcel can either be delivered to the customer's home or to a selected pick-up facility. Shared locations are incapacitated and customer preferences are modeled through penalties in the objective function. Time windows are not considered. This problem is solved with a multi-population genetic algorithm that embeds an ad-hoc local search.

The most closely related study seems to be the one by Yuan et al. [2019], on the GVRPTW. Their metaheuristic method is based on a Set Partitioning formulation of the problem, coupled with an LNS heuristic and a local search that works with dual information.

4 Solution method

The Large Neighborhood Search (LNS) metaheuristic was first proposed by Shaw [1998] in a constraint programming context. In LNS, the current solution is iteratively improved by ruining it (i.e. removing a part of it) and recreating it (i.e reinserting the removed parts). This process is repeated until a stopping criterion (usually a time limit). The potential of LNS for solving a large variety of vehicle routing problems was revealed by Ropke and Pisinger who proposed an adaptive version of LNS, known as ALNS, consisting of multiple search operators adaptively selected according to their past performance [Ropke and Pisinger, 2006a,b, Pisinger and Ropke, 2007]. LNS has been successfully applied to many variants of vehicle routing problems [Pisinger and Ropke, 2019] and the literature is abundant. Recently, Turkeš et al. [2019] compiled many papers using an LNS heuristic in a meta-analysis that concludes that the adaptive component proposed by Ropke and Pisinger [2006a] has, at best, a small impact.

We have selected a few papers of major importance with respect to the VRPDO: Shaw [1998] (VRP, VRPTW), Ropke and Pisinger [2006a,b], Pisinger and Ropke [2007] (diverse VRPTW), Nagata and Bräysy [2009] (VRPTW), Prescott-Gagnon et al. [2009] (VRPTW), Demir et al. [2012] (VRPTW and variants) and Christiaens and Vanden Berghe [2020] (CVRP and variants). Almost all operators from these papers were implemented, so that we have a representative panel of the LNS operators. In addition, ad-hoc operators for the VRPDO were developed in order to integrate synchronized resources and shared locations. The 20 ruin operators implemented are presented in Section 4.2 and the 15 recreate operators implemented are presented in Section 4.3. Numerous tests were performed, using a rigorous methodology, to restrict the operators used in our LNS implementation. The experimental procedure to selected operators is presented in Section 4.4. For the sake of readability, in Sections 4.2 and 4.3, the operators that were selected are listed with an [S] and the ones that were discarded are listed with a [D], with respect to the results explained in Section 4.4. Note that discarded operators may be as efficient as selected ones, but redundant in our configuration. For the sake of traceability, the original names have been used whenever possible.

Sections 4.5 and 4.6 present the acceptance criterion and the set partitioning problem, respectively. The final tuning of the presented algorithm is detailed in Section 4.7.

4.1 Large Neighborhood Search heuristic

Algorithm 1 presents the main steps of the proposed heuristic. It is an LNS metaheuristic with a Set Partitioning component, which we therefore call LNS-SPP. The main loop of the iterative process is from lines (3) to (18). On line (5), a ruin operator σ^- is randomly selected in Σ^-. Each operator has a given constant probability of being selected. The same process is used at line (6) to select a recreate operator σ^+ in Σ^+. At line (7), the size Φ of the destruction is randomly chosen in a given interval $[\delta, \Delta]$. The destruction size is the percentage of customers to be removed from the current solution.

The selected operators are applied to the current solution s' at line (8). First, $\Phi\%$ of the customers are removed from the solution with the chosen ruin operator σ^-. These customers are placed in the so-called request bank. Second, the customers from the request bank are inserted in the solution by the recreate operator σ^+. At line (9), the routes of the newly generated solution are stored into a set of routes: the pool of routes R. On line (10), an acceptance criterion is used to decide whether the new solution becomes the current solution for the next LNS iteration. The routes of pool R are recombined every η iterations by solving a Set Partitioning Problem (SPP) on line (14). After this combination, the pool of routes R is emptied, on line (15).

The VRPDO has two lexicographic objectives: (1) to minimize the number of vehicles, and (2) to minimize the routing cost (described by the objective function 1.1). Similarly to Ropke and Pisinger [2006b], Algorithm 1 is run twice, with half of the time budget for each part. During the first phase, the number of vehicles is decreased by removing the smallest route from the solution each time a feasible solution is found (the customers are placed in the request bank). During the second phase, the cost of the solution is minimized, using the minimum number of vehicles found in a feasible solution obtained during the first phase. The configuration of our algorithm does not change between the two phases.

Algorithm 1: LNS-SPP

Parameters: a set Σ^+ of recreate operators, a set Σ^- of ruin operators, a frequency η
Input: an initial solution s

pool of routes $R = \emptyset$
nbIt $= 1$
while the time budget is not reached **do**
  $s' \leftarrow s$
  randomly select a ruin operator $\sigma^- \in \Sigma^-$
  randomly select a recreate operator $\sigma^+ \in \Sigma^+$
  randomly select a destruction size $\Phi \in [\delta, \Delta]$
  $s' \leftarrow \sigma^+(\sigma^-(s', \Phi))$
  store the routes of s' into pool R
  if s' meets the acceptance criterion **then**
    $s \leftarrow s'$
  end if
  if nbIt$\%\eta = 0$ **then**
    $s =$ best combination of the routes of R found with a Set Partitioning Problem
    $R = \emptyset$
  end if
  nbIt $=$ nbIt $+ 1$
end while
return the best feasible solution found

4.2 Ruin operators

Ruin operators use different rules to remove customers from the solutions. Upon removal, customers are placed in the request bank of the solution.

We divide ruin operators from the literature into two categories, denoted local ruin operators and large ruin operators. A local ruin operator deletes customers so that even if few customers are deleted, it is likely that it is possible to improve the solution. On the contrary, a large ruin operator requires that more customers be deleted. Indeed, with a large ruin operator, if too few customers are removed, it is likely that they will be re-inserted in the same position.

4.2.1 Local ruin operators

The local ruin operators implemented from the literature are:

[S] *Distance-related removal* [Ropke and Pisinger, 2006a] : removes customers that are close to each other with respect to the Euclidean distance.

[D] *Node neighborhood removal* [Demir et al., 2012]: removes customers that are close to each other with respect to the infinity norm.

[D] *Proximity removal* [Prescott-Gagnon et al., 2009]: removes customers that are close both from a spatial and a temporal point of view, according to a parameterless formula. This is an extension of Shaw removal [Shaw, 1998].

[S] *(Split) String removal* [Christiaens and Vanden Berghe, 2020]: removes sequences of customers in the routes of the current solution, either conserving, or not, a sub-string in the middle.

4.2.2 Large ruin operators

The large ruin operators implemented from the literature are:

[S] *Random removal* [Ropke and Pisinger, 2006a]: randomly removes customers.

[D] *Demand-related removal* [Demir et al., 2012]: removes customers that have demands of similar size.

[S] *Time-related removal* [Pisinger and Ropke, 2007]: removes customers that are served at approximately the same time.

[S] *Zone removal* [Demir et al., 2012]: randomly removes customers into predefined fixed rectangular zones.

[S] *Cluster removal* [Pisinger and Ropke, 2007]: removes customers that are served by the same route in the current solution. A route is randomly selected and the Kruskal's algorithm is run on the arcs of this route until two clusters remain. All the customers in one of them, randomly chosen, are removed.

[S] *Route removal* [Nagata and Bräysy, 2009]: removes all the customers of a route.

[D] *Distance worst removal* [Ropke and Pisinger, 2006b]: iteratively removes the customer with the highest individual service cost. The individual service cost of a customer is the cost of the arcs that enter and exit the location where the given customer is served in the current solution, minus the cost of going directly from the previous location on the route to the next one. If it is a shared location, this cost is divided by the number of customers served at this location on the same route.

[D] *Time worst removal* [Demir et al., 2012]: removes the customers that cause the largest time loss.

[D] *Neighborhood removal* [Demir et al., 2012]: first, the cost of each route divided by the number of customers served by this route is computed. The customers are then removed sequentially by decreasing order of the difference between their individual service cost and the average service cost of their route.

[D] *Node-pair history removal* [Pisinger and Ropke, 2007]: memorizes the cost of the best solution that uses each arc. The operator removes the nodes that are reached via arcs with the largest score.

[S] *Historical knowledge node removal* [Demir et al., 2012]: this history removal memorizes the lowest individual service cost of each customer. The operator removes the customers with the largest difference between their current individual service cost and their lowest individual service cost. It can be see as a history-biased worst removal.

Ruin operator for the VRPTW can be adapted to the VRPDO in two ways: option-based or customer-based. Hence, the ruin operators that we are using for the VRPDO can be split between these two categories. An option-based operator only takes in account the options that are currently visited to serve the customers. A customer-based operator takes all the options

into account. Let us describe an example with *distance-related removal*. The distance between two customers in *distance option-based related removal* is the distance between the options that are currently visited to serve these customers. On the contrary, in *distance customer-based related removal*, it is the minimal distance between any two options of these customers. That is to say, the option-based version will remove customers that are currently served in close locations and the customer-based one will delete customers that are potentially served in close locations.

4.2.3 VRPDO specific ruin operators

The new ruin operators specifically developed for the VRPDO are:

[D] *Preference-oriented random removal*: randomly selects customers and deletes them with a probability based on the preference level of the options currently visited to serve them. The probability of deleting a selected customer, currently served with option o, is $(1/P+1-p_o)^3$. When there are three preference levels, the probability of deleting a customer served with an option of level 3, 2 and 1 is 1.0, 0.125 and 0.04, respectively.

[S] *SDL-oriented random removal* (Shared Delivery Location-oriented random removal): randomly selects customers in the solution. If the selected customer is served in an individual location, the probability of being deleted is only 10%. Otherwise, if customers are served in a shared location, they are always deleted.

[D] *Random SDL removal*: randomly selects a shared delivery location and removes all the customers served at this location.

[D] *SDL-related removal*: selects a shared delivery location and randomly deletes customers that have an option at this location.

[D] *SDL-worst removal*: a *distance worst removal* where the detour cost is fully assigned to all the customers served at the shared delivery locations. The detour cost is not divided by the number of customers served at this location.

All these ad-hoc ruin operators are large ruin operators. With the exception of the *SDL related removal*, they are all option-based.

4.3 Recreate operators

Most recreate operators follow the best insertion principle: any given customer is inserted at the position that minimizes the routing cost increase. To compute the best insertion of customer c in route r, we try to insert all customer options in all positions of route r. Only feasible insertions are performed by the algorithm. Hence, a solution always satisfies the vehicle capacity constraints, time windows, shared location capacities, and service level constraints. The only form of infeasibility considered in LNS-SPP is the fact that not all the customers are served, i.e the request bank can be non-empty.

The forward time slacks [Savelsbergh, 1992] of all routes are stored in order to evaluate insertions in constant time with respect to time windows. The usage of each shared location and of each preference level is stored. Hence, testing the validity of insertions with respect to synchronized resources is done in constant time. When an insertion is performed, the forward time slacks of the corresponding route are updated in linear time with respect to the length of the route. The update of capacity usage is done in constant time.

4.3.1 Recreate operators from the literature

We divide the recreate operators from the literature into two categories: list heuristics and others. Most of the operators proposed in the literature may evaluate the insertion of a given customer into a given route multiple times. Typically, the insertion of each customer in each route is evaluated once at the beginning. After each modification, the insertion of the remaining customers in the modified route is then re-evaluated. In a list heuristic, the insertion of a customer into a route is evaluated at most once. Typically, the customers in the request bank are sorted once and inserted in this order. Consequently, list heuristics are very fast recreate operators.

The list heuristics implemented from the literature are:

[S] *Random order best insertions* [Christiaens and Vanden Berghe, 2020]: sequentially inserts the customers in the request bank at their best insertion position in a random order.

[D] *Oldest first best insertions* [Christiaens and Vanden Berghe, 2020]: sequentially inserts the customers in the request bank at their best insertion position in non-increasing order of the number of iterations since the last time a given customer was served.

[S] *Largest first best insertions* [Christiaens and Vanden Berghe, 2020]: sequentially inserts the customers in the request bank at their best insertion position in non-increasing order of their demand.

[D] *Farthest first best insertions* [Christiaens and Vanden Berghe, 2020]: sequentially inserts the customers in the request bank at their best insertion position in non-increasing order of their distance to the depot.

[D] *Closest first best insertions* [Christiaens and Vanden Berghe, 2020]: sequentially inserts the customers in the request bank at their best insertion position in increasing order of their distance to the depot.

The other operators implemented from the literature are:

[D] *Best temporal insertions* [Demir et al., 2012]: inserts the customers so that the loss of time is minimal. The loss of time is defined as the waiting time at the inserted option plus the waiting time at the next option on the route. Customers can be processed either in random order or by decreasing order of demand, respectively.

[D] *Greedy best insertion* [Ropke and Pisinger, 2006b]: iteratively computes the cheapest insertion for each customer and inserts the customers that have the lowest insertion cost.

[S] *k-regret* [Ropke and Pisinger, 2006b]: iteratively computes the best insertion cost on each route for each customer and inserts the one that has the largest difference between its best insertion cost and next $(k-1)$ route's best insertion costs.

[S] *Ejection search* [Nagata and Bräysy, 2009]: first, all the customers in the request bank are placed in a FIFO structure. The customers from this structure are inserted into the solution by allowing some customers from the solution to be removed and put in the queue. As in Curtois et al. [2018] the procedure is heuristically sped up. First, to insert one customer, at most two customers can be removed from the solution. Second, insertions are tested with an increasing number of removed customers. If a feasible insertion is found, insertions with more removals will not be tested. Third, when customers must

be removed to insert the customer in question, the insertion that removes the customers with the smallest score is chosen. The score of a customer is the number of times where no feasible insertion was found for this customer during all the calls to *ejection search*. Finally, the number of iterations of *ejection search*, at each call, is limited to five times the initial size of the request bank.

4.3.2 VRPDO specific recreate operators

The operators specifically developed for the VRPDO are:

[S] *Preferred best insertion*: this operator considers the preferred insertion. The preferred insertion of a customer is the cheapest feasible insertion at the best available preference level. The customer with the cheapest preferred insertion is inserted first with this insertion.

[D] *Preference regret*: the best insertion is computed for each customer and for each preference level. Let C_p^i be the cost of the best insertion of a customer i with an option of level p or lower. The *preference regret* score of customer i is $\sum_{p=1}^{P-1}(C_P^i - C_p^i)$. Customers are always inserted at their cheapest position and the customer with the largest regret is inserted first.

[D] *Normalized best insertion*: the equalized insertion cost is the real cost of insertion, divided by the capacity of the location. *Normalized best insertion* is a *greedy best insertion* that uses an equalized insertion cost to select the insertion possibility for each customer and select the first customer to insert.

[S] *SDL-regret* (Shared Delivery Location regret): customers are inserted at their cheapest insertion by decreasing order of their regret. In this version, the regret of a customer is the difference between the insertion cost when all options are allowed or the cost when only individual locations are authorized. For example, let us consider the partial solution represented in Figure 8 (same instance as in Figure 3). Customers 5 and 6 are not served. Both can be served through locker II, where customer 4 is currently being served. But this locker only has a capacity of two, as in Figure 5. For both customers 5 and 6, the cheapest insertion is in this locker, with a cost of 0. Figures 9 and 10 show the cheapest insertion of customers 5 and 6 without considering shared locations. Hence, the SDL-regret for customer 6 is higher than that of customer 5. In this case, the *SDL-regret* will select customer 6 first and insert him/her in the locker.

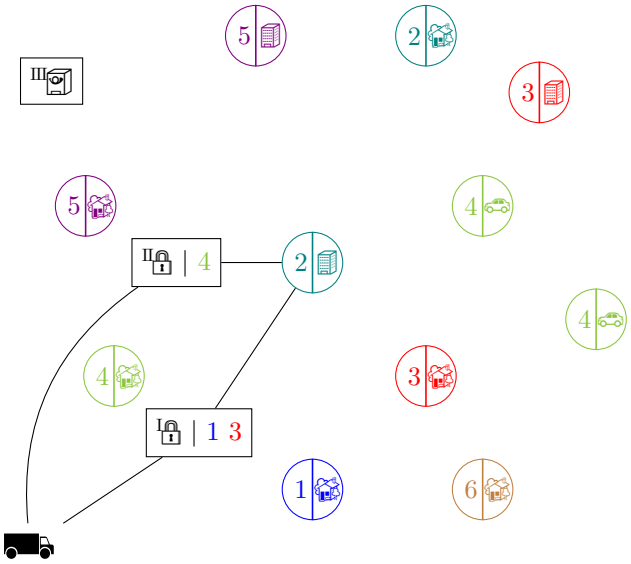

Figure 8: Example of a partial solution

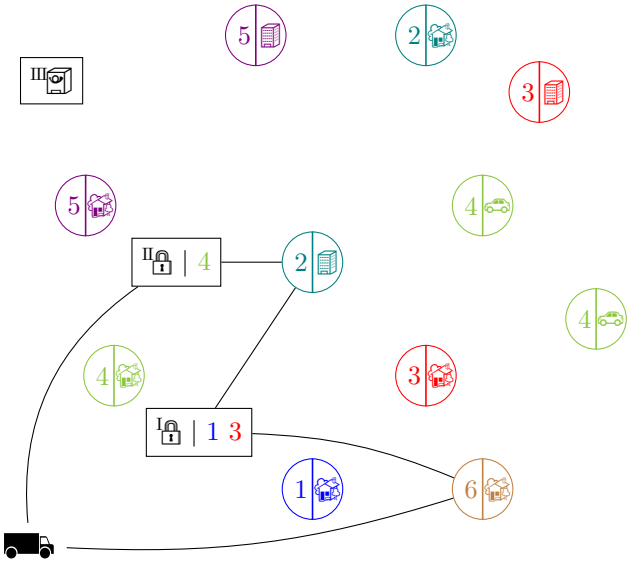

Figure 9: Insertion of the brown(6) customer without lockers

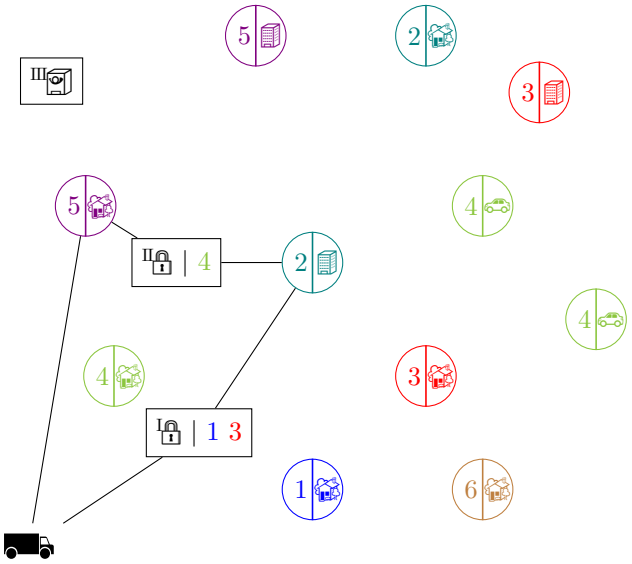

Figure 10: Insertion of the purple(5) customer without lockers

4.4 Operator selection

As described in sections 4.2 and 4.3, a number of operators have been implemented. Because it does not seem useful to keep them all, we searched for a configuration of LNS-SPP with fewer operators. In this section, we define a *configuration* as a subset of ruin operators and a subset of recreate operators.

A statistical study was performed to choose a configuration from the 20 ruin operators and the 15 recreate operators that were implemented. Tests were performed on a representative set of randomly generated VRPDO instances. To determine if one configuration is significantly different from another, we use the Wilcoxon pairwise test [Wilcoxon, 1945] with a threshold of 5%. This test compares two populations of results. In our case, it compares the results obtained by two configurations on each instance of a set of VRPDO instances. This methodology is inspired from Stützle [2018]. Note that we also tried the automatic configuration package IRACE [López-Ibánez et al., 2011], but it did not converge after several days of computation.

We define a class of operators as a group of operators that have similar purposes. First, we split the ruin operators into 4 classes: random removals, related removals, worst removals and history removals. Second, we split the recreate operators into 4 classes: list heuristics, time best insertion heuristics, regret heuristics and ejection search. This classification is detailed in Tables 1 and 2.

This statistical study is decomposed in two phases: (1) a study of the impact of each class of operators; (2) a study of the impact of each individual operator.

To study classes of operators, the reference configurations are: the full configuration with all the operators, and a minimal configuration with as few operators as possible. All the operators of each class are removed from the full configuration and added to the minimal configuration. All these "sub-configurations" were tested on all instances of the test set and compared. This first phase determines whether the operators in question are redundant with other operators of the full configuration, and whether they improve the results of the minimal configuration.

Based on the previous results, we build an intermediate configuration. The operators from this configuration are changed one by one. If a given operator was used, then we deactivate it, otherwise we add it to the configuration. All these alternative configurations are compared with the intermediate configuration. It determines if the assessed operator significantly improves the

results, or if it is redundant with used operators.

The results of the operators are summarized in Tables 1 and 2. Operators marked with $^{++}$ are considered essential. Operators marked with $^{+}$ slightly improve the results. Operators marked with $^{-}$ are useless or redundant with already kept operators, and adding them does not improve the results. The selected operators are those rated $^{++}$ and $^{+}$.

	Option-based operators	Customer-based operators
Random removals	$^{-}$ preference-oriented random removal $^{+}$ SDL-oriented random removal $^{-}$ random SDL removal	$^{+}$ random option removal
Related removals	$^{-}$ distance-related option removal $^{-}$ node neighborhood removal $^{-}$ time-related option removal $^{++}$ (split) string removal $^{++}$ cluster removal $^{+}$ route removal	$^{+}$ distance-related customer removal $^{+}$ zone removal $^{+}$ time-related customer removal $^{-}$ proximity customer removal $^{-}$ demand-related removal $^{-}$ SDL-related removal
Worst removals	$^{-}$ distance worst removal $^{-}$ time worst removal $^{-}$ neighborhood removal $^{-}$ SDL worst removal	
History removals	$^{-}$ node-pair history removal	$^{++}$ historical knowledge node removal

Table 1: Overview of configuration experiments for ruin operators

	VRPTW operators	VRPDO operators
List heuristics	$^{++}$ random order best insertion $^{-}$ oldest first best insertion $^{++}$ largest first best insertion $^{-}$ farthest first best insertion $^{-}$ closest first best insertion	$^{-}$ normalized best insertion $^{+}$ preferred best insertion
Time best insertions	$^{-}$ time best insertion	
Regrets	$^{-}$ greedy best insertion $^{++}$ 2-regret $^{-}$ 3-regret $^{-}$ 4-regret	$^{+}$ SDL-regret $^{-}$ preference regret
Ejection search	$^{+}$ ejection search	

Table 2: Overview of configuration experiments for recreate operators

4.5 Acceptance Criterion

On line 10 of Algorithm 1, we determine whether the newly generated solution should be accepted as the current solution at the next iteration.

Ropke and Pisinger [2006a] use the Metropolis criterion from Simulated Annealing [Kirkpatrick et al., 1983]. To deal with partial solutions, a modified cost is proposed in Pisinger and Ropke [2007]. The modified cost of Equation (1) penalizes the unserved customers with factor β. In this formula, B is the request bank of the current solution, *cost* is its routing cost (described by 1.1) and N is the set of customers.

$$\text{modified cost} \;=\; \text{cost} \times \left(1 + \beta.\frac{|\text{B}|}{|\text{N}|}\right) \tag{1}$$

Santini et al. [2018] conducted a comprehensive study on the acceptance criteria for the LNS metaheuristic. In their conclusion about the CVRP, they advocate for the record-to-record criterion [Dueck, 1993]. With this criterion, the solution is accepted if its modified cost is less than $T\%$ larger than the modified cost of the best known solution. Furthermore, they propose decreasing the acceptance threshold T during the algorithm. T decreases linearly between its initial value at the beginning and 0, when the time limit is reached. They conclude that the best values of T and β depend of the type and size of instance.

In our implementation, we use the record-to-record criterion with modified cost (1). To avoid tuning parameters and get a reliable acceptance criterion, we propose a simple adaptive procedure.

Our experiments empirically show that LNS-SPP performs well if the ratio of accepted solutions is between 4% and 14%. T and β are changed so as to maintain the ratio of accepted solutions in this target. The ratio of accepted solutions is periodically evaluated. If the ratio of accepted solutions is less than 4%, T is multiplied by 1.5 and β is divided by 1.5. On the contrary, if it is larger than 14%, T is divided by 1.5 and β is multiplied by 1.5.

4.6 Set Partitioning Problem

The utilization of the Set Partitioning Problem (SPP) to solve vehicle routing problems was first introduced by Foster and Ryan [1976]. It is now widely used in column generation methods [Toth and Vigo, 2014] for many routing problems [Archetti et al., 2014]. It can also be used to recombine routes that are produced by a heuristic. The SPP is used as a post-optimization technique [Rochat and Taillard, 1995, Mancini, 2017, Gschwind and Drexl, 2019], as well as inside hybrid heuristics [Prescott-Gagnon et al., 2009, Groër et al., 2011, Mendoza and Villegas, 2013, Subramanian et al., 2013, Parragh and Schmid, 2013, Yıldırım and Çatay, 2015, Grangier et al., 2017a, Tellez et al., 2018].

We solve the SPP by solving a Set Covering Problem (SCP) and by repairing the solution if a customer is served more than once. Yıldırım and Çatay [2015] show that solving an SCP instead of a SPP slightly shortens solving time. Additionally, an exact repair is rarely needed and a greedy procedure finds the optimal reparation almost all the time.

The SCP model for the VRPDO is represented in Model 2. R is a set of routes that are valid with respect to time windows and vehicle capacity. Let us define R as the pool of routes generated in Algorithm 1. Indicator α_r^o has value 1 if the option $o \in O$ is visited by the route $r \in R$; it is equal to 0 otherwise. The cost of route $r \in R$ is denoted w_r. Let z^r be a binary variable that indicates whether route $r \in R$ is used in the solution.

The objective function (2.1) minimizes the total cost of the solution, i.e the sum of the cost of the routes used. Constraints (2.2) state that each customer must be served at least once. Constraints (2.3) express the capacity of the shared locations. Constraints (2.4) are the service level constraints. Because it is a set covering formulation, more than one option may be used to serve a customer. Consequently, we express the service level constraint as a capacity constraint. Constraints (2.3) and (2.4) are the synchronized resource constraints. Constraints (2.5) set the upper bound on the number of vehicles used in the solution to K.

Model 2: SCP for the VRPDO

$$min \sum_{r \in R} w_r z_r \tag{2.1}$$

$$s.c \sum_{r \in R} \sum_{o \in O_c} \alpha_r^o z_r \geqslant 1 \qquad \forall c \in N \tag{2.2}$$

$$\sum_{r \in R} \sum_{o \in O_l} \alpha_r^o z_r \leqslant C^l \qquad \forall l \in L \tag{2.3}$$

$$\sum_{r \in R} \sum_{o \in O | p_o \geqslant p} \alpha_r^o z_r \leqslant (1 - \beta_{p+1}) \times |N| \qquad \forall p \in \{1, ..., P-1\} \tag{2.4}$$

$$\sum_{r \in R} z_r \leqslant K \tag{2.5}$$

$$z_r \in \{0, 1\} \qquad \forall r \in R$$

Model 2 is solved every η iterations by an ILP solver. The set of routes R is then composed of the routes that have been generated by Algorithm 1 during the last η iterations. As proposed by Tellez et al. [2018], frequency η can be adapted. The value of η is reduced by a quarter when the solver does not succeed in proving optimality, nor in improving the best known solution, twice in a row. We extend this procedure as follows. η is increased by a quarter if the optimality is proven, or if the best-known solution is improved, twice in a row.

In addition, we observe that it is not necessary to solve the SPP every η iterations. While the LNS still improves the current solution, it is better to wait for stabilization. Hence, the SPP is solved only if LNS cannot improve the cost of the best-known solution's cost by more than $\rho\%$ during the last η iterations.

We observe that the SCP defined for the VRPDO is more difficult to solve than the pure VRPTW set covering formulation. Thus, an extension of Model 2 is proposed to reduce solving time. We introduce binary variables y_o that indicate whether option $o \in O$ is visited. These variables allow the solver to branch on options, hence discarding a lot of routes.

Constraints (3.1), (3.2), and (3.3) reformulate constraints (2.2), (2.3), and (2.4) with the y variables, respectively. Constraints (3.4) bind the two sets of variables by stating that if an option is visited, then the corresponding y_o must be set at value 1.

The following constraints are added to Model 2

$$\sum_{o \in O_c} y_o = 1 \qquad \forall c \in N \tag{3.1}$$

$$\sum_{o \in O_l} y_0 \leqslant C_l \qquad \forall l \in L \tag{3.2}$$

$$\sum_{o \in O | p_o \leqslant p} y_o \geqslant \beta_p \times |N| \qquad \forall p \in \{1, ..., P\} \tag{3.3}$$

$$\sum_{r \in R} \alpha_r^o z_r \leqslant y_o \qquad \forall o \in O \tag{3.4}$$

$$y_o \in \{0, 1\} \qquad \forall o \in O$$

4.7 Final parameter tuning

In our LNS-SPP, the probability of selecting each operator is constant throughout the algorithm. In the proposed implementation, all ruin operators are equiprobable. The probability of selecting each recreate operator is inversely proportional to its average running time. Furthermore, Christiaens and Vanden Berghe [2020] propose to perform a huge number of small and fast iterations in LNS in order to compensate the lack of local search in this metaheuristic. We added this functionality as a special case: if a list heuristic is selected, there is a high probability ϕ that the destruction size will be small (between δ_{mini} and Δ_{mini} percent of the customer) and that the ruin operator will be local, i.e *string removal*, *split string removal* or *distance-related customer removal*. The probability of each operator, δ, Δ, δ_{mini}, Δ_{mini} and ϕ was tuned according to the recommendations of IRACE [López-Ibánez et al., 2011].

Christiaens and Vanden Berghe [2020] introduced the *blink* principle for recreate operators. It randomly ignores certain insertions with a given probability during the computation of the best insertion. In the proposed implementation, this feature did not prove to have a significant impact. It introduces diversification through randomization. Nevertheless, this principle has been applied to the ruin operators; for each removal evaluated there is a given probability of simply ignoring it.

To summarize, on the one hand, the list heuristics and the small destructions favor a high number of iterations. On the other hand, using regret heuristics and ejection search tends to reduce the number of iterations. As observed by Christiaens and Vanden Berghe [2020], the small destruction and list heuristic can compensate for a lack of local search. Furthermore, the numerous iterations coupled with blink provide a good exploration of the search space in the CVRP. To deal with time windows, we observe that it is worthwhile to perform larger destruction and to take some time to anticipate constraint violation, like Ropke and Pisinger [2006a] and Demir et al. [2012]. Hence the final operator configuration in the proposed LNS-SPP combines fast and slow iterations in addition to long intensification phases with the SPP component.

To conclude this section, we indicate the values of the parameters (tuned with the help of IRACE) :

- The probability of selecting each ruin operator is the same for all operators. The probability to blink a deletion possibility is 30%.

- The probability of selecting a recreate operator is inversely proportional to its running time. The probability of selecting list heuristics (*random order best insertion*, *largest first best insertion*) is 0.4. The probability of selecting the other recreate operator (*2-regret*, *ejection search*, *SDL-regret* and *preferred best insertion*) is 0.05.

- The destruction size and removal operator selection rule is different for the list heuristics and the other recreate operators. In general the destruction size is between $\delta = 10\%$ and $\Delta = 20\%$ of the number of customers. For the list heuristic there is a 30% probability of performing a classical destruction (using any ruin operator) and a $\phi = 70\%$ probability of performing a small, local destruction. That is to say, only between $\delta_{\text{mini}} = 1\%$ and $\Delta_{\text{mini}} = 10\%$ of customers are removed, and a local removal operator (*distance-related removal*, *(split) string removal*) is used.

- The initial values of the record-to-record acceptance criteria are $T = 0.18$ and $\beta = 9$. These values are adjusted every 4500 iterations by a factor of 1.5, as described in section 4.5.

- The initial call frequency to the SPP component is $\eta = 20\ 000$. It is adjusted by a factor of 1.25. Furthermore, the time budget for the solver is 30 seconds and $\rho = 1\%$.

5 Experiments

The method is coded in C++ and is compiled with g++ 5.4.0. We use IBM Ilog CPLEX 12.8.0 [IBM, 2018] as the MIP solver. The experiments were performed using Linux, Ubuntu 16.04 LTS, running on an Intel Xeon X5650 @ 2.57 GHz. A single core is used by our code and the third-party solvers. We use the following options of CPLEX: branch up first and emphasis on hidden feasible solutions.

Section 5.1 validates the proposed matheuristic on related problems. Section 5.2 presents the generated VRPDO instances and Section 5.3 presents managerial insights.

5.1 Validation of LNS-SPP

We evaluate LNS-SPP on 120 benchmark instances of the VRPRDL proposed by Reyes et al. [2017] and Ozbaygin et al. [2017]. Two versions of the set of instances are used. In the first version, denoted VRPRDL, the time windows associated with the options of each customer are disjointed, and only trunk deliveries are considered. In the second version, denoted VRPHRDL, the first option of each customer is considered to be the home option and it has no time window. The others remain unchanged. Instances 1 to 40 come from Reyes et al. [2017], but were modified by Ozbaygin et al. [2017] to satisfy triangle inequality. Instances 41 to 50, v1 and v2, come from Ozbaygin et al. [2017].

The solutions produced by LNS-SPP are compared with those produced by the B&P of Ozbaygin et al. [2017]. Their method does not minimize the number of vehicles. Tables 3 and 4 provide a summary of the results. Each line represents an instance group, whose features are described by the first two columns. The last line is the total over all the instances. Columns 3 and 4 depict the total number of routes and cost for all the instances of the group. A "*" indicates that the B&P could not prove the optimality of all the solutions in the set. The results of LNS-SPP are summarized in the remaining columns, and are based on five runs on each instance. In columns 5 and 6, the cost is optimized with the number of routes that used in the solution in Ozbaygin et al. [2017]. Notice that it is possible that the solution found by

LNS-SPP with this number of vehicles comprises empty routes. Column 5 is the sum, over the instances of the group, of the average cost over 5 runs on each instance with LNS-SPP. Column 6 is the sum, over the instances of the group, of the lowest cost over 5 runs on each instance with LNS-SPP. Columns 7 to 10 provide the results of LNS-SPP when the number of routes is minimized. Column 7, 8 and 9, 10 are the total number of vehicles and cost, for the instances of the group on average on each instance or the best solution obtained on each instance, respectively. Finally, the computational time for the instances of each class is provided. The results are detailed in Tables 9, 10, 11 and 12 in A.

When the number of routes is set to the same value: the cost of the best solution of LNS-SPP is never higher than the cost of the solution provided by Ozbaygin et al. [2017]. The B&P proves the optimality of the cost on 93 instances out of 120. Based on the best results out of five runs (column 6), the cost is the same on 97 instances and it is improved on 23 instances by LNS-SPP.

When the number of vehicles is minimized first (column 9 and 10): the number of vehicles is reduced, which results in a higher cost, on 12 instances. On these instances, the total number of vehicles is decreased from 128 to 110 (14%) and the routing costs increase from 33 543 to 34 250 (+2.1%), knowing that 10 of these 12 instances have a cost proven optimal. In addition, the number of vehicles is reduced on 3 instances with the same cost, and both the number of vehicles and the cost were improved on 14 instances.

Unfortunately, no fair comparison can be made with the metaheuristic of Reyes et al. [2017] because the instances were modified by Ozbaygin et al. [2017]. Furthermore, running times are not indicated in Reyes et al. [2017]. Their method does not succeed in finding optimal solutions for small instances with 15 and 20 customers (with respect to optimal solutions provided by Gurobi 5.6).

The results on the VRPRDL and VRPHRDL show that the proposed LNS-SPP is clearly able to deal with delivery options, even with quite a short time budget.

		B&P		LNS-SPP						
				Fixed nbRoutes		Average of 5		Best of 5		
Instances	Customers	Routes	Cost	Avg. cost	Best of 5 cost	Routes	Cost	Routes	Cost	Time (s)
1-5	15	24	6 072	6 072	6 072	22	6 119	22	6 119	3
6-10	20	28	6 848	6 848	6 848	27	6 848	27	6 848	4
11-20	30	68	18 595	18 595	18 595	67	18 651.4	67	18 639	8
21-30	60	129	37 213	37 213	37 213	127	37 535	127	37 535	12
31-40	120	195	53 881 *	53 738.4	53 738	178.8	53 826	178	53 918	60
41-50_v1	40	94	29 842 *	29 838	29 838	93	29 855.4	93	29 838	10
41-50_v2	40	75	21 863	21 864.4	21 863	71	21 928.4	71	21 927	10
Total		613	174 314	174 168.8	174 167	585.8	174 763.2	585	174 824	

Table 3: Summary of the results on the VRPRDL instances of Reyes et al. [2017] and Ozbaygin et al. [2017]

Total number of routes and total cost over the instances of each group, and over all the instances, with the two considered algorithms for the VRPRDL instances

		B&P		LNS-SPP						
				Fixed nbRoutes		Average of 5		Best of 5		
Instances	Customers	Routes	Cost	Avg. cost	Best of 5 cost	Routes	Cost	Routes	Cost	Time (s)
1-5	15	19	5 450	5 450	5 450	19	5 450	19	5 450	3
6-10	20	20	5 604	5 604	5 604	20	5 604	20	5 604	4
11-20	30	52	15 128 *	15 128	15 128	51	15 228	51	15 212	8
21-30	60	83	26 829 *	26 800	26 800	83	26 800	83	26 800	12
31-40	120	132	38 610 *	37 310.4	37 252	115.6	37 423	115	37 373	60
41-50_v1	40	88	27 997 *	27 996	27 996	87	27 996	87	27 996	10
41-50_v2	40	67	20 977 *	20 958	20 958	67.6	20 958	67	20 958	10
Total		461	142 595	139 259.2	139 188	443.2	139 459	442	139 393	

Table 4: Summary of the results on the VRPHRDL instances of Reyes et al. [2017] and Ozbaygin et al. [2017]

Total number of routes and total cost over the instances of each group, and over all the instances, with the two considered algorithms for the VRPHRDL instances

5.2 Instances of the VRPDO

VRPDO instances were randomly generated, because we did not find any suitable existing instances. Three types of instances were generated: U, V and UBC. For each type, instances with 50, 100 and 200 customers were generated.

All the delivery locations were randomly generated in a 50×50 square. The depot is located at the bottom left-hand corner. Euclidean distances are considered. A unit of distance costs 1 and takes a unit of time to be crossed. We consider a time horizon of 12 hours, i.e. 720 time units.

In the U and UBC instances, each customer has between 1 and 3 options, with an average of 2 options per customer. In the V instance, each customer has 1 or 2 options, with an average of 1.5 options per customer.

In the U and V instances, the capacities are tight, both for the vehicles and the lockers. A locker can accept between 3 and 5 parcels, and vehicle capacity is such that a route can serve around 10 customers. In the UBC instances (U with Big Capacity), the vehicle capacity is such that a route can serve around 25 customers. Furthermore, there are five times fewer lockers and their capacity is five times larger.

The time window of individual locations can be either: the morning ($[0; 360]$), the afternoon ($[360; 720]$), random in the morning (i.e $[a_i, b_i]$ such that $0 \leqslant a_i \leqslant 240$ and $b_i = a_i + 120$), random in the afternoon (i.e $[a_i, b_i]$ such that $360 \leqslant a_i \leqslant 600$ and $b_i = a_i + 120$) or random in the whole day (i.e $[a_i, b_i]$ such that $0 \leqslant a_i \leqslant 480$ and $b_i = 240$). The time window of a shared location can be either: random in the day (i.e $[a_i, b_i]$ such that $0 \leqslant a_i \leqslant 240$ and $b_i = 480$) or the full day ($[0; 720]$).

The characteristics of instance classes are summarized in Table 5. For each size and each class, 10 instances were generated, leading to a total of 90 instances. All the instances are available upon request.

Table 6 summarizes the results obtained by LNS-SPP on these instances. This table depicts the total number of vehicles and the total cost for all instances of each class. These results are detailed in Tables 13, 14 and 15 in B. By default, all these tests were conducted with a service level of $80\% - 90\%$, i.e at least 80% of the customers are served with their level 1 option and at least 90% of the customers are served with an option of level 1 or 2. The time budget of the algorithm only depends on the instance size: 30 seconds for 50 customers, 90 seconds for 100,

Instance type	Capacity	Avg. option per customer	Time windows	
			Individual locations	Shared Locations
U	medium	2	2 to 6 hours	8 to 12 hours
UBC	big	2	2 to 6 hours	8 to 12 hours
V	medium	1.5	2 to 6 hours	8 to 12 hours

Table 5: VRPDO instance classes

and 300 seconds for 200 customers.

		Average of 5		Best of 5		
Type	Customers	Routes	Cost	Routes	Cost	Time (s)
U	50	54	3 913.88	54	3 864.48	30
	100	105	6 577.74	105	6 502.99	90
	200	205	14 082.36	205	13 641.13	300
UBC	50	20	2 301.31	20	2 293.53	30
	100	40	3 666.90	40	3 608.39	90
	200	80	6 534.72	80	6 384.29	300
V	50	54	3 759.76	54	3 742.59	30
	100	104.6	7 172.94	104	7 089.06	90
	200	205	15 261.14	205	14 781.23	300
Total		867.6	63 270.75	867	61 907.68	

Table 6: Summary of the results obtained by LNS-SPP on the VRPDO instances

5.3 Managerial insights for the VRPDO

To quantify the impact of the delivery options, we compare the solution of the VRPDO instances with their VRPTW counterpart. To transform a VRPDO instance into a VRPTW instance, we consider only the home option. No preference level is taken into account. In the VRPDO instances, we assume that home delivery options are the preferred individual locations. In the case of customers that only have a locker option, a random location is added as a home location.

The total number of vehicles and the total cost for each instance class are depicted in Table 7. The instances are grouped by line, the first line indicating the type of instance and the second column the number of customers. Each group is composed of 10 instances and the "Total" line is the sum over all the instances. Columns 3 and 4 indicate the total number of vehicles and the total cost over the instances of the group when only home delivery is considered. Columns 5 and 6 indicate the total number of vehicle and the total cost over the instances of the group when all options are considered. Column 7 is the relative savings on the route length obtained by using delivery options. In our instances, considering delivery options leads to a cost reduction of 29.2%, on average. Furthermore, on the UBC instance, with large lockers, the savings are even larger. The number of routes is not reduced, because it is determined by the binding vehicles' capacities.

Type	nbCustomers	1 option (home)		With options		Gap (%)
U	50	54	5 504.07	54	3 864.48	29.8
	100	105	8 662.22	105	6 502.99	24.9
	200	205	16 223.30	205	13 641.13	15.9
UBC	50	20	5 253.71	20	2 293.53	56.3
	100	40	7 052.06	40	3 608.39	48.8
	200	80	10 681.93	80	6 384.29	40.2
V	50	54	5 192.10	54	3 742.59	27.9
	100	104	9 877.77	104	7 089.06	28.2
	200	205	18 945.38	205	14 781.23	22.0
Total		867	87 392.53	867	61 907.68	29.2

Table 7: Economic impact of delivery options

Total number of routes and total cost of the solutions of each VRPDO instance group, and over all the VRPDO instances, with and without delivery options

We performed a sensitivity analysis by modifying the width of individual locations' time windows. Considering a time window $[a_i, b_i]$ at location $i \in L$, the modified time window is still centered at time $0.5 \times (a_i + b_i)$ but its width is reduced by a factor α as shown in formula (2).

$$[a'_i, b'_i] = \left[\frac{a_i + b_i}{2} - \frac{b_i - a_i}{2\alpha} \; ; \; \frac{a_i + b_i}{2} + \frac{b_i - a_i}{2\alpha} \right]. \tag{2}$$

We conduct the same experiments as in Table 7 with these smaller time windows for individual locations. Figure 11 summarizes these results. The graphs present the total cost and the total number of vehicles based on the time window width. Along the the x-axis, we show the time window width in time units while the y-axis shows the cost and the number of vehicles, respectively. The time windows are reduced by a factor α between 1 and 10 according to Equation 2. Thus the average time window width range from 180 time units down to 20 time units. Because some instances become infeasible when the time windows are too tight, not all instances are taken into account in this figure.

With the VRPTW counterpart, both the number of vehicles and the cost grow by 33% when the individual locations' time window width is divided by 10. For the VRPDO, the number of vehicles only increases from 598 to 604 and the cost grows by only 10%. That is to say, with time windows of about 20 minutes, the cost savings of the VRPDO is 44.3%, while the number of vehicles decreases by 21.8%, on average over all the considered instances.

On these instances, considering delivery options and shared delivery locations makes it possible to to serve customers with very narrow time windows without significant cost increase, as observed in the VRPTW.

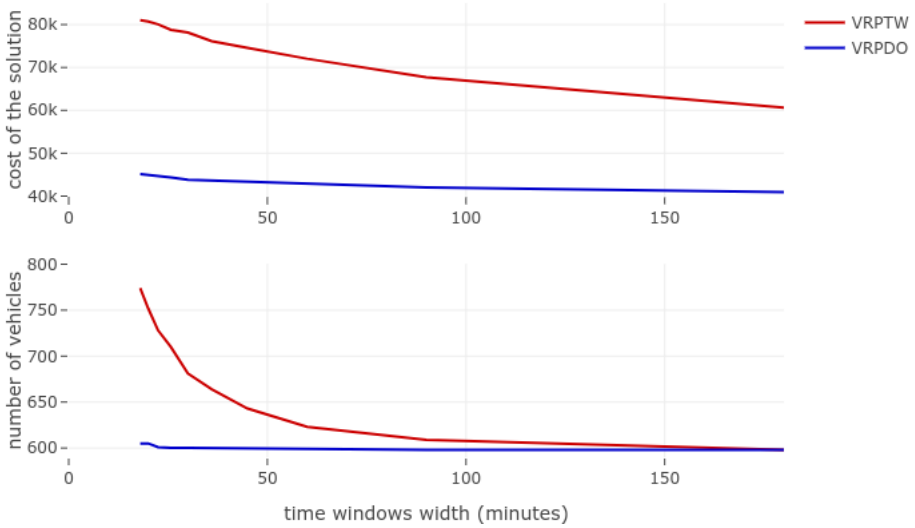

Figure 11: Impact of time window width with and without delivery options

Total number of vehicles and total cost with the VRPDO and the VRPTW with respect to average individual locations time window width

In order to quantify the impact of the required service level, we perform some experiments with different values. Table 8 presents the total number of vehicles and the total cost, for all the VRPDO instances, with respect to the required service level. Each line sets a different minimal percentage of options of level 1 and 2. Each column sets a different minimal percentage of customers served via their preferred option.

This table shows that the total cost does not significantly increase when the required service level is more strict. When the service level becomes too strict (e.g with $\beta_1 = 90$ and $\beta_2 = 100$), some instances become infeasible.

		min percentage of options of level 1 (β_1)					
		70		80		90	
min percentage of options of level 1 or 2 (β_2)	80	866	60 215.29	866	61 475.93	NA	NA
	90	866	60 393.85	867	61 907.68	867	66 155.67
	95	866	60 710.38	867	61 980.41	867	66 574.68
	100	866	61 892.21	867	63 172.80	NA	NA

Table 8: Impact of the required service level

Total number of routes and total cost, over all VRPDO instances, based on the required service level

The proposed experiments show that considering delivery options and shared delivery locations can reduce cost. In addition, such options make it possible to guarantee a very high quality of service, with respect to both the time window width and service level.

6 Conclusion

In this paper we have proposed a new extension of the VRPTW allowing multiple delivery options for each customer. The Vehicle Routing Problem with Delivery Options (VRPDO)

includes the use of shared delivery locations, such as lockers, and takes in account customer preferences. In addition, the model is general enough to include new modes of delivery, such as trunk delivery. The VRPDO is theoretically challenging because it introduces synchronized resources and a new structure of the search space, due to the delivery locations for each customer.

The numerical experiments show that VRPDO can save considerable amounts of money compared with VRPTW. Moreover, for a small cost increase, a very high quality of service can be achieved, especially with respect to the time window width.

The VRPDO is solved with an LNS meta-heuristic combined with a set partitioning component. After implementing a large number of ruin and recreate operators, we led a comprehensive tuning process that resulted in the selection of a few relevant operators. The experiments show that combining local impact fast operators and global impact slower operators is an efficient strategy. Alternating between large and small removal operators help perform intensification as well as diversification.

For future research, we plan to use dynamic programming, as in Moccia et al. [2012] and Reyes et al. [2017], to improve promising solutions. The difficulty in efficiently applying these methods to the VRPDO is the combinatorial explosion of the number of labels induced by the synchronized resources. Another perspective is to integrate delivery options into multi-echelon [Grangier et al., 2017b] or multi-modal [Masson et al., 2017] city logistics systems, or to combine it with the use of autonomous vehicles [Boysen et al., 2018].

Acknowledgements

This work has been supported by ANR-DFG under the OPUSS (Optimization of Urban Synchromodal Systems - OPUSS; ANR-17-CE22-0015) project.

We authors thank Stefan Irnich, Katharina Olkis and Christian Tilk from Johannes Gutenberg University Mainz for the generation of the VRPDO instances.

References

H Murat Afsar, Christian Prins, and Andréa Cynthia Santos. Exact and heuristic algorithms for solving the generalized vehicle routing problem with flexible fleet size. *International Transactions in Operational Research*, 21(1):153–175, 2014. doi:https://doi.org/10.1111/itor.12041.

Niels Agatz, Ann Campbell, Moritz Fleischmann, and Martin Savelsbergh. Time slot management in attended home delivery. *Erasmus Research Institute of Management*, 45, 01 2008. doi:10.2307/23018537.

Somayeh Allahyari, Majid Salari, and Daniele Vigo. A hybrid metaheuristic algorithm for the multi-depot covering tour vehicle routing problem. *European Journal of Operational Research*, 242(3):756–768, 2015. doi:https://doi.org/10.1016/j.ejor.2014.10.048.

Julian Allen, Maja Piecyk, and Marzena Piotrowska. Analysis of the parcels market and parcel carriers' operations in the UK. Technical report, University of Westminster, 2016.

Claudia Archetti, M Grazia Speranza, and Daniele Vigo. Chapter 10: Vehicle routing problems with profits. In *Vehicle Routing: Problems, Methods, and Applications, Second Edition*, pages 273–297. SIAM, 2014. doi:https://doi.org/10.1137/1.9781611973594.ch10.

Slim Belhaiza, Pierre Hansen, and Gilbert Laporte. A hybrid variable neighborhood tabu search heuristic for the vehicle routing problem with multiple time windows. *Computers & Operations Research*, 52:269–281, 2014. doi:https://doi.org/10.1016/j.cor.2013.08.010.

Raquel Bernardino and Ana Paias. Metaheuristics based on decision hierarchies for the traveling purchaser problem. *International Transactions in Operational Research*, 25(4):1269–1295, 2018. doi:https://doi.org/10.1111/itor.12330.

Nils Boysen, Stefan Schwerdfeger, and Felix Weidinger. Scheduling last-mile deliveries with truck-based autonomous robots. *European Journal of Operational Research*, 271(3):1085 – 1099, 2018. doi:https://doi.org/10.1016/j.ejor.2018.05.058.

Olli Bräysy and Michel Gendreau. Vehicle routing problem with time windows, part I: Route construction and local search algorithms. *Transportation Science*, 39(1):104–118, 2005a. doi:https://doi.org/10.1287/trsc.1030.0056.

Olli Bräysy and Michel Gendreau. Vehicle routing problem with time windows, part II: Metaheuristics. *Transportation Science*, 39(1):119–139, 2005b. doi:https://doi.org/10.1287/trsc.1030.0057.

Jan Christiaens and Greet Vanden Berghe. Slack induction by string removals for vehicle routing problems. *Transportation Science*, 2020.

Jean-François Cordeau, Guy Desaulniers, Jacques Desrosiers, Marius M. Solomon, and François Soumis. 7. vrp with time windows. In *The Vehicle Routing Problem*, chapter 7, pages 157–193. SIAM, 2002.

Timothy Curtois, Dario Landa-Silva, Yi Qu, and Wasakorn Laesanklang. Large neighbourhood search with adaptive guided ejection search for the pickup and delivery problem with time windows. *EURO Journal on Transportation and Logistics*, 7(2):151–192, 2018. doi:https://doi.org/10.1007/s13676-017-0115-6.

Emrah Demir, Tolga Bektaş, and Gilbert Laporte. An adaptive large neighborhood search heuristic for the pollution-routing problem. *European Journal of Operational Research*, 223 (2):346–359, 2012. doi:https://doi.org/10.1016/j.ejor.2012.06.044.

Michael Drexl. Synchronization in vehicle routing—a survey of VRPs with multiple synchronization constraints. *Transportation Science*, 46(3):297–316, 2012. doi:https://doi.org/10.1287/trsc.1110.0400.

Gunter Dueck. New optimization heuristics: The great deluge algorithm and the record-to-record travel. *Journal of Computational physics*, 104(1):86–92, 1993. doi:https://doi.org/10.1006/jcph.1993.1010.

Daniela Favaretto, Elena Moretti, and Paola Pellegrini. Ant colony system for a VRP with multiple time windows and multiple visits. *Journal of Interdisciplinary Mathematics*, 10(2): 263–284, 2007. doi:https://doi.org/10.1080/09720502.2007.10700491.

Alexandre M. Florio, Dominique Feillet, and Richard F. Hartl. The delivery problem: Optimizing hit rates in e-commerce deliveries. *Transportation Research Part B: Methodological*, 117: 455 – 472, 2018. doi:https://doi.org/10.1016/j.trb.2018.09.011.

Brian A Foster and David M Ryan. An integer programming approach to the vehicle scheduling problem. *Journal of the Operational Research Society*, 27(2):367–384, 1976. doi:https://doi.org/10.1057/jors.1976.63.

Aurélien Froger, Jorge E Mendoza, Ola Jabali, and Gilbert Laporte. A matheuristic for the electric vehicle routing problem with capacitated charging stations. Technical Report CIRRELT-2017-31, CIRRELT, 2017.

Gianpaolo Ghiani and Gennaro Improta. An efficient transformation of the generalized vehicle routing problem. *European Journal of Operational Research*, 122(1):11–17, 2000. doi:https://doi.org/10.1016/S0377-2217(99)00073-9.

Philippe Grangier, Michel Gendreau, Fabien Lehuédé, and Louis-Martin Rousseau. A matheuristic based on large neighborhood search for the vehicle routing problem with cross-docking. *Computers & Operations Research*, 84:116–126, 2017a. doi:https://doi.org/10.1016/j.cor.2017.03.004.

Philippe Grangier, Michel Gendreau, Fabien Lehuédé, and Louis-Martin Rousseau. The vehicle routing problem with cross-docking and resource constraints. *Journal of Heuristics*, pages 1–31, 2017b. doi:http://dx.doi.org/10.1007/s10732-019-09423-y.

Chris Groër, Bruce Golden, and Edward Wasil. A parallel algorithm for the vehicle routing problem. *INFORMS Journal on Computing*, 23(2):315–330, 2011. doi:https://doi.org/10.1023/A:1018948011707.

Timo Gschwind and Michael Drexl. Adaptive large neighborhood search with a constant-time feasibility test for the dial-a-ride problem. *Transportation Science*, 53(2):480–491, 2019. doi:https://doi.org/10.1287/trsc.2018.0837.

Christoph Hempsch and Stefan Irnich. Vehicle routing problems with inter-tour resource constraints. In *The Vehicle Routing Problem: Latest Advances and New Challenges*, pages 421–444. Springer, 2008. doi:https://doi.org/10.1007/978-0-387-77778-8_19.

Maaike Hoogeboom and Wout Dullaert. Vehicle routing with arrival time diversification. *European Journal of Operational Research*, 275:93–107, May 2019. doi:https://doi.org/10.1016/j.ejor.2018.11.020.

IBM. Cplex, 2018. URL https://www.ibm.com/analytics/data-science/prescriptive-analytics/cplex-optimizer.

Manel Kammoun, Houda Derbel, Mostapha Ratli, and Bassem Jarboui. An integration of mixed VND and VNS: the case of the multivehicle covering tour problem. *International Transactions in Operational Research*, 24(3):663–679, 2017. doi:https://doi.org/10.1111/itor.12355.

Scott Kirkpatrick, C Daniel Gelatt, and Mario P Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983. doi:10.1126/science.220.4598.671.

Manuel López-Ibánez, Jérémie Dubois-Lacoste, Thomas Stützle, and Mauro Birattari. The IRACE package, iterated race for automatic algorithm configuration. Technical Report TR/IRIDIA/2011-004, Université Libre de Bruxelles, 2011.

Simona Mancini. A combined multistart random constructive heuristic and set partitioning based formulation for the vehicle routing problem with time dependent travel times. *Computers & Operations Research*, 88:290–296, 2017. doi:https://doi.org/10.1016/j.cor.2017.06.021.

Renaud Masson, Anna Trentini, Fabien Lehuédé, Nicolas Malhéné, Olivier Péton, and Houda Tlahig. Optimization of a city logistics transportation system with mixed passengers and goods. *EURO Journal on Transportation and Logistics*, 6(1):81–109, 2017.

Matt McFarland. Amazon now delivers to the trunk of your car. *CNN business*, 2018. URL https://money.cnn.com/2018/04/24/technology/amazon-key-in-car-delivery-review/index.html.

Jorge E Mendoza and Juan G Villegas. A multi-space sampling heuristic for the vehicle routing problem with stochastic demands. *Optimization Letters*, 7(7):1503–1516, 2013. doi:https://doi.org/10.1007/s11590-012-0555-8.

Luigi Moccia, Jean-François Cordeau, and Gilbert Laporte. An incremental tabu search heuristic for the generalized vehicle routing problem with time windows. *Journal of the Operational Research Society*, 63(2):232–244, 2012. doi:https://doi.org/10.1057/jors.2011.25.

Eleonora Morganti, Saskia Seidel, Corinne Blanquart, Laetitia Dablanc, and Barbara Lenz. The impact of e-commerce on final deliveries: alternative parcel delivery services in France and Germany. *Transportation Research Procedia*, 4:178–190, 2014. doi:https://doi.org/10.1016/j.trpro.2014.11.014.

E. Moyou. Chiffre d'affaires annuel du e-commerce en france de 2005 à 2018, 2019. URL `https://fr.statista.com/statistiques/474685/chiffre-d-affaires-e-commerce-france/`.

Yuichi Nagata and Olli Bräysy. A powerful route minimization heuristic for the vehicle routing problem with time windows. *Operations Research Letters*, 37(5):333–338, 2009. doi:https://doi.org/10.1016/j.orl.2009.04.006.

Gizem Ozbaygin, Oya Ekin Karasan, Martin Savelsbergh, and Hande Yaman. A branch-and-price algorithm for the vehicle routing problem with roaming delivery locations. *Transportation Research Part B: Methodological*, 100:115–137, 2017. doi:https://doi.org/10.1016/j.trb.2017.02.003.

Sophie N Parragh and Verena Schmid. Hybrid column generation and large neighborhood search for the dial-a-ride problem. *Computers & Operations Research*, 40(1):490–497, 2013. doi:https://doi.org/10.1016/j.cor.2012.08.004.

David Pisinger and Stefan Ropke. A general heuristic for vehicle routing problems. *Computers and Operations Research*, 34(8):2403–2435, 2007. doi:https://doi.org/10.1016/j.cor.2005.09.012.

David Pisinger and Stefan Ropke. Large neighborhood search. In Michel Gendreau and Jean-Yves Potvin, editors, *Handbook of Metaheuristics*, pages 99–127. Springer International Publishing, 2019.

Eric Prescott-Gagnon, Guy Desaulniers, and Louis-Martin Rousseau. A branch-and-price-based large neighborhood search algorithm for the vehicle routing problem with time windows. *Networks*, 54(4):190–204, 2009. doi:https://doi.org/10.1002/net.20332.

Damián Reyes, Martin Savelsbergh, and Alejandro Toriello. Vehicle routing with roaming delivery locations. *Transportation Research Part C: Emerging Technologies*, 80:71–91, 2017. doi:https://doi.org/10.1016/j.trc.2017.04.003.

Yves Rochat and Éric D Taillard. Probabilistic diversification and intensification in local search for vehicle routing. *Journal of Heuristics*, 1(1):147–167, 1995. doi:https://doi.org/10.1007/BF02430370.

Stefan Ropke and David Pisinger. A unified heuristic for a large class of vehicle routing problems with backhauls. *European Journal of Operational Research*, 171(3):750–775, 2006a. doi:https://doi.org/10.1016/j.ejor.2004.09.004.

Stefan Ropke and David Pisinger. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science*, 40(4):455–472, 2006b. doi:https://doi.org/10.1016/j.cor.2016.01.018.

Alberto Santini, Stefan Ropke, and Lars Magnus Hvattum. A comparison of acceptance criteria for the adaptive large neighbourhood search metaheuristic. *Journal of Heuristics*, 24(5): 783–815, 2018. doi:https://doi.org/10.1007/s10732-018-9377-x.

Martin WP Savelsbergh. Local search in routing problems with time windows. *Annals of Operations Research*, 4(1):285–305, 1985. doi:https://doi.org/10.1007/BF02022044.

Martin WP Savelsbergh. The vehicle routing problem with time windows: Minimizing route duration. *ORSA Journal on Computing*, 4(2):146–154, 1992. doi:https://doi.org/10.1287/ijoc.4.2.146.

Paul Shaw. Using constraint programming and local search methods to solve vehicle routing problems. In *International Conference on Principles and Practice of Constraint Programming*, pages 417–431. Springer, 1998. doi:https://doi.org/10.1007/3-540-49481-2_30.

Pawel Sitek and Jarosław Wikarek. Capacitated vehicle routing problem with pick-up and alternative delivery (CVRPPAD): model and implementation using hybrid approach. *Annals of Operations Research*, pages 1–21, 2017. doi:https://doi.org/10.1007/s10479-017-2722-x.

Wouter Souffriau, Pieter Vansteenwegen, Greet Vanden Berghe, and Dirk Van Oudheusden. The multiconstraint team orienteering problem with multiple time windows. *Transportation Science*, 47(1):53–63, 2013. doi:https://doi.org/10.1287/trsc.1110.0377.

Thomas Stützle. Heuristic optimization, 2018. URL `http://iridia.ulb.ac.be/~stuetzle/Teaching/HO/`.

Anand Subramanian, Eduardo Uchoa, and Luiz Satoru Ochi. A hybrid algorithm for a class of vehicle routing problems. *Computers & Operations Research*, 40(10):2519–2531, 2013. doi:https://doi.org/10.1016/j.cor.2013.01.013.

Y. Takada, Y. Hu, H. Hashimoto, and M. Yagiura. An iterated local search algorithm for the multi-vehicle covering tour problem. In *IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)*, pages 1242–1246, Dec 2015. doi:https://doi.org/10.1109/IEEM.2015.7385846.

Oscar Tellez, Samuel Vercraene, Fabien Lehuédé, Olivier Péton, and Thibaud Monteiro. The fleet size and mix dial-a-ride problem with reconfigurable vehicle capacity. *Transportation Research Part C: Emerging Technologies*, 91:99–123, 2018. doi:https://doi.org/10.1016/j.trc.2018.03.020.

Paolo Toth and Daniele Vigo. *Vehicle routing: problems, methods, and applications*. SIAM, 2014. doi:https://doi.org/10.1137/1.9781611973594.

Fabien Tricoire, Martin Romauch, Karl Doerner, and Richard Hartl. Heuristics for the multi-period orienteering problem with multiple time windows. *Computers and Operations Research*, 37:351–367, 2010. doi:https://doi.org/10.1016/j.cor.2009.05.012.

Renata Turkeš, Kenneth Sörensen, Lars Magnus Hvattum, Eva Barrena, Hayet Chentli, Leandro Coelho, Iman Dayarian, Alex Grimault, Anders Gullhav, Çağatay Iris, Merve Keskin, Alexander Kiefer, Richard Lusby, Geraldo Mauri, Marcela Monroy-Licht, Sophie Parragh, Juan-Pablo Riquelme-Rodríguez, Alberto Santini, Gandra Martins Santos Vinicius, and Charles

Thomas. Meta-analysis of metaheuristics: Quantifying the effect of adaptiveness in adaptive large neighborhood search. Working paper d/2019/1169/002, University of Antwerp, 2019.

Pieter Vansteenwegen, Wouter Souffriau, and Dirk Van Oudheusden. The orienteering problem: A survey. *European Journal of Operational Research*, 209(1):1–10, 2011. doi:https://doi.org/10.1016/j.ejor.2016.04.059.

Leticia Vargas, Nicolas Jozefowiez, and Sandra Ulrich Ngueveu. A selector operator-based adaptive large neighborhood search for the covering tour problem. In *International Conference on Learning and Intelligent Optimization*, pages 170–185. Springer, 2015. doi:https://doi.org/10.1007/978-3-319-19084-6_16.

Thibaut Vidal, Teodor Gabriel Crainic, Michel Gendreau, and Christian Prins. A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time windows. *Computers & Operations Research*, 40(1):475–489, 2013. doi:https://doi.org/10.1016/j.cor.2012.07.018.

Frank Wilcoxon. Individual comparisons by ranking methods. *Biometrics bulletin*, 1(6):80–83, 1945. doi:10.2307/3001968.

Umman Mahir Yıldırım and Bülent Çatay. An ant colony-based matheuristic approach for solving a class of vehicle routing problems. In *International Conference on Computational Logistics (ILCL)*, pages 105–119. Springer, 2015.

Yuan Yuan, Diego Cattaruzza, Maxime Ogier, and Frédéric Semet. A branch-and-cut algorithm for the generalized traveling salesman problem with time windows. In *ROADEF 2018*, Lorient, France, February 2018.

Yuan Yuan, Diego Cattaruzza, Maxime Ogier, Frédéric Semet, and Daniele Vigo. The generalized vehicle routing problem with time windows. In *VeRoLog 2019*, Seville, Spain, June 2019.

Lin Zhou, Roberto Baldacci, Daniele Vigo, and Xu Wang. A multi-depot two-echelon vehicle routing problem with delivery options arising in the last mile distribution. *European Journal of Operational Research*, 265(2):765–778, 2018. doi:https://doi.org/10.1016/j.ejor.2017.08.011.

A Detailed results on the VRPRDL and VRPHRDL instances

		B&P			LNS-SPP						
					Fixed nbRoutes		Average of 5		Best of 5		
Instances	Customers	Routes	Cost		Avg. cost	Best of 5 cost	Routes	Cost	Routes	Cost	Time (s)
1	15	4	901		901	901	4	901	4	901	3
2	15	5	1 286		1 286	1 286	5	1 286	5	1 286	3
3	15	4	991		991	991	3	999	3	999	3
4	15	5	1 062		1 062	1 062	4	1 101	4	1 101	3
5	15	6	1 832		1 832	1 832	6	1 832	6	1 832	3
Group avg		4.8	1 214.4		1 214.4	1 214.4	4.4	1 223.8	4.4	1 223.8	
6	20	5	1 294		1 294	1 294	5	1 294	5	1 294	4
7	20	4	1 155		1 155	1 155	4	1 155	4	1 155	4
8	20	6	1 455		1 455	1 455	6	1 455	6	1 455	4
9	20	5	1 260		1 260	1 260	5	1 260	5	1 260	4
10	20	8	1 684		1 684	1 684	7	1 684	7	1 684	4
Group avg		5.6	1 369.6		1 369.6	1 369.6	5.4	1 369.6	5.4	1 369.6	
11	30	7	1 922		1 922	1 922	7	1 922	7	1 922	8
12	30	8	2 324		2 324	2 324	8	2 324	8	2 324	8
13	30	6	1 747		1 747	1 747	6	1 747	6	1 747	8
14	30	6	1 273		1 273	1 273	5	1 329.4	5	1 317	8
15	30	6	1 694		1 694	1 694	6	1 694	6	1 694	8
16	30	7	1 938		1 938	1 938	7	1 938	7	1 938	8
17	30	8	1 965		1 965	1 965	8	1 965	8	1 965	8
18	30	7	1 827		1 827	1 827	7	1 827	7	1 827	8
19	30	7	2 083		2 083	2 083	7	2 083	7	2 083	8
20	30	6	1 822		1 822	1 822	6	1 822	6	1 822	8
Group avg		6.8	1 859.5		1 859.5	1 859.5	6.7	1 865.1	6.7	1 863.9	
21	60	13	3 761		3 761	3 761	13	3 761	13	3 761	12
22	60	10	2 828		2 828	2 828	10	2 828	10	2 828	12
23	60	16	4 440		4 440	4 440	16	4 440	16	4 440	12
24	60	11	3 378		3 378	3 378	11	3 378	11	3 378	12
25	60	11	3 161		3 161	3 161	11	3 161	11	3 161	12
26	60	16	4 536		4 536	4 536	16	4 536	16	4 536	12
27	60	10	2 865		2 865	2 865	9	2 976	9	2 976	12
28	60	14	4 173		4 173	4 173	14	4 173	14	4 173	12
29	60	14	3 964		3 964	3 964	13	4 175	13	4 175	12
30	60	14	4 107		4 107	4 107	14	4 107	14	4 107	12
Group avg		12.9	3 721.3		3 721.3	3 721.3	12.7	3 753.5	12.7	3 753.5	
31	120	18	4 935		4 935	4 935	16	4 938	16	4 938	60
32	120	19	5 278	*	5 258	5 258	17	5 273.4	17	5 263	60
33	120	18	5 083	*	5 061	5 061	17	5 061	17	5 061	60
34	120	17	5 218		5 218	5 218	17	5 218	17	5 218	60
35	120	20	5 519	*	5 498	5 498	18	5 528.6	18	5 526	60
36	120	22	6 498		6 498	6 498	21	6 498	21	6 498	60
37	120	17	4 845	*	4 830	4 830	17	4 830	17	4 830	60
38	120	21	5 608	*	5 604	5 604	19	5 604	19	5 604	60
39	120	24	5 849	*	5 841	5 841	19.8	5 869.8	19	5 985	60
40	120	19	5 048	*	4 995.4	4 995	17	5 005.2	17	4 995	60
Group avg		19.5	5 388.1		5 373.8	5 373.8	17.9	5 382.6	17.8	5 391.8	
Overall avg		11.1	3 065.2		3 061.6	3 061.6	10.6	3 074.5	10.5	3 076.5	

Table 9: Comparison with the results of Ozbaygin et al. [2017] on the VRPRDL instances of Reyes et al. [2017]

		B&P			LNS-SPP						
					Fixed nbRoutes		Average of 5		Best of 5		
Instances	Customers	Routes	Cost		Avg. cost	Best of 5 cost	Routes	Cost	Routes	Cost	Time (s)
1	15	3	773		773	773	3	773	3	773	3
2	15	4	1 065		1 065	1 065	4	1 065	4	1 065	3
3	15	3	988		988	988	3	988	3	988	3
4	15	3	914		914	914	3	914	3	914	3
5	15	6	1 710		1 710	1 710	6	1 710	6	1 710	3
Group avg		3.8	1 090.0		1 090.0	1 090.0	3.8	1 090	3.8	1 090	
6	20	4	1 099		1 099	1 099	4	1 099	4	1 099	4
7	20	3	996		996	996	3	996	3	996	4
8	20	5	1 346		1 346	1 346	5	1 346	5	1 346	4
9	20	4	997		997	997	4	997	4	997	4
10	20	4	1 166		1 166	1 166	4	1 166	4	1 166	4
Group avg		4.0	1 120.8		1 120.8	1 120.8	4	1 120.8	4	1 120.8	
11	30	5	1 587		1 587	1 587	5	1 593.6	5	1 587	8
12	30	6	1 808		1 808	1 808	6	1 808	6	1 808	8
13	30	6	1 563		1 563	1 563	5	1 647	5	1 647	8
14	30	4	1 058		1 058	1 058	4	1 058	4	1 058	8
15	30	5	1 347		1 347	1 347	5	1 347	5	1 347	8
16	30	5	1 517	*	1 517	1 517	5	1 517	5	1 517	8
17	30	5	1 445		1 445	1 445	5	1 445	5	1 445	8
18	30	5	1 627		1 627	1 627	5	1 636.4	5	1 627	8
19	30	5	1 461		1 461	1 461	5	1 461	5	1 461	8
20	30	6	1 715		1 715	1 715	6	1 715	6	1 715	8
Group avg		5.2	1 512.8		1 512.8	1 512.8	5.1	1 522.8	5.1	1 521.2	
21	60	8	2 580		2 580	2 580	8	2 580	8	2 580	12
22	60	7	2 213	*	2 206	2 206	7	2 206	7	2 206	12
23	60	10	3 363		3 363	3 363	10	3 363	10	3 363	12
24	60	8	2 569	*	2 569	2 569	8	2 569	8	2 569	12
25	60	8	2 400	*	2 378	2 378	8	2 378	8	2 378	12
26	60	9	2 845	*	2 845	2 845	9	2 845	9	2 845	12
27	60	8	2 518		2 518	2 518	8	2 518	8	2 518	12
28	60	8	2 758		2 758	2 758	8	2 758	8	2 758	12
29	60	9	2 892	*	2 892	2 892	9	2 892	9	2 892	12
30	60	8	2 691		2 691	2 691	8	2 691	8	2 691	12
Group avg		8.3	2 682.9		2680.0	2 680.0	8.3	2 680.0	8.3	2 680.0	
31	120	14	3 984	*	3 666	3 666	11	3 666	11	3 666	60
32	120	13	3 958	*	3 885	3 885	12	3 886.4	12	3 885	60
33	120	13	3 630	*	3 543.6	3 543	11.4	3 567	11	3 544	60
34	120	13	3 891	*	3 711.8	3 694	12	3 784.8	12	3 783	60
35	120	11	3 255	*	3 184	3 184	10	3 184	10	3 184	60
36	120	15	4 525	*	4 311	4 273	13.2	4 326	13	4 304	60
37	120	11	3 395	*	3 217	3 217	10	3 217	10	3 217	60
38	120	14	3 976	*	3 935.8	3 935	12	3 935	12	3 935	60
39	120	15	4 316	*	4 300	4 300	13	4 300	13	4 300	60
40	120	13	3 680	*	3 556.2	3 555	11	3 556.8	11	3 555	60
Group avg		13.2	3 861.0		3 731.0	3 725.2	11.6	3 742.3	11.5	3 737.3	
Overall avg		7.6	2 290.5		2 257.3	2 255.8	7.2	2 262.6	7.2	2 261	

Table 10: Comparison with the results of Ozbaygin et al. [2017] on the VRPHRDL instances of Reyes et al. [2017]

	B&P			LNS-SPP						
				Fixed nbRoutes		Average of 5		Best of 5		
Instances	Routes	Cost		Avg. cost	Best of 5 cost	Routes	Cost	Routes	Cost	Time (s)
41_v1	10	3 203		3 203	3 203	10	3 219.4	10	3 203	10
41_v2	7	2 133		2 134.4	2 133	7	2 134.4	7	2 133	10
42_v1	9	2 799		2 799	2 799	9	2 799	9	2 799	10
42_v2	7	1 946		1 946	1 946	6	1 946	6	1 946	10
43_v1	8	2 607	*	2 603	2 603	8	2 604	8	2 603	10
43_v2	8	1 966		1 966	1 966	7	1 967	7	1 967	10
44_v1	7	2 261		2 261	2 261	7	2 261	7	2 261	10
44_v2	6	1 610		1 610	1 610	5	1 614	5	1 614	10
45_v1	10	3 217		3 217	3 217	10	3 217	10	3 217	10
45_v2	8	2 478		2 478	2 478	8	2 478	8	2 478	10
46_v1	9	2 805		2 805	2 805	9	2 805	9	2 805	10
46_v2	8	2 469		2 469	2 469	8	2 469	8	2 469	10
47_v1	10	3 339		3 339	3 339	10	3 339	10	3 339	10
47_v2	7	1 946		1 946	1 946	6	2 005	6	2 005	10
48_v1	10	3 325		3 325	3 325	10	3 325	10	3 325	10
48_v2	8	2 380		2 380	2 380	8	2 380	8	2 380	10
49_v1	11	3 534		3 534	3 534	11	3 534	11	3 534	10
49_v2	8	2 492		2 492	2 492	8	2 492	8	2 492	10
50_v1	10	2 752		2 752	2 752	9	2 752	9	2 752	10
50_v2	8	2 443		2 443	2 443	8	2 443	8	2 443	10
Avg	8.4	2 585.2		2 585,1	2 585.0	8.2	2 589.2	8.2	2 588.2	

Table 11: Comparison with the results of Ozbaygin et al. [2017] on the VRPRDL instances of Ozbaygin et al. [2017]

	B&P			LNS-SPP						
				Fixed nbRoutes		Average of 5		Best of 5		
Instances	Routes	Cost		Avg. cost	Best of 5 cost	Routes	Cost	Routes	Cost	Time (s)
41_v1	8	2 662		2 662	2 662	8	2 662	8	2 662	10
41_v2	6	1 998		1 998	1 998	6.6	1 998	6	1 998	10
42_v1	8	2 610		2 610	2 610	8	2 610	8	2 610	10
42_v2	6	1 946	*	1 927	1 927	6	1 927	6	1 927	10
43_v1	7	2 260		2 260	2 260	7	2 260	7	2 260	10
43_v2	6	1 830		1 830	1 830	6	1 830	6	1 830	10
44_v1	7	2 147		2 147	2 147	7	2 147	7	2 147	10
44_v2	5	1 478		1 478	1 478	5	1 478	5	1 478	10
45_v1	10	3 172		3 172	3 172	10	3 172	10	3 172	10
45_v2	8	2 466		2 466	2 466	8	2 466	8	2 466	10
46_v1	8	2 616		2 616	1 616	8	2 616	8	2 616	10
46_v2	8	2 388		2 388	2 388	8	2 388	8	2 388	10
47_v1	9	3 011	*	3 010	3 010	9	3 010	9	3 010	10
47_v2	6	1 848		1 848	1 848	6	1 848	6	1 848	10
48_v1	10	3 278		3 278	3 278	10	3 278	10	3 278	10
48_v2	7	2 264		2 264	2 264	7	2 264	7	2 264	10
49_v1	11	3 514	*	3 514	3 514	11	3 514	11	3 514	10
49_v2	8	2 457		2 457	2 457	8	2 457	8	2 457	10
50_v1	10	2 727		2 727	2 727	9	2 727	9	2 727	10
50_v2	7	2 302		2 302	2 302	7	2 302	7	2 302	10
Avg	7.7	2 448.7		2 447.7	2 447.7	7.7	2 447.7	7.7	2 447.7	

Table 12: Comparison with the results of Ozbaygin et al. [2017] on the VRPHRDL instances of Ozbaygin et al. [2017]

B Detailed results on the VRPDO instances

			Average of 5		Best of 5	
Instance	Customers	Time (s)	Routes	Cost	Routes	Cost
U_50_1	50	30	6	434.773	6	433.081
U_50_2	50	30	6	424.665	6	423.228
U_50_3	50	30	5	329.138	6	329.138
U_50_4	50	30	5	448.321	5	427.183
U_50_5	50	30	6	354.322	6	353.179
U_50_6	50	30	5	403.979	5	399.620
U_50_7	50	30	5	336.428	5	332.481
U_50_8	50	30	5	387.648	5	384.635
U_50_9	50	30	5	360.930	5	359.888
U_50_10	50	30	6	433.679	6	422.048
Total			54	3 913.883	54	3 864.481
U_100_1	100	90	11	484.226	11	479.972
U_100_2	100	90	10	687.754	10	683.365
U_100_3	100	90	11	634.604	11	630.215
U_100_4	100	90	10	610.063	10	592.860
U_100_5	100	90	10	678.523	10	673.844
U_100_6	100	90	10	597.610	10	586.810
U_100_7	100	90	11	761.344	11	746.400
U_100_8	100	90	11	856.017	11	847.533
U_100_9	100	90	10	690.985	10	687.659
U_100_10	100	90	11	576.614	11	574.328
Total			105	6 577.739	105	6 502.986
U_200_1	200	300	21	1 706.754	21	1 687.420
U_200_2	200	300	21	1 326.250	21	1 287.250
U_200_3	200	300	20	1 603.076	20	1 565.130
U_200_4	200	300	21	968.057	21	943.988
U_200_5	200	300	21	1 164.854	21	1 112.500
U_200_6	200	300	20	1 310.448	20	1 261.360
U_200_7	200	300	21	1 166.970	21	1 108.010
U_200_8	200	300	20	1 345.542	20	1 259.660
U_200_9	200	300	20	1 623.972	20	1 575.460
U_200_10	200	300	20	1 866.434	20	1 840.350
Total			205	14 082.357	205	13 641.128

Table 13: Detailed results on the U instances of the VRPDO

			Average of 5		Best of 5	
Instance	Customers	Time (s)	Routes	Cost	Routes	Cost
UBC_50_1	50	30	2	256.042	2	256.042
UBC_50_2	50	30	2	305.678	2	301.963
UBC_50_3	50	30	2	241.696	2	241.696
UBC_50_4	50	30	2	226.627	2	223.522
UBC_50_5	50	30	2	208.172	2	208.096
UBC_50_6	50	30	2	242.484	2	241.846
UBC_50_7	50	30	2	246.186	2	246.186
UBC_50_8	50	30	2	196.079	2	196.079
UBC_50_9	50	30	2	184.054	2	183.804
UBC_50_10	50	30	2	194.297	2	194.297
Total			20	2 301.314	20	2 293.531
UBC_100_1	100	90	4	383.743	4	374.862
UBC_100_2	100	90	4	357.865	4	351.220
UBC_100_3	100	90	4	330.514	4	323.464
UBC_100_4	100	90	4	335.395	4	334.615
UBC_100_5	100	90	4	376.359	4	371.859
UBC_100_6	100	90	4	364.341	4	357.611
UBC_100_7	100	90	4	339.716	4	337.902
UBC_100_8	100	90	4	426.310	4	419.714
UBC_100_9	100	90	4	394.706	4	386.871
UBC_100_10	100	90	4	357.947	4	350.272
Total			40	3 666.896	40	3 608.39
UBC_200_1	200	300	8	609.565	8	601.353
UBC_200_2	200	300	8	525.744	8	511.301
UBC_200_3	200	300	8	851.712	8	842.696
UBC_200_4	200	300	8	670.960	8	634.515
UBC_200_5	200	300	8	674.374	8	669.268
UBC_200_6	200	300	8	665.084	8	649.461
UBC_200_7	200	300	8	649.339	8	630.953
UBC_200_8	200	300	8	655.330	8	640.022
UBC_200_9	200	300	8	582.294	8	562.377
UBC_200_10	200	300	8	650.315	8	562.377
Total			80	6 534.717	80	6 384.287

Table 14: Detailed results on the UBC instances of the VRPDO

			Average of 5		Best of 5	
Instance	Customers	Time (s)	Routes	Cost	Routes	Cost
V_50_1	50	30	5	417.903	5	416.349
V_50_2	50	30	6	345.635	6	345.635
V_50_3	50	30	5	401.676	5	398.662
V_50_4	50	30	5	378.415	5	374.501
V_50_5	50	30	5	328.692	5	328.387
V_50_6	50	30	5	390.432	5	387.196
V_50_7	50	30	5	368.295	5	365.985
V_50_8	50	30	6	387.645	6	387.645
V_50_9	50	30	6	366.000	6	366.000
V_50_10	50	30	6	375.068	6	372.228
Total			54	3 759.762	54	3 742.588
V_100_1	100	90	11	652.508	11	647.566
V_100_2	100	90	10.2	905.803	10	895.205
V_100_3	100	90	10.4	743.616	10	733.412
V_100_4	100	90	10	596.960	10	594.605
V_100_5	100	90	10	795.629	10	765.491
V_100_6	100	90	11	597.567	11	597.288
V_100_7	100	90	11	705.792	11	702.181
V_100_8	100	90	11	745.717	11	744.855
V_100_9	100	90	10	812.543	10	794.041
V_100_10	100	90	10	616.804	10	614.412
Total			104.6	7 172.939	104	7 089.056
V_200_1	200	300	21	1 321.538	21	1 288.280
V_200_2	200	300	20	1 526.820	20	1 420.190
V_200_3	200	300	21	1 369.850	21	1 346.210
V_200_4	200	300	21	1 545.578	21	1 488.720
V_200_5	200	300	21	1 445.074	21	1 420.810
V_200_6	200	300	21	1 562.392	21	1 519.580
V_200_7	200	300	20	1 204.788	20	1 171.590
V_200_8	200	300	20	1 851.056	20	1 799.830
V_200_9	200	300	20	1 837.234	20	1 798.400
V_200_10	200	300	20	1 596.814	20	1 527.620
Total			205	15 261.144	205	14 781.23

Table 15: Detailed results on the V instances of the VRPDO