The Doubly Open Park-and-loop Routing Problem

Nicolás Cabrera[a], Jean-François Cordeau[a], Jorge E. Mendoza[a]

[a]*HEC Montréal, 3000 chemin de la Côte-Sainte-Catherine, Montréal, H3T 2A7 Canada*

Abstract

The purpose of this paper is to introduce a highly efficient heuristic for the Doubly Open Park-and-loop Routing Problem (DOPLRP). This problem is an extension of the classical Vehicle Routing Problem (VRP) in which routes may involve a main tour that is completed using a vehicle and subtours that are carried out on foot after parking the vehicle. In addition, routes do not start and end at a central depot, but at customer locations. We propose a matheuristic based on a split procedure that generates high quality solutions fast. We present computational experiments on a set of real instances with up to 3,800 customers. We also illustrate the use of the matheuristic on a related problem called the Vehicle Routing Problem with Transportable Resources, where the method is capable of finding 32/40 new best solutions.

Keywords: vehicle routing problem, heuristics, sampling, logistics

1. Introduction

On a daily basis, public utilities must visit a large number of customers to perform on-site services (e.g., installation, maintenance work, meter readings). In most cases, these customers are located in densely populated areas (e.g., city centers and small villages). Operating in these areas is not easy. One reason is that they are usually subject to heavy traffic congestion and accessibility constraints. Another reason is that access to parking is very limited. Therefore, utilities often design vehicle routes following a park-and-loop structure (Parragh & Cordeau, 2017). These routes involve a main tour that is completed using a vehicle and subtours that are carried out on foot after parking the vehicle. Another difficulty faced by utilities is the high number of customers simultaneously requesting service. Indeed, utilities often lack the resources to perform all requests by themselves. As a result, they sometimes outsource some of these tasks to a third-party contractor. There can be

many types of agreements between the two parties. For example, in Mendoza et al. (2009) the authors discuss the case of the water and sewer company in Bogotá, Colombia, where the third-party operator is directly responsible for designing their routes. The latter implies a strong relation between the parties, because the utility must provide the third-party with access to its information systems (e.g., ERP). Other utilities prefer lighter agreements, in which they are responsible for the routing and the third-party only executes the field operations. In this setting, the routes usually do not start and end at a single depot, but at customer locations. This leads to the doubly open park-and-loop routing problem (DOPLRP) studied in this paper.

We focus our attention on the case of ENEDIS, a subsidiary of the French electricity giant EDF. Everyday, the company must visit thousands of customers located mostly in highly populated residential areas. Part of this vast operation is outsourced to third-party contractors. Under the current agreement, ENEDIS provides the third-party with a route plan that minimizes the number of technicians and the driven distance. According to the agreement, the routes may start and end at any given customer and contain park-and-loop subroutes. However, a subtour has a duration limit. In addition, the total daily walking distance is bounded. Finally, the routes should satisfy a maximum duration constraint, associated to the work shift of the technicians.

The resulting problem is closely related to the truck and trailer routing problem (TTRP). In this variant of the VRP, a fleet composed of trucks pulling trailers has to serve a set of customers, some of which are only accessible by truck (without the trailer). With this in mind, a subset of all customers are set as *decoupling* points (i.e., parking places) where the trailer can be detached from the truck. The truck can then visit customers with accessibility constraints following a park-and-loop structure. To the best of our knowledge, this problem was originally introduced by Semet (1995), who proposed a two-phase heuristic method. Since then, a significant body of literature addressing the TTRP through different solution strategies has emerged. The problem is typically solved using metaheuristic (Chao, 2002; Sheuerer, 2006; Lin et al., 2009; Villegas et al., 2011; Derigs et al., 2013) and matheuristic algorithms (Villegas et al., 2013). Most of this work has focused on finding solutions to relatively small instances with up to 100 customers and 50 (or fewer) parking places. It is worth noting that the latter play a major role on the complexity of the problem. As opposed to the TTRP, in the case of ENEDIS, the vehicle can be parked at any customer. Additionally, in

the TTRP routes are only limited by one resource, namely, the capacity of both the truck and the trailer. In contrast, ENEDIS routes are constrained by three resources, namely, the daily walking distance, the route duration and the subtour duration limit.

More recently, some authors have focused their attention on TTRP variants such as the TTRPTW, which includes time windows (Derigs et al., 2013; Parragh & Cordeau, 2017). The methodology proposed by Parragh & Cordeau (2017), which enabled them to provide the first optimal solutions for instances in the literature with up to 100 customers, is a branch-and-price algorithm that uses specific enhancements in the pricing scheme and alternative lower bound computations.

Another related problem is the Swap-body routing problem (SVRP). This variant of the VRP considers a homogeneous fleet consisting of trucks, semi-trailers and swap bodies. The problem also considers swap locations, special vertices where the swap bodies can be coupled and decoupled. Sparked by the 2014 VeRoLog Challenge, the SVRP has received substantial attention from the research community. Absi et al. (2017) developed a relax and repair heuristic based on the principle of solving a relaxed version of the problem and then deriving a feasible solution by repairing the relaxed one. Huber & Geiger (2017) proposed an Iterated Local Search (ILS) based heuristic, in which an initial random solution is modified using intra- and inter-tour local search. Miranda-Bront et al. (2017) considered a cluster-first route-second approach including a GRASP and a ILS metaheuristic. Todosijević et al. (2017) proposed a method based on a two general variable neighborhood search heuristics. Huber et al. (2020) proposed a column generation-based matheuristic that combines a variable neighborhood search with a set partitioning formulation. Most of these approaches were capable of finding solutions to instances with up to 500 customers and 100 swap locations.

To the best of our knowledge, the problem most closely related to our DOPLRP is the VRP with transportable resources (VRPTR) introduced by Coindreau et al. (2019). In this problem, a set of technicians has to serve a set of customers. The technicians can either walk or drive to their next location and are allowed to carpool (i.e., share a vehicle). To solve their VRPTR, Coindreau et al. (2019) propose a variable neighborhood search algorithm and a fast insertion mechanism. They tested their method on instances with up to 50 customers. The authors also studied a version of their problem in which carpooling is not allowed. The latter is a special case of the DOPLRP faced by ENEDIS, in which i) the duration of the subtours is not constrained,

ii) the routes start and end at a pre-defined node (i.e., a centre depot), and *iii*) the routes serve a considerably smaller number of customers.

In this paper we introduce and solve the DOPLRP. Our method is a matheuristic following the route-first, assemble-second principle. In the first phase, the method uses a modified split procedure to generate a set of high-quality routes. In the second phase, the method assembles the final solution by recombining some of the routes found in the first phase using a set partitioning model. We present extensive computation experiments on real instances with up to 3,800 customers provided by ENEDIS. Our results suggest that our algorithm is able to improve by 11.58% (on average) the solutions delivered by the routing software currently used by the company. To foster future research on this new problem variant, we made the instances publicly available, and developed an open online solution verification and visualization tool. We also present state-of-the-art results on the non-carpooling version of the VRPTR introduced in Coindreau et al. (2019). Our algorithm produced 32 (out of 40) new best known solutions for that problem and yielded average improvements of 2.07% in solution quality as well as speed ups of up to 40 times with respect to the previous state-of-the-art method.

The remainder of the paper is organized as follows. Section 2 presents the problem and the corresponding mathematical formulation. Section 3 provides a detailed description of the proposed matheuristic. Section 4 reports and discusses the results of our computational experiments. Finally, Section 5 concludes the paper and outlines directions for future work.

2. Problem description and formulation

The DOPLRP can be defined on a complete and directed graph $\mathcal{G} = (\mathcal{N}, \mathcal{A})$, where $\mathcal{N} = \{1, ..., n\}$ is the node set and $\mathcal{A} = \{(i, j) : i, j \in \mathcal{N}, i \neq j\}$ is the arc set. Nodes represent the company customers. Each customer $i \in \mathcal{N}$ has a service time p_i. In addition, each arc $(i, j) \in \mathcal{A}$ has four main attributes: the walking distance δ_{ij}, the driving distance d_{ij}, the walking time η_{ij}, and the driving time τ_{ij}. Customers are served using a set $\mathcal{E}$ of homogeneous technicians who may drive or walk to their next location. Each technician $e \in \mathcal{E}$ is hired for a whole working day that lasts τ_M units of time, and has a maximum daily walking distance d_M. The maximum distance that a technician may walk between two points is κ_M. The parking time is ν. Hiring a technician involves a fixed cost c^f. In addition, driving involves a variable cost c^v per unit of distance. The objective of the DOPLRP is to

find a set of routes (starting and ending at any customer) that minimizes the total costs such that: each customer is served exactly once; the total duration of all routes performed by any technician does not exceed the working day duration; the walking distance of all routes does not exceed the walking distance limit; and the total duration of a walking subtour (i.e., subroute) does not exceed the time limit γ_M. Because it is a generalization of the well-know vehicle routing problem (VRP), the DOPLRP is NP-hard.

A solution of the DOPLRP may have three types of routes: pure vehicle routes performed by a technician driving between customers; pure walking routes performed by a technician walking between customers; and finally vehicle routes with walking subtours. The latter are composed of a main tour performed by the technician while driving the car, and one or more walking subtours, in which the technician parks the car, visits one or more customers on foot, and then returns. The parking place of the car is called the root of the subtour. Figure 1 shows a feasible solution of the DOPLRP with 15 customers. Customers 1 to 4 are served by a pure vehicle route, while customers 5 to 7 are served by a pure walking route. Finally, customers 8 to 15 are served by a vehicle route with walking subtours. Note that a technician may start the route with a walking subtour (between customers 8 and 9), and may also end by a walking subtour (serving customers 13, 14 and 15).

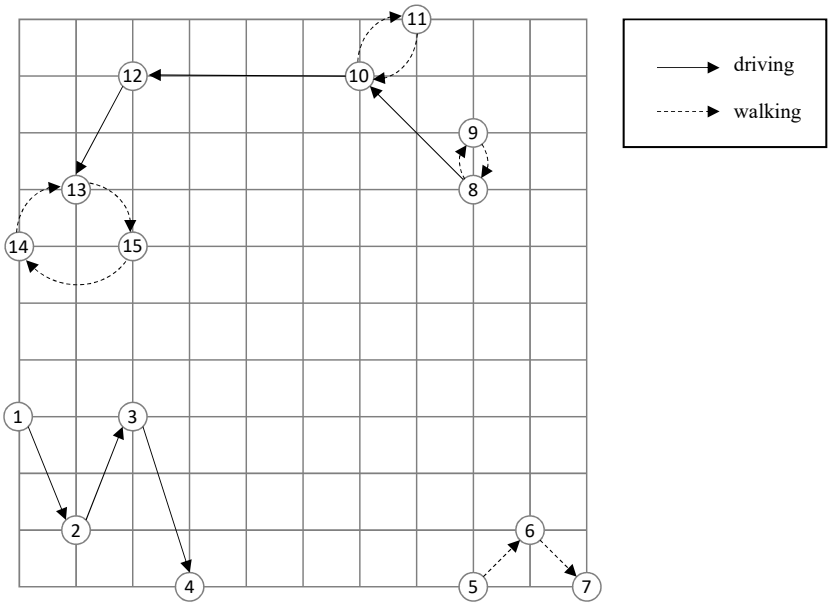


Figure 1: OPLRP solution example.

Even though our solution methodology is heuristic, we provide a mathematical formulation of the DOPLRP with the intent of formally stating the problem being addressed. We define an additional set: $\mathcal{N}^+ = \{n+1, ..., 2n\}$ as a set of duplicated nodes, where node $i \in \mathcal{N}$ and node $i+n \in \mathcal{N}^+$ represent the same physical location. Using the sets $\mathcal{N}$ and $\mathcal{N}^+$, it is possible to represent different actions that a technician may perform at a customer. On the other hand, the set $\mathcal{N}$ allows to represent both customers served in a route and the starting node of a walking subtour where the technician parks the car (i.e., parking place). On the other hand, the set $\mathcal{N}^+$ represents the ending nodes of walking subtours where technicians recover their car. In addition, we define the following sets of arcs: $\mathcal{A}^1 = \{(i,j) : i \in \mathcal{N} \cup \mathcal{N}^+, j \in \mathcal{N}\}$ are the arcs along which a technician may drive and $\mathcal{A}^2 = \{(i,j) : i \in \mathcal{N}, j \in \mathcal{N} \cup \mathcal{N}^+ | \delta_{ij} \leq \kappa_M\}$ are the arcs along which a technician may walk. The integer programming formulation of the DOPLRP uses the following variables:

-$x_{ije} = 1$ if technician $e \in \mathcal{E}$ drives from node i to node j, such that

$(i,j) \in \mathcal{A}^1$ and 0, otherwise,

-$h_{ije} = 1$ if technician $e \in \mathcal{E}$ walks from node i to node j, such that $(i,j) \in \mathcal{A}^2$ and 0, otherwise,

-$g_{ie} = 1$ if technician $e \in \mathcal{E}$ parks at node $i \in \mathcal{N}$ and 0, otherwise,

-$y_{ie} = 1$ if technician $e \in \mathcal{E}$ starts his route at node $i \in \mathcal{N}$ and 0, otherwise,

-$z_{ie} = 1$ if technician $e \in \mathcal{E}$ ends his route at node $i \in \mathcal{N} \cup \mathcal{N}^+$ and 0, otherwise,

-$b_{ie} = 1$ if technician $e \in \mathcal{E}$ services customer at node $i \in \mathcal{N}$ and 0, otherwise,

-w_{ie} is the arrival time of technician $e \in \mathcal{E}$ at node $i \in \mathcal{N}$,

-$o_e = 1$ if technician $e \in \mathcal{E}$ performs a pure walking route and 0, otherwise,

-$a_e = 1$ if technician $e \in \mathcal{E}$ is hired and 0, otherwise.

The DOPLRP can be formulated as follows:

$$\min \sum_{e \in \mathcal{E}} a_e \cdot c^f + \sum_{e \in \mathcal{E}} \sum_{(i,j) \in \mathcal{A}^1} x_{ije} \cdot d_{ij} \tag{1}$$

subject to

$$\sum_{i \in \mathcal{N}} y_{ie} = a_e, \quad \forall e \in \mathcal{E} \tag{2}$$

$$\sum_{i \in \mathcal{N}} z_{ie} = a_e, \quad \forall e \in \mathcal{E} \tag{3}$$

$$\sum_{(i,j) \in \mathcal{A}^1} x_{ije} + \sum_{(i,j) \in \mathcal{A}^2} h_{ije} \leq a_e \cdot |\mathcal{A}^1 \cup \mathcal{A}^2|, \quad \forall e \in \mathcal{E} \tag{4}$$

$$\sum_{e \in \mathcal{E}} b_{ie} = 1, \quad \forall i \in \mathcal{N} \tag{5}$$

$$b_{ie} \geq \sum_{j:(j,i) \in \mathcal{A}^1} x_{jie} + \sum_{j:(j,i) \in \mathcal{A}^2} h_{jie}, \quad \forall i \in \mathcal{N}, e \in \mathcal{E} \tag{6}$$

$$b_{ie} \leq y_{ie} + \sum_{j:(j,i) \in \mathcal{A}^1} x_{jie} + \sum_{j:(j,i) \in \mathcal{A}^2} h_{jie}, \quad \forall i \in \mathcal{N}, e \in \mathcal{E} \tag{7}$$

$$\sum_{j:(i,j) \in \mathcal{A}^2} h_{ije} \leq g_{ie} + (1 - g_{ie}) + o_e, \quad \forall i \in \mathcal{N}, e \in \mathcal{E} \tag{8}$$

$$\sum_{j:(i,j)\in\mathcal{A}^2} h_{ije} \geq g_{ie} - (1 - g_{ie}) - o_e, \quad \forall i \in \mathcal{N}, e \in \mathcal{E} \tag{9}$$

$$\begin{aligned} y_{ie} - z_{ie} = & \sum_{j:(i,j)\in\mathcal{A}^1} x_{ije} + \sum_{j:(i,j)\in\mathcal{A}^2} h_{ije} \\ & - \sum_{j:(j,i)\in\mathcal{A}^1} x_{jie} - \sum_{j:(j,i)\in\mathcal{A}^2} h_{jie}, \quad \forall i \in \mathcal{N} \cup \mathcal{N}^+, e \in \mathcal{E} \end{aligned} \tag{10}$$

$$\sum_{j:(i,j)\in\mathcal{A}^1} x_{ije} + \sum_{j:(i,j)\in\mathcal{A}^2} h_{ije} \leq 1, \quad \forall i \in \mathcal{N} \cup \mathcal{N}^+, e \in \mathcal{E} \tag{11}$$

$$x_{ije} + h_{ije} \leq 1, \quad \forall (i,j) \in \mathcal{A}^1 \cap \mathcal{A}^2, e \in \mathcal{E} \tag{12}$$

$$\begin{aligned} & w_{ie} + (\tau_{ij} + \nu) \cdot x_{ije} + \eta_{ij} \cdot h_{ije} + b_{ie} \cdot p_i \\ & \quad \leq w_{je} + (1 - x_{ije} - h_{ije}) \cdot \tau_M, \quad \forall (i,j) \in \mathcal{A}^1 \cap \mathcal{A}^2, e \in \mathcal{E} \end{aligned} \tag{13}$$

$$\begin{aligned} & w_{ie} + (\tau_{ij} + \nu) \cdot x_{ije} + b_{ie} \cdot p_i \\ & \quad \leq w_{je} + (1 - x_{ije}) \cdot \tau_M, \quad \forall (i,j) \in \mathcal{A}^1 \setminus (\mathcal{A}^1 \cap \mathcal{A}^2), e \in \mathcal{E} \end{aligned} \tag{14}$$

$$\begin{aligned} & w_{ie} + \eta_{ij} \cdot h_{ije} + b_{ie} \cdot p_i \\ & \quad \leq w_{je} + (1 - h_{ije}) \cdot \tau_M, \quad \forall (i,j) \in \mathcal{A}^2 \setminus (\mathcal{A}^1 \cap \mathcal{A}^2), e \in \mathcal{E} \end{aligned} \tag{15}$$

$$(1 - y_{ie}) \cdot \tau_M \geq w_{ie}, \quad \forall i \in \mathcal{N} \cup \mathcal{N}^+, e \in \mathcal{E} \tag{16}$$

$$w_{i+ne} - w_{ie} - p_i \cdot b_{ie} \leq g_{ie} \cdot \gamma_M + (1 - g_{ie}) \cdot \tau_M, \quad \forall i \in \mathcal{N}, e \in \mathcal{E} \tag{17}$$

$$\sum_{(i,j)\in\mathcal{A}^2} h_{ije} \cdot \delta_{ij} \leq a_e \cdot d_M, \quad \forall e \in \mathcal{E} \tag{18}$$

$$o_e \cdot \gamma_M + (1 - o_e) \cdot \tau_M \geq w_{ie}, \quad \forall e \in \mathcal{E} \tag{19}$$

$$(1 - o_e) \cdot |\mathcal{A}^1| \geq \sum_{(i,j)\in\mathcal{A}^1} x_{ije}, \quad \forall e \in \mathcal{E} \tag{20}$$

$$a_e \in \{0,1\}, \quad \forall e \in \mathcal{E} \tag{21}$$

$$o_e \in \{0,1\}, \quad \forall e \in \mathcal{E} \tag{22}$$

$$g_{ie} \in \{0,1\}, \quad \forall i \in \mathcal{N}, e \in \mathcal{E} \tag{23}$$

$$y_{ie} \in \{0,1\}, \quad \forall i \in \mathcal{N}, e \in \mathcal{E} \tag{24}$$

$$b_{ie} \in \{0,1\}, \quad \forall i \in \mathcal{N}, e \in \mathcal{E} \tag{25}$$

$$z_{ie} \in \{0,1\}, \quad \forall i \in \mathcal{N} \cup \mathcal{N}^+, e \in \mathcal{E} \tag{26}$$

$$w_{ie} \geq 0, \quad \forall i \in \mathcal{N} \cup \mathcal{N}^+, e \in \mathcal{E} \tag{27}$$

$$x_{ije} \in \{0,1\}, \quad \forall (i,j) \in \mathcal{A}^1, e \in \mathcal{E} \tag{28}$$

$$h_{ije} \in \{0,1\}, \quad \forall (i,j) \in \mathcal{A}^2, e \in \mathcal{E} \tag{29}$$

The objective function (1) comprises two terms: the first one represents the fixed cost of hiring technicians and the second one the total variable cost associated to the distance traveled by car. Constraints (2) and (3) state that a hired technician must start and end her or his route at a customer. Constraints (4) guarantee that only hired technicians may drive or walk through the graph. Constraints (5) state that all customers must be serviced by one technician. Constraints (6) and (7) guarantee that a customer is only served by technicians that arrived at the corresponding node. Constraints (8) and (9) state that after parking the car, the technician must leave on foot. Constraints (10) are the connectivity constraints. Constraints (11) and (12) state that a technician only drives or walks through an arc, but not both. Constraints (13)-(15) are subtour elimination constraints. Constraints (16) state that the departure time of a technician is always at the beginning of the day, and also that the arrival to any customer must be before the day ends. Constraints (17) guarantee that any walking subtour must be completed before the limit. Constraints (18) state that the total walking distance of any technician must be less than the limit. Constraints (19) guarantee that the duration of a pure walking route does not exceed the limit. Constraints (20) state that a pure walking route is only composed by walking.

3. Solution method

Since the DOPLRP generalizes the VRP, it is clear that solving formulation (1)-(29) with a general-purpose MIP solver is not adequate for instances in the range of the thousands of customers (as required by ENEDIS). Consequently, we turned to a matheuristic approach combining the computational efficiency of heuristics with the capability to exploit certain properties of the problem via dynamic and mixed-integer programming. Our approach uses randomized heuristics for the traveling salesman problem and a tour partitioning procedure to sample the solution space. These procedures allow us to create a subset $\mathcal{R} \subseteq \Omega$, where Ω denotes the set of all existing routes. Then, a set partitioning formulation is used to assemble a high-quality solution for the problem by selecting a subset of routes from $\mathcal{R}$. Algorithm 1 presents the main logic of the proposed matheuristic. Line 1 initializes subset $\mathcal{R}$. Line 2 initializes the iteration number. The sampling phase (lines 3 to 14) runs for a given number of iterations (T) or a maximum execution time (Q) – whatever condition is met first. During the sampling phase, the algorithm populates $\mathcal{R}$ (lines 4-13) using a set of TSP heuristics $\mathcal{H}$ and the splitting procedure $split\langle\cdot,\cdot\rangle$. Line 4 randomly selects a TSP heuristic h from $\mathcal{H}$. Line 5 generates a giant path visiting all customers using h. Line 6 generates a solution to the DOPLRP, denoted as s^t, composed by pure vehicle routes, pure walking routes, and vehicle routes with walking subtours. Line 7 joins the routes in solution s^t to subset $\mathcal{R}$. Lines 8-11 update the incumbent solution. Line 15 solves a set partitioning model over $\mathcal{R}$ using $f(s^*)$ as an upper bound on the objective function value in (1). Finally, line 16 returns the solution $\overline{\mathcal{R}}$.

Algorithm 1 `Matheuristic` function

Require: $\mathcal{G}$, graph; $\mathcal{H}$, heuristic set; T iteration limit; Q time limit.
Ensure: final solution $\overline{\mathcal{R}}$

1: $\mathcal{R} \leftarrow \emptyset$
2: $t \leftarrow 1$
3: **while** $t < T \wedge samplingTime < Q$ **do**
4: $\quad h \leftarrow \mathcal{H}_j$
5: $\quad \pi^t \leftarrow h(\mathcal{G})$
6: $\quad s^t \leftarrow split\langle \mathcal{G}, \pi^t \rangle$ ▷ *see §3.1*
7: $\quad \mathcal{R} \leftarrow \mathcal{R} \cup s^t$
8: $\quad$ **if** t $= 1$ **then**
9: $\quad\quad s^* \leftarrow s^t$
10: $\quad$ **else if** $f(s^t) < f(s^*)$ **then**
11: $\quad\quad s^* \leftarrow s^t$
12: $\quad$ **end if**
13: $\quad t \leftarrow t + 1$
14: **end while**
15: $\overline{\mathcal{R}} \leftarrow$ `setPartitioning`$(\mathcal{G}, \mathcal{R}, s^*)$ ▷ *see §3.2*
16: **return** $\overline{\mathcal{R}}$

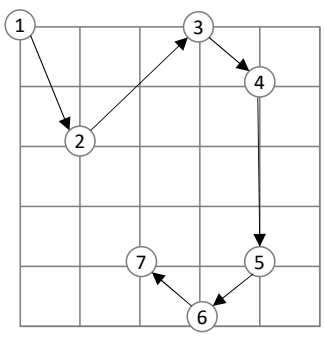
One of the key steps in Algorithm 1 is to find a giant path visiting all the customers (line 6). To this end, we use randomized versions of four classical TSP constructive heuristics, namely, randomized nearest neighbor (RNN), randomized nearest insertion (RNI), randomized farthest insertion (RFI), and randomized best insertion (RBI). We borrowed the randomization strategies for the four heuristics from Mendoza & Villegas (2013). It is worth noting that TSP heuristics consider a depot where the tour starts and ends. Recall, however, that the routes in our DOPLRP are doubly-open. We therefore introduce mild modifications to the heuristics to account for this feature.

The remainder of this section is organized as follows. Section 3.1 explains the procedure used to extract feasible routes from the previously generated giant path. Section 3.2 presents the mathematical model used to find the final solution.

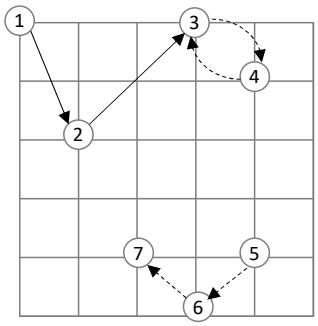
3.1. Split

To extract a solution s^t composed by pure vehicle routes, pure walking routes and vehicle routes with walking subtours from π^t (line 6 in Algorithm

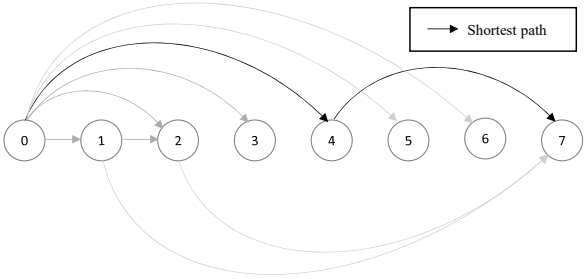
1) our method uses an adaptation of the Prins (2004) splitting procedure (hereafter labeled Split). The Split procedure follows two steps: first, it constructs a directed acyclic graph $\mathcal{G}' = (\mathcal{N}', \mathcal{A}')$ defined by a set of nodes $\mathcal{N}' = (v_0, v_1, ..., v_i, ..., v_n)$ and the set of arcs $\mathcal{A}'$. Node v_0 is a dummy node, while nodes numbered 1 to n represent the customer in the i-th position of π^t. Each arc $(i, j) \in \mathcal{A}'$ represents a feasible route $r_{(v_{i+1}, v_j)}$ with value $e_{r_{(v_{i+1}, v_j)}}$ visiting customers $(v_{i+1}, ..., v_j)$. The evaluation of route $r_{(v_{i+1}, v_j)}$ is the contribution to the objective function (1). Second, the procedure finds the shortest path from v_0 to v_n in $\mathcal{G}'$. Note that the set of arcs (i.e., routes) along the shortest path corresponds to a feasible solution s^t. Figure 2 illustrates the idea of the Split procedure on a simple example.



(a) Node location and giant path



(b) Optimal split



(c) Split graph

Figure 2: Split procedure example.

Since the graph $\mathcal{G}'$ is directed and acyclic, building the graph and finding the shortest path can be done simultaneously using Bellman's algorithm

(Prins, 2004). Algorithm 2 shows the body of the Split procedure. Lines 1 to 4 initialize the shortest path labels. Then, we enter the outer loop (lines 5-21) where line 6 sets the tail node. Afterwards, we use the inner loop (lines 8-20) to build all arcs sharing the tail node. At the start of the inner loop we build a new arc expanding the last generated arc. In the next step, we evaluate if the arc should be added to $\mathcal{G}'$. These tasks are performed by the evaluate function in line 11. If the arc is added to the graph, lines 13 to 16 update the shortest path and predecessor labels and line 18 updates the head node. If an arc was added and $j \leq |\mathcal{N}|$ the inner loop continues. After completing the outer loop, we retrieve the solution using the giant path and the predecessor labels.

To evaluate if an arc should be added to $\mathcal{G}'$ (line 12) we first check if $i + 1 = j$. If the condition holds, the arc represents a route that only visits customer v_j, the evaluation e_r is equal to the fixed cost of hiring a technician c^f and the feasibility f_r is set to $true$. If $i+1 \neq j$, we solve a subproblem that can be seen as a multi-resource version of the single truck and trailer routing problem with satellite depots (STTRPSD) proposed by Villegas et al. (2010). In our version of the problem, the satellite depots correspond to customers in $r_{(v_{i+1},v_j)}$.

We define the subproblem on an auxiliary graph that is similar to that introduced by Villegas et al. (2010). To simplify the notation, we will use $\hat{\imath}$ to refer to $i + 1$. The subproblem is defined on an auxiliary state graph $\mathcal{Y} = (\mathcal{X},\mathcal{U})$. The node set $\mathcal{X}$ includes a node for each state $[l, m]$ such that $\hat{\imath} \leq l \leq j \wedge l \leq m \leq j | v_l, v_m \in r_{(v_{\hat{\imath}},v_j)}$. State $[l, m]$ represents the use of customer v_l as the root of a subtour (parking place) ending at customer v_m. The state $[\hat{\imath}, \hat{\imath}]$ represents the service of the first customer, and acts as the source node of $\mathcal{Y}$. In addition we include a dummy node θ that acts as the sink node for $\mathcal{Y}$. The arc set $\mathcal{U}$ is composed by four types of arcs, that is, $\mathcal{U} = \bigcup_{y=1}^{4} \mathcal{U}_y$, where:

- $\mathcal{U}_1$ are the incoming arcs of θ. They represent the end of the route. Their cost w_u, walking distance λ_u, total subtour time μ_u and total time β_u are set to 0.

- $\mathcal{U}_2$ are the arcs between states $[l, l]$ and $[h, h]$ such that $h = l + 1$. They represent the movement of the vehicle between two consecutive customers. Their cost and total time are defined as $w_u = d_{v_l v_h} \cdot c^v$ and $\beta_u = p_{v_h} + \tau_{v_l v_h} + \nu$. The walking distance λ_u and the total subtour time μ_u are set to 0.

Algorithm 2 `Split` function

Require: $\mathcal{G}$, graph; π^t, giant path.
Ensure: solution s^t

$c_0 \leftarrow 0$ ▷ c: shortest path labels
for $i = 1$ **to** $i = |\mathcal{N}|$ **do**
    $c_i \leftarrow \infty$
end for
for $i = 0$ **to** $i = |\mathcal{N}| - 1$ **do**
    $j \leftarrow i + 1$
    `continue` $\leftarrow true$
    while `continue` $\wedge\ j \leq |\mathcal{N}|$ **do**
        $r \leftarrow r_{(v_{i+1}, v_j)}$
        `continue` $\leftarrow false$
        $\langle e_r, f_r \rangle \leftarrow$ `evaluate`(r) ▷ e_r: evaluation, f_r: feasibility
        if $f_r = true$ **then**
            if $c_{i-1} + e_r \leq c_j$ **then**
                $c_j \leftarrow c_{i-1} + e_r$
                $q_j \leftarrow i - 1$ ▷ q: predecessor labels
            end if
            `continue` $\leftarrow true$
            $j \leftarrow j + 1$
        end if
    end while
end for
$s^t \leftarrow$ `recoverSolution`(π^t, q)
return s^t

- $\mathcal{U}_3$ are the arcs between states $[l, m]$ and $[l, k]$ such that $m < k$ they represent a walking subtour visiting customers $(v_{m+1}, ..., v_k)$ using v_l as parking place. The walking distance and total time are defined as $\lambda_u = \sum_{y=m}^{k} \delta_{v_y v_{y+1}}$ and $\beta_u = \sum_{y=m+1}^{k} p_{v_y} + \sum_{y=m}^{k} \eta_{v_y v_{y+1}}$. The time of the subtour $\mu_u = \beta_u$. Finally, the cost w_u is set to 0.

- $\mathcal{U}_4$ are the arcs between states $[l, m]$ and $[h, h]$ such that $l < h$. They represent a walking subtour visiting customers $(v_{m+1}, ..., v_h)$ using v_l as parking place. Then, a walking subtour is performed visiting customers $(v_{m+1}, ..., v_h)$. The cost w_u is set to $d_{v_l v_h} \cdot c^v$ which is the cost

of driving between the parking place v_l and the parking place v_h. The walking distance is $\lambda_u = \sum_{y=m+1}^{h} \delta_{v_y v_{y+1}}$. Finally, the total time is $\beta_u = \sum_{y=m+1}^{h} p_{v_y} + \sum_{y=m}^{h} \eta_{v_y v_{y+1}} + \tau_{v_l v_h} + \nu$ and the time of the subtour is $\mu_u = \beta_u - \tau_{v_l v_h} - \nu$.

If an arc of the groups $\mathcal{U}_2$, $\mathcal{U}_3$ or $\mathcal{U}_4$ starts at state $[\hat{\imath}, \hat{\imath}]$ we add the service time $p_{v_{\hat{\imath}}}$ to the total time. An arc is only included in $\mathcal{U}$ if $(\beta_u \leq \tau_M) \wedge (\lambda_u \leq d_M) \wedge (\mu_u \leq \gamma_M)$. Figure 3 illustrates the structure of the auxiliary state graph $\mathcal{Y}$.

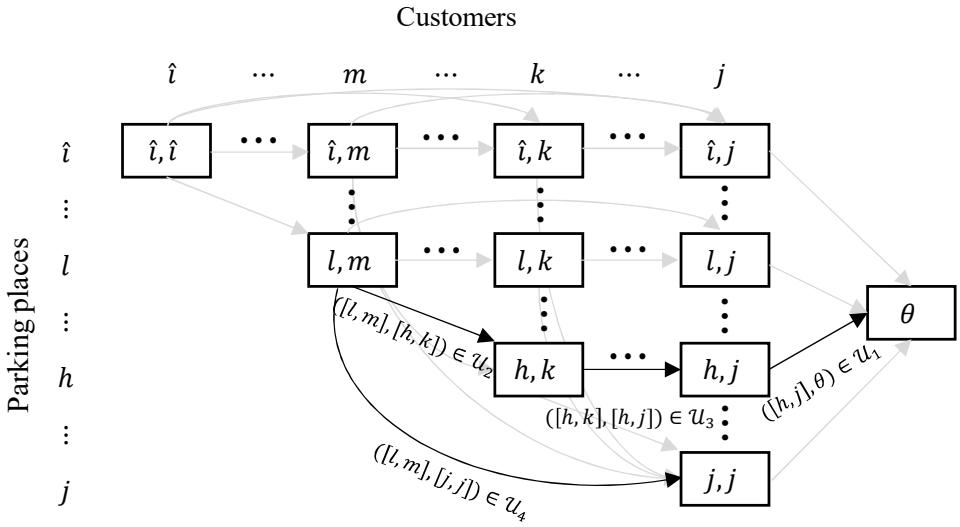


Figure 3: Auxiliary state graph structure.

Note that all paths from state $[\hat{\imath}, \hat{\imath}]$ to the dummy node θ represent a route in which the technician may perform several subtours. However, the majority of these paths are not feasible in terms of either the walking distance limit or the route duration limit (i.e, resources). Therefore, once we have the auxiliary graph $\mathcal{Y}$ we solve a resource-constrained shortest path problem (RCSPP) from the state $[\hat{\imath}, \hat{\imath}]$ to the dummy node θ. If it is not possible to find a path from $[\hat{\imath}, \hat{\imath}]$ to the dummy node θ that satisfies the resources constraints, the feasibility f_r is set to $false$. Otherwise, the evaluation e_r is equal to the cost of the minimum cost path that satisfies both constraints plus the fixed cost c^f and the feasibility f_r is set to $true$.

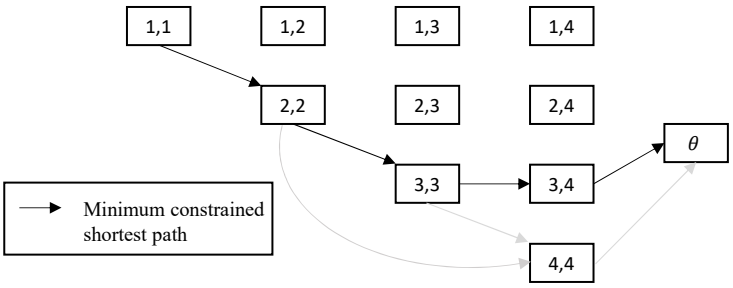
Figure 4 presents an example of a solution to the RCSPP that arises when the arc from node 0 to node 4 of the Split graph in Figure 2 is considered.

First, note that states $[2,1],[3,1],[3,2],[4,1],[4,2]$ and $[4,3]$ were not added to the graph $\mathcal{Y}$ represented in Figure 4a due to the definition of the node set $\mathcal{X}$. Then, the reader must consider that several arcs of the auxiliary state graph did not pass the feasibility check, thus, they were not added to the arc set. Arcs from the minimum constrained shortest path from $[1,1]$ to θ are highlighted in black. The arc from state $[1,1]$ to state $[2,2]$ represents the movement of the technician between customers 1 and 2 using the vehicle, as depicted in Figure 4b. Similarly, the arc from state $[2,2]$ to state $[3,3]$ represents the movement of the technician between customers 2 and 3 using the vehicle. The arc from state $[3,3]$ to state $[3,4]$ represents a walking subtour in which the technician serves customer 4 using customer 3 as parking place. Finally, the arc from state $[3,4]$ to node θ represents the end of the route at customer 3. Arcs that belong to the arc set $\mathcal{U}$, but were not part of the minimum constrained shortest path are colored in grey.

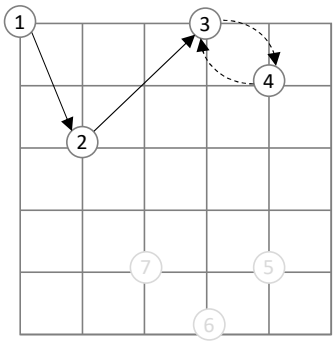
To solve the RCSPP in $\mathcal{Y}$, we use the pulse algorithm (PA) proposed by Lozano & Medaglia (2013), a recursive depth first search based on the idea of propagating a pulse through the network. In our particular case, we improve the PA performance taking advantage of $\mathcal{Y}$ structure. First, note that the minimum cost path from the state $[\hat{\imath},\hat{\imath}]$ to the dummy node θ represents the same solution as the minimum cost path from the dummy node θ to the state $[\hat{\imath},\hat{\imath}]$ on the reversed auxiliary graph $\mathcal{Y}'$. Second, note that $\mathcal{Y}$ is acyclic. Therefore, it is possible to find lower bounds on the resource consumption and the cost from the state $[\hat{\imath},\hat{\imath}]$ to the dummy node θ while building the state graph θ using Bellman's algorithm. Based on these observations, we adapted the PA to run in the *backward* direction, creating the graph and performing the initialization step simultaneously. In addition, we included two of the acceleration strategies for the PA outlined in Cabrera et al. (2020). Namely, path completion and pulse queuing. The path completion strategy enables the PA to construct feasible paths in initial stages of the search. The pulse queuing strategy modifies the algorithm's behavior, enabling the algorithm to perform an adjustable depth-first (or breadth-first) search.

3.2. Set partitioning

After the sampling phase, we use a set partitioning model to try to improve the best solution found s^* using parts of other solutions in $\mathcal{R}$ (line 15). The model is $\min_{\overline{\mathcal{R}}\in\mathcal{R}}\left\{\sum_{r\in\overline{\mathcal{R}}} e_r : \bigcup_{r\in\overline{\mathcal{R}}} = \mathcal{N}; r_i \bigcap r_j = \emptyset \ \forall r_i, r_j \in \overline{\mathcal{R}}\right\}$. The objective of this model is to minimize the total cost, guaranteeing that a customer is served exactly by one route. We delegate the task of solving



(a) Auxiliary state graph



(b) Solution on the original graph

Figure 4: Split procedure example.

this formulation to a general-purpose solver. To speed up the algorithm we use the objective function of the best solution found s^* as an initial upper bound.

4. Computational experiments

The proposed algorithm was implemented in Java and compiled using Eclipse SDK version 4.8.0. The experiments were performed on a 2 x Intel Xeon CPU E5-2637 v4 @ 3.50GHz processor with 512GB of RAM allocated to the memory heap size of the Java Virtual Machine. We used CPLEX 12.9 to solve the set-partitioning model in the assembly phase.

To assess the efficiency and effectiveness of our approach, we conducted two sets of experiments. For validation purposes, in the first set of exper-

iments we tested our method on the closely-related VRPTR (without carpooling) and compared its results to those delivered by the state-of-the-art method (i.e., the VNS introduced by Coindreau et al. (2019)). In the second set of experiments we tested our approach on real instances provided by ENEDIS and compared its performance against that of the software currently used by the company.

4.1. Experiments 1: the VRP with transportable resources

For these experiments we used the instances proposed by Coindreau et al. (2019). Each instance considers a number of customers $n = \{20, 30, 40, 50\}$ located inside a square grid of 10 km by 10 km. The depot is located at the center of the grid. The distance between nodes (customers and depot) is the Euclidian distance. In addition, to compute driving and walking times they consider a driving speed of 30 km/h and a walking speed a 4 km/h. The customer service times range from 20 to 35 minutes. The parking time is zero. Finally, the maximum daily walking distance for each worker is 5 km and the day duration is 7 hours (i.e., 420 minutes). It is worth recalling that similar to our DOPLRP, in the VRPTR, any customer location can be used as a parking space. The objective is to minimize the total driving distance (which constitutes the total transportation cost). An instance is referred to as "n_A_i", where n stands for the number of customers. A total of 10 instances are considered for each instance size (i.e., number of customers). Note that these instances include a depot. Accordingly, we adapted our method to guarantee that each technician departs from and returns to the depot.

Table 1 compares the performance of the proposed matheuristic (labeled MA) against the VNS of Coindreau et al. (2019) in terms of solution quality. Each row corresponds to an instance size (i.e., number of customers). Columns 2 and 3 report the number of BKSs delivered by each approach (recall that each instance family comprises 10 instances). Columns 4 to 6 report the minimum, average, and maximum gap in the objective function of the solutions returned MA with respect to those delivered by VNS. More specifically, for each instance in the set, the gap was computed as

$$\frac{f(MA) - f(VNS)}{f(VNS)}, \tag{30}$$

were $f(\cdot)$ denotes the objective function of the solution delivered by a given approach. In this experiment, MA ran with the number of iterations (T) set

to 2,500 and the time limit for the sampling phase (Q) set to infinity (i.e., no time limit). The depth limit in the pulse algorithm was set to 2.

Table 1: Solution quality on Coindreau et al (2019) instances.

# Customers	VNS	MA			
	BKS	BKS	Min. Δ cost %	Max. Δ cost %	Avg. Δ cost %
20	1/10	10/10	-8.16%	0.00%	-3.85%
30	0/10	10/10	-5.53%	-0.07%	-3.01%
40	3/10	10/10	-4.98%	0.00%	-1.17%
50	4/10	8/10	-5.55%	0.91%	-1.14%
Average	8/10	38/40	-6.06%	0.21%	-2.29%

As the results show, MA found 38 of the 40 best known solutions, while this figure is only 8 for VNS. The data also show that MA delivered solutions having (on average) 2.29% shorter total distance (i.e., cost) than VNS. Moreover, MA found solutions that decrease the total distance by up to 8.16%. In the two instances in which MA was not able to match or improve the solution reported by VNS, it delivered a solution with an objective function that is (in the worst case) only 0.91% higher.

Table 2 compares the two approaches in terms of computational efficiency. Similar to Table 1, the results are grouped by instance size. Column 1 shows the number of customers in the instance family. Columns 2 and 3 shows the average CPU time reported by VNS and MA, respectively. Columns 4 to 6 report the minimum, average, and maximum CPU time employed by MA to solve instances in a family and Column 7 reports the average speedup achieved by MA with respect to VNS.

Table 2: Computational times on Coindreau et al (2019) instances.

# Customers	VNS	MA			
	Avg. CPU	Avg. CPU	Min. CPU	Max. CPU	Avg. Speedup
20	71.00	13.41	12.33	14.31	5.29
30	381.00	35.97	32.88	39.80	10.59
40	1993.00	56.06	38.72	115.05	35.55
50	6779.00	110.51	59.76	162.56	61.35
Average	2306.00	53.99	35.92	82.93	42.71

While a perfect head-to-head comparison is impossible to make because of differences in the programming languages and testing environment, the

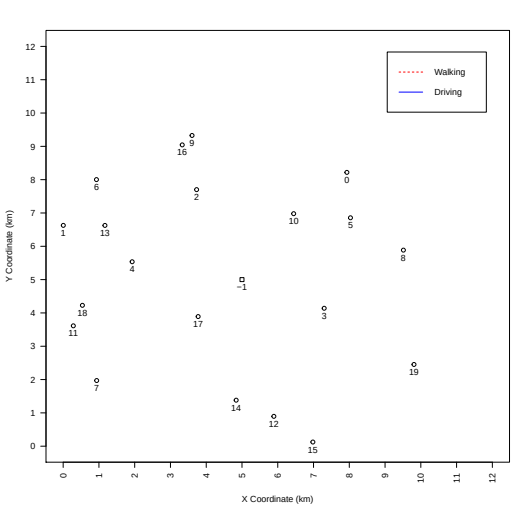
results suggest that MA is consistently faster than VNS. More specifically, MA was up to 60 times faster on instances with 50 customers and was roughly 42 times faster on average. In conclusion, the results obtained in terms of both solution quality and computational efficiency set MA as the new state-of-the-art approach to solve the VRPTR without carpooling.

To foster future research on this problem and facilitate comparisons with our results we have created a website where researchers can download the instances, access instance-by-instance results, download the detailed solutions reported by both VNS and MA, and do online solution checking and plotting. The website is available at: `https://nicolascabrera.shinyapps.io/VRPTR/`. As an example, Figures 5a, 5b and 5c show the customer locations for instance 20_A_7, the solution delivered by VNS, and the solution delivered by MA, respectively. The VNS solution has a cost of 40.11 km, while the MA solution has a cost of 38.67 km.

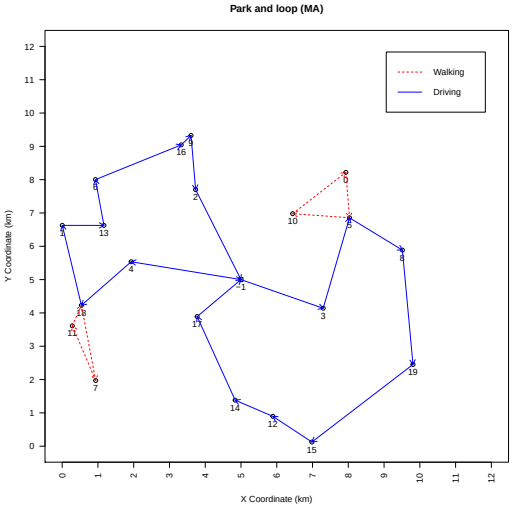
4.2. Experiments 2: DOPLRP real-world instances

As we mentioned in Section 1 we considered 60 real instances provided by ENEDIS. Each instance corresponds to one day of operation of a given *service zone* located in eastern France. The number of customers per instance ranges from 2,200 to 3,800. The distances between customers are Euclidian. In addition, to compute driving and walking times we assume a driving speed of 30 km/h and a walking speed of 4 km/h. The service time of each customer is 2 minutes (average time of reading an electricity meter). The parking time is 2 minutes. Finally, the maximum walking distance between two customers is 3.5 km, the maximum walking daily distance for each worker is 16 km, the time limit for each walking subtour is 2 hours and the day duration is 8 hours (i.e., 480 minutes). The fixed cost of hiring a technician is 375 € and the driving cost is 0.5 €/km. We made the instances publicly available at VRP-REP.ORG (Mendoza et al., 2014). The VRP-REP-ID is 2021-0006. We divided the set of instances in 10 categories. Table 3 shows the minimum and maximum number of customers in each category. Note that category 1 has the lowest number of customers while category 10 has the highest number of customers.

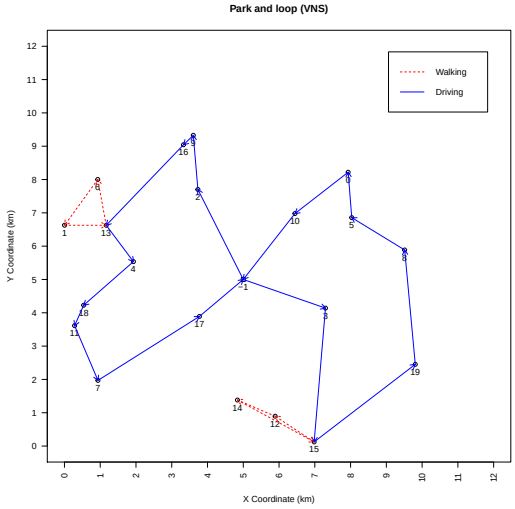
Table 4 compares the performance of MA against the routing software currently used by the company. Each row corresponds to an instance category. Columns 2 and 5 show the average number of technicians (i.e., number of routes) used. Columns 3 and 6 show the average variable cost related to driving a vehicle. Columns 4 and 7 show the average total cost. Column 8



(a) Instance 20_A_6



(b) MA solution



(c) VNS solution

Figure 5: VRPTR visualization tool example

shows the average difference in the number of technicians. Finally, column 9 shows the average percentage difference between the cost of the solutions found by each algorithm. In this experiment, the number of iterations T was

Table 3: ENEDIS instance information

Category	Customers_min	Customers_max	# Instances
1	2,200	2,560	6
2	2,561	2,650	6
3	2,651	2,740	6
4	2,741	2,860	6
5	2,861	2,975	6
6	2,976	3,050	6
7	3,051	3,150	6
8	3,151	3,220	6
9	3,221	3,400	6
10	3,401	3,800	6

set to 50 and the time limit for the sampling phase Q to 1 hour.

Table 4 shows that the MA is always capable of finding solutions with a smaller number of technicians. Note that the ENEDIS solutions use on average 18.15 technicians while the MA uses only 16.03 (on average). This reduction in the number of technicians translates into average savings of 795€ per day of operation, that is, nearly 50,000€ over the two months of operations covered by our testbed. Moreover, the MA solutions are also more efficient in terms of variable cost. More specifically, the MA solutions have an average variable cost of 147.32€ while the ENEDIS software solutions have an average variable cost of 164.70€. The decrease in both the fixed cost and the variable cost leads to a decrease in the total cost of 11.64% on average. In conclusion, MA can potentially bring considerable savings to ENEDIS if adopted and deployed to other service zones.

Similar to the data produced in Experiment 1, we share all the data produced in these experiments with the community at `https://nicolascabrera.shinyapps.io/OpenVRP/`. Following this link, the interested reader can find the detailed MA solutions for each instance and upload her or his own solutions for checking and plotting[1].

[1]The solutions must follow a predefined format explained on the website

Table 4: Computational results on ENEDIS instances.

Category	ENEDIS			MA			Δ Avg.	Δ Avg.
	Avg. # Technicians	Avg. Variable cost (€)	Avg. Total cost (€)	Avg. # Technicians	Avg. Variable cost (€)	Avg. Total cost (€)	# Technicians	Total cost %
1	15.50	148.20	5,960.70	13.50	130.87	5,193.37	-2.00	-12.87%
2	16.67	178.72	6,428.72	14.17	150.63	5,463.13	-2.50	-15.02%
3	16.50	151.49	6,338.99	14.83	139.23	5,701.73	-1.67	-10.05%
4	17.50	158.90	6,721.40	15.17	144.32	5,831.82	-2.33	-13.24%
5	17.83	141.98	6,829.48	15.83	132.19	6,069.69	-2.00	-11.13%
6	18.00	157.36	6,907.36	16.17	150.44	6,212.94	-1.83	-10.05%
7	19.17	212.78	7,400.28	17.00	179.79	6,554.79	-2.17	-11.43%
8	19.17	150.53	7,338.03	17.00	143.81	6,518.81	-2.17	-11.16%
9	20.00	188.15	7,688.15	17.50	167.30	6,729.80	-2.50	-12.47%
10	21.17	158.90	8,096.40	19.17	134.64	7,322.14	-2.00	-9.56%
Overall Avg.	18.15	164.70	6,970.95	16.03	147.32	6,159.82	-2.12	-11.64%

5. Concluding remarks

In this paper we presented a matheuristic for the the doubly open park-and-loop routing problem faced on a daily basis by a French utility. In this problem, routes can be completed by car, by walking, or both. In the latter, the routes include one or more subtours that are covered on foot after parking the car at a customer's location. Our matheuristic is capable of finding high quality solutions in a reasonable amount of time for instances with up to 3,800 customers. The matheuristic is based on a route-first assemble-second approach. Specifically, it uses four constructive heuristics for the traveling salesman problem and a tailored split procedure to populate a set of high-quality routes that is later use to assemble a final solution. To generate vehicle routes with walking subtours the method solves a particular resource constrained shortest path problem using an enhanced version of the pulse algorithm.

Extensive computational experiments carried on real instances provided by the utility suggest that the proposed method is able to improve the solutions currently used by the company by close to 12% (on average). This cost reduction comes as a result of i) using fewer technicians (i.e., fixed cost) and ii) better balancing the use of the driving and walking modes (i.e., variable cost).

In experiments carried out on standard instances for the closely related VRP with transferable resources (on its non-carpooling version) the proposed method yielded or matched 38 out of 40 best known solutions. In the two instances for which the method did not improve or match the best existing solution, the maximum gap with respect to the state-of-the-art method, the VNS by Coindreau et al. (2019), was less than 1%. These experiments also showed that the proposed method offers speedups of up to 60 times with respect to the VNS.

To encourage future research on these two problems, we developed two online tools that allow the members of the community to visualize our solutions and upload their own solutions for checking and plotting.

Future research will focus on incorporating the carpooling option, a setting that may yield additional savings to ENEDIS.

Acknowledgements

The authors would like to sincerely thank Thomas Triboulet from EDF Lab at Paris-Saclay for sharing the problem, instances, and solutions with us.

None of this work would have been possible without his precious assistance. This research was partially funded by the Canada First Research Excellence Fund through IVADO and by HEC Montréal through the Chair in Logistics and Transportation.

References

Absi, N., Cattaruzza, D., Feillet, D., & Housseman, S. (2017). A relax-and-repair heuristic for the swap-body vehicle routing problem. *Annals of Operations Research*, *253*, 957–978.

Cabrera, N., Medaglia, A. L., Lozano, L., & Duque, D. (2020). An exact bidirectional pulse algorithm for the constrained shortest path. *Networks*, *76*, 128–146.

Chao, I.-M. (2002). A tabu search method for the truck and trailer routing problem. *Computers and Operations Research*, *29*, 33–51.

Coindreau, M., Gallay, O., & Zufferey, N. (2019). Vehicle routing with transportable resources: Using carpooling and walking for on-site services. *European Journal of Operational Research*, *279*, 996–1010.

Derigs, U., Pullmann, M., & Vogel, U. (2013). Truck and trailer routing - problems, heuristics and computational experience. *Computers and Operations Research*, *40*, 536–546.

Huber, S., Cordeau, J., & Geiger, M. (2020). A matheuristic for the swap body vehicle routing problem. *OR Spectrum*, *42*, 111—-160.

Huber, S., & Geiger, M. (2017). Order matters – a variable neighborhood search for the swap-body vehicle routing problem. *European Journal of Operational Research*, *263*, 419–445.

Lin, S., Yu, V., & Chou, S. (2009). Solving the truck and trailer routing problem based on a simulated annealing heuristic. *Computers and Operations Research*, *36*, 1683–1692.

Lozano, L., & Medaglia, A. L. (2013). On an exact method for the constrained shortest path problem. *Computers and Operations Research*, *40*, 378–384.

Mendoza, J., Guéret, C., Hoskins, M., Lobit, H., Pillac, V., Vigo, D., & Vidal, T. (2014). VRP-REP: a vehicle routing community repository. In *Third meeting of the EURO Working Group on Vehicle Routing and Logistics Optimization (VeRoLog). Oslo (Norway)*.

Mendoza, J., Medaglia, A., & Velasco, N. (2009). An evolutionary-based decision support system for vehicle routing: The case of a public utility. *Decision Support Systems*, *46*, 730–742.

Mendoza, J., & Villegas, J. (2013). A multi-space sampling heuristic for the vehicle routing problem with stochastic demands. *Optimization Letters*, *7*, 1503–1516.

Miranda-Bront, J., Curcio, B., Méndez-Díaz, I., Montero, A., Pousa, F., & Zabala, P. (2017). A cluster-first route-second approach for the swap body vehicle routing problem. *Annals of Operations Research*, *253*, 935–956.

Parragh, S., & Cordeau, J. (2017). Branch-and-price and adaptive large neighborhood search for the truck and trailer routing problem with time windows. *Computers and Operations Research*, *83*, 28–44.

Prins, C. (2004). A simple and effective evolutionary algorithm for the vehicle routing problem. *Computers & Operations Research*, *31*, 1985–2002.

Semet, F. (1995). A two-phase algorithm for the partial accessibility constrained vehicle routing problem. *Annals of Operations Research*, *61*, 45–65.

Sheuerer, S. (2006). A tabu search heuristic for the truck and trailer routing problem. *Computers and Operations Research*, *33*, 894–909.

Todosijević, R., Hanafi, S., Urosević, D., Jarboui, B., & Gendron, B. (2017). A general variable neighborhood search for the swap-body vehicle routing problem. *Computers and Operations Research*, *78*, 468–479.

Villegas, J., Prins, C., Medaglia, A., & Velasco, N. (2010). Grasp/vnd and multi-start evolutionary local search for the single truck and trailer routing problem with satellite depots. *Engineering Applications of Artificial Intelligence*, *23*, 780–794.

Villegas, J., Prins, C., Medaglia, A., & Velasco, N. (2013). A matheuristic for the truck and trailer routing problem. *European Journal of Operational Research*, *230*, 231–244.

Villegas, J., Prins, C., Prodhon, C., Medaglia, A., & Velasco, N. (2011). A grasp with evolutionary path relinking for the truck and trailer routing problem. *Computers and Operations Research*, *38*, 1319–1334.