CIRRELT

CIRRELT-2019-40

The Delivery and Installation Routing Problem

Ousmane Ali
Jean-François Côté
Leandro C. Coelho

September 2019

Document de travail également publié par la Faculté des sciences de l'administration de l'Université Laval, sous le numéro FSA-2019-013.

Bureau de Montréal
Université de Montréal
C.P. 6128, succ. Centre-Ville
Montréal (Québec) H3C 3J7
Tél. : 1-514-343-7575
Télécopie : 1-514-343-7121

Bureau de Québec
Université Laval,
2325, rue de la Terrasse
Pavillon Palasis-Prince, local 2415
Québec (Québec) G1V 0A6
Tél. : 1-418-656-2073
Télécopie : 1-418-656-2624

The Delivery and Installation Routing Problem

Ousmane Ali, Jean-François Côté*, Leandro C. Coelho

Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT) and Department of Operations and Decision Systems, 2325 de la Terrasse, Université Laval, Québec, Canada G1V 0A6

Abstract. This paper proposes a variant of the vehicle routing problem with time windows (VRPTW) in which one disposes of two heterogeneous fleets to serve the customers; the first fleet is in charge of deliveries and the second one performs the installations. The customers receive some furniture, electronics and home appliances, and may require an installation service according to the items received. Installation can be performed by the deliverymen or by a dedicated installation fleet which needs to be synchronized with the delivery one. Each installer has different skills, and thus different installation times. We first formulate the problem as a mixed-integer linear programming model and solve it through a branch-and-bound procedure. We also design a tailored adaptive large neighborhood search heuristic to quickly find good solutions. Several computational experiments with new instances generated for this problem are used to assess the performance of both methods. Finally, we also prove the performance of both methods on the test instances of the related VRP with multiple synchronization constraints and the VRP with time windows and driver-specific times, yielding high-quality solutions in a short time using our heuristic and providing new lower bounds with the exact method.

Keywords.: Vehicle routing, delivery, installation, synchronization, heterogeneous fleet.

Acknowledgements. Financial support for this work was provided by the Natural Sciences and Engineering Research Council of Canada (NSERC) under grants 2015-04893 and 2019-00094. This support is gratefully acknowledged.

---

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

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

1 Introduction

In furniture and electronics retail, planning for home deliveries is a complex problem. Many have already addressed this problem from different perspectives, such as city logistics traffic [Coelho et al., 2016], time windows assignment [Agatz et al., 2011], or loading of the truck [Côté et al., 2014]. An issue receiving increasing attention recently is that of minimizing total costs by separating and synchronizing delivery from installation/assembly and having customers potentially be visited twice: once for the delivery, and once for the assembly of the products. In the literature, this problem has been addressed mainly by its two extreme cases:

1. a delivery is made, in which installation/assembly is performed by the same workers. This problem can be modeled as a vehicle routing problem with time windows (VRPTW) [Desaulniers et al., 2014]. When assuming that workers have different installation service times, one can model this problem as a vehicle routing problem with time windows and driver-specific times (VRPTWDST) [Schneider, 2016].
2. a delivery is made, but later a new crew arrives to perform installation/assembly. This problem can be modeled as a vehicle routing problem with multiple synchronization constraints (VRPMS) [Hojabri et al., 2018].

In this paper, we address a generalization of both cases, in which two fleets exist: one for the delivery and another for the installation. Moreover, installation can be performed by the deliverymen or by specialized installers. Three different categories of products (furniture, electronics and home appliances) are considered and each installer has different performances in terms of time required to install each type of product. This problem is thus called Delivery and Installation Routing Problem (DIRP). This paper is the first to deal with a delivery problem with such a flexible setting for installation services. We also evaluate our methods on these two extreme cases.

The DIRP is inspired by a real-world problem encountered by a logistic company delivering large furniture, home appliances and electronics products. This company's actual distribution strategy is to let its deliverymen perform some additional services needed to ensure that the delivered products are operational. For example, furniture must be assembled, home appliances need electric and electronic connections and may require some plumbing operations. Electronic products also need some electric connections and network configuration. These tasks require some skills that deliverymen do not necessarily have and are a source of delay in the delivery operations. The company also

has difficulty hiring qualified staff who are good in both delivery and installation. A potential solution to these issues is what we consider in this paper. The DIRP is then proposed to explore the different strategies where (i) deliverymen install all products, or (ii) where some specialized workers are used to install, or (iii) where deliverymen and installers are both used to install.

In the DIRP, one disposes of two distinct fleets. The first one performs deliveries and is composed of relatively large trucks. The second fleet is used to route the installation crew and is composed of small passenger vehicles, hence faster and cheaper. Different workers (deliverymen and installers) have different skills and thus different installation times. Deliverymen can perform the installation but are less skilled causing their installation times to be higher.

According to the classification of synchronization problems by Drexl [2012], the DIRP is a synchronization problem with temporal precedence constraints. The goal is to construct routes for two different types of vehicles while ensuring that the installation vehicle visits the customer location after the end of delivery if the installation has not been performed by the deliverymen.

Related problems are discussed next. Hojabri et al. [2018] studied the delivery of large items in which deliverymen and installers perform distinct activities and the main constraint is the synchronization of the two at a certain number of customer locations: the installer must visit the customer within ten units of time after the start of the delivery service. The authors proposed a constraint programming-based formulation of the problem before solving it with an adaptive large neighborhood search (ALNS) heuristic. The search power of ALNS combined with the constraint propagation of constraint programming (CP) allowed the authors to consider instances containing up to 200 customers. Bae and Moon [2016] studied a variant of the problem involving several depots. The authors determined a set of depots from which routes should start to satisfy both delivery and installation requests such that the fixed costs of the depots, delivery and installation vehicles, as well as transportation and labor costs, are minimized. Besides, a service level is considered, which implies that the arrival times of delivery and installation vehicles at a customer location must not exceed a maximum interval. A mixed integer linear model and a genetic algorithm (GA) were proposed. The authors also show the trade-off between having separate delivery and installation services versus the use of a single vehicle. When time windows are tight and high service levels are required, the integrated optimization of the two fleets performs better. Recently, Sun [2018] solved a similar problem in the electronics industry. An additional constraint to satisfy the expected service level is also considered. The author proposed a hybrid technique using an ant colony optimization (ACO) for the routing of the delivery vehicles and a GA for the routing of the installation vehicles. Computational experiments on randomly

generated instances with up to 150 customers show that the proposed hybrid algorithm outperforms two pure metaheuristic approaches.

The VRPTW and multiple deliverymen (VRPTWMD) [Pureza et al., 2012] is related to our problem as it is also relevant to situations where the service time is a function of the vehicle crew rather than being fixed for a given request. In order to reduce service times, the authors assigned more than one deliveryman to each route. They presented a mathematical programming formulation as well as a tabu search (TS) and an ACO. Alvarez and Munari [2017] later proposed a hybrid method combining a branch-price-and-cut algorithm with an iterated local search and a large neighborhood search metaheuristic to solve the VRPTWMD. In the same context, we can mention the VRPTWDST of Schneider [2016], where drivers have specific travel and service times in order to model their familiarity with the customers to visit. A TS metaheuristic is implemented with the main objective being the minimization of the number of vehicles used and a secondary objective being considered in case of a tie. Two alternative secondary objectives are independently addressed: minimization of traveled distance and of working duration. It is shown that considering the knowledge of drivers about the location of customers improves the efficiency of the routing. They observe benefits of the familiarity when drivers are familiar with customers geographically contiguous, or when a higher number of drivers are familiar with a larger region. Comparatively, a scenario where each driver is only familiar with a dedicated (smaller) region does not work quite well.

A problem involving the synchronization of two types of routes is introduced by Domínguez-Martín et al. [2018] in the driver and vehicle routing problem. In this situation, vehicles must depart from one depot and arrive at another one, while drivers should leave and return to the same depot and their routes cannot exceed a given duration. This requires a change of vehicle for the drivers in order to return to their base depots and synchronization of vehicles and drivers at some nodes. The problem was modeled as a VRP with two depots and two types of routes (one for drivers and the other for vehicles) and solved with a branch-and-cut algorithm.

Routing problems with synchronization constraints are well known in home health care (HHC) routing and scheduling problems. Some constraints are similar to those of the DIRP, notably temporal constraints linking routes of different care workers (in our case, deliverymen and installers). This constraint was addressed by Bredström and Rönnqvist [2008] solving a HHC staff scheduling problem modeled as a VRPTW with precedence and synchronization constraints. The authors developed a mixed integer programming model and an optimization-based heuristic to solve real size instances. A similar problem with a multi-criteria objective is solved by Rasmussen et al. [2012]. The criteria are the minimization of uncovered visits, maximization of home

care-visit preferences and minimization of the total travelling costs, presented here in decreasing order of priority. Temporal dependencies are modeled with generalized precedence constraints and the problem is solved using a branch-and-price algorithm. A mixed-integer programming formulation of a similar problem where customers can require more than two synchronized visits is proposed in Liu et al. [2019]. An ALNS algorithm is designed and tested on medium size instances of HHC problem and on large scale instances generated from VRPTW benchmarks. The proposed ALNS outperforms existing solution methods and finds new best known solutions within shorter computation time.

Other constraints relevant to our problem are the skills requirement. Each care worker or vehicle can have several skills as described in Bertels and Fahle [2006]. Patient's and care worker's preferences are considered; qualifications required to take care of a patient and each care worker's skills were also added to the model. A hybrid algorithm based on CP, linear programming (LP), TS and simulated annealing (SA) is developed to solve the problem. Computational experiments on 120 synthetic instances with up to 50 care workers, 200 patients, and 600 daily tasks show that the combination of CP and TS outperforms the SA and CP combination and the standalone CP, TS or SA. In the same context, Ciré and Hooker [2012] scheduled a set of medical services within a certain time window to be given to patients over a single-day horizon. The services are represented by a set of jobs, each requiring a qualification level defining the minimum skill a care worker must have to perform the job. A care giver also has a qualification level and can perform any job requiring a lower level. A heuristic adaptation of logic-based Benders decomposition is proposed, and the master problem is solved with a constructive greedy heuristic that is guided by a CP-based propagation of subproblem constraints and Benders cuts. This method is reported by the authors as being faster than a pure CP method. Finally, Rasmussen et al. [2012] have also incorporated particular skills that influence service times for the drivers in their model.

In this paper, we introduce the DIRP which generalizes some previous problems. Notably, our setting is flexible enough to allow the items to be installed by the deliverymen, or by more skilled specialized workers later. We propose a mixed integer linear programming model and an ALNS framework to solve this general distribution and installation problem. We also provide managerial insights about the best strategy to implement with respect to installation service.

The remainder of the paper is organized as follows. Section 2 provides a formal definition and introduces a mixed integer linear formulation. In Section 3, we present the ALNS designed for the problem. Computational experiments are presented in Section 4 and our conclusions follow in Section 5.

2 Problem description and mathematical formulation

In the distribution of large items, delivery is often followed by installation and the latter can be done either by a deliveryman or by a specialized worker. The DIRP is a variant of the heterogeneous VRPTW where two fleets are used to perform delivery and installation at customer locations. Different types of products are distributed, and each installer has a different skill for each type of product. A deliveryman performing an installation is less skilled thus with higher installation time. Moreover, the delivery fleet is composed of relatively large trucks while the installation fleet is composed of faster and cheaper passenger vehicles.

All customers receive deliveries, and some require an installation. Each customer requiring only delivery has his location represented by a single vertex. Customers requiring both delivery and installation have their location represented by two vertices, one for the delivery service and another for the installation one. Besides the classical constraints of the VRPTW, a solution to our problem must consider that:

1. Each customer is served by exactly one delivery vehicle;
2. A delivery vehicle must start its service at a customer within its time window;
3. An installation vehicle must start its service at a customer within its time window and once the delivery is completed;
4. Each vehicle can wait up to the opening of the time window of the customer if it arrives earlier;
5. Each customer requiring an installation is either served by exactly one delivery vehicle, or by exactly one delivery and one installation vehicle.
6. A service time is associated with each delivery and installation tasks.

Figure 1 shows an illustration of a solution of the DIRP. In this example, we have seven customers of whom five require an installation. We use two delivery and one installation vehicles. One delivery vehicle (bold solid line) serves three customers and performs installation at customer 6 location. The other delivery vehicle (thin solid line) serves the remaining customers and the installation vehicle installs at the customers who need an installation (dotted line). We notice on the figure that customers 3 and 7 do not require an installation.

We now introduce a formal definition for the DIRP. It is defined on a complete directed graph $G = (V, A)$ where $V = \{0, 1, ..., n + m + 1\}$ is the set of nodes with nodes 0 and $n+m+1$ indicating the depot, n being the number of

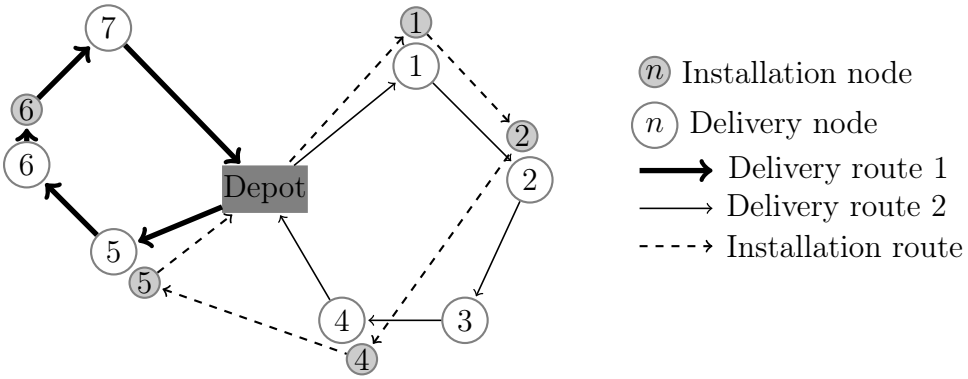


Figure 1: Solution of a DIRP with seven customers

customers and $m \leq n$ being the number of customers requiring an installation. $N = \{1, ..., n+m\}$ is the set of customer nodes and is divided in two subsets: D for the deliveries and I for installations. Let D^I be the subset of delivery nodes that require an installation ($D^I \subseteq D$). Each installation node $i \in I$ is linked to a delivery node $n(i) \in D^I$, with $l(j)$ being the installation node associated with delivery node j. The arc sets are $A = \{(i,j)|i,j \in V\}$, $A^D = \{(i,j)|i,j \in D \cup \{0, n+m+1\}\}$ and $A^I = \{(i,j)|i,j \in I \cup \{0, n+m+1\}\}$. The outcoming arc sets of node i are $\delta_D^+(i) = \{(i,j)|j \in D\}$ and $\delta_I^+(i) = \{(i,j)|j \in I\}$. The incoming arc sets of node i are $\delta_D^-(i) = \{(j,i)|j \in D\}$ and $\delta_I^-(i) = \{(j,i)|j \in I\}$. A time window $[a_i, b_i]$ is associated to each node i, with a_i and b_i being respectively the earliest and latest time the services must begin at node i.

Let K be the set of vehicles partitioned into the subset K^D of delivery vehicles and K^I of installation vehicles. For each arc (i,j), a cost c_{ijk} and a time t_{ijk} are known. They are respectively the transportation cost and the travelling time from customer i to customer j by vehicle k. Each vehicle is assigned to one particular deliveryman or installer. The capacity of each delivery vehicle is Q.

There are three product categories: (1) furniture, (2) home appliance and (3) electronic products. Each delivery node i has a total demand q_i which is the sum of the specific demand q_i^p of each product of type p that the customer will receive. The service time s_i^D for each delivery node i is standard for all delivery vehicles. The service time of an installation node i depends on the delivered products and the driver that performs it. Knowing that vehicle k performs the installation of product p, the service time will be s_{ikp}^I. All products of a customer must be installed by the same worker. Thus, the total installation time is $s_{ik}^I = \sum_{p \in P} s_{ikp}^I$ for vehicle k and customer i.

We formulate the problem with the following decision variables. Let binary

variable x_{ijk} be equal to 1 if a deliveryman travels from i to j using vehicle k, and 0 otherwise. Binary variable y_{ijk} equals 1 if an installer travels from i to j using vehicle k, and 0 otherwise. Binary variable z_i equals 1 if the installation service at customer i is done by a deliveryman, and 0 if it is done by an installer. The use of z_i allows our formulation to be flexible; by setting z_i equal to 1, we have a heterogeneous VRPTW model and by setting it to 0, it is a VRPTW model with temporal precedence and synchronization constraints. Finally, let S_i be the beginning time of service (delivery or installation) at node i. The objective function of the DIRP is to minimize the total transportation costs of all the routes traveled by the vehicle fleets. The mathematical model of the DIRP is as follows:

$$\min \sum_{k\in K^D} \sum_{(i,j)\in A^D} c_{ijk}x_{ijk} + \sum_{k\in K^I} \sum_{(i,j)\in A^I} c_{ijk}y_{ijk} \tag{1}$$

$$\sum_{j\in\delta_D^+(0)} x_{0jk} \leq 1 \qquad k \in K^D \tag{2}$$

$$\sum_{j\in\delta_I^+(0)} y_{0jk} \leq 1 \qquad k \in K^I \tag{3}$$

$$\sum_{k\in K^D} \sum_{j\in\delta_D^+(i)} x_{ijk} = 1 \qquad i \in D \tag{4}$$

$$\sum_{j\in\delta_D^-(i)} x_{jik} - \sum_{j\in\delta_D^+(i)} x_{ijk} = 0 \qquad i \in N,\ k \in K^D \tag{5}$$

$$\sum_{j\in\delta_I^-(i)} y_{jik} - \sum_{j\in\delta_I^+(i)} y_{ijk} = 0 \qquad i \in I,\ k \in K^I \tag{6}$$

$$\sum_{i\in D} q_i \sum_{j\in\delta_D^+(i)} x_{ijk} \leq Q \qquad k \in K^D \tag{7}$$

$$\sum_{k\in K^I} \sum_{j\in\delta_I^+(i)} y_{ijk} = 1 - z_i \qquad i \in I \tag{8}$$

$$S_j \geq S_i + s_i^D + \sum_{k\in K^D} t_{ijk}x_{ijk} - M\left(1 - \sum_{k\in K^D} x_{ijk}\right) (i,j) \in A^D, i \notin D^I \tag{9}$$

$$S_j \geq S_i + s_i^D + \sum_{k\in K^D} (s_{ik}^I + t_{ijk})x_{ijk} - M\left(2 - \sum_{k\in K^D} x_{ijk} - z_{l(i)}\right) (i,j) \in A^D, i \in D^I \tag{10}$$

$$S_j \geq S_i + \sum_{k\in K^I} (s_{ik}^I + t_{ijk})y_{ijk} - M\left(1 - \sum_{k\in K^I} y_{ijk}\right) \qquad (i,j) \in A^I \tag{11}$$

$$S_i \geq S_{n(i)} + s_{n(i)} \qquad i \in I \qquad (12)$$
$$a_i \leq S_i \leq b_i \qquad i \in N \qquad (13)$$
$$x_{ijk} \in \{0,1\} \qquad (i,j) \in A, k \in K^D \qquad (14)$$
$$y_{ijk} \in \{0,1\} \qquad (i,j) \in A, k \in K^I \qquad (15)$$
$$z_i \in \{0,1\} \qquad i \in I. \qquad (16)$$

The objective function (1) minimizes the sum of the transportation costs of all routes. Constraints (2) and (3) impose that a delivery/installation vehicle may be used at most once. Constraints (4) set that each customer is assigned to exactly one delivery route. Constraints (5) and (6) are degree constraints, and constraints (7) guarantee that the vehicle capacity is respected. Constraints (8) allow an installation to be performed by a deliveryman, and constraints (9)–(13) define the timing of the visits and time window at the nodes. Finally, constraints (14)–(16) define the nature and domain of the variables.

3 An ALNS heuristic for the DIRP

We now present our implementation of the ALNS, inspired from Ropke and Pisinger [2006]. It works by using sets of destruction and reconstruction heuristics or operators that compete to create a new solution by destroying and repairing a given solution. This metaheuristic proves itself to be highly efficient to solve various optimization problems including many VRP variants (Pisinger and Ropke [2007], Hemmelmayr et al. [2012], Naccache et al. [2018]).

3.1 Destroy operators

A destroy operator is used to change a solution by removing some nodes according to a certain criterion. This criterion is first applied to a delivery node and in case of successful removal, the installation node associated with it is also removed. We have derived one random and one related operator tailored for the DIRP.

3.1.1 Random delivery removal

The random removal operator selects c delivery nodes at random and removes them from the solution. If a delivery node selected has a linked installation node, it is also removed from its route.

3.1.2 Related delivery removal

This algorithm is inspired by the Shaw removal heuristic [Shaw, 1997] and adapted for the DIRP. Unlike Shaw [1997], we consider that the relatedness between two nodes is the euclidean distance between them, and we also decrease the time complexity by not sorting an array at each iteration but only once. This operator removes c delivery nodes related to each other and any installation node associated with them. As described in Algorithm 1, a delivery node $n(i)$ is randomly selected in line 1 and removed from the solution in line 2. In lines 3 and 4, an installation node linked to $n(i)$ is also removed. From lines 8 to 14, $c-1$ delivery nodes related to $n(i)$ are removed together with their installation associated nodes.

Algorithm 1 *Related removal for DIRP*

Input: L: List of all delivery nodes in the solution
c: number of delivery nodes to remove

```
1  Select a random node n(i) from L
2  Remove n(i) from the solution
3  if n(i) has an installation node i then
4  |  Remove i from the solution
5  end
6  Sort L in descending order of distance from n(i)
7  counter ← 0
8  while counter < c − 1 do
9  |  Node n(i) ← L[counter]
10 |  Remove n(i) from the solution
11 |  if n(i) has an installation node i then
12 |  |  Remove i from the solution
13 |  end
14 |  counter ← counter + 1
15 end
```

3.2 Repair operators

A repair (insert) operator is used to recreate a solution after the destruction step. For the DIRP we designed two insertion heuristics as repair operators. These operators repair a solution by iteratively inserting a delivery node into a delivery route, and if necessary an installation node into the same delivery route or into an installation route. When investigating the combined delivery and installation at a customer by a deliveryman, both nodes are merged, and a single node is inserted into the route.

Inserting a node into a route can cause the arrival time at subsequent nodes of the route out of their time windows and thus can make the route infeasible. Also, inserting a delivery node can modify the arrival times in the installation routes.

We used a preprocessing technique to check the feasibility of insertion into a route in $\mathcal{O}(1)$, hence decreasing the complexity of our operators. For each node i, we define the forward time slack (F_i) [Savelsbergh, 1992], which indicates how far the arrival time at node i can be shifted forward in time without causing the route to become infeasible. When inserting i before j, we compute the difference between the new arrival time at j and its actual arrival time. If this difference is greater than F_j then the route will be infeasible, and this position is ignored. With γ being the number of nodes in a route, 0 and $\gamma+1$ indicating the departure and arrival depot, s_i being the service time and the computed A_i, B_i, W_i being respectively the arrival time, beginning of service, and waiting time at node i, from Cordeau et al. [2004], we define the forward time slack as:

$$F_i = \min_{i \leq r \leq \gamma+1} \sum_{i \leq p \leq r} W_p + (b_r - B_r). \tag{17}$$

The waiting time is computed as $W_i = \max\{0, a_i - A_i\}$ for a delivery node i and as $W_{l(i)} = \max\{0, B_i + s_i - A_{l(i)}\}$ for its installation node $l(i)$. It is assumed that B_0 is the departure time and $B_{\gamma+1}$ the arrival time at the depot. Thus F_0 measures how far the departure of a vehicle can be shifted forward in time.

The slack computed with equation (17) is sufficient to check the feasibility in one of these three cases: when inserting the delivery and installation nodes of a customer in the same route, when inserting a customer who does not require installation service, or when inserting an installation node. However, it does not work in the case where delivery and installation nodes are on different routes. In this case, the value of F_i of the delivery node has to incorporate s_i and $F_{l(i)}$ and the violation of the latter is a sufficient condition to drop the insertion. The procedure to obtain F_i is as follows:

1. compute the arrival time at all delivery nodes of the route
2. compute the arrival time at all installation nodes of the route
3. compute $F_{l(i)}$ using equation (17)
4. compute F_i as:

$$F_i = \min\left\{ \min_{i<k\leq\gamma+1} \sum_{i<p\leq k} W_p + (b_k - B_k),\ b_i - B_i - s_i,\ F_{l(i)} - W_{l(i)} \right\} + W_i. \tag{18}$$

An illustration of the calculation of the slacks is shown in Figure 2. We consider three customers 1, 2 and 3 with customers 1 and 3 requiring an

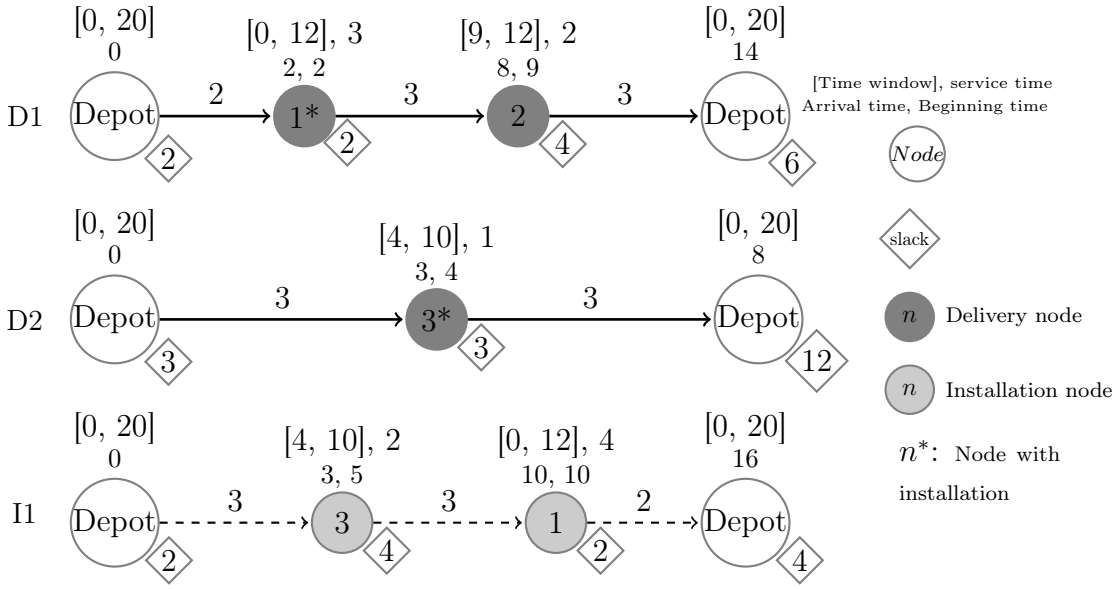


Figure 2: Computation of the forward time slacks

installation (nodes $l(1)$ and $l(3)$). Two deliverymen ($D1$ and $D2$) and one installer ($I1$) are used to serve them. $D1$ visits 1 and 2 at $t = 2$ and $t = 8$ and comes back to the depot at $t = 14$. $D2$ visits only 3 at $t = 4$. $I1$ starts its service at 3 and 1 at $t = 5$ and $t = 10$ and comes back to the depot at $t = 16$. The slacks for the installation nodes are first computed starting from the depot and are equal respectively to $\min\{12 - 10, 20 - 16\} = 2$ and $\min\{10 - 5, 2\} + 2 = 4$ for $l(1)$ and $l(3)$. Finally, $F_2 = \min\{12 - 9, 20 - 14\} + 1 = 4$, $F_1 = \min\{4, 12 - 2 - 3, 2\} = 2$ and $F_3 = \min\{20 - 8, 10 - 3 - 1, 4 - 2\} + 1 = 3$.

We note that if only equation (17) was used, $F_3 = \min\{10 - 3 - 1, 20 - 8\} = 6$, but an increment of 6 postpones the start of service at $l(1)$ out of its time window and then makes the route infeasible, confirming that (17) does not work in this case, justifying the use of (18).

Considering these procedures to check whether an insertion is feasible, our repair operators are described next.

3.2.1 Greedy sequential insertion

This operator consists of iteratively inserting a node in the cheapest possible route until all customers have been inserted or no more insertions are feasible. The steps of this procedure are showcased in Algorithm 2. In lines 2 and 3, a customer without installation is inserted in the best position among all delivery routes. If an installation is required, both delivery i and installation $l(i)$ nodes are inserted either in the same route or in different routes according

to the insertion with the cheapest cost. In the lines 5 and 6, we respectively investigate the best insertions of i and $l(i)$ in the same route and in different routes. From lines 7 to 11, we implement the insertions which have the least cost.

Algorithm 3 is used to determine the best pair of delivery and installation routes where to place i and $l(i)$. This algorithm starts by enumerating the list of all insertions of i ($l(i)$) in all delivery (installation) routes in line 1 (2). These lists are next sorted by increasing order of cost in the lines 3 and 4. Then, for each pair of insertion in a delivery and installation routes, the synchronization constraints (line 10) and the slack checking (line 16) are tested before saving the best insertion, cost and slack obtained so far (lines 17 to 21). Finally, the two insertions whose sum of the cost is the least are returned in line 23.

Algorithm 2 *Greedy sequential insertion for DIRP*

Input: L: List of all unassigned delivery nodes

```
1  for each node i in L
2      if i does not require an installation then
3          Insert i in the best position among all delivery routes
4      else
5          Delivery and installation by the same driver:
             Insert_1 ← best insertion of i among all delivery routes
6          Delivery and installation by different drivers:
             Insert_2 ← DeliverAndInstall(...)      // See Algorithm 3
7          if Cost(Insert_1) < Cost(Insert_2) and Insert_1 is feasible then
8              Apply Insert_1 and insert l(i) after i
9          else if Insert_2 is feasible then
10             Apply Insert_2
11         end
12     end
```

3.2.2 Regret-k insertion

In the regret heuristic [Potvin and Rousseau, 1993], a regret value is computed for all delivery nodes. The regret value is the cost difference between the best insertion position (the cheapest route) and the $(k-1)^{th}$ best route for $k > 2$. If $k = 2$, the regret value is the cost difference between the best and second-best insertion position. Algorithm 4 shows the outline of this operator adapted for the DIRP. First, from lines 1 to 13, all the best insertions of delivery (installation) nodes in all delivery (installation) routes are computed. At line 4, we find the best insertion for a node without installation. At line 6, a

Algorithm 3 *Deliver and install with different drivers*

Function `DeliverAndInstall()`:

Input:
i: delivery node to insert in a delivery route
$l(i)$: installation node to insert in an installation route
Output: Insertion $I_{best} = \{(P_i^{best}, P_{l(i)}^{best}, c_i^{best})\}$

$List\_del \leftarrow$ all insertions of i in delivery routes.
$List\_install \leftarrow$ all insertions of $l(i)$ in installation routes.
Sort $List\_del$ by increasing order of cost
Sort $List\_install$ by increasing order of cost
$best\_cost \leftarrow \infty$
$best\_slack \leftarrow 0$
for each *insertion* P_i *in* $List\_del$
  for each *insertion* $P_{l(i)}$ *in* $List\_install$
    $Arr_{l(i)}$: Arrival time at $l(i)$'s position.
    if $Arr_{l(i)} > b_i$ **then**
      Continue
    end
    $l(j)$: node subsequent to $l(i)$
     $Slack_{l(j)}$: slack of $l(j)$
     $Cur_{l(j)}$: actual arrival time at $l(j)$'s position.
    $Next_{l(j)} \leftarrow$ next arrival time at $l(j)$'s position
    $Diff_{l(j)} \leftarrow Next_{l(j)} - Cur_{l(j)}$
    if $diff_{l(j)} \leq Slack_{l(j)}$ ***And*** $best\_cost > Cost(P_i) + Cost(P_{l(i)})$ ***Or*** $best\_cost = Cost(P_i) + Cost(P_{l(i)})$ ***And*** $Slack_{l(j)} - diff_{l(j)} > best\_slack$ **then**
      $best\_cost \leftarrow \text{Cost}(P_i) + \text{Cost}(P_{l(i)})$
      $best\_slack \leftarrow Slack_{l(j)} - diff_{l(j)}$
      $c_i^{best} \leftarrow best\_cost$
      $P_i^{best} \leftarrow P_i$
      $P_{l(i)}^{best} \leftarrow P_{l(i)}$
    end
return I_{best}

delivery node and its installation node are inserted in the same route while they are inserted separately in line 7. From lines 8 to 12, the insertion with the least cost is stored. Then, lines 17 to 30 are repeated in a loop until all nodes are inserted in a route. The delivery node with the maximum regret value not yet inserted is selected between lines 17 and 26. It is then inserted if possible together with its installation node at line 28. The best insertions of the remaining nodes are finally updated in line 30.

3.3 ALNS algorithm

An outline of the ALNS implemented is provided in Algorithm 5. Starting from an initial solution (line 1) and until a stop criterion is met, a new solution $x^{'}$ is created at each iteration by destroying and reconstructing the precedent solution (line 8) using destroy d and repair r operators. In line 7, d and r are selected using a roulette wheel selection based on their respective weight ω. Given n operators in a destroy or repair set, each operator j within the set is selected with probability $\omega_j / \sum_{i=1}^{n} \omega_i$. From lines 9 to 17, according to the cost of $x^{'}$, any operator j involved has a score π_j which is updated. Initially set to 0, the score is incremented by θ_1 if $x^{'}$ is the new best one, or by θ_2 if it is only better than the previous one. A worse new solution can be accepted and in this case, the score is incremented by θ_3. The acceptance criterion of the ALNS is the same used for simulated annealing [Kirkpatrick et al., 1983]. A new solution whose cost value C' is worse than the cost C of the current solution is accepted with probability $e^{-\left(\frac{C'-C}{T}\right)}$, where T is the temperature that decreases at each iteration by a cooling rate c (line 19). The temperature is initially set to T_{init} and when it reaches a minimum threshold T_{min}, it is reset to T_{init}, in a procedure called reheating (line 21). Between lines 23 and 27, all operators weights are updated at every φ iterations using their scores which are then reset to zero.

4 Computational experiments

In this section, we present and discuss the results obtained in the computational experiments of the ALNS algorithm described in Section 3 and the branch-and-bound (B&B) applied to the model of Section 2. The different algorithms were coded in C++ and ran on a processor Intel 2.4 GHz Gold 6148 Skylake. For the exact method, we have used IBM Concert Technology and CPLEX 12.8 as the MIP solver. A single thread was used with a time limit of one hour. No benchmark instances for the DIRP exist so we have generated

Algorithm 4 Regret-k insertion for DIRP

Input: k: Number of routes involved in the computation of the regret value
L: List of all unassigned delivery nodes
U: List of all unassigned nodes
M: Matrix of insertions of all delivery nodes

```
for each node i in L
    for each delivery route d
        if i does not require an installation then
            M_id ← best insertion of n in d
        else
            Delivery and installation by the same driver:
             Insert_1 ← best insertion of n(i) among all delivery routes
            Delivery and installation by different drivers:
             Insert_2 ← DeliverAndInstall(...)        // See Algorithm 3
            if Cost(Insert_1) < Cost(Insert_2) then
                M_id ← Insert_1
            else
                M_id ← Insert_2
            end
        end
while U is not empty do
    Best: Best insertion at current iteration
    Max: Biggest regret computed at current iteration; Max ← −∞
    for each delivery node i in U
        Compute the regret value for i (r_j = cost of the best insertion in the
         j^th best route): regret ← ( Σ_{j=2}^{k} r_j ) - (k − 1)r_1
        Ins ← best insertion of i
        if Ins is feasible then
            if regret > Max or regret = Max and Cost of Ins < Cost of Best
             then
                Best ← Ins; Max ← regret
            end
        else
            Remove i and any l(i) from U
        end
    if Best is feasible then
        Apply Best
    end
    Update M for the remaining unassigned nodes
end
```

Algorithm 5 ALNS metaheuristic for DIRP

Input: D: Set of destroy operators; R: Set of repair operators
T_{init}: initial temperature; T_{min}: minimum threshold temperature
c: cooling rate
φ: Number of iterations before updating the parameters

Create an initial solution x; set $x^* = x$
Initiate score π_d and π_r for each destroy d and repair r operator
Initiate weight ω_d and ω_r for each destroy d and repair r operator
T $\leftarrow T_{init}$
while *stopping criterion is not met* **do**
    Generate a random number q of delivery nodes to remove
    Select destroy d and repair r operators using a roulette wheel based on their weights
    Create solution $x^{'}$
        $x^{'} \leftarrow r\,(d(x, q))$
    if *Cost($x^{'}$) < Cost(x^*)* **then**
        $x^* \leftarrow x \leftarrow x^{'}$
        Increase the scores of d and r by θ_1
    else if *Cost($x^{'}$) < Cost(x)* **then**
        $x \leftarrow x^{'}$
        Increase the scores of d and r by θ_2
    else if *$x^{'}$ can be accepted* **then**
        $x \leftarrow x^{'}$
        Increase the scores of d and r by θ_3
    end
    $T \leftarrow T \times c$
    if $T < T_{min}$ **then**
        $T \leftarrow T_{init}$
    end
    if *number of iterations is multiple of* φ **then**
        for each *operator i*
            Update the weight ω_i using the score π_i and the number of times i has been used during the last φ iterations
            $\pi_i \leftarrow 0$
    end
end
return x^*

the dataset for the experiments as follows.

4.1 Generation of instances

We have generated test instances for the DIRP using the VRPTW instances of Solomon [1987]. From these instances, the capacity of the vehicles, the coordinates, demand and time window of the customers were kept, and we generated installation and delivery service times randomly. The dataset contains three different classes of instances: class R with randomly located customers, class C with clustered customers, and class RC where customers are both randomly located and clustered. Each class contains instances of type 1 and 2 which have respectively tight and large time windows. Given the high installation service time of the deliverymen, the time windows of some of Solomon's instances are too tight for the DIRP so we only use the 27 instances of type 2. The upper time window of each customer node with an installation is updated by subtracting its delivery service time.

We introduced speed and cost parameters for each vehicle. Delivery vehicles take twice the time needed by an installation vehicle to travel between a pair of nodes. A cost C_{del} associated with the usage of a delivery vehicle means that the travel distance of a deliveryman between a pair of nodes is C_{del} times the distance traveled by an installer. Each instance file contains the number of vehicles available, the cost and speed of each vehicle, and the installation efficiency of the different drivers. We create 18 sets based on the performance of the deliverymen when installing, on the percentage of customers having installations and on the cost of using a delivery vehicle. Efficiency of the installers are generated between [85%, 100%]. The other characteristics are the following:

- Number of customers: 15, 25, 50, 100
- Speeds of each delivery/installation vehicle: 1/2
- C_{del}: 2, 5 and 10
- Efficiency (η) of the deliverymen when installing: [30%, 50%], [60%, 80%]
- Percentage ($\%Inst$) of customers requiring installation: 20%, 50% and 75%.

In total we have generated 1944 instances that can be obtained upon request.

Table 1: Quality of the solution when varying the number of iterations

# of iterations	Dev (%)	Time (s)
25,000	1.31	49.03
50,000	1.06	98.01
100,000	0.61	200.15
200,000	0.14	404.68
400,000	0.00	806.76

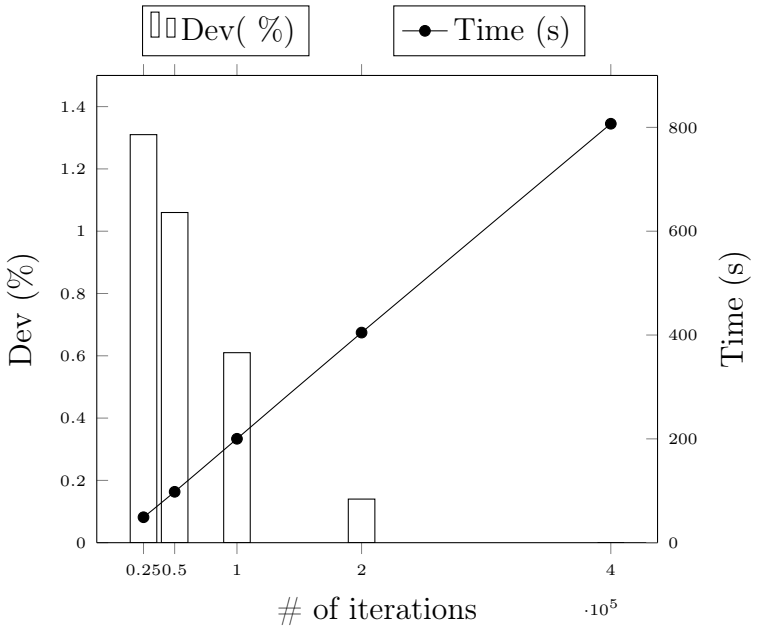
4.2 ALNS parameters tuning

In this section, we have selected the appropriate parameters to have a good performance of the ALNS for the DIRP. The first parameter chosen is the number of iterations allowed to run before stopping. We selected a set of 27 instances with 100 customers and performed tests with 25,000, 50,000, 100,000, 200,000 and 400,000 iterations, each run once. The results are shown in Table 1 and illustrated in Figure 3. For each instance and each iteration value, the average deviation ($Dev(\%)$) between its solution and the best known solution (BKS) is calculated. As shown in Figure 3, the gap decreases slowly from 25,000 to 100,000 iterations. From 100,000 to 200,000, we observe a large decrease of the gap and the best gap is obtained for 400,000 iterations. As for the execution time, it almost doubles with each variation of iterations between 25,000 and 400,000. We choose 200,000 as our maximal value of iterations as it gives the best trade-off between quality of the solutions and runtime of the algorithm.

For the next tuning phase, we have selected the insertion and removal operators used in the ALNS. In Section 3.1, we introduced the two destroy operators we intended to use. Sequential heuristics (seq), regret-3, regret-m were selected as well as repair operators. We obtained two more repair operators (regret-3n and regret-mn) by adding some randomness to the regret heuristics. The aim of this analysis is to determine the best combinations of operators adapted to our problem while considering the different interactions between them. Having two removal and five repair operators, thus 2^7 combinations, we applied a fractional factorial design [Montgomery, 2009] to reduce the number of combinations to $3 \times 2^3 = 24$.

For each combination, we report in Table 2 the average runtime and deviation computed with respect to the BKS. The best deviation is obtained for all operators but the random removal. Adding the random removal to this combination or switching the related removal by the random removal degraded not only the gap but also the average running time. We can conclude that using all operators does not guarantee a better solution and that the inter-

Figure 3: Effect of the number of iterations on the performance of the ALNS



action between operators is important. The performance of the ALNS does not necessarily take into account the number of operators used but rather the appropriate combination adapted for the problem.

The operators selected for the DIRP were then all operators but the random removal.

4.3 Results for the DIRP

The computational results of our instances solved with the ALNS and the B&B algorithms are summarized in Tables 3, 4 and 5. We set 3600 seconds of running time and 200,000 iterations as stop criteria for the ALNS. To speed up the search, we started with an initial solution created with the sequential insertion. The best solution obtained in 10 runs is then given as upper bound (UB) to CPLEX to solve the B&B for 3600s. The BKS is then established as either the best of the 10 executions of our ALNS or the UB from CPLEX, if it was able to improve the ALNS solution. In the following tables, *%Inst* is the percentage of customers with an installation, η is the installation efficiency of deliverymen, C_{del} is the cost of using a delivery vehicle while *Gap best (%), Gap avg (%)* report the average gap between the BKS and respectively the best and average solution found by the ALNS within ten runs. The average number of deliverymen and installers used and the runtime in seconds are reported in columns Del, Ins and $Time(s)$.

Table 2: Quality of the solution when combining the operators

Combination of operators	Dev (%)	Time(s)
seq, regret-3, regret-m, regret-3n, regret-mn, related	1.05	640.92
seq, regret-m, regret-mn, random, related	1.20	675.07
seq, regret-3, regret-m, regret-3n, regret-mn, random, related	1.23	646.73
seq, regret-3, regret-m, regret-3n, regret-mn, random	1.29	647.51
seq, regret-m, regret-mn, random	1.29	689.39
seq, regret-m, regret-mn, related	1.35	663.37
regret-m, regret-3n, random	1.44	728.61
regret-3, regret-mn, random, related	1.46	714.90
regret-3n, regret-mn, random, related	1.47	712.84
regret-3, regret-m, random, related	1.48	711.87
regret-m, regret-3n, random, related	1.48	718.56
regret-3, regret-mn, related	1.55	706.48
regret-3, regret-m, related	1.55	706.97
regret-3n, regret-mn, related	1.55	709.82
regret-m, regret-3n, related	1.60	706.79
regret-3, regret-m, random	1.66	731.21
regret-3, regret-mn, random	1.66	723.32
seq, regret-3, regret-3n, related	1.66	527.98
seq, regret-3, regret-3n, random, related	1.68	533.89
regret-3n, regret-mn, random	1.70	732.33
seq, random, related	1.82	134.58
seq, related	1.91	132.58
seq, random	1.92	133.68
seq, regret-3, regret-3n, random	1.92	532.74

Table 3 shows the average results for the heuristic. There is only a slight difference between *Gap best* and *Gap avg*, which attests of the robustness of the ALNS. The average runtime is good for instances with up to 100 customers and it also appears that CPLEX has not significantly improved the ALNS solutions given that *Gap best* is close to zero.

Table 4 contains the results for optimal instances with the number of those instances solved with the B&B and ALNS in the columns *Opt* and *Opt ALNS*. Of all the instances, 24.4% (475 over 1944) are solved optimally with CPLEX. Our ALNS also found a large number (347) of those solutions. CPLEX solved these instances in 2311 seconds in average (*Time(s)*), whereas ALNS found feasible or optimal solutions in at most 27 seconds, showing that the proposed heuristic fulfilled its role to find a good solution for a DIRP instance in short time. *Gap(%)* is the average gap between the UB and lower bound (LB) for unsolved instances and is about 24.3%.

For each instance, several combinations of $\%Inst$, C_{del} and efficiency of the

Table 3: Summary results for the ALNS

%Inst	C_{del}	$\eta \in [30\%, 50\%]$					$\eta \in [60\%, 80\%]$				
		Gap best (%)	Gap avg(%)	Del	Ins	Time (s)	Gap best(%)	Gap avg(%)	Del	Ins	Time (s)
20%	2	0.07	0.19	3.3	0.2	15.5	0.08	0.24	3.2	0.1	15.6
	5	0.06	0.20	3.3	0.3	15.6	0.08	0.24	3.2	0.2	15.6
	10	0.05	0.16	3.2	0.4	15.6	0.08	0.24	3.2	0.3	15.6
50%	2	0.13	0.41	3.5	0.4	27.9	0.10	0.28	3.4	0.2	28.1
	5	0.09	0.31	3.4	0.7	28.5	0.06	0.24	3.3	0.4	28.5
	10	0.07	0.29	3.3	0.9	29.0	0.03	0.22	3.2	0.6	28.7
75%	2	0.12	0.52	3.7	0.6	37.8	0.03	0.31	3.6	0.3	38.1
	5	0.10	0.41	3.5	1.0	39.3	0.08	0.33	3.5	0.7	39.0
	10	0.08	0.40	3.4	1.3	40.5	0.07	0.35	3.3	1.0	39.9
Average		0.08	0.32	3.4	0.6	27.7	0.07	0.27	3.3	0.4	27.7

deliverymen are tested. For instances solved optimally with all these combinations, we compute the average number of deliverymen and installers used in Del and Ins. The same number of deliverymen is used and an increase of the number of installations induces an increase in the required number of installers. As expected, the number of installers is inversely proportional to the efficiency and cost of the deliverymen especially when a huge number of installations are required for the latter case.

In Table 5, we present the number of workers used with respect to the different classes of instances. The number of instances solved for all efficiency parameters is reported in column Nb. From the results, clustered customers needed more installers followed by the customers located randomly. Customers of class RC do not appear to be served by installers even with a high value of $\%Inst$, but only a few number of instances of this class are solved optimally.

The managerial insight derived from the resolution of the DIRP is discussed next. The economic potential of using two fleets of installers and deliverymen (who can install) is then evaluated as a distribution strategy. We consider different scenarios by varying the number of customers with installation, the performance of the deliverymen when installing and the cost of using a delivery vehicle. Furthermore, we compare the total cost of those obtained for the existing distribution strategies where deliverymen are in charge of all installations or where there is a dedicated fleet for the installation. Through Tables 4 and 5, we found that using more efficient deliverymen reduces the number of installers needed, thus decreasing services cost. Also, applying the DIRP is

Table 4: Summary results for optimal solutions

		$\eta \in [30\%, 50\%]$						$\eta \in [60\%, 80\%]$					
%Inst	C_{del}	Opt	Opt ALNS	Gap (%)	Del	Ins	Time (s)	Opt	Opt ALNS	Gap (%)	Del	Ins	Time (s)
20%	2	30	23	22.72	2.4	0.4	2006.8	35	27	22.53	2.4	0.3	2217.7
	5	29	22	22.47	2.4	0.4	2413.2	28	23	21.33	2.4	0.3	2402.6
	10	28	24	22.78	2.4	0.4	2498.2	28	26	21.53	2.4	0.3	2189.0
50%	2	31	21	26.53	2.4	0.5	2423.6	31	22	25.24	2.4	0.4	2304.7
	5	28	17	25.18	2.4	0.7	2402.0	27	16	24.22	2.4	0.4	2264.0
	10	26	16	24.30	2.4	0.8	2547.3	23	16	23.06	2.4	0.5	2478.0
75%	2	26	19	27.69	2.4	0.6	2555.7	23	18	26.35	2.4	0.4	2378.8
	5	21	14	26.52	2.4	0.9	2425.6	21	15	25.22	2.4	0.8	2270.5
	10	20	12	25.09	2.4	1.0	2535.7	20	16	24.82	2.4	0.9	1153.5
Average				24.8	2.4	0.6	2413.6			23.9	2.4	0.5	2208.5

more relevant for clustered followed by random located customers.

We obtained different costs by solving three exact models with various values of z_i. With $z_i = 1$ (*model 1*), we force the deliverymen to install all commodities, $z_i = 0$ (*model 2*) compels them to only deliver, while $z_i \in \{0, 1\}$ (DIRP) makes the model decide to let them install or not. For all instances solved optimally regardless of the model used, Tables 6 and 7 report the average costs in columns *Mod1*, *Mod2* and *DIRP* and n is the number of customers served.

From these tables, average costs show that it is better to use two fleets of vehicles with deliverymen allowed to install. In the case where one cannot apply the DIRP, allowing the deliverymen to install all products is a good choice regardless of the efficiency for a small number of customers and low to medium cost of usage. Even when the number of installations increases, C_{del} is low and the efficiency is high, this choice still seems to be a good one. On the other hand, using two distinct fleets is preferable in situations where deliverymen have low efficiency and their cost range from medium to high or when their efficiency and cost are both high.

In the next subsection, the performance of both B&B and ALNS algorithms are assessed on the VRPMS [Hojabri et al., 2018] and VRPTWDST [Schneider, 2016]. These two related problems are embedded in the general delivery and installation problem solved by the DIRP.

Table 5: Summary results for optimal solutions per instance classes

		C					R					RC			
			[30%, 50%]		[60%, 80%]			[30%, 50%]		[60%, 80%]			[30%, 50%]		[60%, 80%]	
%Inst	C_{del}	Nb	Del	Ins	Del	Ins	Nb	Del	Ins	Del	Ins	Nb	Del	Ins	Del	Ins
	2	13	1.9	0.6	1.8	0.5	12	2.6	0.0	2.6	0.0	4	2.3	0.0	2.3	0.0
20%	5	13	1.9	0.6	1.8	0.5	11	2.5	0.1	2.5	0.1	4	2.3	0.0	2.3	0.0
	10	12	1.8	0.8	1.8	0.5	11	2.5	0.1	2.5	0.1	4	2.3	0.0	2.3	0.0
	2	12	2.2	0.9	2.0	0.8	12	2.9	0.0	2.9	0.0	4	2.3	0.0	2.3	0.0
50%	5	12	2.1	1.1	2.0	0.8	9	2.9	0.4	2.9	0.0	4	2.3	0.0	2.3	0.0
	10	11	2.0	1.2	1.9	1.0	8	2.3	0.6	2.3	0.3	4	2.3	0.5	2.3	0.0
	2	8	2.1	1.3	2.1	0.8	9	2.8	0.1	2.7	0.0	4	2.3	0.0	2.3	0.0
75%	5	8	2.3	2.0	2.3	1.8	6	2.5	0.5	2.5	0.5	2	2.0	0.0	2.0	0.0
	10	10	2.0	1.9	2.0	1.7	6	2.7	0.7	2.7	0.7	2	2.0	0.0	2.0	0.0
Average			2.0	1.1	2.0	0.9		2.6	0.2	2.6	0.1		2.2	0.1	2.2	0.0

Table 6: Comparison between different distribution strategies where delivery-men efficiency are within [30%, 50%]

		20%				50%				75%			
n	C_{del}	Opt	Mod1	Mod2	DIRP	Opt	Mod1	Mod2	DIRP	Opt	Mod1	Mod2	DIRP
	2	21	463.4	533.0	461.1	20	490.9	593.9	480.4	14	503.4	637.6	486.8
15	5	21	1158.5	1216.3	1146.6	17	1196.6	1245.2	1143.3	15	1275.5	1340.5	1190.2
	10	21	2304.9	2343.1	2274.8	19	2368.5	2339.6	2243.3	14	2557.4	2456.8	2315.7
	2	7	623.7	684.5	610.6	6	712.3	830.7	666.4	4	799.4	922.3	716.3
25	5	6	1639.8	1647.2	1574.8	6	1780.8	1762.8	1603.4	3	2037.4	1776.7	1607.1
	10	5	3504.5	3395.8	3327.3	5	3483.5	3143.3	3003.3	3	4074.8	3269.1	3109.6
	2	1	934.8	875.9	856.2	1	1206.1	992.2	951.9	2	1512.5	1668.6	1326.4
50	5	1	2337.0	1961.3	1941.6	2	3503.4	3257.2	3018.9	1	3561.3	2206.3	2137.7
	10	1	4674.1	3770.3	3750.6	1	6030.4	3886.5	3855.2	1	7122.7	4015.3	3947.0
	2	1	1760.7	1467.6	1467.6	0	-	-	-	0	-	-	-
100	5	1	4401.7	3234.2	3234.2	0	-	-	-	0	-	-	-
	10	1	8803.3	6178.6	6178.6	0	-	-	-	0	-	-	-
Average			1575.9	1552.7	1486.5		1581.5	1560.6	1443.5		1705.7	1610.8	1453.3

Table 7: Comparison between distribution strategies where deliverymen efficiency are within [60%, 80%]

		20%				50%				75%			
n	C_{del}	Opt	Mod1	Mod2	DIRP	Opt	Mod1	Mod2	DIRP	Opt	Mod1	Mod2	DIRP
	2	21	461.6	533.0	459.8	20	465.1	585.4	460.7	15	474.2	628.2	468.9
15	5	20	1138.7	1200.0	1127.9	18	1120.2	1215.4	1095.8	13	1227.6	1339.1	1178.9
	10	20	2277.4	2322.8	2250.7	17	2272.6	2328.6	2209.2	14	2406.5	2415.4	2267.4
	2	6	640.9	715.1	632.2	6	673.0	831.4	654.6	5	750.5	951.2	707.6
25	5	6	1602.1	1647.2	1564.3	6	1682.6	1764.5	1593.7	5	1876.2	1940.4	1702.2
	10	6	3204.3	3200.8	3117.9	4	3437.4	3350.6	3192.4	4	3702.5	3421.9	3208.8
	2	3	1075.9	1196.3	1054.4	1	990.8	954.0	883.3	2	1379.1	1677.3	1315.3
50	5	1	2258.4	1961.3	1924.2	1	2476.9	2039.4	1968.7	1	2922.3	2263.4	2184.0
	10	1	4516.7	3770.3	3733.2	1	4953.8	3848.3	3777.7	1	5844.7	4138.8	4059.3
	2	1	1481.6	1424.6	1380.1	0	-	-	-	0	-	-	-
100	5	1	3704.0	3191.3	3146.7	0	-	-	-	1	5092.2	3733.9	3523.6
	10	1	7408.1	6135.7	6091.1	0	-	-	-	1	10184.5	6678.3	6474.0
	Average		1532.1	1552.9	1477.6		1410.9	1472.8	1343.0		1798.5	1757.4	1578.3

4.4 Results for the VRPMS instances set

Synchronization of two types of vehicles is the main constraint in the VRPMS. This is in line with our exact model when setting the binary variable z_i to 0 or when constraining the repair operators of the ALNS not to insert installation nodes in delivery routes. We adapt our model to the VRPMS by constraining an installer to start its service at most ten units of time after the delivery service time. The 684 instances used by Hojabri et al. [2018] and derived from Solomon [1987] and Homberger and Gehring [1999] are solved with our ALNS with a stopping criterion of 200,000 iterations. The best solution obtained in ten runs for each instance is next provided to CPLEX with a time limit of one hour to compute the LB and UB. Computational results are reported in Tables 8 and 9. In these tables, n is the number of customers, *Syn* the number of customers with synchronized installations, *Del* and *Ins* the number of deliverymen and installers used. *Gap heur (%)* contains the gap between the UB and the best solution found by a heuristic and *Time(s)* reports the computational time.

Table 8 contains the results for all instances while Table 9 reports only the instances solved to optimality by CPLEX. From 133 optimal solutions, our ALNS found 112 and Hojabri et al. [2018] found 42. The average gap for the unsolved instances by the exact model (*Gap(%)*) is 24.98%. Our heuristic method outperforms the ALNS of Hojabri et al. [2018] for all instances in both the value of the solutions and in the runtime. We have a much better runtime as we used fast heuristics for repairing purposes instead of the time-consuming

Table 8: Performance of the ALNS on the VRPMS instances

		Hojabri et al. [2018]				ALNS				
n	Syn	Gap heur(%)	Del	Ins	Time (s)	Gap best(%)	Gap avg(%)	Del	Ins	Time (s)
	2	1.70	3.4	1.0	13,788.7	0.00	0.03	3.2	1.0	3.5
25	7	2.76	3.4	2.0	8,255.0	0.08	0.09	3.3	1.9	6.4
	13	2.63	3.4	2.5	7,762.3	0.08	0.08	3.2	2.4	11.6
	3	4.35	5.7	1.2	25,629.1	0.07	0.13	5.2	1.1	11.8
50	13	5.15	5.9	2.9	20,942.7	0.00	0.08	5.3	2.7	24.9
	25	5.92	5.9	3.9	20,339.2	0.05	0.11	5.3	3.7	52.0
	5	6.40	9.8	1.7	80,777.8	0.00	0.51	8.6	1.4	42.3
100	25	7.65	10.0	4.4	84,518.0	0.00	0.43	8.9	4.1	98.0
	50	9.76	10.0	6.7	78,931.0	0.01	0.37	8.8	6.1	231.4
	10	10.19	14.4	2.5	137,220.1	0.00	0.57	12.7	2.4	112.3
200	50	10.43	14.2	6.2	144,486.2	0.00	0.70	13.0	5.6	321.8
	100	14.04	14.5	9.5	141,521.5	0.01	4.11	13.2	8.6	753.4
Average		6.84	8.5	3.8	65,038.8	0.03	0.62	7.6	3.5	143.6

CP model applied by these authors. The utilization of forward time slack helps us also further decrease the time complexity of our insertion heuristics.

We noticed some issues with five solutions of Hojabri et al. [2018] which are better than the LB found with CPLEX. These instances are all from the class C ($C101, C105, C106, C201$) with $n = 25$ and $Syn = 2$ for the first three, $n = 50$ and $Syn = 3, 13$ for the last one. Furthermore, of the 684 instances, ten identical pairs were found. These instances are mainly for the class C with $C102$ equal to $C103$ for $n = 25$ and $Syn = 2, 7, 13$, $n = 50$ and $Syn = 3, 13, 25$, $n = 100$ and $Syn = 5, 25, 50$. The last pair is $RC2\_2\_9$ with $RC2\_2\_10$ [Homberger and Gehring, 1999] for $n = 200$ and $Syn = 50$. We contacted the authors who have confirmed these issues.

4.5 Results for the VRTWDST instances set

In the VRPTWDST, Schneider [2016] solved a delivery problem where drivers have specific travel and service times according to their familiarity with the customers. We constrained our ALNS to only use the delivery fleet and solved the VRPTWDST benchmark instances generated with the random and cluster mechanisms. We summarize the computational results obtained in Tables 10

Table 9: Summary of optimal solutions for VRPMS instances set.

		Hojabri et al. [2018]				ALNS				Exact method				
n	Syn	Gap heur(%)	Del	Ins	Time (s)	Gap heur(%)	Del	Ins	Time (s)	Opt	Gap (%)	Del	Ins	Time (s)
	2	1.26	3.3	1.0	18,604.2	0.00	3.2	1.1	3.1	27	17.06	3.2	1.1	357.6
25	7	1.96	3.4	2.0	6,628.1	0.16	3.3	2.0	5.4	28	17.59	3.3	2.0	459.3
	13	1.78	3.2	2.4	5,790.6	0.07	3.2	2.3	8.9	21	22.25	3.2	2.4	209.3
	3	2.41	4.8	1.2	44,038.9	0.38	4.6	1.3	8.1	11	24.61	4.7	1.6	140.5
50	13	0.62	5.1	3.1	15,581.1	0.00	4.9	3.1	14.6	10	23.38	4.9	3.1	336.4
	25	1.47	5.4	4.2	12,731.3	0.34	5.2	4.1	26.5	9	28.13	5.2	4.3	375.4
	5	2.53	8.9	2.0	25,202.4	0.00	8.4	1.4	22.1	7	23.96	8.4	1.4	9.2
100	25	2.67	8.5	5.0	34,233.8	0.00	8.2	5.0	62.5	6	24.20	8.2	5.0	11.2
	50	4.34	8.5	6.5	32,725.6	0.11	8.2	6.2	165.2	6	28.11	8.2	6.3	31.4
	10	4.31	22.0	4.3	33,484.2	0.04	21.0	4.0	32.5	3	25.32	21.0	4.0	149.5
200	50	6.31	17.5	7.5	117,215.4	0.13	15.5	7.0	142.8	2	27.31	15.0	7.0	84.5
	100	5.16	18.7	12.3	54,023.2	0.10	17.0	12.0	252.9	3	29.42	16.7	11.7	322.5
Average		2.05	5.5	2.7	19,845.5	0.11	5.2	2.7	26.7		24.98	5.2	2.7	279.0
Solved			42/684				112/684				133/684			

and 11. Here, Γ is a familiarity factor used to compute the travel and service time for familiar drivers. Of the 560 instances, our ALNS found 165 best known and 256 new best solutions. With the exact method, we solved 64 instances optimally, while both heuristics found 54 optimal solutions each. The significantly better average *Gap heur* and runtime of the ALNS show that it outperforms the TS used by the author.

5 Conclusion

This paper presents an exact model and an ALNS heuristic for the Delivery and Installation Routing Problem. The problem investigates a distribution strategy where two fleets of installers and deliverymen who can install are used in order to deliver and install furniture, home appliances, etc. Several instances generated based on the efficiency of the deliverymen when installing, on the percentage of customers having installations, and on the cost of using a delivery vehicle are used to assess the value of the DIRP as a novel delivery method. Computational results show that while it is better to apply the proposed method, it is also good to make deliverymen install all products for companies with a small number of customers with few installations, regardless of their efficiency or costs. This is also the case where the number of installation increases and the deliverymen have low cost and high installation performance. Regarding using one fleet of deliverymen, having two distinct

Table 10: Results of the VRPTWDST instances generated with the Random mechanism when minimizing the distance

		Schneider [2016]			ALNS			Exact	
Γ		Gap heur(%)	Opt	Time (s)	Gap heur(%)	Opt	Time (s)	Gap (%)	Opt
0.1	C1	0.00	3/9	63.6	0.00	3/9	26.1	30.05	3/9
	C2	0.00	3/8	46.0	0.00	3/8	35.5	5.35	3/8
	R1	1.09	0/12	43.1	0.14	0/12	29.8	33.69	1/12
	R2	2.15	0/11	62.4	0.30	0/11	48.2	21.69	0/11
	RC1	2.01	0/8	48.2	0.54	0/8	27.0	42.60	0/8
	RC2	5.37	0/8	30.6	0.23	0/8	41.1	30.50	0/8
0.2	C1	0.00	3/9	39.9	0.00	3/9	26.1	30.93	3/9
	C2	0.00	3/8	59.5	0.01	3/8	36.6	6.22	3/8
	R1	2.55	0/12	51.1	0.13	0/12	29.7	33.32	1/12
	R2	4.96	0/11	47.5	0.46	0/11	52.3	21.01	0/11
	RC1	2.38	0/8	54.1	0.64	0/8	27.1	42.36	0/8
	RC2	6.83	0/8	34.0	0.68	0/8	41.6	30.18	0/8
0.3	C1	0.00	3/9	74.9	0.00	3/9	26.5	32.57	3/9
	C2	0.00	3/8	44.7	0.00	3/8	36.2	6.77	3/8
	R1	2.35	0/12	39.5	0.48	0/12	30.1	32.86	1/12
	R2	8.29	0/11	69.6	0.30	0/11	49.0	20.43	0/11
	RC1	3.66	0/8	41.8	0.23	0/8	27.2	42.15	0/8
	RC2	6.77	0/8	25.8	0.73	0/8	37.0	29.45	0/8
0.4	C1	0.00	3/9	76.6	0.00	3/9	26.8	34.03	3/9
	C2	0.00	2/8	39.6	0.06	2/8	34.7	6.61	2/8
	R1	2.43	0/12	37.5	0.44	0/12	29.8	32.88	1/12
	R2	8.11	0/11	73.5	0.62	0/11	48.1	20.32	0/11
	RC1	4.63	0/8	30.6	0.09	0/8	27.1	41.85	0/8
	RC2	14.95	0/8	29.4	0.37	0/8	39.1	29.25	0/8
0.5	C1	0.00	3/9	67.5	0.02	3/9	26.5	37.68	3/9
	C2	0.00	2/8	53.7	0.00	2/8	35.8	6.89	2/8
	R1	4.18	0/12	39.5	0.60	0/12	29.3	32.62	1/12
	R2	6.66	0/11	59.9	0.98	0/11	48.7	19.98	0/11
	RC1	5.42	0/8	50.9	0.00	0/8	25.1	41.66	0/8
	RC2	12.09	0/8	24.0	0.32	0/8	43.0	28.82	0/8
Average		3.56		52.8	0.29		35.0	24.97	
Solved			28/280			28/280			33/280

Table 11: Results of the VRPTWDST instances generated with the Cluster mechanism when minimizing the distance

		Schneider [2016]			ALNS			Exact	
Γ		Gap heur(%)	Opt	Time (s)	Gap heur(%)	Opt	Time (s)	Gap (%)	Opt
0.1	C1	0.00	3/9	48.2	0.00	3/9	25.1	30.01	3/9
	C2	0.00	3/8	77.7	0.13	3/8	32.6	5.21	3/8
	R1	0.81	0/12	30.7	1.25	0/12	28.6	33.65	1/12
	R2	0.44	0/11	53.6	0.32	0/11	50.4	21.65	0/11
	RC1	1.95	0/8	53.8	1.72	0/8	26.9	42.81	0/8
	RC2	4.95	0/8	23.1	0.49	0/8	41.7	30.05	0/8
0.2	C1	0.00	3/9	62.5	0.00	3/9	25.2	32.34	3/9
	C2	0.00	3/8	61.2	0.00	3/8	33.4	6.14	3/8
	R1	1.89	0/12	38.7	0.74	0/12	29.9	32.73	1/12
	R2	7.00	0/11	55.6	0.36	0/11	50.4	20.65	0/11
	RC1	1.01	0/8	43.5	1.09	0/8	27.0	41.37	0/8
	RC2	8.55	0/8	40.3	0.45	0/8	34.3	29.40	0/8
0.3	C1	0.00	3/9	52.5	0.00	3/9	26.1	33.38	3/9
	C2	0.00	2/8	52.3	0.00	2/8	32.2	5.86	2/8
	R1	0.73	0/12	43.9	1.07	0/12	30.4	31.22	1/12
	R2	9.73	0/11	41.4	0.45	0/11	51.2	19.67	0/11
	RC1	5.32	0/8	36.8	0.64	0/8	27.4	40.98	0/8
	RC2	9.57	0/8	51.8	0.47	0/8	35.9	28.51	0/8
0.4	C1	0.00	3/9	67.1	0.00	3/9	25.3	34.55	3/9
	C2	0.00	2/8	53.6	0.00	2/8	34.5	6.42	2/8
	R1	1.30	0/12	44.5	1.22	0/12	30.8	30.91	1/12
	R2	7.45	0/11	56.0	0.40	0/11	48.4	19.12	0/11
	RC1	2.26	0/8	37.3	1.33	0/8	27.8	40.25	0/8
	RC2	15.17	0/8	44.5	0.40	0/8	38.4	28.13	0/8
0.5	C1	0.00	2/9	72.7	0.02	2/9	25.4	32.17	2/9
	C2	0.00	2/8	61.2	0.00	2/8	34.1	6.42	2/8
	R1	2.27	0/12	33.0	1.32	0/12	29.7	30.05	1/12
	R2	5.97	0/11	69.7	0.44	0/11	46.0	18.61	0/11
	RC1	1.50	0/8	33.7	0.73	0/8	28.1	39.84	0/8
	RC2	13.93	0/8	30.9	0.00	0/8	34.0	27.67	0/8
Average		3.45		45.2	0.53		34.1	24.32	
Solved			26/280			26/280			31/280

fleets is better with low efficiency and cost ranging from medium to high or with high performance and cost.

We have also assessed the quality of our algorithms on the instances of the related VRPMS [Hojabri et al., 2018] and VRPTWDST [Schneider, 2016]. Several instances have been solved optimally by both methods in very short times and new lower bounds and several best known solutions are found for the VRPMS and the VRPTWDST. A promising application for the DIRP is in the home care staff scheduling problem where one needs to choose between a specialist and a generalist and is suggested as future work in this area.

Acknowledgments

Financial support for this work was provided by the Canadian Natural Sciences and Engineering Research Council (NSERC) under grants 2015-04893 and 2019-00094. This support is gratefully acknowledged.

References

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

A. Alvarez and P. Munari. An exact hybrid method for the vehicle routing problem with time windows and multiple deliverymen. *Computers & Operations Research*, 83:1 – 12, 2017.

H. Bae and I. Moon. Multi-depot vehicle routing problem with time windows considering delivery and installation vehicles. *Applied Mathematical Modelling*, 40(13):6536 – 6549, 2016.

S. Bertels and T. Fahle. A hybrid setup for a hybrid scenario: combining heuristics for the home health care problem. *Computers & Operations Research*, 33(10):2866 – 2890, 2006.

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

A. A. Ciré and J. N. Hooker. A heuristic logic-based benders method for the home health care problem. *Presented at Matheuristics, Angra dos Reis, Brazil*, 2012.

L. C. Coelho, J.-P. Gagliardi, J. Renaud, and A. Ruiz. Solving the vehicle routing problem with lunch break arising in the furniture delivery industry. *Journal of the Operational Research Society*, 67(5):743–751, 2016.

J.-F. Cordeau, G. Laporte, and A. Mercier. Improved tabu search algorithm for the handling of route duration constraints in vehicle routing problems with time windows. *Journal of the Operational Research Society*, 55(5): 542–546, 2004.

J.-F. Côté, M. Gendreau, and J.-Y. Potvin. An exact algorithm for the two-dimensional orthogonal packing problem with unloading constraints. *Operations Research*, 62(5):1126–1141, 2014.

G. Desaulniers, O. B.G. Madsen, and S. Ropke. The vehicle routing problem with time windows. In *P. Toth and D. Vigo (eds.), Vehicle Routing: Problems, Methods, and Applications, Second Edition*, MOS-SIAM Series on Optimization, chapter 5, pages 119–159. SIAM, Philadelphia, 2014.

B. Domínguez-Martín, I. Rodríguez-Martín, and J.-J. Salazar-González. The driver and vehicle routing problem. *Computers & Operations Research*, 92: 56 – 64, 2018.

M. Drexl. Synchronization in vehicle routing - a survey of VRPs with multiple synchronization constraints. *Transportation Science*, 46(3):297–316, 2012.

V. C Hemmelmayr, J.-F. Cordeau, and T. G. Crainic. An adaptive large neighborhood search heuristic for two-echelon vehicle routing problems arising in city logistics. *Computers & Operations Research*, 39(12):3215–3228, 2012.

H. Hojabri, M. Gendreau, J.-Y. Potvin, and L.-M. Rousseau. Large neighborhood search with constraint programming for a vehicle routing problem with synchronization constraints. *Computers & Operations Research*, 92:87 – 97, 2018.

J. Homberger and H. Gehring. Two evolutionary metaheuristics for the vehicle routing problem with time windows. *INFOR: Information Systems and Operational Research*, 37(3):297–318, 1999.

S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

R. Liu, Y. Tao, and X. Xie. An adaptive large neighborhood search heuristic for the vehicle routing problem with time windows and synchronized visits. *Computers & Operations Research*, 101:250 – 262, 2019.

D. C. Montgomery. *Design and Analysis of Experiments*. Wiley, Hoboken, NJ, 7th edition, 2009.

S. Naccache, J.-F. Côté, and L. C. Coelho. The multi-pickup and delivery problem with time windows. *European Journal of Operational Research*, 269(1):353 – 362, 2018.

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

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

V. Pureza, R. Morabito, and M. Reimann. Vehicle routing with multiple deliverymen: Modeling and heuristic approaches for the VRPTW. *European Journal of Operational Research*, 218(3):636 – 647, 2012.

M. S. Rasmussen, T. Justesen, A. Dohn, and J. Larsen. The home care crew scheduling problem: Preference-based visit clustering and temporal dependencies. *European Journal of Operational Research*, 219(3):598 – 610, 2012.

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

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

M. Schneider. The vehicle-routing problem with time windows and driver-specific times. *European Journal of Operational Research*, 250(1):101–119, 2016.

P. Shaw. A new local search algorithm providing high quality solutions to vehicle routing problems. *APES Group, Dept of Computer Science, University of Strathclyde, Glasgow, Scotland, UK*, 1997.

M. M. Solomon. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, 35(2):254–265, 1987.

J. U. Sun. A hybrid computational intelligence technique for vehicle routing problem. *Information*, 21(1):139 – 148, 2018.