# Metaheuristic, Models and Software for the Heterogeneous Fleet Pickup and Delivery Problem with Split Loads

Diógenes Gasque, Pedro Munari*

*Federal University of São Carlos, Production Engineering Department,*
*Rod. Washington Luís Km 235, São Carlos - SP, CEP 13565-905, Brazil*

**Abstract**

This paper addresses a rich variant of the vehicle routing problem (VRP) that involves pickup and delivery activities, customer time windows, heterogeneous fleet, multiple products and the possibility of splitting a customer demand among several routes. This variant generalizes traditional VRP variants by incorporating features that are commonly found in practice. We present two mixed-integer programming models and propose a metaheuristic based on Adaptive Large Neighborhood Search for the addressed problem. Additionally, to facilitate the use of the proposed approaches in real-world decision-making, we develop an open-source, publicly available web interface that allows one to set and solve VRP variants with the mentioned features. Computational experiments using benchmark instances with up to 150 customers show that the approaches can be used to obtain good-quality solutions in a reasonable time frame in practice.

*Keywords:* pickup and delivery, heterogeneous fleet, split load, metaheuristic, compact model

## 1. Introduction

Vehicle routing problems arise in many real-world situations and have a large impact on our day-to-day lives [1, 2, 3, 4, 5]. In this paper, we are particularly interested in problems that involve pickup and delivery activities, motivated by the interaction with companies that seek to determine the best set of routes to collect goods from certain locations and deliver them to other locations, while satisfying vehicle capacity and time windows of each location. In this context, emerges the pickup and delivery problem with time windows (PDPTW), a variant of the vehicle routing problem with time windows (VRPTW) in which the set of customers is partitioned into two groups, one with pickup and the other with delivery customers. Each pickup customer is paired with exactly one delivery customer, which is commonly referred to as paired demand. Both must be visited by the same vehicle and the pickup customer must be visited before the corresponding delivery. Real-

*Corresponding author
*Email addresses:* diogeneshfg@gmail.com (Diógenes Gasque), munari@dep.ufscar.br (Pedro Munari)

world examples for this type of problem arise in the transportation of people, animals, goods, and in situations involving school buses, taxi services, groceries delivery, among others [6, 7, 8, 9, 10].

The literature on the PDPTW includes various compact and extensive mathematical formulations for the traditional variant of the problem, which assumes a homogeneous fleet, single product and no split of customer demands [9, 11, 12, 13]. Different exact solution approaches have been proposed, including branch-and-cut [12, 13] and branch-and-price [14, 15, 16, 17] methods, the latter being the most effective currently. Nevertheless, the computational time required by exact methods to solve large-scale, realistic instances of the problem is typically elevated and prohibitive in practice. Thus, heuristic methods are needed and have been used with success to obtain good-quality solutions within a reasonable computational time [18, 19, 20, 21, 22, 23]. Additionally, heuristics are more appropriate to deal with additional real-world features. In fact, Bettinelli et al. [16] and Qu and Bard [17] addressed the PDPTW with a heterogeneous fleet, while Ríos-Mercado et al. [24], Qu and Bard [8, 25], Ghilas et al. [26], Nowak et al. [27] and Haddad et al. [28] further considered multiple products and split pickup and delivery. Several of them were motivated by applications from the beverage industry, transportation of elderly and disabled people to nursing centers, passenger urban transportation and air cargo carriers.

Incorporating rich characteristics into pickup and delivery variants is of ultimate relevance, as they are found in many applications in practice [29, 30]. Variants with a heterogeneous fleet leads to solutions that use the most suitable vehicles regarding their capacity, among other vehicle characteristics. The main motivation for multiple products is that certain goods have distinct qualities and demands. As examples we can mention the transportation of olive oil, raw milk, petroleum products, and glass waste collection [31, 32, 33, 34]. Finally, by allowing split pickups and deliveries, customer demands may be served by combining different routes, whereby fewer vehicles may be necessary to service all customers, leading to significant cost reductions [27, 35, 36, 37].

In this paper, we address the heterogeneous-fleet PDPTW with multiple products and split pickup and split delivery (HPS-PDPTW). Additionally, we consider a variant without splitting (HP-PDPTW). The main contributions are as follows: (*i*) a three-index vehicle flow model for the HPS-PDPTW and a two-index vehicle flow model for the HP-PDPTW; (*ii*) a metaheuristic based on Adaptive Large Neighborhood Search (ALNS) for both variants; and (*iii*) an open-source, publicly available web interface to ease the use of the proposed models and methods and serve as a framework to aid decision making in practice, for situations involving pickup and delivery problems.

The remainder of this paper is organized as follows. Section 2 presents the main elements of the addressed variants, defines the mathematical notation, and states the proposed models. Section 3 describes the proposed metaheuristic. Section 4 shows the results of computational experiments

with the models and the metaheuristic using benchmark instances from the literature. Section 5 describes the web interface that integrates the proposed approaches. Finally, Section 6 presents our concluding remarks and ideas for future work.

## 2. Problem definition and mathematical formulations

Let $n$ be the number of requests and $G = (N, A)$ be a directed graph where $N = \{0, \ldots, 2n+1\}$ is the set of nodes and $A = \{(i, j) : i, j \in N, i \neq j\}$ is the set of arcs. The nodes 0 and $2n+1$ represent the initial and final depot, respectively, where the vehicle fleet must depart from and arrive to (they often represent the same location in practice). Each request consists of one pickup customer and one paired delivery customer. The subsets $P = \{1, \ldots, n\}$ and $D = \{n+1, \ldots, 2n\}$ represent the pickup and delivery customers, respectively. A set of multiple products is defined as $L = \{1, \ldots, l\}$ and each pickup customer $i \in P$ has a positive demand $d_{il}, \forall l \in L$. The pickup customer $i$ is associated to the delivery customer $n+i \in D$ and $d_{(n+i)l} = -d_{il}, \forall l \in L$. The vehicle fleet is composed of $m$ vehicles with distinct capacities and is represented by the set $K = \{1, \ldots, m\}$. A capacity $q_k$ is associated to each vehicle $k \in K$. A service time $s_i$ and a time window $[w_i^a, w_i^b]$ are related to every customer $i \in P \cup D$. Finally, associated to each arc $(i, j) \in A$ there is a cost $c_{ij}$ and a travel time $t_{ij}$. We assume that the triangle inequality holds for costs and travel times.

Every customer should be visited at least once, and the routes must satisfy time windows and vehicle capacities. All the demand available in a pickup customer must be transported to the corresponding delivery customer, possibly by more than one route. Customers can be visited at any order as long as the pickup customer is visited before its delivery customer (precedence relation) and both are visited by the same route(s) (pairing relation). The visit to a delivery customer does not have to be right after its corresponding pickup customer. The problem consists of determining a set of least-cost routes that satisfy all these requirements. In the following subsections, we propose two mixed-integer programming (MIP) models, one for the HP-PDPTW and another for the HPS-PDPTW.

### *2.1. Two-index vehicle flow model for the HP-PDPTW*

We first propose a two-index vehicle flow model for the HP-PDPTW, the variant that does not allow split pickups nor split deliveries. Hence, each customer must be visited exactly once and fully serviced in this visit. The model extends the formulation originally proposed by Furtado et al. [11] for the classic PDPTW. The authors presented a novel strategy to impose pairing and precedence relations by inserting new continuous variables $v_i, \forall i \in P \cup D$. Basically, these variables propagate the index of the first customer visited in a route. Then, using a polynomial number of constraints, the index of this first customer is assigned to other variables $v_j$ such that $j$ is

in the same route. Computational results have shown that this formulation promotes a superior performance of general-purpose optimization software in comparison to using other two- and three-index formulations available in the literature [9, 14, 13]. Hence, we adapt this formulation to include heterogeneous-fleet and multiple products.

Consider the following decision variables: $v_i$, a continuous variable that represents the index of the first customer visited in the route that services customer $i \in P \cup D$; $x_{ij}$, a binary variable that assumes the value of 1 if and only if there is a route that goes from customer $i$ directly to $j$, for each arc $(i, j) \in A$; $y_{il}$, a continuous variable that represents the load of product $l$ in the vehicle after servicing customer $i$, for each $i \in N$ and $l \in L$; and $w_i$, a continuous variable that indicates the exact time that the service starts at customer $i$, for each $i \in N$.

An interesting advantage is that this formulation does not rely on a vehicle-index $k$ to explicitly differentiate between the vehicles, yet we consider a heterogeneous fleet. Formulations with vehicle-indexed variables often have an increased number of variables and suffer from solution symmetry, which negatively affects the performance of general-purpose mixed-integer optimization solvers [38, 39, 40]. To be able to consider different vehicle capacities without such variables, we resort to a modeling trick that consists of adding $2m$ artificial nodes to the network [12]. Recall that each vehicle $k$ has a capacity $q_k$ and let $q = max_{k \in K} q_k$ be the largest vehicle capacity in the fleet. We create two additional sets, $P_a$ and $D_a$, with $m$ artificial pickup and delivery customers, respectively. Then, the sets of pickup and delivery customers are redefined as $P := P_a \cup P$ and $D := D_a \cup D$, and the number of requests becomes $n := n + m$. An artificial product $l_a$ with the same unity of measure of $q$ must be created too. Consequently, the demand for product $l_a$ in the artificial nodes is set as $d_{i,l_a} = q - q_i$ and $d_{n+i,l_a} = -d_{i,l_a}$, $\forall i \in P_a$, where $q_i$ is the capacity of vehicle $i$. Finally, artificial customers do not have demand for real products and original customers have no demand for the artificial product. We establish that $c_{0i} = 0$ and $c_{ij} = c_{0j}$ for all artificial pickup customers $i$ and original pickup customers $j$. In the same way, $c_{n+i,2n+1} = 0$ and $c_{n+j,n+i} = c_{n+j,2n+1}$ for all artificial delivery customers $n + i$ and all original delivery customers $n + j$. The cost between any two artificial customers is zero. Figure 1 shows a graph with original nodes for a problem with two requests ($n = 2$), and a second one right after the insertion of an artificial request ($m = 1$). After this procedure, we have a problem with three requests where real pickup nodes are represented by 1 and 2 and real delivery nodes are denoted by 4 and 5. The pickup artificial node is represented by 3 and the delivery artificial node by 6. Note that as we add a number $m$ of artificial requests to the problem this causes the real delivery nodes to shift $m$ positions ahead.

With the described changes, the modeling of the heterogeneous fleet is done by means of a homogeneous fleet with capacity $q$, but we ensure that the artificial pickup customers are the first

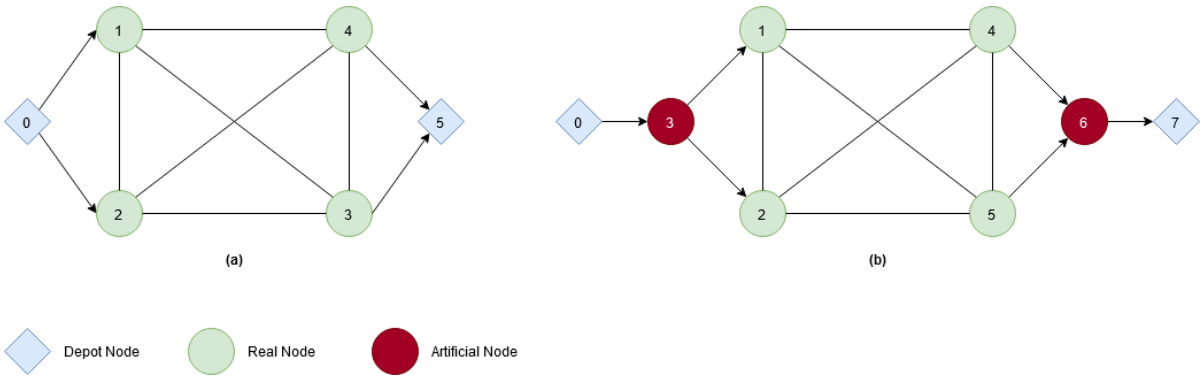



Figure 1: Graph with real nodes and after the insertion of artificial ones.

ones to be visited in any route, and the artificial delivery customers are the last ones in the same route. This way, each artificial request enforces the right capacity of a specific vehicle, as after visiting an artificial pickup customer $i$, the amount $d_{i,l_a}$ is subtracted from the vehicle capacity and is not released until the artificial delivery customer $n+i$ is visited by the same vehicle. With all these definitions, we state the two-index formulation for the HP-PDPTW as follows:

$$\min \sum_{i \in N} \sum_{j \in N} c_{ij} x_{ij} \tag{1}$$

$$\text{s.t.} \sum_{i \in N} x_{ij} = 1 \qquad \forall j \in P \cup D, \tag{2}$$

$$\sum_{j \in N} x_{ij} = 1 \qquad \forall i \in P \cup D, \tag{3}$$

$$w_j \geq w_i + t_{ij} - M(1 - x_{ij}) \qquad \forall i \in N; j \in N, \tag{4}$$

$$w_i^a \leq w_i \leq w_i^b \qquad \forall i \in N, \tag{5}$$

$$y_{jl} \geq y_{il} + d_{jl} - M(1 - x_{ij}) \qquad \forall i \in N; j \in N; l \in L, \tag{6}$$

$$\sum_{l \in L} y_{il} \leq q \qquad \forall i \in N, \tag{7}$$

$$w_{n+i} \geq w_i + s_i + t_{i,i+n} \qquad \forall i \in P, \tag{8}$$

$$v_{n+i} = v_i \qquad \forall i \in P, \tag{9}$$

$$v_j \geq j.x_{0j} \qquad \forall j \in P \cup D, \tag{10}$$

$$v_j \leq j.x_{0j} - n(x_{0j} - 1) \qquad \forall j \in P \cup D, \tag{11}$$

$$v_j \geq v_i + n(x_{ij} - 1) \qquad \forall i, j \in P \cup D, \tag{12}$$

$$v_j \leq v_i + n(1 - x_{ij}) \qquad \forall i, j \in P \cup D, \tag{13}$$

$$x_{0i} = 0 \qquad \forall i \in N \setminus P_a, \tag{14}$$

$$x_{ij} = 0 \qquad \forall i \in P_a; j \in N \setminus P \cup D_a, \tag{15}$$

$$x_{ij} = 0 \qquad \forall i \in N \setminus D \cup P_a; j \in D_a, \tag{16}$$

$$x_{i,2n+1} = 0 \qquad \forall i \in N \setminus D_a, \tag{17}$$

$$v_j, w_j \geq 0 \qquad \forall j \in N, \tag{18}$$

$$y_{jl} \geq 0 \qquad \forall j \in N, l \in L, \tag{19}$$

$$x_{ij} \in \{0, 1\} \qquad \forall i \in N; j \in N. \tag{20}$$

The objective function (1) consists of minimizing the total routing costs. Constraints (2) and (3) ensure that customers must be visited exactly once. Constraints (4) and (5) guarantee the consistency of time variables, impose time windows and prevent sub-tours. Constraints (6) ensure the correct load balance in the vehicles after each visit. Constraints (7) prevent the vehicle capacity to be exceeded. Constraints (8) to (13) ensure precedence and pairing relations as originally proposed by Furtado et al. [11]. Constraints (14) to (17) guarantee that artificial pickup and delivery customers are the first and last nodes in each route, respectively. Finally, constraints (18)–(20) impose the domain of the decision variables.

### *2.2. Three-index vehicle flow model for the HPS-PDPTW*

Formulation (1)–(20) cannot be straightforwardly adapted to allow split pickups and split deliveries. The difficulty lies in allowing multiple visits to customers using variables indexed by arcs only, as there may be several routes arriving to a same node from different arcs and thus each incoming arc must be paired with exactly one outgoing arc. Otherwise, it compromises the correct propagation of load and time flow through a route. We can overcome this by adding the vehicle index $k$ to the decision variables. Namely, let $x_{ijk}$ be a binary variable that assumes the value of 1 if and only if vehicle $k$ goes from customer $i$ directly to $j$, for each arc $(i, j) \in A$; $y_{ikl}$ be a continuous variable that represents the load of product $l$ in vehicle $k$ after servicing customer $i$, for each $i \in N$ and $l \in L$; and $w_{ik}$, a continuous variable that indicates the exact time that the service starts at customer $i$ when it is visited by vehicle $k$, for each $i \in N$. These variables are similar to the ones defined in model (1)–(20). Additionally, to include the possibility of split pickup and split delivery, we define the continuous variable $z_{ikl}$ that represents the amount of product $l$ collected/delivered by vehicle $k$ when it services customer $i$. For the ease of notation, we assume that the demand $d_{(n+i)l}$ for each delivery customer $n + i$ and for each product $l$ is positive and hence $d_{(n+i)l} = d_{il}$. Using these definitions, we state the three-index formulation for the HPS-PDPTW as follows:

$$\min \sum_{k \in K} \sum_{i \in N} \sum_{j \in N} c_{ij} x_{ijk} \tag{21}$$

$$\text{s.t.} \sum_{k\in K}\sum_{j\in N} x_{ijk} \geq 1 \qquad \forall i \in P, \qquad (22)$$

$$\sum_{j\in N} x_{ijk} - \sum_{j\in N} x_{n+i,j,k} = 0 \qquad \forall k \in K; i \in P, \qquad (23)$$

$$\sum_{j\in N} x_{0jk} \leq 1 \qquad \forall k \in K, \qquad (24)$$

$$\sum_{j\in N} x_{0jk} - \sum_{i\in N} x_{i,2n+1,k} = 0 \qquad \forall k \in K, \qquad (25)$$

$$\sum_{j\in N} x_{jik} - \sum_{j\in N} x_{ijk} = 0 \qquad \forall k \in K; i \in P \cup D, \qquad (26)$$

$$w_{ik} + s_i + t_{i,n+i} \leq w_{n+i,k} \qquad \forall k \in K; i \in P \cup D, \qquad (27)$$

$$w_{jk} \geq w_{ik} + (s_i + t_{ij})x_{ijk} - M(1 - x_{ijk}) \qquad \forall k \in K; i, j \in N, \qquad (28)$$

$$w_i^a \leq w_{ik} \leq w_i^b \qquad \forall k \in K; i \in P \cup D, \qquad (29)$$

$$y_{jkl} \geq y_{ikl} + z_{jkl} - M(1 - x_{ijk}) \qquad \forall k \in K; i \in N; j \in P; l \in L, \qquad (30)$$

$$y_{jkl} \geq y_{ikl} - z_{jkl} - M(1 - x_{ijk}) \qquad \forall k \in K; i \in N; j \in D; l \in L, \qquad (31)$$

$$\sum_{l\in L} y_{ikl} \leq q_k \qquad \forall k \in K; i \in N; l \in L, \qquad (32)$$

$$z_{ikl} \leq d_{il} \sum_{j\in N} x_{jik} \qquad \forall k \in K; i \in N; l \in L, \qquad (33)$$

$$\sum_{k\in K} z_{ikl} = d_{il} \qquad \forall i \in N; l \in L, \qquad (34)$$

$$w_{jk} \geq 0 \qquad \forall j \in N; k \in K, \qquad (35)$$

$$y_{jkl}, z_{jkl} \geq 0 \qquad \forall j \in N; k \in K; l \in L, \qquad (36)$$

$$x_{ijk} \in \{0, 1\} \qquad \forall i, j \in N; k \in K. \qquad (37)$$

The objective function (21) consists of minimizing the total routing cost. Constraints (22) guarantee that customers are visited at least once. Constraints (23) ensure that the same vehicle visits the pickup and its paired delivery customer. Constraints (24) and (25) state that a vehicle can leave the initial depot only once and, if it leaves, it must return to the final depot. Constraints (26) ensure that if a vehicle visits a customer $i$ it must leave that customer after servicing it. Constraints (27) guarantee the precedence between pickup and delivery visits. Constraints (28) and (29) ensure the consistency for time variables, impose time windows and prevents sub-tours. Constraints (30) and (31) establish the load balance for vehicles after visiting pickup and delivery customers, respectively. Constraints (32) ensure that the vehicle capacity is not exceeded. Constraints (33) guarantee that the amount loaded or unloaded of a product by any vehicle at any customer should be less than or equal to the demand of that product of that customer. Constraints (34) ensure that

the demand of each customer is met. Finally, constraints (35)–(37) impose the domain of variables.

Formulation (21)–(37) can be easily adapted to model the HP-PDPTW. We just need to impose exactly one visit to each pickup and delivery customer and ensure that the demand is fully satisfied by the only vehicle that service the pickup and the delivery node. However, as it will be shown in Section 4, the resulting model leads to an inferior performance of general-purpose optimization software in comparison with using formulation (1)–(20). For this reason, we use the latter for modeling the HP-PDPTW.

## 3. Metaheuristic

We propose an Adaptive Large Neighborhood Search (ALNS) metaheuristic for both variants, HP-PDPTW and HPS-PDPTW, based on the methods developed by Ropke and Pisinger [18], Qu and Bard [25, 8] and Ghilas et al. [26]. The ALNS of Ropke and Pisinger [18] was proposed for the classic PDPTW and consists of a Large Neighborhood Search with destroy and repair operators that are applied to large portions of the solution [41]. The operators are selected according to an adaptive mechanism, based on their performance on previous iterations of the method. In addition to incorporate additional features of the problem, our implementation differs from the original method proposed by Ropke and Pisinger [18] in several ways. The authors divided their algorithm into two phases, such that the first one minimizes the number of vehicles and then the second minimizes the routing costs. As recognized by them, this two-phase approach would not be effective to address heterogeneous fleet variants. Hence, we propose a one-phase approach and develop a specific removal heuristic to reduce the number of vehicles [25].

Algorithm 1 describes the proposed ALNS search procedure. First, an initial solution $S$ is provided to ALNS and set as the best global, $S_{best}$ (line 2), and the incumbent solution, $S_*$ (line 4), respectively. To provide the initial solution $S$, we first assign every vehicle to an empty route (i.e., a route that traverses arc $(0, 2n+1)$ only) and then randomly choose requests and insert them into the best possible position and route. The search procedure is divided into segments, each one corresponding to 100 iterations, and at the end of each segment we update the heuristic weights (lines 5-7) that are used in the adaptive mechanism. Let $\bar{h}$ be the total number of insertion and removal heuristics. For a given heuristic $i \in \{1, 2, \ldots, \bar{h}\}$, we update its weight $W_i$ as follows:

$$W_i := (1 - \rho) * W_i + \rho \frac{\pi_i}{\theta_i},$$

where $\rho$ is an input parameter known as reaction factor; $\pi_i$ is the score of heuristic $i$ and $\theta_i$ is the number of times heuristic $i$ was chosen in the last segment. We update the score $\pi_i$ using one of the following values: $\sigma_1$, if the solution found is better than the global one; $\sigma_2$, if a solution was

**Algorithm 1:** Proposed ALNS metaheuristic for the HP-PDPTW and HPS-PDPTW.

```
   input  : S, maxIterations ∈ ℕ, timeToSpend ∈ ℕ;
   output: S_best;
1  countIt = 0;
2  S_best = S;
3  while maxIterations ≥ 0 and timeToSpend > 0 do
4      S_* = S;
5      if countIt = 100 then
6          update weights W_i for each i ∈ {1, 2, ..., h̄};
7          countIt = 0;
8      choose a number μ between 0.1n and 0.4n;
9      select a removal heuristic h_r and an insertion heuristic h_i;
10     remove μ requests from S_* using h_r and insert them into U;
11     insert requests from U into S_* using h_i;
12     if S_* ≤ S_best then
13         S_best = S_*;
14     if accepted(S_*, S) then
15         S = S_*;
16     countIt = countIt + 1;
17     maxIterations = maxIterations − 1;
18     reduce timeToSpend using the time spent in the current iteration;
```

accepted and its costs is better than the current one; and $\sigma_3$, if a solution was accepted and its costs is worse than the current one. These three values $\sigma_1$, $\sigma_2$ and $\sigma_3$ are input parameters.

At each iteration of our ALNS, a random number $\mu$ is chosen between 10 and 40% of $n$, the number of requests (line 8). After that, one removal and one insertion heuristic are selected based on their weights (line 9). A given heuristic $i \in \{1, 2, \ldots, \bar{h}\}$ is chosen with probability

$$\frac{W_i}{\sum_{j=1}^{\bar{h}} W_j}.$$

The selected removal heuristic removes $\mu$ requests from the incumbent solution and places them into the request bank $U$ (line 10). The request bank allows infeasible solutions to be visited and helps with the search for better solutions. See Section 3.1 for further details on the removal heuristics. Insertion heuristics try to insert all the requests in $U$ into the incumbent solution (line 11). They are detailed in Section 3.2. After they are invoked, the algorithm checks if the new solution is better than the best global solution and if it is, assigns it as the new best global (lines 12 and 13).

We resort to the Simulated Annealing (SA) acceptance criterion [42, 43] (line 14) to accept a solution with probability

$$e^{-\frac{S_* - S}{T}},$$

where $T$ is the initial temperature, defined as in [18]. The initial temperature is calculated for a

solution $S_w$ that is $w\%$ worse than the initial $S$ and it is accepted with a 50% probability. We use a scalar $\alpha$ to cool down $T$ at each iteration using $T := \alpha.T$, with $T \geq 0$. We only accept solutions in which all requests are serviced. The SA acceptance criterion helps the algorithm to escape from local optimal solutions. All these steps are repeated unless the running time exceeds $timeToSpend$ or the number of iterations exceeds $maxIterations$.

We evaluate the cost of a solution using the objective function

$$c(S) = w_1 \sum_{r \in S} \sum_{(i,j) \in r} t_{ij} + w_2 \sum_{r \in S} I_r + w_3 \sum_{i \in U} H_i, \tag{38}$$

where $r$ is a feasible route that belongs to $S$; $t_{ij}$ is the travel time between customers $i$ and $j$; $I_r$ is a variable that assumes the value of 0 if route $r$ is empty, and 1 otherwise; $H_i$ is a variable that assumes the value of 1 if a request $i$ is not serviced in $S$, and 0 otherwise. The terms in the cost $c(S)$ are weighted in a lexicographic way, $w_1 \ll w_2 \ll w_3$, in order to increase the chances of obtaining feasible solutions by the ALNS [18, 25, 8].

### *3.1. Removal heuristics*

The removal heuristics are used to destroy a given solution $S_*$ and this way explore the search space through their neighborhood. At each call, they remove exactly $\mu$ requests from the solution. We used eight removal heuristics selected from several previous approaches [18, 26, 25], which work as follows:

- *Shaw's removal* (SWR): This heuristic removes requests that are similar to each other in terms of travel time, demand and service start time. First, a random request $i$ is randomly selected from $S$ and a relation factor $R_{ij} = \varphi(t_{ij} + t_{n+i,n+j}) + \chi(|w_i - w_j| + |w_{n+i} - w_{n+j}|) + \psi|\sum_{l=1}^{L} d_{il} - \sum_{l=1}^{L} d_{jl}|$ is calculated for every request $j \in S, i \neq j$. Then, all requests are inserted into an ordered list $OL$ in non-descending order of $R_{ij}$ (the smaller is $R_{ij}$, the more similar are the requests). We then remove the request in position $p^{\delta}.|OL|$ of $OL$, where $p \in [0,1]$ is randomly chosen and $\delta$ is a parameter that promotes randomness in the process. The weights $\varphi$, $\chi$ and $\psi$ are normalized.

- *Travel time related removal* (TTR): removes requests that are similar to each other in terms of travel times. It is a special case of SWR heuristic with weights $\varphi = 1$ and $\chi = \psi = 0$. Recall that we assume $t_{ij} = c_{ij}$ for every $(i, j) \in A$.

- *Demand related removal* (DER): removes requests that are similar to each other in terms of customer demand. It is a special case of SWR heuristic with weights $\psi = 1$ and $\varphi = \chi = 0$.

- *Service start time related removal* (STR): removes requests that are similar to each other in terms of the service start time at a customer. This is a special case of SWR heuristic with weights $\chi = 1$ and $\varphi = \psi = 0$.

- *Late arrival removal* (LAR): removes requests that have the largest difference between the service start time and the opening of the time window. This lateness measure for a request $i$ is calculated by $LA_i = |w_i - w_i^a| + |w_{n+i} - w_{n+i}^a|$. The same mechanism of SWR is used to select the next request to be removed.

- *Random Removal* (RAR): removes requests randomly.

- *Worst removal* (WR): let $c(S, -i)$ be the cost of a solution $S$ where the request $i$ is completely removed and let $\Delta c(S, -i) = c(S) - c(S, -i)$ be the cost variation for $S$ if $i$ is removed, where $c(S)$ is defined in (38). This heuristic removes requests that have the largest $\Delta c(S, -i)$. The requests are inserted into an ordered list and removed using the same mechanism as in the SWR heuristic.

- *Random Route Removal* (RRR): randomly selects a route and removes all requests from it, leaving the route empty (we keep only the initial and final depots to ensure feasibility). As in [25], we use this specific removal heuristic to take into account the heterogeneity of the fleet.

### *3.2. Insertion heuristic*

The proposed ALNS resorts to six insertion heuristics. Five of them are *regret* heuristics based on [18] and the remaining one is proposed to incorporate split pickup and split delivery. Each one tries to re-insert all the requests that are in the request bank $U$. The *regret* heuristics are an attempt to look ahead when a request is tested to be inserted into a solution. For a given request $i$, we evaluate in each route the cost of the best insertion position of $i$ in that route. We denote as $c(S, +i)_k$ the cost of the $k$th best insertion position of $i$ considering all routes (one position per route), and $c(S, +i)_{best} = c(S, +i)_1$ is the cost of the best insertion position. In each iteration, the heuristic tries to insert the request $i$ with the maximum regret value, given by $max_{i \in U} \left\{ \sum_{j=1}^{k} c(S, +i)_j - c(S, +i)_{best} \right\}$. We can obtain a whole family of regret heuristics by varying the value of $k$. In our implementation, we used $k \in \{1, 2, 3, 4, m\}$, where $m$ is the number of routes in the solution. Notice that *regret-1* is a greedy heuristic, while *regret-m* is the heuristic that considers all $m$ routes of a solution.

The insertion heuristic related to split pickup and split delivery is based on the *regret-1* heuristic. For a given request $i$, the heuristic splits the total customer demand over two routes only. First, we select the number of products that will be split inside the interval $[1, |L|]$. For each product $l$

chosen, we randomly define the fraction of demand $d_{il}^1$ in the interval $[1, d_{il} - 1]$. Therefore, to one route we assign the pickup and delivery amount $d_{il}^1$, while to another we assign $d_{il}^2 = d_{il} - d_{il}^1$.

We additionally considered modified versions of these six heuristics in which we add a *noise* to the computation of the cost of a solution, given as $c(S)^* = max\{0, c(S) + \xi\}$, where $c(S)$ is defined in (38) and $\xi$ is randomly chosen in the interval $[-maxR, maxR]$, for $maxR = \eta.max_{i,j \in N}\{t_{ij}\}$ and a parameter $\eta$ that controls the noise level. This way, we have 12 heuristics in total, and the adaptive mechanism chooses if the version with or without noise is the one used. This addition of *noise* together with the SA acceptance criterion seeks to avoid the ALNS to get stuck in local optimal solutions.

## 4. Computational results

We present the results of computational experiments with the proposed models and metaheuristic, using benchmark instances from the literature. The approaches were coded in language C++ and the MIP models were solved using the general-purpose optimization software IBM CPLEX Optimization Studio v.12.7 in its default settings. All experiments were run in a PC with Intel Core i7-7500U 2.70 GHz processor, 16 GB of RAM and operating system Windows 10. For the MIP models, we imposed the time limit of 3600 seconds for each instance. The metaheuristic was run five times for each instance, each one with a time limit of 600 seconds and iteration limit of 25000.

As benchmark instances we selected the ones in classes AA, BB, CC and DD, with up to 150 customers, proposed by Ropke et al. [12] for the classic PDPTW. We also included the instances proposed by Furtado et al. [11]. Since these instances consider homogeneous fleet and single product, we modified them as follows. First, to create multiple products, we took the original demand $d_i$ of each request $i$ from the benchmark instances and divided it by three, leading to three products for each request. If the remainder of the integer division was positive, then this value was added to one of the products, ensuring that $\sum_{l \in L} d_{il} = d_i \ \forall i \in N$. Then, we created a fleet of vehicles for the HP-PDPTW and another for the HPS-PDPTW. The fleet for the HP-PDPTW was created by randomly generating the parameter $q_k$ in the range [$maxd$,1.2$maxd$], where $maxd$ is the biggest demand considering all instances of a class. It is composed of seven vehicles for classes AA and CC, with capacities $15, 18, 17, 15, 15, 17$ and 16; and nine vehicles for classes BB and DD, with capacities $23, 22, 21, 20, 24, 24, 20, 21$ and 22. We add the suffix *-HP* to the name of instances defined with these fleets. For the HPS-PDPTW a more restricted fleet was created to force demands to be split. Hence, the parameter $q_k$ was randomly chosen in [0.8$maxd$,$maxd$]. For classes AA and CC, we define seven vehicles with capacities $15, 14, 12, 14, 13, 13$ and 12; and for classes BB and DD, nine

vehicles with capacities 19, 18, 18, 20, 16, 17, 16, 16 and 19. For instances using these fleets, we add the suffix *-HPS* to their names.

Following the common approach in the PDPTW literature, we penalized the vehicles departure from the initial depot in $10^4$. This value is also assigned to parameter $w_2$ in the ALNS. Parameters $w_1$ and $w_3$ are set to 1 and $10^6$, respectively. All parameters were calibrated using I-Race [44] and the procedure was done in 12 instances with 15, 35, 55 and 75 requests from classes AA, BB, CC and DD. It was given to I-Race a budget of 1000 experiments. The resulting parameters are presented in Table 1.

| Parameters | $w$ | $c$ | $\sigma_1$ | $\sigma_2$ | $\sigma_3$ | $r$ | $\varphi$ | $\chi$ | $\psi$ | $\delta$ | $\eta$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Values | 1.3 | 0.9997 | 27 | 30 | 22 | 0.1 | 0.3 | 0.4 | 0.3 | 6 | 0.2 |

Table 1: ALNS parameter choices.

It is worth mentioning that we also ran experiments with our ALNS adapted to the classic PDPTW, in order to verify its overall performance with respect to other metaheuristics [14]. Considering the same time limits, the results showed that our method found solutions with the same objective values as reported by Ropke and Cordeau [14] for 30 out of 60 instances. Only for six instances our ALNS found solutions with objective value larger than 1% of the best ones.

### *4.1. HP-PDPTW results*

We first present the results of the proposed approaches applied to the HP-PDPTW. For this variant, we ran experiments with the two- and three-index models of Sections 2.1 and 2.2, and with the proposed ALNS. Table 2 shows the results obtained by CPLEX using the two models, for the first five instances of classes AA and BB and first four instances of CC and DD (CPLEX could not obtain even a feasible solution for the remaining instances). The first and eighth columns in the table give the instance names (Instance). Then, for each model and instance, the table presents the objective value of the best solution found within the time limit (UB); the relative optimality gap (Gap) in percentage, as provided by CPLEX; and the total computational time (T(s)) in seconds. When CPLEX reached the time limit for a given instance, we report the total time as 'TL'. If no feasible solution was obtained by CPLEX, we use the symbol '-' in the columns of the corresponding model and instance.

The results in Table 2 indicate that the two-index model is superior to the three-index model. With this model, the computational times were shorter, more instances were solved to optimality, and more feasible solutions were obtained. These results support the ones presented by Furtado et al. [11] for the classical PDPTW. Nevertheless, as expected, the models could find feasible solutions only for small-sized instances, with up to 25 customers.

| | Three-index model | | | Two-index model | | | | Three-index model | | | Two-index model | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Instance | UB | Gap | T(s) | UB | Gap | T(s) | Instance | UB | Gap | T(s) | UB | Gap | T(s) |
| AA05-HP | 10184.80 | 0.0% | 0.2 | 10184.80 | 0.0% | 0.1 | BB05-HP | 10339.70 | 0.0% | 0.1 | 10339.70 | 0.0% | 0.1 |
| AA10-HP | 10383.60 | 0.0% | 2.7 | 10383.60 | 0.0% | 0.4 | BB10-HP | 20512.50 | 0.0% | 25.5 | 20512.50 | 0.0% | 0.9 |
| AA15-HP | 20542.40 | 0.0% | TL | 20542.40 | 0.0% | 11.7 | BB15-HP | 20667.60 | 0.0% | TL | 20667.60 | 0.0% | 15.9 |
| AA20-HP | 20762.90 | 1.0% | TL | 20757.90 | 0.3% | TL | BB20-HP | – | – | – | 20786.00 | 0.0% | 106.5 |
| AA25-HP | – | – | – | 30976.20 | 33.0% | TL | BB25-HP | – | – | – | 20946.10 | 0.6% | TL |
| CC05-HP | 10212.50 | 0.0% | 5.6 | 10212.50 | 0.0% | 0.2 | DD05-HP | 10324.80 | 0.0% | 0.3 | 10324.80 | 0.0% | 0.1 |
| CC10-HP | 10394.30 | 1.0% | TL | 10394.30 | 0.0% | 6.5 | DD10-HP | 20567.80 | 49.0% | TL | 20567.80 | 0.0% | 41.5 |
| CC15-HP | – | – | – | 20557.80 | 49.0% | TL | DD15-HP | – | – | – | 20616.40 | 49.0% | TL |
| CC20-HP | – | – | – | 20748.90 | 49.0% | TL | DD20-HP | – | – | – | 20679.20 | 49.0% | TL |

Table 2: Results for the three- and two-index models for instances of the HP-PDPTW.

| Instance | AvgS | Cvar | AvgT | BestS | BestT | Instance | AvgS | Cvar | AvgT | BestS | BestT |
|---|---|---|---|---|---|---|---|---|---|---|---|
| AA05-HP | 10184.80 | 0.00 | 0.0 | 10184.8 | 0 | BB05-HP | 10339.70 | 0.00 | 0.0 | 10339.7 | 0 |
| AA10-HP | 10383.60 | 0.00 | 0.0 | 10383.6 | 0 | BB10-HP | 20512.50 | 0.00 | 0.0 | 20512.5 | 0 |
| AA15-HP | 20542.40 | 0.00 | 1.8 | 20542.4 | 0 | BB15-HP | 20667.60 | 0.00 | 0.2 | 20667.6 | 0 |
| AA20-HP | 20757.90 | 0.00 | 17.2 | 20757.9 | 0 | BB20-HP | 20786.00 | 0.00 | 1.8 | 20786.0 | 0 |
| AA25-HP | 21041.08 | *0.00 | 4.2 | 21041.0 | 1 | BB25-HP | 20947.40 | *0.00 | 6.4 | 20946.1 | 0 |
| AA30-HP | 31117.20 | *0.00 | 126.0 | 31116.0 | 136 | BB30-HP | 31038.14 | *0.00 | 127.2 | 31036.7 | 127 |
| AA35-HP | 31279.10 | *0.00 | 286.2 | 31277.1 | 182 | BB35-HP | 31228.68 | *0.00 | 137.0 | 31225.0 | 174 |
| AA40-HP | 31448.90 | *0.00 | 272.6 | 31448.7 | 195 | BB40-HP | 31449.06 | *0.00 | 177.0 | 31447.3 | 46 |
| AA45-HP | 31673.46 | *0.00 | 404.2 | 31666.6 | 445 | BB45-HP | 41535.02 | *0.00 | 447.4 | 41522.4 | 450 |
| AA50-HP | 41770.50 | *0.00 | 475.8 | 41733.8 | 487 | BB50-HP | 41681.20 | *0.00 | 511.2 | 41652.6 | 513 |
| AA55-HP | 41939.18 | *0.00 | 540.2 | 41930.9 | 436 | BB55-HP | 41866.54 | *0.00 | 462.0 | 41853.7 | 555 |
| AA60-HP | 42133.80 | *0.00 | 410.6 | 42108.4 | 599 | BB60-HP | 62361.98 | *0.00 | 579.4 | 62344.2 | 583 |
| AA65-HP | 42268.96 | *0.00 | 312.6 | 42238.6 | 449 | BB65-HP | 62673.42 | *0.00 | 498.6 | 62635.2 | 596 |
| AA70-HP | 52491.64 | *0.00 | 405.0 | 52448.5 | 150 | BB70-HP | 62987.24 | *0.00 | 356.4 | 62920.8 | 544 |
| AA75-HP | 52625.50 | *0.00 | 341.2 | 52575.1 | 495 | BB75-HP | 71104.30 | 0.06 | 280.0 | 63031.1 | 175 |
| CC05-HP | 10212.50 | *0.00 | 0.0 | 10212.5 | 0 | DD05-HP | 10324.80 | 0.00 | 0.0 | 10324.8 | 0 |
| CC10-HP | 10394.30 | 0.00 | 0.0 | 10394.3 | 0 | DD10-HP | 20567.80 | 0.00 | 0.0 | 20567.8 | 0 |
| CC15-HP | 20554.90 | 0.00 | 0.8 | 20554.9 | 0 | DD15-HP | 20616.40 | 0.00 | 0.0 | 20616.4 | 0 |
| CC20-HP | 20721.40 | 0.00 | 20.4 | 20721.4 | 2 | DD20-HP | 20679.20 | 0.00 | 15.8 | 20679.2 | 11 |
| CC25-HP | 20908.80 | 0.00 | 64.8 | 20908.8 | 38 | DD25-HP | 20902.84 | *0.00 | 84.0 | 20902.3 | 66 |
| CC30-HP | 31036.02 | *0.00 | 94.0 | 31010.2 | 93 | DD30-HP | 21016.68 | *0.00 | 113.6 | 21015.1 | 65 |
| CC35-HP | 31168.04 | *0.00 | 270.2 | 31105.0 | 576 | DD35-HP | 31095.08 | *0.00 | 209.4 | 31083.5 | 201 |
| CC40-HP | 31283.42 | *0.00 | 294.8 | 31276.4 | 335 | DD40-HP | 31209.54 | *0.00 | 260.6 | 31202.2 | 280 |
| CC45-HP | 31430.64 | *0.00 | 351.0 | 31419.7 | 423 | DD45-HP | 31326.74 | *0.00 | 413.2 | 31310.9 | 442 |
| CC50-HP | 41671.38 | *0.00 | 492.8 | 41628.8 | 505 | DD50-HP | 31444.38 | *0.00 | 508.8 | 31430.2 | 536 |
| CC55-HP | 41771.42 | *0.00 | 551.0 | 41745.0 | 523 | DD55-HP | 31620.22 | *0.00 | 484.2 | 31581.6 | 515 |
| CC60-HP | 42012.26 | *0.00 | 423.8 | 41997.8 | 438 | DD60-HP | 41794.32 | *0.00 | 549.0 | 41762.3 | 513 |
| CC65-HP | 52196.38 | *0.00 | 452.6 | 52147.6 | 562 | DD65-HP | 42145.56 | *0.00 | 488.6 | 42129.8 | 442 |
| CC70-HP | 52382.60 | *0.00 | 433.4 | 52353.6 | 534 | DD70-HP | 42249.50 | *0.00 | 330.2 | 42228.2 | 446 |
| CC75-HP | 52564.38 | *0.00 | 364.0 | 52519.8 | 369 | DD75-HP | 50446.74 | 0.09 | 485.6 | 42462.7 | 478 |

*Coefficient of variation not null but less than 0.01.

Table 3: Results of the proposed ALNS for instances of the HP-PDPTW.

Table 3 presents the results of the proposed ALNS considering all instances of the HP-PDPTW. In the tables, column Instance denotes the instance name; AvgS is the average solution value for the five runs; Cvar is the coefficient of variation, computed by dividing the standard deviation for the average of the solutions; AvgT is the average computational time for the five runs; BestS is the best solution value found by the metaheuristic in the five runs; and BestT is the shortest computational time to obtain the best solution found by the method. Additionally, Figure 2 compares the ALNS solution values and time with both two- and three-index models only for instances that CPLEX reported results to at least to one of the models. The time axis was limited to 30 seconds because no resolution took more than 15 seconds with our ALNS. For the sake of readability, time labels are presented for models only.

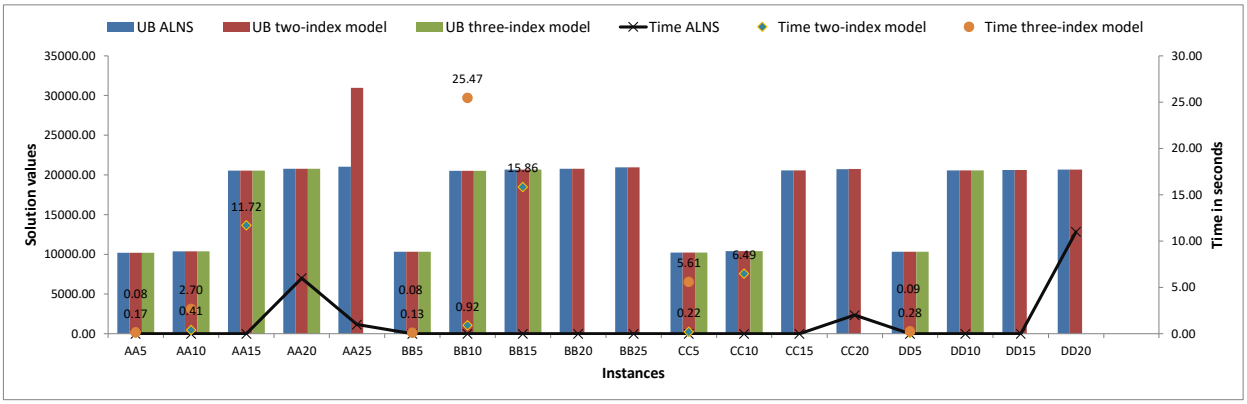



Figure 2: ALNS and two- and three-index models comparison for the HP-PDPTW

From the presented results, we observe that the proposed ALNS method found solutions for every instance in the classes. The coefficient of variation was zero or very small for the majority of instances, indicating that the method is stable. For two instances, namely BB75 and DD75, it found solutions with fewer vehicles than the average, which leads to a coefficient of variation slightly higher than the other ones. As expected, the method found feasible solutions for more instances than the models and in shorter computational times. These solutions were never worse than the ones obtained using the models. For example, for instance AA25 (see Fig. 2), the proposed ALNS reported a solution with two vehicles in negligible time, while the two-index model found a solution with three vehicles and reached the time limit of 3600 seconds.

### *4.2. HPS-PDPTW results*

Table 4 shows the results for the HPS-PDPTW using the three-index model. CPLEX reported feasible solutions for only 12 instances. For most of them, the time limit was reached. This indicates that this vehicle-indexed compact model is not suitable even for small-sized instances. Table 5 describes the results obtained with the proposed ALNS for the HPS-PDPTW. As for the

| Instance | UB | Gap | T(s) | Instance | UB | Gap | T(s) |
|---|---|---|---|---|---|---|---|
| AA05-HPS | 10184.80 | 0.0% | 0.3 | BB05-HPS | 10339.70 | 0.0% | 0.1 |
| AA10-HPS | 10383.60 | 0.0% | 13.8 | BB10-HPS | 20525.40 | 0.0% | 8.2 |
| AA15-HPS | 20544.70 | 0.0% | 2028.9 | BB15-HPS | 30721.60 | 33.0% | TL |
| AA20-HPS | 30778.80 | 64.0% | TL | BB20-HPS | 40957.40 | 50.0% | TL |
| CC05-HPS | 10212.50 | 0.0% | 27.6 | DD05-HPS | 10324.80 | 0.0% | 1.3 |
| CC10-HPS | 20393.30 | 49.0% | TL | DD10-HPS | 20588.70 | 48.0% | TL |

Table 4: Results for the three-index model for instances of the HPS-PDPTW.

| Instance | AvgS | Cvar | AvgT | BestS | BestT | Instance | AvgS | Cvar | AvgT | BestS | BestT |
|---|---|---|---|---|---|---|---|---|---|---|---|
| AA05-HPS | 10184.84 | 0 | 0 | 10184.84 | 0 | BB05-HPS | 10339.67 | 0.00 | 0.0 | 10339.67 | 0 |
| AA10-HPS | 10383.62 | 0 | 0 | 10383.62 | 0 | BB10-HPS | 20525.43 | 0.00 | 0.0 | 20525.43 | 0 |
| AA15-HPS | 20544.67 | 0 | 2 | 20544.67 | 4 | BB15-HPS | 20737.32 | 0.00 | 0.6 | 20737.32 | 1 |
| AA20-HPS | 20764.26 | 0 | 8.4 | 20764.26 | 5 | BB20-HPS | 30870.56 | *0.00 | 4.6 | 30870.56 | 1 |
| AA25-HPS | 30987.84 | *0.00 | 85 | 30986.92 | 57 | BB25-HPS | 31031.07 | 0.00 | 16.2 | 31031.07 | 5 |
| AA30-HPS | 31121.34 | *0.00 | 125.8 | 31121.16 | 138 | BB30-HPS | 31164.12 | *0.00 | 60.8 | 31162.69 | 63 |
| AA35-HPS | 31308.36 | *0.00 | 175.2 | 31303.82 | 96 | BB35-HPS | 39399.61 | 0.11 | 155.2 | 31455.34 | 14 |
| AA40-HPS | 31562.34 | *0.00 | 280.6 | 31551.35 | 327 | BB40-HPS | 41550.23 | *0.00 | 321.4 | 41512.71 | 354 |
| AA45-HPS | 41715.71 | *0.00 | 397.6 | 41701.65 | 365 | BB45-HPS | 45655.58 | 0.12 | 433.2 | 41607.57 | 532 |
| AA50-HPS | 41876.04 | *0.00 | 542.8 | 41850.15 | 596 | BB50-HPS | 49890.57 | 0.17 | 422.8 | 41893.83 | 250 |
| AA55-HPS | 42098.23 | *0.00 | 427.2 | 42087.35 | 586 | BB55-HPS | 64116.78 | 0.07 | 364.0 | 62092.37 | 350 |
| AA60-HPS | 52317.29 | *0.00 | 463.4 | 52290.98 | 472 | BB60-HPS | 80759.21 | 0.06 | 391.2 | 72753.79 | 87 |
| AA65-HPS | 52477.2 | *0.00 | 492.4 | 52446.23 | 566 | BB65-HPS | 81057.89 | 0.06 | 320.4 | 72973.82 | 48 |
| AA70-HPS | 52688.43 | *0.00 | 253 | 52667.81 | 261 | BB70-HPS | 83353.63 | 0.09 | 450.8 | 73294.68 | 578 |
| AA75-HPS | 52823.34 | *0.00 | 277 | 52808.04 | 482 | BB75-HPS | 83487.35 | 0.08 | 401.8 | 73448.66 | 508 |
| CC05-HPS | 10212.53 | 0.00 | 0.0 | 10212.53 | 0 | DD05-HPS | 10324.77 | 0.00 | 0.0 | 10324.77 | 0 |
| CC10-HPS | 20385.50 | 0.00 | 0.0 | 20385.50 | 0 | DD10-HPS | 20572.97 | 0.00 | 0.2 | 20572.97 | 0 |
| CC15-HPS | 20571.18 | 0.00 | 2.6 | 20571.18 | 1 | DD15-HPS | 20680.33 | 0.00 | 1.8 | 20680.33 | 0 |
| CC20-HPS | 20764.42 | 0.00 | 3.8 | 20764.42 | 7 | DD20-HPS | 20777.65 | *0.00 | 26.8 | 20776.23 | 71 |
| CC25-HPS | 20965.74 | *0.00 | 40.0 | 20961.74 | 81 | DD25-HPS | 21009.28 | 0.00 | 122.0 | 21009.28 | 119 |
| CC30-HPS | 31149.34 | *0.00 | 87.6 | 31128.32 | 99 | DD30-HPS | 21150.37 | *0.00 | 104.2 | 21137.71 | 138 |
| CC35-HPS | 31285.71 | *0.00 | 237.4 | 31272.88 | 365 | DD35-HPS | 31254.80 | *0.00 | 224.4 | 31250.76 | 259 |
| CC40-HPS | 31417.45 | *0.00 | 310.4 | 31415.43 | 317 | DD40-HPS | 31387.26 | *0.00 | 372.6 | 31382.67 | 516 |
| CC45-HPS | 33569.02 | 0.13 | 303.2 | 31561.76 | 342 | DD45-HPS | 31512.85 | *0.00 | 449.2 | 31510.66 | 293 |
| CC50-HPS | 41775.38 | *0.00 | 486.4 | 41765.28 | 594 | DD50-HPS | 31643.95 | *0.00 | 534.0 | 31628.68 | 543 |
| CC55-HPS | 41991.89 | *0.00 | 467.0 | 41976.97 | 470 | DD55-HPS | 39829.68 | 0.11 | 486.6 | 31905.41 | 311 |
| CC60-HPS | 52217.06 | *0.00 | 533.8 | 52199.99 | 535 | DD60-HPS | 42053.71 | *0.00 | 491.4 | 42022.93 | 506 |
| CC65-HPS | 52400.97 | *0.00 | 369.0 | 52365.03 | 538 | DD65-HPS | 46351.74 | 0.12 | 531.4 | 42341.40 | 438 |
| CC70-HPS | 52599.09 | *0.00 | 462.4 | 52567.34 | 578 | DD70-HPS | 52574.08 | *0.00 | 284.8 | 52560.46 | 347 |
| CC75-HPS | 62847.57 | *0.00 | 289.2 | 62817.20 | 525 | DD75-HPS | 52764.83 | *0.00 | 460.8 | 52743.20 | 544 |

*Coefficient of variation not null but less than 0.01.

Table 5: Results of the proposed ALNS for instances of the HPS-PDPTW.

other variant, the method found solutions to instances in all classes. The coefficient of variation for most instances is zero or very close to zero, which again indicates it is a stable method. For 11 instances, the coefficient of variation was larger than 0.01%, due to solutions with smaller travel times and less vehicles than the others. This behavior can be observed in instances from the classes

BB, CC and DD. In Fig. 3 we compare the results of the ALNS with the results obtained with the three-index model for the HPS-PDPTW, only for those instances for which CPLEX reported a feasible solution. Again, we see that our ALNS outperformed the solver using the three-index model in all instances as for solution values and time. As an example, the ALNS method took no more than five seconds to provide a solution for the instance AA20, while CPLEX exceeded the time limit of one hour and provided a worse solution.

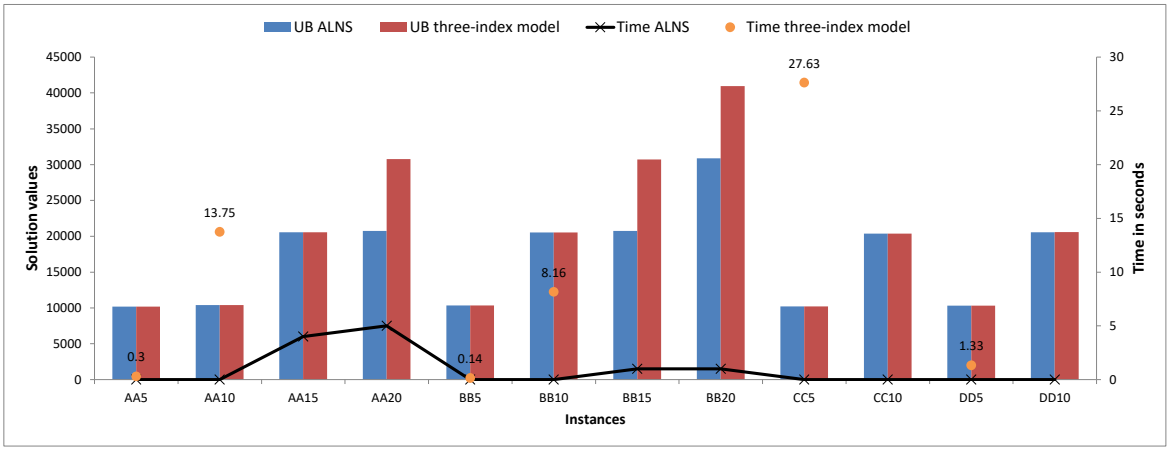



Figure 3: ALNS and three-index model comparison HPS-PDPTW

Finally, we mention that we conducted an additional experiment, providing an initial solution to the three-index model as follows. We ran the ALNS for up to 600 seconds and the best solution obtained was provided as a starting point for CPLEX, which then ran for additional 3000 seconds. The overall results showed that the model was not able to improve the initial solutions and finished with an average gap of 64%. The only positive aspect was that, for some small-sized instances with up to 20 requests, the model was able to prove the optimality of the solutions provided by the ALNS.

## 5. Web interface

We developed a web interface to facilitate the use of the proposed approaches. This software is open-source and publicly available to researchers and practitioners. It attempts to hide to the end user all the complications involved in an implementation of these approaches, serving as a decision-making tool. Additionally, since the source code is publicly available, it can be extended to other VRP variants and even help solving problems in other contexts.

The user is able to execute CRUD (Create, Read, Update and Delete) operations for customers, products and vehicles through the interface, as illustrated in Figures 4 to 7. Figure 4 presents a customer registration process, where the user can search for a customer right on the map or try to search it by address or geographic coordinates (longitude and latitude). It is possible to assign

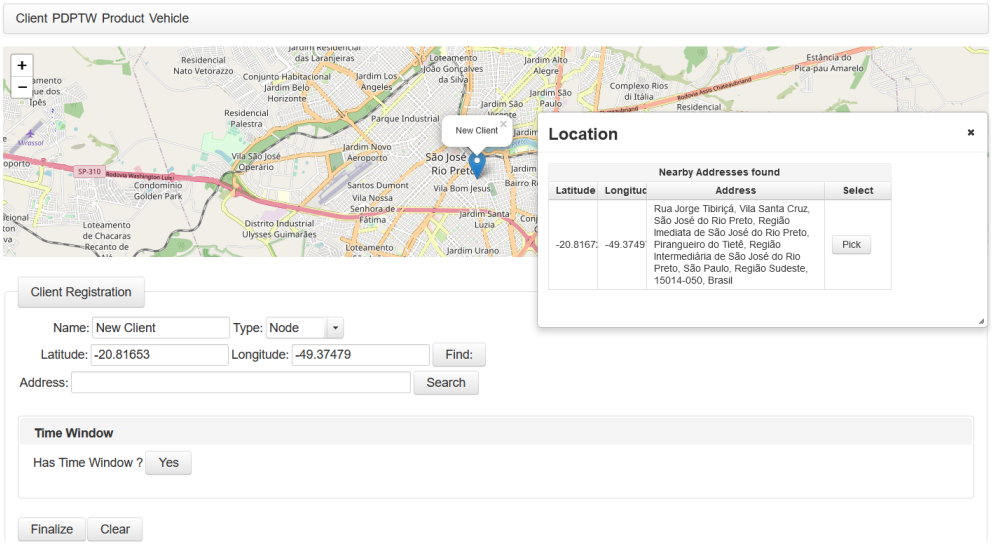



Figure 4: Interface showing customer registration.

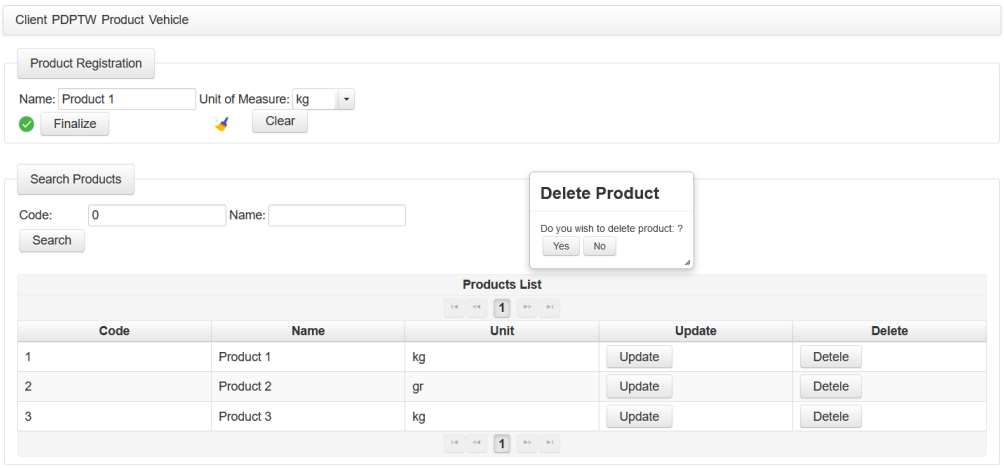



Figure 5: Interface showing product registration.

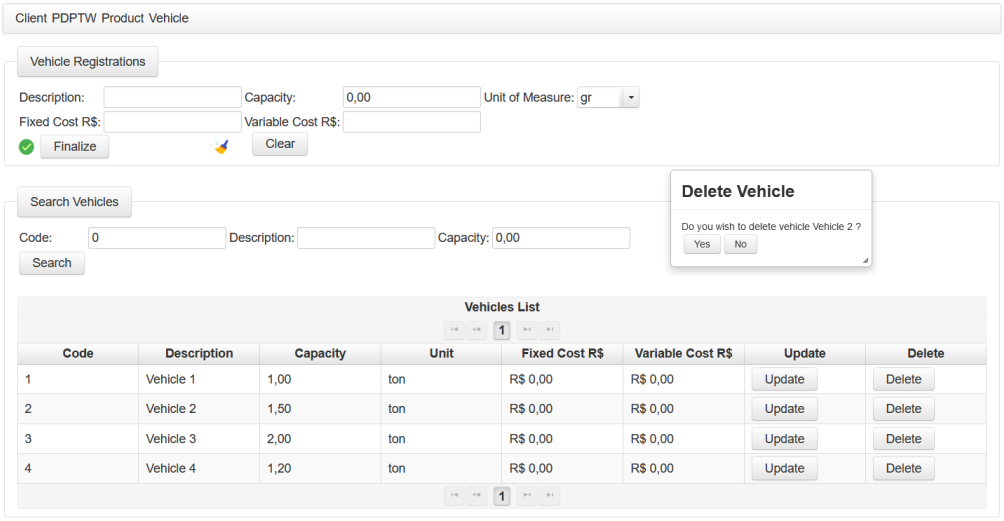
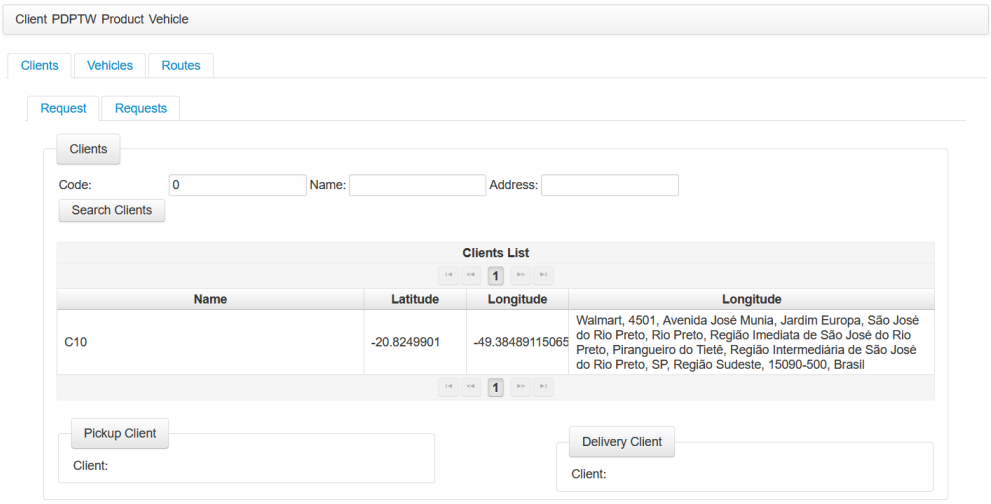



Figure 6: Interface showing vehicle registration.

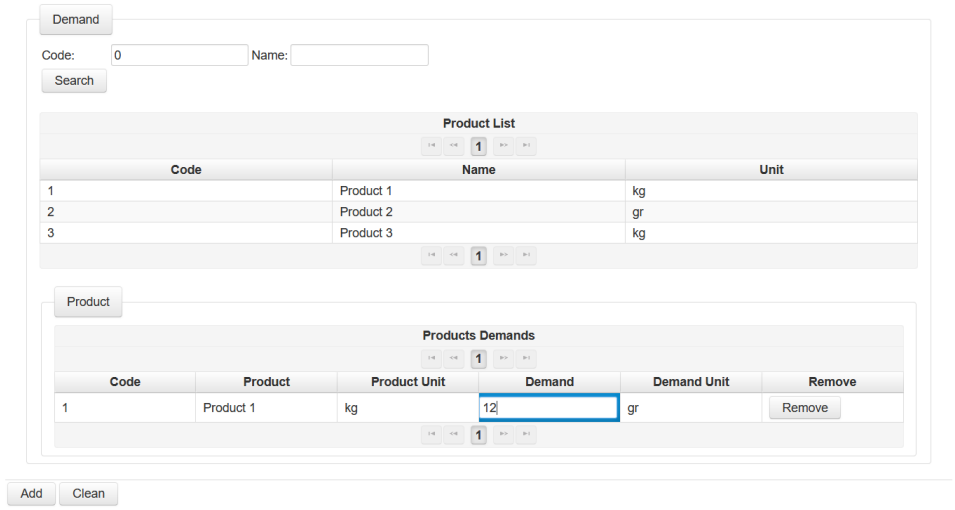
time windows to a customer and label it as a depot, if this is the case. Customers can be searched by name or code (unique identifier) and it is possible to update or delete any customer from the result table. Figure 5 presents product registration, which only requires product name and unit of measure. Products can be searched by name and code, and it is also possible to update or delete products. Figure 6 shows the vehicle registration section, which can be used to specify the available fleet. The fields are a brief description of the vehicle, the vehicle capacity and its corresponding unit of measure. Finally, we must provide the fixed and variable cost of each vehicle. Any inserted vehicle can be searched, updated or deleted.

The user can create a routing plan by adding pickup and delivery customers to a request, which must have a demand for one or more products. It is possible to add one or more vehicles to the routing plan as initial fleet. These steps are demonstrated in Figure 7. To illustrate the output for a given problem instance, we created a toy example with five requests, in which pickup and delivery locations were randomly chosen. We defined three products, with demands equal to one, two and three units, respectively. Time windows for all nodes were defined using the time interval from 08:00 AM to 06:00 PM. Three vehicles are available, and their capacities were set in a way to not restrict the pickups. In Figure 8, we present three prints of our interface, one for each route in the optimal solution of the toy example, considering the HPS-PDPTW.

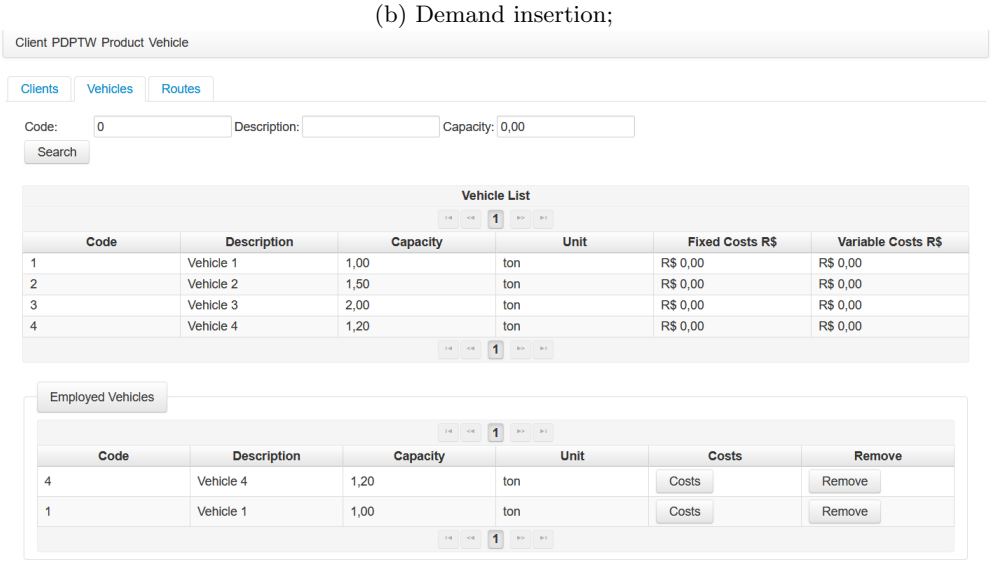
The software was coded in Java with Java Server Faces (JSF) 2.2 and Primefaces 6.2 as web interface frameworks. The web tool runs on Apache Tomcat 9.0 server with MySQL Community 5.7 as a relational data base server. Other frameworks and libraries are employed, such as Hibernate 5.2.13 for mapping objects to relational data bases, Java Native Access 4.5.2 (JNA) to communicate with the methods and models coded in C++, and Apache HttpComponents used to deal with HTTP protocol. We used the Javascript tool Leaflet 1.3.4 to interact with maps, JSON with JSON-Java library as a data exchange format, and Open Street Map (OSM) to provide the maps and services for searching for customer addresses, providing latitude and longitude information. A C++ Server called Open Source Routing Machine (OSRM) [45] was used to extract distance and travel time matrices from the maps provided by the OSM. The OSRM is also employed to draw the routes found by our tools into the maps. These technologies are integrated with the Java counterpart by HTTP protocol and JSON. The source code as well the database scripts are fully available at `http://github.com/diogeneshfg/vrp-webapp`. It is still an ongoing project, yet we believe it can benefit other researchers and practitioners from the VRP community looking for a graphical tool to aid decision making, which can be even adapted to their needs.




(a) Request insertion;

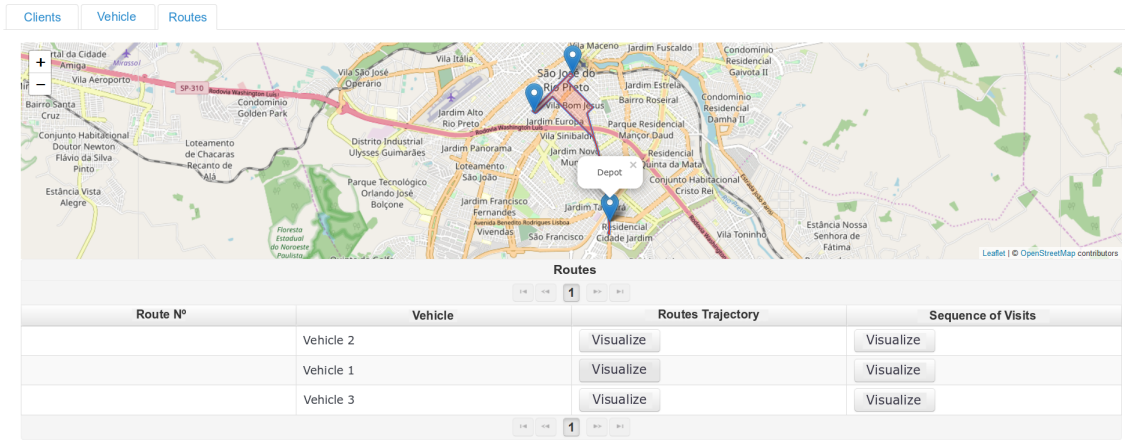



(b) Demand insertion;

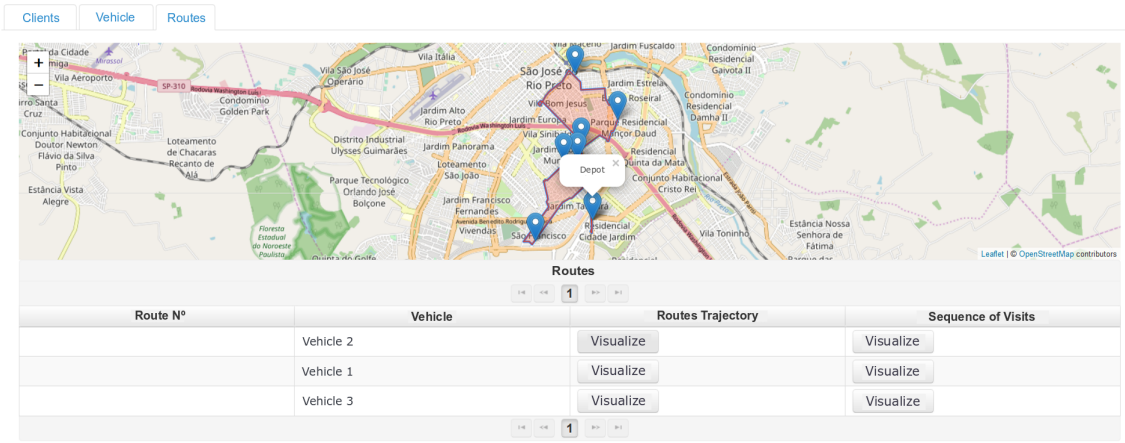



(c) Vehicle insertion.

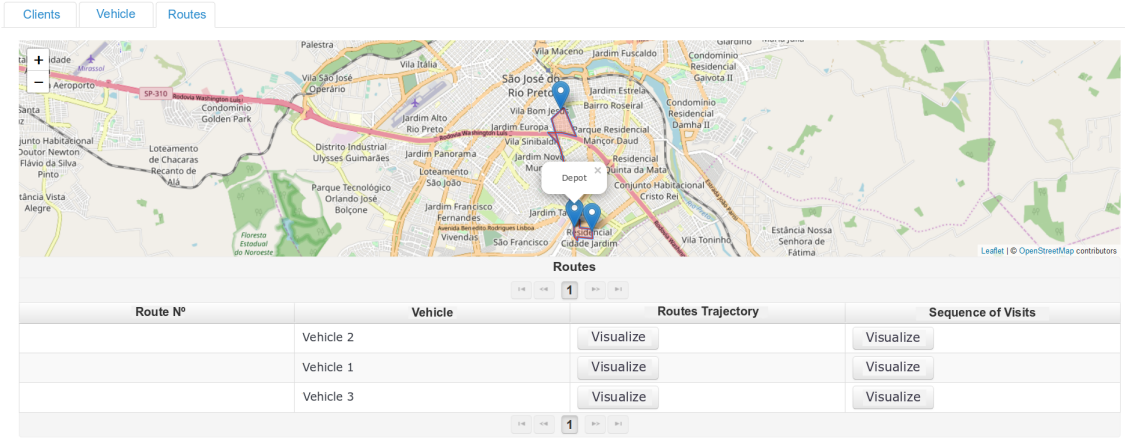
Figure 7: Interface for a routing plan registration.




(a) Vehicle 1;




(b) Vehicle 2;




(c) Vehicle 3.

Figure 8: Interface showing the optimal routes for vehicles 1 to 3.

## 6. Conclusions

We presented compact formulations and an effective metaheuristic for a practical variant of the pickup and delivery problem with time windows, including a heterogeneous fleet, multiple products and split pickup and split delivery. We ran computational experiments using instances with up to 150 customers, adapted from the benchmark instances of the classic variant. The results indicate that the models are suitable for small-sized instances, while the metaheuristic was able to find good-quality solutions for all instances within reasonable times. An open-source, publicly available web interface was developed to facilitate the use of the proposed formulations and metaheuristic. It can be a valuable tool in the decision-making processes involving pickup and delivery activities and can be used as framework for other variants of the vehicle routing problem. An interesting direction for future research is improving the metaheuristic by adding new local search heuristics as well as combining it with other exact approaches, such as branch-and-cut and branch-and-price methods, leading to an exact hybrid method [23].

## 7. Acknowledgments

This research was supported by the São Paulo Research Foundation (FAPESP) [grant numbers 19/23596-2, 16/24763-1, 16/01860-1, 13/07375-0]; the National Council for Scientific and Technological Development (CNPq-Brazil) [grant number 313220/2020-4]; and Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brazil (CAPES) - Finance Code 001.

## References

[1] P. Toth, D. Vigo, Vehicle Routing: Problems, Methods and Applications, Second ed., MOS-SIAM Series in Optimization, 2014.

[2] K. Braekers, K. Ramaekers, I. Van Nieuwenhuyse, The vehicle routing problem: State of the art classification and review, Computers & Industrial Engineering 99 (2016) 300–313.

[3] M. A. Mohammed, M. K. A. Ghani, R. I. Hamed, S. A. Mostafa, M. S. Ahmad, D. A. Ibrahim, Solving vehicle routing problem by using improved genetic algorithm for optimal solution, Journal of Computational Science 21 (2017) 255–262.

[4] P. Munari, A. Alvarez, Aircraft routing for on-demand air transportation with service upgrade and maintenance events: Compact model and case study, Journal of Air Transport Management 75 (2019) 75 – 84.

[5] R. Goel, R. Maini, S. Bansal, Vehicle routing problem with time windows having stochastic customers demands and stochastic service times: Modelling and solution, Journal of Computational Science 34 (2019) 1–10.

[6] M. Sigurd, D. Pisinger, M. Sig, Scheduling transportation of live animals to avoid the spread of diseases, Transportation Science 38 (2004) 197–209.

[7] J.-F. Cordeau, G. Laporte, The dial-a-ride problem: models and algorithms, Annals of Operations Research 153 (2007) 29–46.

[8] Y. Qu, J. F. Bard, The heterogeneous pickup and delivery problem with configurable vehicle capacity, Transportation Research Part C: Emerging Technologies 32 (2013) 1–20. doi:10.1016/j.trc.2013.03.007.

[9] M. Battarra, J.-F. Cordeau, M. Iori, Pickup-and-delivery problems for goods transportation, in: P. Toth, D. Vigo (Eds.), Vehicle routing: Problems, methods, and applications, MOS/SIAM Ser Optim, 2014, pp. 161–191.

[10] M. G. S. Furtado, P. Munari, R. Morabito, The pickup and delivery problem with time windows in the oil industry: model and branch-and-cut methods, Gestão & Produção 24 (2017) 501–513.

[11] M. G. S. Furtado, P. Munari, R. Morabito, Pickup and delivery problem with time windows: A new compact two-index formulation, Operations Research Letters 45 (2017) 334–341. doi:10.1016/J.ORL.2017.04.013.

[12] S. Ropke, J.-F. Cordeau, G. Laporte, Models and branch-and-cut algorithms for pickup and delivery problems with time windows, Networks 49 (2007) 258–272.

[13] Q. Lu, M. Dessouky, An exact algorithm for the multiple vehicle pickup and delivery problem, Transportation Science 38 (2004) 503–514.

[14] S. Ropke, J.-F. Cordeau, Branch and cut and price for the pickup and delivery problem with time windows, Transportation Science 43 (2009) 267–286.

[15] R. Baldacci, E. Bartolini, A. Mingozzi, An exact algorithm for the pickup and delivery problem with time windows, Operations Research 59 (2011) 414–426.

[16] A. Bettinelli, A. Ceselli, G. Righini, A branch-and-price algorithm for the multi-depot heterogeneous-fleet pickup and delivery problem with soft time windows, Mathematical Programming Computation 6 (2014) 171–197. doi:10.1007/s12532-014-0064-0.

[17] Y. Qu, J. F. Bard, A branch-and-price-and-cut algorithm for heterogeneous pickup and delivery problems with configurable vehicle capacity, Transportation Science 49 (2015) 254–270. doi:10.1287/trsc.2014.0524.

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

[19] Q. Lu, M. M. Dessouky, A new insertion-based construction heuristic for solving the pickup and delivery problem with time windows, European Journal of Operational Research 175 (2006) 672–687.

[20] R. Bent, P. V. Hentenryck, A two-stage hybrid algorithm for pickup and delivery vehicle routing problems with time windows, Computers and Operations Research 33 (2006) 875–893. doi:10.1016/j.cor.2004.08.001.

[21] H. Li, A. Lim, A metaheuristic for the pickup and delivery problem with time windows, in: Proceedings 13th IEEE International Conference on Tools with Artificial Intelligence. ICTAI 2001, IEEE Comput. Soc, 2001, pp. 160–167. doi:10.1109/ICTAI.2001.974461.

[22] J. F. Ehmke, A. Steinert, D. C. Mattfeld, Advanced routing for city logistics service providers based on time-dependent travel times, Journal of Computational Science 3 (2012) 193–205.

[23] A. Álvarez, P. Munari, An exact hybrid method for the vehicle routing problem with time windows and multiple deliverymen, Computers & Operations Research 83 (2017) 1–12.

[24] R. Z. Ríos-Mercado, J. F. López-Pérez, A. Castrillón-Escobar, A GRASP for a multi-depot multi-commodity pickup and delivery problem with time windows and heterogeneous fleet in the bottled beverage industry, in: D. Pacino, S. Voß, R. M. Jensen (Eds.), Computational Logistics: 4th International Conference, ICCL 2013, Copenhagen, Denmark, September 25-27, 2013. Proceedings, Springer Berlin Heidelberg, 2013, pp. 143–157. doi:10.1007/978-3-642-41019-2_11.

[25] Y. Qu, J. F. Bard, A GRASP with adaptive large neighborhood search for pickup and delivery problems with transshipment, Computers and Operations Research 39 (2012) 2439–2456. doi:10.1016/j.cor.2011.11.016.

[26] V. Ghilas, E. Demir, T. Van Woensel, An adaptive large neighborhood search heuristic for the Pickup and

Delivery Problem with Time Windows and Scheduled Lines, Computers and Operations Research 72 (2016) 12–30.

[27] M. Nowak, O. Ergun, I. I. I. White Chelsea C., Pickup and Delivery with Split Loads, Transportation Science 42 (2008) 32–43.

[28] M. N. Haddad, R. Martinelli, T. Vidal, S. Martins, L. S. Ochi, M. J. F. Souza, R. Hartl, Large neighborhood-based metaheuristic and branch-and-price for the pickup and delivery problem with split loads, European Journal of Operational Research 270 (2018) 1014 – 1027. doi:10.1016/j.ejor.2018.04.017.

[29] R. Lahyani, M. Khemakhem, F. Semet, Rich vehicle routing problems: From a taxonomy to a definition, European Journal of Operational Research 241 (2015) 1–14. doi:10.1016/j.ejor.2014.07.048.

[30] S. Irnich, M. Schneider, D. Vigo, Chapter 9: Four variants of the vehicle routing problem, in: Vehicle Routing: Problems, Methods, and Applications, Second Edition, SIAM, 2014, pp. 241–271.

[31] T. Henke, M. G. Speranza, G. Wäscher, The multi-compartment vehicle routing problem with flexible compartment sizes, European Journal of Operational Research 246 (2015) 730–743.

[32] F. Cornillier, F. F. Boctor, G. Laporte, J. Renaud, Discrete Optimization A heuristic for the multi-period petrol station replenishment problem, European Journal of Operational Research 191 (2008) 295–305.

[33] R. Lahyani, L. C. Coelho, M. Khemakhem, G. Laporte, A multi-compartment vehicle routing problem arising in the collection of olive oil in Tunisia, Omega 51 (2015) 1–10. doi:10.1016/j.omega.2014.08.007.

[34] G. Paredes-Belmar, V. Marianov, A. Bronfman, C. Obreque, A. Lüer-Villagra, A milk collection problem with blending, Transportation Research Part E: Logistics and Transportation Review 94 (2016) 26–43.

[35] C. Archetti, M. G. Speranza, Vehicle routing problems with split deliveries, International Transactions in Operational Research 19 (2012) 3–22.

[36] P. Munari, M. Savelsbergh, Compact Formulations for Split Delivery Routing Problems, Technical Report, 7500, Operations Research Group, Production Engineering Department, Federal University of Sao Carlos, 2019.

[37] P. Munari, M. Savelsbergh, A column generation-based heuristic for the split delivery vehicle routing problem with time windows, SN Operations Research Forum 1 (2020) 1–24.

[38] J. M. Belenguer, E. Benavent, A cutting plane algorithm for the capacitated arc routing problem, Computers & Operations Research 30 (2003) 705–728.

[39] C. Archetti, N. Bianchessi, S. Irnich, M. G. Speranza, Formulations for an inventory routing problem, International Transactions in Operational Research 21 (2014) 353–374.

[40] N. Bianchessi, S. Irnich, Branch-and-cut for the split delivery vehicle routing problem with time windows, Transportation Science 53 (2019) 442–462.

[41] G. Schrimpf, J. Schneider, H. Stamm-Wilbrandt, G. Dueck, Record Breaking Optimization Results Using the Ruin and Recreate Principle, Journal of Computational Physics 159 (2000) 139–171.

[42] S. Kirkpatrick, C. D. Gelatt Jr, M. P. Vecchi, Optimization by Simulated Annealing, Science 220 (1983) 671–680.

[43] A. Alvarez, P. Munari, R. Morabito, Iterated local search and simulated annealing algorithms for the inventory routing problem, International Transactions in Operational Research 25 (2018) 1785–1809.

[44] M. López-Ibáñez, J. Dubois-Lacoste, L. Pérez Cáceres, M. Birattari, T. Stützle, The irace package: Iterated racing for automatic algorithm configuration, Operations Research Perspectives 3 (2016) 43–58. doi:10.1016/j.orp.2016.09.002.

[45] D. Luxen, C. Vetter, Real-time routing with openstreetmap data, in: Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, GIS '11, ACM, New York, NY, USA, 2011, pp. 513–516. doi:10.1145/2093973.2094062.