



## Discrete Optimization

## An ILS-based algorithm to solve a large-scale real heterogeneous fleet VRP with multi-trips and docking constraints

V. N. Coelho<sup>a,\*</sup>, A. Grasas<sup>b</sup>, H. Ramalhinho<sup>c</sup>, I. M. Coelho<sup>d,f</sup>, M. J. F. Souza<sup>e</sup>, R. C. Cruz<sup>a</sup><sup>a</sup> Department of Control and Automation Engineering, Federal University of Ouro Preto, Ouro Preto, MG, 35400-000, Brazil<sup>b</sup> Department of Marketing, Operations and Supply, EADA Business School, Barcelona, 08011, Spain<sup>c</sup> Department of Economics and Business, Universitat Pompeu Fabra, Barcelona, 08005, Spain<sup>d</sup> Institute of Computing, Fluminense Federal University, Niterói, 24210-240, Brazil<sup>e</sup> Department of Computer Science, Federal University of Ouro Preto, Ouro Preto, MG, 35400-000, Brazil<sup>f</sup> Department of Computer Science, State University of Rio de Janeiro, Rio de Janeiro, Brazil

## ARTICLE INFO

## Article history:

Received 25 September 2014

Accepted 16 September 2015

Available online 1 October 2015

## Keywords:

VRP

Heterogeneous fleet

Multiple trips

Docking constraints

Iterated local search

## ABSTRACT

Distribution planning is crucial for most companies since goods are rarely produced and consumed at the same place. Distribution costs, in addition, can be an important component of the final cost of the products. In this paper, we study a VRP variant inspired on a real case of a large distribution company. In particular, we consider a VRP with a heterogeneous fleet of vehicles that are allowed to perform multiple trips. The problem also includes docking constraints in which some vehicles are unable to serve some particular customers, and a realistic objective function with vehicles' fixed and distance-based costs and a cost per customer visited. We design a trajectory search heuristic called GILS-VND that combines Iterated Local Search (ILS), Greedy Randomized Adaptive Search Procedure (GRASP) and Variable Neighborhood Descent (VND) procedures. This method obtains competitive solutions and improves the company solutions leading to significant savings in transportation costs.

© 2015 Elsevier B.V. and Association of European Operational Research Societies (EURO) within the International Federation of Operational Research Societies (IFORS). All rights reserved.

## 1. Introduction

Vehicle routing problems (VRP) seek to find routes to deliver goods from a central depot to a set of geographically dispersed customers. These problems, faced by many companies, are crucial in distribution and logistics due to the need of finding cost-effective routes providing high customer satisfaction. The classical routing problem, first proposed by Dantzig and Ramser (1959) and known as Capacitated VRP, has the objective of minimizing the total distance traveled by a homogeneous fleet of vehicles to serve the demands of all customers. Although this problem has been studied for more than five decades (Laporte, 2009), real applications remain a challenge. They feature a variety of operational restrictions and rules that complicate the problem and may have a significant impact on the solution. These additional considerations may affect customers, depots, and vehicles, for example.

In this paper, we study a real VRP variant of a major distribution company in Europe that serves around 400 customers. This version of the problem addresses the following considerations:

1. Limited heterogeneous fleet of vehicles: the company owns a fleet composed of different vehicle types;
2. Possibility of vehicles performing multiple trips;
3. Docking constraints that restrict certain customers to be served by certain types of vehicles;
4. For each vehicle, a fixed and variable (transportation) cost.

This problem is a variant of the Heterogeneous Fleet Multi-trip Vehicle Routing Problem (HFMVRP) introduced by Prins (2002). This new variant includes docking constraints and a different objective function. The goal is to minimize the total cost composed of (i) a fixed cost for using each vehicle, (ii) a fixed cost per customer visited, and (iii) a variable vehicle-dependent cost per distance traveled. Besides minimizing total distribution costs, for managerial reasons the company is also concerned about three other routing indicators, namely, (i) the total number of routes employed, (ii) the total distance traveled, and (iii) the vehicles' idle capacity. Their purpose, apart from saving costs, is to have the least number of routes with full truckload vehicles.

\* Corresponding author. Tel.: +55 3135514407.

E-mail addresses: [vncoelho@gmail.com](mailto:vncoelho@gmail.com) (V.N. Coelho), [agrasas@eada.edu](mailto:agrasas@eada.edu) (A. Grasas), [helena.ramalhinho@upf.edu](mailto:helena.ramalhinho@upf.edu) (H. Ramalhinho), [imcoelho@ic.uff.br](mailto:imcoelho@ic.uff.br) (I.M. Coelho), [marcone@iceb.ufop.br](mailto:marcone@iceb.ufop.br) (M.J.F. Souza), [phaelcarlos@gmail.com](mailto:phaelcarlos@gmail.com) (R.C. Cruz).

The HFMVRP is an  $\mathcal{NP}$ -Hard problem, and, as such, exact methods have restricted applicability to obtain good solutions. Heuristic methods, like the one presented in this paper, are the most common approach to solve this type of problems. In particular, we use a heuristic algorithm, the GILS-VND, that combines three different procedures:

1. Iterated Local Search (ILS) (Lourenço, Martin, & Stützle, 2003; Stützle, 2006);
2. Greedy Randomized Adaptive Search Procedure (GRASP) (Feo & Resende, 1995; Resende & Ribeiro, 2010);
3. Variable Neighborhood Descent (VND) (Mladenović & Hansen, 1997).

We test our algorithm using real instances provided by the company. The algorithm proved to be fast and reliable, and the solutions obtained were better than those implemented by the company in all instances and dimensions. Overall, the major contributions of the current work are:

- The study of a routing problem of a real company that includes docking constraints, a heterogeneous fleet and multi-trips, and with a realistic cost function based on distance, type of vehicle and customers visited.
- The design of a trajectory search metaheuristic combining ILS, GRASP and VND.
- The development of a new multi-trip constructive method inspired by the Clarke and Wright Savings procedure.
- The application of efficient Auxiliary Data Structures to optimize the search process in the proposed neighborhood structures.

The remainder of this paper is organized as follows. Section 2 reviews the literature on heterogeneous VRPs. Section 3 defines formally the HFMVRP with docking constraints. Section 4 describes the GILS-VND algorithm used to solve this problem. Section 5 presents some computational experiments, and finally Section 6 draws the final considerations.

## 2. Heterogeneous VRPs

VRPs with heterogeneous fleet (HVRP) can be divided according different features (Penna, Subramanian, & Ochi, 2013), including the vehicle availability (limited or unlimited) and vehicle costs (fixed or variable). When the fleet is limited, the number of vehicles and their capacity are known beforehand, and solution routes must consider this availability. In the case of unlimited fleet, however, the required number of vehicles to meet customer demands is unknown initially, and the problem must determine the fleet composition considering the vehicles' cost and capacity.

To the best of our knowledge, the first paper in the literature that involves an unlimited fleet with fixed costs was proposed by Golden, Assad, Levy, and Gheysens (1984). This problem is also referred as the Fleet Size and Mix VRP. The authors designed two heuristic methods to solve the problem: one based on best insertion and the other based on the classical Clarke and Wright Savings (CWS) heuristic (Clarke & Wright, 1964). The latter outperformed the former. They also developed a mathematical formulation for the variant with dependent costs, and obtained the first lower bounds for the VRP with unlimited fixed fleet. More studies on HVRPs with unlimited fleet came thereafter. Gendreau, Laporte, Musaraganyi, and Taillard (1999) included investment costs in the medium term and short-term operating costs that fluctuated according to the specific customers attended per day. The authors suggested an algorithm based on Tabu Search (TS) with a tour construction phase and an improvement phase that considered variable costs. They, however, assumed Euclidean problems only, where nodes were located in the same plane. Choi and Tcha (2007) obtained lower bounds for all variants of the unlimited fleet problem using a column generation approach based on the set covering problem. Baldacci and Mingozzi (2009) proposed a variant based on

a set partitioning problem that used bounds provided by a procedure based on the Linear and Lagrangian relaxation. The procedure was applied to solve the main variants of the problem involving limited and unlimited fleet, with costs and dependent variables. The proposed method was able to solve instances with up to 100 customers, presenting itself as the state-of-the-art exact algorithm applied to the problem. Brandão (2009) followed the basic ideas of Gendreau et al. (1999) using a deterministic TS algorithm for the fleet size and mix VRP.

Among the heuristic approaches presented in the literature, noteworthy are those based on Evolutionary Algorithms. Ochi, Vianna, Drummond, and Victor (1998) developed an algorithm that combines a Genetic Algorithm (GA) with Scatter Search. Liu, Huang, and Ma (2009) proposed a hybrid GA with a hybrid local search procedure. Prins (2009) presented two Memetic Algorithms. The first approach used a chromosome encoded as a giant tour, and a split procedure that performed the optimal distribution of vehicles and routes. The second algorithm used distance calculation strategies in order to diversify the search in the solution space. Tütüncü (2010) proposed a visual interactive approach based on a greedy randomized adaptive memory programming search algorithm to study an HVRP variant with backhauls. Penna et al. (2013) devised an algorithm based on ILS which used a random VND in the local search phase. More studies on different variants of the HVRP are compiled by Baldacci, Battarra, and Vigo (2008), Imran, Salhi, and Wassan (2009) and Vidal, Crainic, Gendreau, and Prins (2013).

The HVRP is gaining attention from researchers due to its applicability in real cases. In the past years, a variety of papers, including this one, have addressed more realistic setups involving a heterogeneous fleet with additional constraints. Belfiore, Yoshizaki, and Tsugunobu (2009) studied a real-life HVRP with time windows and split deliveries in a major Brazilian retail group. The authors generated some initial solutions that were improved by scatter search. Kritikos and Ioannou (2013) addressed an HVRP with time windows, in which some vehicles were loaded above their nominal capacity (overloads). The authors developed a sequential insertion heuristic with a component in the selection criteria of the non-routed customers and a penalty in the objective function for overloads. Leung, Zhang, Zhang, Hua, and Lim (2013) analyzed a two-dimensional loading HVRP using a simulated annealing with a heuristic local search. Ribeiro, Desaulniers, Desrosiers, Vidal, and Vieira (2014) studied the workover rig routing problem, which can be seen as a variation of the VRP without a depot. In the workover rig routing problem, routes of a heterogeneous fleet of rigs need to be found to minimize the total production loss of the wells over a finite horizon. The authors proposed and compared four heuristics: a variable neighborhood search heuristic, a branch-price-and-cut heuristic, an adaptive large neighborhood search heuristic and a hybrid GA. Another application of the adaptive large neighborhood search method was presented by Amorim, Parragh, Sperandio, and Almada-Lobo (2014). The authors considered a real heterogeneous fleet site dependent VRP with multiple time windows faced by a Portuguese food distribution company. Jiang, Ng, Poh, and Teo (2014) also studied an HVRP with time windows and employed a two-phase TS algorithm. Dayarian, Crainic, Gendreau, and Rei (2015) designed a branch-and-price methodology to tackle a real-life milk collection problem with heterogeneous fleet, multi-depot and other resource constraints. Another real application was handled by de Armas and Melián-Batista (2015), who proposed a variable neighborhood search algorithm to solve an HVRP with multiple and soft time windows and customers' priorities. Although the problems exposed in this paragraph and our problem have different constraints, most of the algorithms designed to solve them rely on heuristic searches like ILS or variable neighborhood search.

The HVRP variants that most resemble the variant we study in this paper are by Prins (2002) and Caceres-Cruz, Grasas, Ramalhinho, and Juan (2014). The reader will find a detailed comparison of both in

Caceres-Cruz et al. (2014, Section 3). Our problem and these other two problems are routing problems that include a heterogeneous fleet with the possibility of performing multiple trips (HFMVRP). On one hand, Prins (2002) studied a large-scale real case of a French furniture manufacturer with 775 stores. In Prins' problem, a time restriction of 5 hours on routes was imposed and the largest demand of a store could not exceed the smallest vehicle's capacity. The author developed a CWS-based algorithm with two heuristics, New Merge Heuristic (MER) and MER2, to address the single- and the multi-trip problems, respectively. The algorithm solutions were improved via a steepest descent local search and a tabu search with a simple bi-objective approach: minimizing the total duration of all trips and the number of vehicles. The results obtained outperformed the solutions used by the furniture company. On the other hand, Caceres-Cruz et al. (2014) also studied an HVRP with multi-trips inspired on the same distribution company that is discussed in the present work. The authors built a randomized hybrid algorithm based on the CWS heuristic, called Rand-MER. The Rand-MER is a biased randomized version of the MER algorithm that also uses local search methods based on cache and splitting techniques. The authors carried out extensive numerical experiments to compare the performance of the MER and the Rand-MER algorithms. To do so, they adapted the instances found in Prins (2002), Golden et al. (1984) and Li, Golden, and Wasil (2007) to make an objective comparison. They also modified the MER algorithm to run 10 company instances analyzed in our paper. The Rand-MER algorithm was able to outperform MER in all experiments conducted. Unlike our paper, both settings considered by Prins (2002) and Caceres-Cruz et al. (2014) did not consider docking constraints and only included distance-based costs in the objective functions.

### 3. Problem definition

The HFMVRP described in this paper can be defined over an undirected graph  $G = (V, E)$ , where  $V = \{0, 1, \dots, n\}$  and  $E = \{(i, j) | i, j \in V, i < j\}$  represent the vertices and the edges of the graph, respectively. The depot is denoted by 0 and vertices  $i \in V \setminus \{0\}$  represent the  $n$  customers, each one with a nonnegative demand  $d_i$ . Each edge  $(i, j) \in E$  has an associated nonnegative cost or distance  $c_{ij}$ . There is a fleet  $T$  composed of  $m$  different types of vehicles, i.e.,  $T = \{1, \dots, m\}$ . For each  $t \in T$  there are  $m_t$  available vehicles with capacity  $q_t$  (in boxes of products), fixed cost  $cf_t$  per vehicle used, and variable cost  $cd_t$  per distance traveled. There is also a fixed cost  $cc_t$  incurred per customer visited.

We let  $S = \{(r, t) | t \in T, r = (v_0, v_1, \dots, v_{n(r)+1})\}$  be a set of valid routes, where  $v_i \in V$  ( $0 \leq i \leq n(r) + 1$ ),  $n(r)$  denotes the number of customers visited in route  $r$  and  $t$  is the vehicle type associated with the route. All routes start and end at the depot, so for each route  $r$ , we have  $v_0 = v_{n(r)+1} = 0$ . Therefore:

- The route total demand is:  $Q_r^t = \sum_{i=1}^{n(r)} d_{v_i}$ ;
- The route total cost is:  $C_r^t = cf_t + n(r) \times cc_t + cd_t \sum_{i=0}^{n(r)} c_{v_i, v_{i+1}}$ ;
- The route residual capacity is:  $E_r^t = q_t - Q_r^t$ .

This set of features characterizes the HFMVRP tackled in this work. The problem seeks to build a set  $S^*$  that minimizes the total cost function given by:

$$TC = \sum_{(r,t) \in S} C_r^t \quad (1)$$

Note that the cost of a route,  $C_r^t$ , is composed of three cost terms. For convenience, we group these terms by type for all routes and express the objective function as  $TC = C_F + C_C + C_D$  where:

- $C_F = \sum_{(r,t) \in S} cf_t$  is the total fixed cost of the vehicles used.
- $C_C = \sum_{(r,t) \in S} n(r) \times cc_t$  is the total cost of the customers visited.
- $C_D = \sum_{(r,t) \in S} cd_t \sum_{i=0}^{n(r)} c_{v_i, v_{i+1}}$  is the total cost of the distance traveled by all vehicles.

A valid route must satisfy the following criteria:

1. Each route must start and end at the depot.
2. Each customer is assigned to exactly one route.
3. Each customer must be compatible with the vehicle type assigned to its route, i.e., given a route  $(r, t)$ ,  $\forall v_i \in r$ ,  $comp(t, v_i) = 1$ , where  $comp(t, v_i)$  is equal to 1 if the vehicle type  $t$  can serve customer  $v_i$ , and 0 otherwise.
4. The sum of customer demands in the route cannot exceed the maximum capacity of the vehicle type  $t$  assigned to that route, i.e.,  $Q_r^t \leq q_t$ .
5. Each vehicle type can perform one or two routes, that is, it can be a single- or a multi-trip vehicle, respectively.

Fig. 1 displays a solution example for an HFMVRP with 12 customers, 2 types of vehicles, and only one vehicle per type. The “Parameters” table shows the parameter values for each vehicle type. The “Compatibility” table indicates the values of  $comp(t, i)$ , for  $t \in \{A, B\}$  and  $i \in V \setminus \{0\}$ . Note that a type B vehicle cannot visit customers 8 or 10. The “Solution” table displays the resulting routes, with the total demand served, distance traveled, total cost and residual capacity. Note that the vehicle type A performs two routes, 1 and 3. The graph shows the sequence of the three routes: vertices have the corresponding customer demand, and arcs the corresponding distance next to them. For this example,  $C_F = 50$ ,  $C_C = 17$ ,  $C_D = 68$ , and the total cost is  $TC = 135$ .

### 4. Methodology

#### 4.1. The GILS-VND algorithm

The algorithm proposed in this paper, dubbed GILS-VND, combines an Iterated Local Search (ILS), a Greedy Randomized Adaptive Search Procedure (GRASP), and a Variable Neighborhood Descent (VND). Its pseudocode is outlined in Algorithm 1. It requires the following input parameters: (i) *GRASPmax* is the number of GRASP executions to construct an initial solution, (ii) *IterMax* is the maximum number of iterations performed at a given perturbation level, (iii)  $\gamma$  restricts the size of the candidate list, and (iv) the objective function  $f(\cdot)$  (Eq. (1) defined in Section 3). The GILS-VND

---

#### Algorithm 1: GILS-VND.

---

**Input:** *GRASPmax*, *IterMax*,  $\gamma$ , Function  $f(\cdot)$

**Output:** Solution  $s$

---

```

1  $s_0 \leftarrow$  best solution in GRASPmax iterations of the procedure
  BuildInitialSavingsSolution( $\gamma$ )
2  $s^* \leftarrow$  VND( $s_0, f$ )
3  $p \leftarrow 0$ 
4 while stop criterion not satisfied do
5    $iter \leftarrow 0$ 
6   while  $iter < IterMax$  and stop criterion not satisfied do
7      $s' \leftarrow$  Refine( $s^*, p, f$ )
8     if  $f(s') < f(s^*)$  then
9        $s^* \leftarrow s'$ ;
10       $p \leftarrow 0$ ;
11       $iter \leftarrow 0$ 
12   end
13   else
14      $iter \leftarrow iter + 1$ 
15   end
16 end
17  $p \leftarrow p + 1$ 
18 end
19 return  $s$ 
```

---

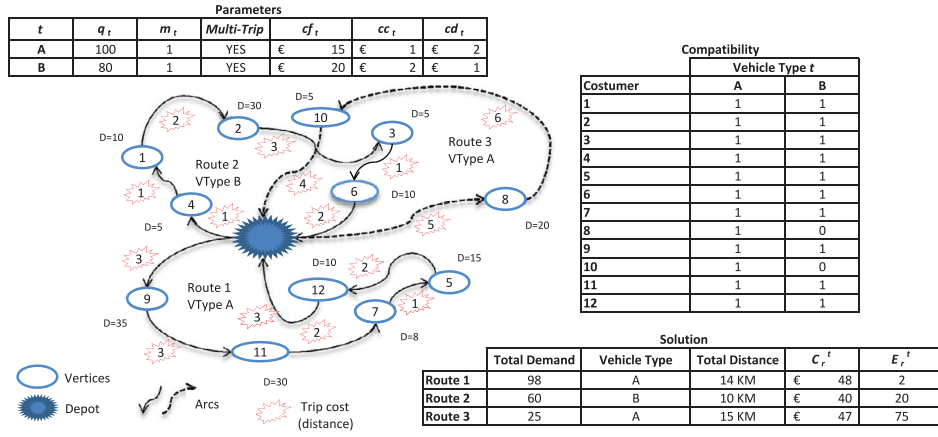


Fig. 1. HFMVRP example.

**Algorithm 2: BuildInitialSavingsSolution.**

**Input:**  $\gamma$ , Function  $f(\cdot)$ , set of vehicle types  $T$ , set of customers  $V$   
**Output:** Solution  $s$

```

1 feasible ← false;
2 while feasible ≠ true do
3   s ← empty solution
4   allocated ← set of customers  $V$  to be visited
5   forall  $vType \in T$  do
6      $vType \leftarrow$  random vehicle type
7      $vTypeDemands \leftarrow$  customer demands that can be served
      by vehicle type  $vType$ 
8      $vTypeRoutes \leftarrow$  Savings( $vType$ ,  $vTypeDemands$ )
9     CL ← set of available routes  $vTypeRoutes$  ordered by its
      residual capacity
10    foreach  $v \in vType$  and  $|CL| \neq 0$  do
11      RCL ← first  $\gamma \times |CL|$  elements of CL
12      s ← allocate a random route  $\in RCL$  to vehicle  $v$ 
13      update allocated with customers served by vehicle  $v$ 
      and erase route from  $vTypeRoutes$ 
14      CL ← set of available routes  $vTypeRoutes$  ordered by
      its residual capacity
15    end
16  end
17  if there are customers  $\in$  allocated to be visited then
18     $vTypeMT \leftarrow$  random vehicle type able to perform
      multi-trip
19     $vTypeDemands \leftarrow$  customer demands that can be served
      by vehicle type  $vTypeMT$ 
20     $vTypeRoutes \leftarrow$  Savings( $vTypeMT$ ,  $vTypeDemands$ )
21    s ← allocate all routes  $\in vTypeRoutes$  to vehicles of type
       $vTypeMT$ 
22    update allocated with new attended customers
23  end
24  if all customers  $\in$  allocated have been visited then
25    feasible = true
26  end
27 end
28 return s

```

**Algorithm 3: VND.**

**Input:**  $r^{Intra}$  intra-route neighborhood structures in random order  
**Input:**  $r^{Inter}$  inter-route neighborhood structures in random order  
**Input:** Solution  $s_0$  and Function  $f(\cdot)$   
**Output:** Solution  $s$  with possibly better quality than initial solution  $s_0$  according to Function  $f(\cdot)$

```

1 s ← s0
2  $k^{Inter} \leftarrow 1$ 
3 while  $k^{Inter} \leq |r^{Inter}|$  do
4   Find the best neighbor  $s' \in N^{(k^{Inter})}(s)$ 
5   if  $f(s') < f(s)$  then
6     s ← s'
7      $k^{Inter} \leftarrow 1$ 
8      $k^{Intra} \leftarrow 1$ 
9     while  $k^{Intra} \leq |r^{Intra}|$  do
10      Find the best neighbor  $s' \in N^{(k^{Intra})}(s)$ 
11      if  $f(s') < f(s)$  then
12        s ← s'
13         $k^{Intra} \leftarrow 1$ 
14      end
15    else
16      setLocalOptimum(s, k)
17       $k^{Intra} \leftarrow k^{Intra} + 1$ 
18    end
19  end
20 end
21 else
22   setLocalOptimum(s, k)
23    $k^{Inter} \leftarrow k^{Inter} + 1$ ;
24 end
25 end
26 return s

```

algorithm works as follows. Line 1 calls the BuildInitialSavingsSolution procedure to obtain the best initial solution after GRASPmax runs (Algorithm 2). Line 2 calls the VND procedure to perform a local search (Algorithm 3). Line 3 initializes  $p$ , a variable that regulates the “power” of diversification, and lines 4–18 perform the main ILS loop

while the stopping condition is not satisfied. Within the ILS, line 7 calls the Refine procedure that perturbs the solution (Algorithm 4). Line 19 then returns the final solution.

The BuildInitialSavingsSolution procedure (Algorithm 2) uses the CWS algorithm (line 8) adapted to the multi-trip case. At each iteration (lines 5–16), the order of vehicle types is chosen randomly. Thus, different sets of routes can be generated according to the capacity of each vehicle type. A restricted candidate list is created (line 11) by sorting the routes in non-decreasing order of idle capacity, that



is, routes with total demand closer to the capacity of the current allocated vehicle type,  $vType$ , are listed first. In this sense, routes that have its demand equal to the capacity of the vehicle are the most coveted. The input parameter  $\gamma$  regulates the size of the candidate list. All available vehicles of the vehicle type  $vType$  are assigned to a route until  $vTypeRoutes$  is empty or vehicles of that type are already allocated (line 10). If there are still customers to be allocated after this first phase, a vehicle type able to perform multi-trips is selected and all customers with demand lower or equal to that vehicle capacity are allocated and grouped according to the same Clarke and Wright Savings algorithm (lines 17 to 23). Line 22 updates the vector *allocated* with the clients attended by multi-trip vehicles of type  $vTypeMT$ . If customers remain still unassigned (line 24), the procedure is repeated.

---

**Algorithm 4:** Refine.

---

**Input:**  $r^{pert}$  perturbation neighborhoods in random order

**Input:** Initial solution  $s$ , Level  $p$  and Function  $f(\cdot)$

**Output:** Solution  $s$

```

1 for  $i \leftarrow 1$  To  $p + 2$  do
2    $k \leftarrow \text{SelectNeighborhood}(r^{pert})$ 
3    $s' \leftarrow \text{Shake}(s, k)$ 
4 end
5  $s \leftarrow \text{VND}(s', f)$ 
6 return  $s$ 

```

---

The VND procedure (Algorithm 3) performs a local search using the neighborhood structures described in Section 4.2. The exploration is done via inter-route movements (line 4) and intra-route movements (line 10). Lines 16 and 22 trigger the setLocalOptimum procedure, which sets neighborhoods as “local-optimum”. This mark is used by the auxiliary data structure  $\text{NeighborhoodStatus}[j][i]$ , described in Section 4.3.

The Refine procedure (Algorithm 4) takes the current solution,  $s^*$ , selects randomly the neighborhood structures,  $r^{pert}$  (line 2), and performs a shake (line 3). This is done iteratively (lines 1–4) according to the variable  $p$ . If a given solution is not improved for a number of *iterMax* iterations (line 6 of Algorithm 1), variable  $p$  is incremented (line 17 of Algorithm 1) so that  $p + 2$  random moves (*shakes*) will be applied to the current solution. This mechanism balances exploration against exploitation.

#### 4.2. Neighborhood structures

Five different neighborhood structures are applied to explore the solution space of the problem. The first three are intra-route movements while the last two are inter-route movements. It is important to note that movements that lead to infeasible solutions are not allowed.

**2-opt move:** A 2-opt move is an intra-route movement that consists in removing two non-adjacent arcs and inserting two new arcs, so that a new route is formed. Fig. 2 exemplifies the movement: edges (4, 6) and (8, 5) of Route 2 are removed and edges (4, 8) and (6, 5) are inserted instead. Note that an inversion took place involving customers 6, 16 and 8, and now the sequence is 8–16–6. For a symmetric problem, the total distance among these customers remains unaffected.

**Or-optk move:** An Or-optk move is an intra-route movement that consists in removing  $k$  consecutive customers from a given route and reinserting them into another position of the same route. This move is a generalization of the Or-opt proposed by Or (1976), in which the removal involves up to three consecutive customers only. Fig. 3 illustrates the movement with  $k = 1$ , where customer 5 is

**Table 1**  
HFMVRP solution.

| Solution representation<br>Route ( $r, t$ ) | Customers visited               |
|---|---------------------------------|
| (1,A)                                       | (0,9,11,7,5,12,0)               |
| (2,B)                                       | (0,4,1,2,3,6,0)                 |
| (3,A)                                       | (0,8,10,0)                      |
| Auxiliary Data Structures (ADSS)            |                                 |
| SumDemand[i]                                | $i = (1, 2, 3)$<br>(98, 60, 25) |
| MinDemand[i]                                | (8, 5, 5)                       |
| MaxDemand[i]                                | (35, 30, 20)                    |
| NeighborhoodStatus[j][i]                    |                                 |
| [2-opt][i]                                  | (1, 0, 1)                       |
| [Or-optk][i]                                | (1, 0, 1)                       |
| [Exchange][i]                               | (1, 0, 1)                       |
| [Shift(1,0)][i]                             | (1, 0, 1)                       |
| [Swap(1,1)][i]                              | (1, 0, 1)                       |

moved to the last position of Route 2. For this particular case when  $k = 1$ , the movement is also known as Reinsertion in the literature (Subramanian, Uchoa, & Ochi, 2013).

**Exchange move:** An Exchange move is an intra-route movement that consists in exchanging two customers in the same route. Fig. 4 shows an Exchange of customers 5 and 18 in Route 2.

**Shift(1,0) move:** A Shift(1,0) move is an inter-route movement that relocates a customer from one route to another.

**Swap(1,1) move:** A Swap(1,1) move is an inter-route movement that exchanges two customers from different routes.

All neighborhood structures are used as a perturbation strategy. The application of these moves occurs randomly with no improvement verification in the objective function. This mechanism is a key to diversify and explore the solution space (exploration-exploitation). After applying a given move, the  $\text{NeighborhoodStatus}[j][i]$  vector (described below) is updated.

#### 4.3. Auxiliary Data Structures (ADSS)

In order to intensify and optimize the search of the neighborhood structures, some Auxiliary Data Structures (ADSS) of Penna et al. (2013) have been adapted and applied to the multi-trip HFMVRP. A brief description is given below:

For  $i \in \{1, \dots, \#nRoutes\}$  and  $j \in \{1, \dots, \#nNeighborhoods\}$  the following data structures are used. Variable  $\#nRoutes$  indicates the total number of routes and  $\#nNeighborhoods$  the number of neighborhood structures (see Section 4.2).

- SumDemand[i]: it stores the sum of all customer demands assigned to route  $i$ .
- MinDemand[i]: it stores the minimum demand among all customers in route  $i$ .
- MaxDemand[i]: it stores the maximum demand among all customers in route  $i$ .
- NeighborhoodStatus[j][i]: it is a boolean value that indicates whether the neighborhood  $j$  is in a local optimum regarding route  $i$ . Upon a full application of all neighboring structures by a local search method, all routes are marked as “local-optimum”. When a solution is “shaked” (Line 3 of Algorithm 4), some “local-optimum” markers are removed from the routes that have been affected by that perturbation.

Table 1 displays the representation and the ADSS of the HFMVRP solution provided in Fig. 1. Note that route 2 is not in a local-optimum as seen in the values of  $\text{NeighborhoodStatus}[j][2]$ , so moves involving this route should be verified.

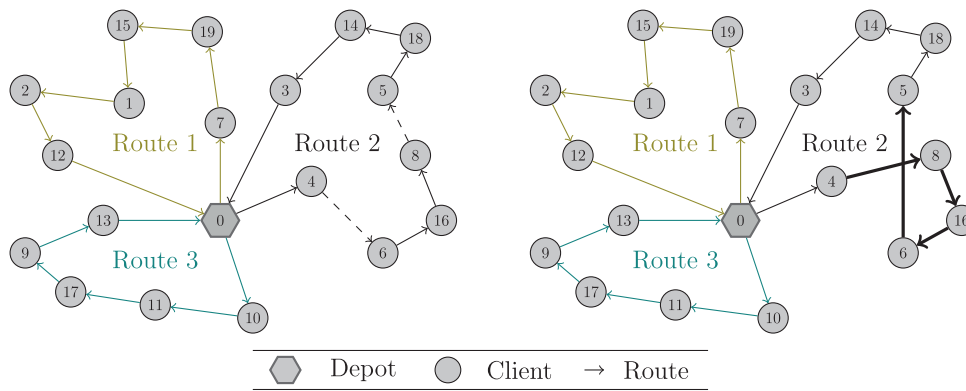


Fig. 2. Example of 2-opt move.

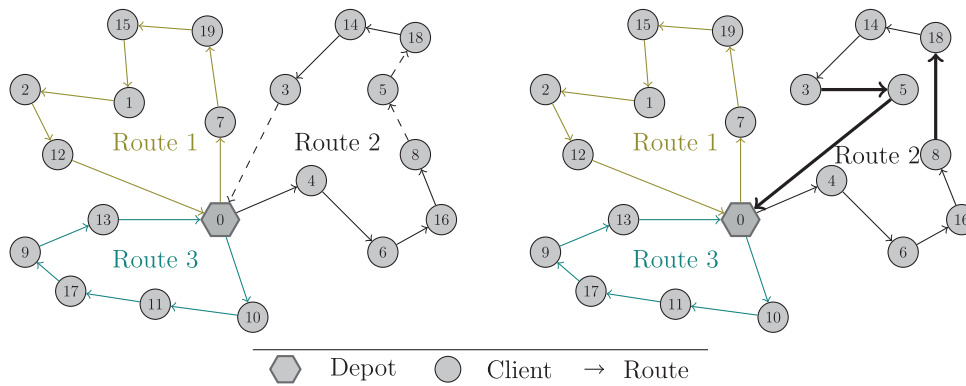


Fig. 3. Example of 0r-optk move.

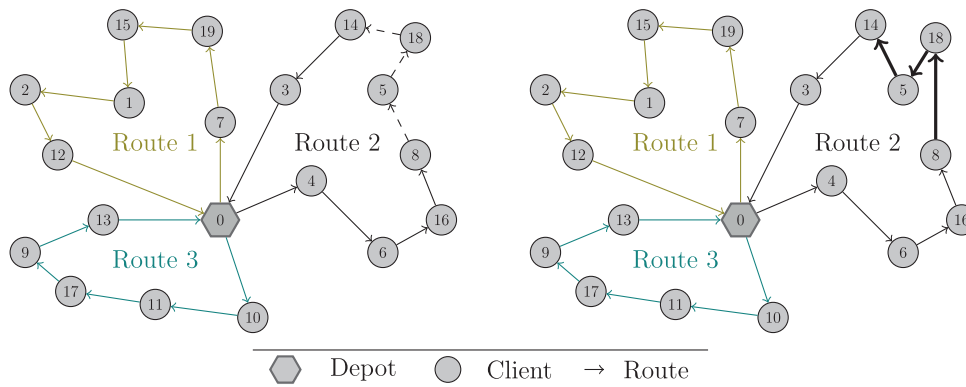


Fig. 4. Example of Exchange move.

## 5. Computational experiments

The GILS-VND algorithm was implemented in C++ with assistance from OptFrame framework (<http://sourceforge.net/projects/optframe/>). This optimization framework has been successfully applied in guiding the implementation of neighborhood structures (see Coelho et al., 2012). In general, frameworks are based on the researchers' experience with the implementation of multiple methods for different problems (Coelho et al., 2011). For instance, Souza et al. (2010) and Coelho et al. (2014) employed OptFrame to an open-pit-mining problem, and to load energy forecasting, respectively. The computational framework OptFrame has, besides standardized optimization structures, statistical and checking modules which are able to provide the ability of verifying the consistency of the new designed method (as will be described in Section 5.1).

The GILS-VND tests were conducted on a Pentium Core 2 Quad (Q6600), 2.4 GHZ with 8 GB of RAM, with operating system Ubuntu 10.10 Kernel 2.6.32-33, and compiled by g++ 4.5.2, using the Eclipse 3.1 IDE. The first step before running computational experiments was to calibrate the input parameters. The algorithm requires the number of iterations to obtain an initial solution (*GRASPmax*), the maximum number of iterations performed at a given perturbation level (*IterMax*), and  $\gamma$  that affects the RCL. The BuildInitialSavingsSolution procedure spends 40 milliseconds on average to build an initial solution (see Table 2 below). Thus, *GRASPmax* was set to be 300, that is, the algorithm generates all initial solutions in less than 15 seconds. Since the total running time will be set to 5 minutes (see below), 15 seconds seem reasonable for the initial solution generation. As for the maximum number of perturbations without improvement at each level of the ILS, we decided to use the same value as in Souza et al. (2010), i.e., *IterMax* = 5,000. Finally, we set  $\gamma$  = 0.4 because that renders a

**Table 2**  
CheckModule Output – computational times.

| Component                                 | #Tests | Average(milliseconds) |
|---|--------|-----------------------|
| Test 1: building an initial solution      |        |                       |
| Constructive                              | 30     | 40,568                |
| Test 2: update cost of the ADS            |        |                       |
| ADSManager                                | 94,885 | 0.1222                |
| Test 3: complete evaluation of a solution |        |                       |
| Evaluator                                 | 94,885 | 0.0385                |
| Test 4: cost of apply method              |        |                       |
| 2-Opt                                     | 256    | 0.0039                |
| Or-opt1                                   | 708    | 0.0039                |
| Exchange                                  | 1144   | 0.0035                |
| Shift(1,0)                                | 12,986 | 0.0074                |
| Swap(1,1)                                 | 33,402 | 0.0071                |
| Test 5: calculating the cost of a move    |        |                       |
| 2-Opt                                     | 128    | 0.0783                |
| 2-Opt-Optimized                           | 128    | 0.0014                |
| Or-opt1                                   | 354    | 0.0799                |
| Or-opt1-Optimized                         | 354    | 0.0014                |
| Exchange                                  | 572    | 0.0790                |
| Exchange-Optimized                        | 572    | 0.0015                |
| Shift(1,0)                                | 6493   | 0.0921                |
| Shift(1,0)-Optimized                      | 6493   | 0.0016                |
| Swap(1,1)                                 | 16,701 | 0.0917                |
| Swap(1,1)-Optimized                       | 16,701 | 0.0016                |

**Table 3**  
CheckModule Output – efficiency of the neighborhood structures.

| Average number of moves from neighborhood in 30 tests |                          |          |           |          |
|---|--------------------------|----------|-----------|----------|
| Neighborhood  | Valid neighborhood moves | Standard | Optimized | Imp. (%) |
| 2-Opt   | 154                      | 869      | 869       | 0.00     |
| Or-opt1   | 426                      | 3,030    | 3,030     | 0.00     |
| Exchange  | 688                      | 3,030    | 3,030     | 0.00     |
| Shift(1,0)  | 7,813                    | 738,630  | 8,106     | 98.90    |
| Swap(1,1)   | 20,097                   | 296,258  | 69,307    | 76.60    |

good balance between cost and residual capacity in the candidate routes.

The next three sections describe the computational experiments conducted to measure the efficiency of the algorithm. Section 5.1 evaluates the neighborhood structure efficiency. Section 5.2 measures the time required to reach the solution currently used by the company, based on run time distributions. Finally, Section 5.3 provides detailed results with all costs involved comparing GILS-VND, Rand-MER and the company solutions.

### 5.1. Detailed results on algorithm implementation

The first experiment aims at verifying the quality and efficiency of the neighborhood structures implemented in the GILS-VND algorithm. Tables 2 and 3 exhibit the typical indicators from checkModule's output of the OptFrame framework. The first column in Table 2 indicates the OptFrame component. All five neighborhood structures are implemented in OptFrame core as sequential neighborhoods. The "Optimized" neighborhood structures have an efficient reimplement that discards inter-route moves that violate maximum capacities of a given vehicle (vectors SumDemand[i], MinDemand[i] and MaxDemand[i] helped in this task). The number of tests for each component and the average time spent in each experiment are displayed in the second and third columns of Table 2, respectively.

Tests 1 and 2 display the computational time to build an initial solution and the ADS, respectively. Thirty initial solutions were generated with an average time of 0.041 seconds per solution, and 94,885 different feasible solutions were considered through the neighborhood structures described in Section 4.2. Test 3 indicates the average

time spent to evaluate a solution. Test 4 shows the time required to apply each move generated by the neighborhood structures. Shift(1,0) move is the most costly to apply, requiring 0.0074 milliseconds. This result is consistent since this move changes the size of routes. Test 5 shows the computational time spent to calculate the cost of the move, i.e., the impact on the evaluation function of changing to the selected neighbor. In the "Optimized" version of each neighborhood, the cost calculation benefits from ADSs, not needing to perform the change in the solution and to recalculate the objective function value. This strategy improved the execution time up to 57.6 times for the neighborhood Shift(1,0), reducing the average time from 0.0921 to 0.0016 milliseconds.

On the other hand, Table 3 shows the average number of solutions generated by each neighborhood structure in 30 tests. The second column indicates the average number of moves that lead to other feasible solutions; the third and fourth columns indicate the average number of moves generated by the standard OptFrame implementation and the OptFrame implementation using ADSs, respectively, and the last column indicates the percentage reduction of ineffective moves calculated using Eq. (2).

$$\text{Improvement (\%)} = \frac{\text{Standard} - \text{Optimized}}{\text{Standard}} * 100 \quad (2)$$

Clearly, the use of standard neighborhood structures provided by OptFrame and the implementation of efficient ADSs led to a drastic reduction in the number of moves to find the same number of valid solutions: 98.90% and 76.60% less moves for Shift(1,0) and Swap(1,1), respectively. This improvement is achieved mainly because most of the infeasible solutions can be disregarded directly before proceeding further just by analyzing the values of the ADSs. Many moves from the Swap and Shift neighborhood structures lead to infeasible solutions, i.e., the vehicle's capacity and/or docking constraint are violated because a new customer from another route is included. Although the proposed algorithm may still reach the same final solution (since infeasible solutions can be discarded after evaluation), OptFrame provides two specific mechanisms to avoid such unnecessary moves and calculations. The first mechanism is to use the pre-computed information in the ADS to disallow the selection of moves that could violate constraints (e.g., by testing whether the customer to be inserted in the route has a bigger demand than the vehicle's idle capacity). In that case, the move is never generated by the iterator nor applied to the current solution. The second mechanism is to avoid the generation of moves corresponding to a local optimum of specific neighborhood structures. For example, if the first route is in a local optimum regarding all intra-route neighborhood structures, a future intra-route move will never be tested in this route unless a perturbation or an inter-route move has removed that route from the local optimum. Note that using ADSs in intra-route neighborhoods does not reduce the number of moves, since vehicle loads remain unchanged.

### 5.2. Time-to-target plot results

In the second experiment, time-to-target plots (TTTplots) (Feo & Smith, 1994) were performed to check the efficiency of the GILS-VND algorithm in reaching the solution currently used by the company. TTTplots display the probability (the ordinate) that an algorithm will find a solution at least as good as a given target value within some given running time (the abscissa). TTTplots are also used by Ribeiro and Resende (2012) as a way to characterize the running times of stochastic algorithms for combinatorial optimization.

Aiex, Resende, and Ribeiro (2007) describe a Perl program to create TTTplots for measured times that are assumed to fit a shifted exponential distribution, as presented for a GRASP algorithm (Aiex, Resende, & Ribeiro, 2002). Such plots are very useful to compare different algorithms or strategies for solving a given problem and

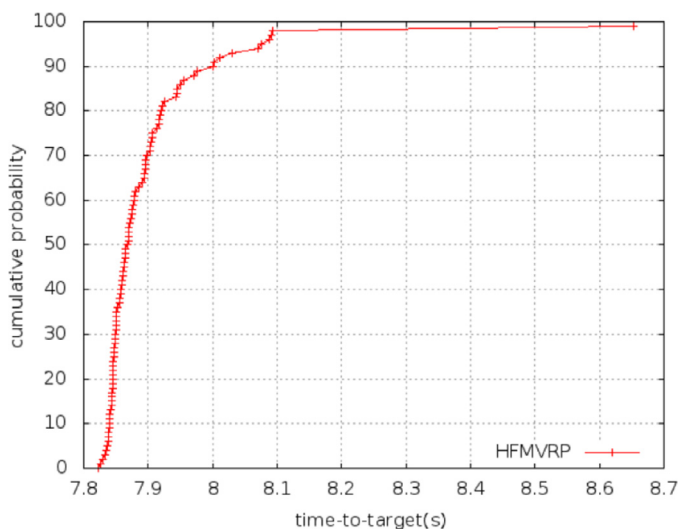


Fig. 5. Time-to-Target Plot - HFMVRP\_1.

Table 4  
Company's fleet composition.

| Veh. type | Cap.      | Avail.    | MT  | Costs (in €) |        |        |
|-----------|-----------|-----------|-----|--------------|--------|--------|
| (t)       | ( $q_t$ ) | ( $m_t$ ) |     | $cf_t$       | $cc_t$ | $cd_t$ |
| a         | 222       | 8         | No  | 88.30        | 8      | 0.2446 |
| b         | 414       | 5         | No  | 115.70       | 8      | 0.3195 |
| c         | 482       | 139       | Yes | 123.29       | 8      | 0.3315 |
| d         | 550       | 3         | No  | 148.87       | 8      | 0.3315 |
| e         | 616       | 6         | No  | 172.23       | 8      | 0.364  |
| f         | 676       | 3         | No  | 178.92       | 8      | 0.364  |
| g         | 752       | 4         | No  | 187.39       | 8      | 0.364  |
| h         | 1,210     | 1         | No  | 238.46       | 8      | 0.364  |

have been widely used as a tool for algorithm design and comparison.

This experiment was run 100 times by the GILS-VND algorithm on a company instance with a total cost of € 32,472.37. This corresponds to instance K (see Tables 6 and 7). The execution ended once the algorithm found the target value (i.e., the same cost). Fig. 5 shows the empirical probability curve. Note that our algorithm was able to find the company solution in all experiments in less than 8.65 seconds. Based on this, we adopted a maximum computational time of 5 minutes in the experiments of Section 5.3.

### 5.3. Benchmark results

To test the algorithm performance, we used 14 real instances that correspond to 14 different working days of the distribution company. We conducted two benchmark experiments: one to compare the performance of our algorithm and the Rand-MER algorithm by Caceres-Cruz et al. (2014), and the other to compare the results obtained by our algorithm with those obtained by the company. For the former, we considered the 10 instances presented in Caceres-Cruz et al. (2014, Table 4) (denoted by A, B, ..., J), and for the latter, we considered five additional instances (denoted by K, L, ..., O). The company distributes its products to 382 customers but it is frequent to have customers with zero demand for a particular day. This may be due to several reasons, for example, a store with low sales in the previous day that can wait another day to replenish, or a store that closes for a local holiday. The company has a fleet of 169 vehicles: Table 4 shows the composition of this fleet by vehicle type, with its capacity (in boxes), the possibility of performing multiple trips, and the corresponding costs (fixed cost per vehicle, variable distance cost, and cost per customer visited, respectively).

Table 5  
Results for real multi-trip instances: GILS-VND vs. Rand-MER (distances in kilometer).

| Instance | #nCustomers | Rand-MER |             | GILS-VND |             | Differences   |               |
|----------|-------------|----------|-------------|----------|-------------|---------------|---------------|
|          |             | Best (1) | Average (2) | Best (3) | Average (4) | Gap (%) (3-1) | Gap (%) (4-2) |
| A        | 372         | 39,534   | 39,841      | 38,995   | 39,572      | -1.36         | -0.68         |
| B        | 366         | 41,072   | 41,399      | 40,670   | 41,243      | -0.98         | -0.38         |
| C        | 371         | 49,669   | 50,082      | 46,116   | 49,639      | -7.15         | -0.89         |
| D        | 364         | 31,378   | 31,543      | 31,300   | 31,600      | -0.25         | 0.18          |
| E        | 372         | 45,485   | 45,836      | 45,300   | 46,206      | -0.41         | 0.80          |
| F        | 373         | 45,275   | 45,681      | 42,100   | 44,942      | -7.01         | -1.64         |
| G        | 372         | 45,165   | 45,493      | 44,983   | 45,883      | -0.40         | 0.85          |
| H        | 374         | 44,386   | 44,909      | 44,230   | 45,114      | -0.35         | 0.46          |
| I        | 370         | 49,053   | 49,354      | 47,345   | 48,292      | -3.48         | -2.20         |
| J        | 372         | 38,973   | 39,252      | 35,366   | 38,074      | -9.25         | -3.10         |

**GILS-VND vs. Rand-MER.** This experiment aims at comparing the performances of the GILS-VND algorithm and the Rand-MER algorithm. As stated in Section 2, the Rand-MER solves a similar problem but does not include docking constraints and it minimizes total distance only. In this first benchmark, the GILS-VND algorithm was adapted to make the comparison consistent, that is, we changed the objective function to consider distances exclusively, and disregarded the docking constraints. The results are displayed in Table 5. The table includes the number of customers with positive demand in that instance, and values are expressed in kilometer. The Rand-MER best and average values are excerpted from Table 4 in Caceres-Cruz et al. (2014). It is worth mentioning that Rand-MER was run for 10 minutes whereas GILS-VND was run for only 5 minutes. The GILS-VND algorithm finds better solutions in all instances, with percentage gaps in the total distance traveled by all vehicles that go from 0.25% (Instance D) to 9.25% (Instance J). The smallest gap happens to be in the instance with the lowest demand, i.e., Instance D with 63,078 boxes delivered to 364 customers. Instance D is also the only instance with no multi-trips. We may infer that the performance of both algorithms is similar when handling single-trip problems. On the contrary, the largest gaps tend to occur in instances with higher demands, i.e., Instances C, F, and I with demand equal to 91,901 boxes, 85,773 boxes and 89,596 boxes, respectively. This may imply a better performance of the GILS-VND algorithm in instances where multi-trips are more decisive. So GILS-VND probably handles demand allocations to different vehicle types more efficiently. Finally, the difference between average values of both algorithms are closer, although GILS-VND beats Rand-MER in 6 out of 10 instances.

**GILS-VND vs. Company.** This second benchmark experiment compares the solutions that the company employed in the corresponding day with those obtained by the GILS-VND algorithm. To do that, five new instances were analyzed: K, L, M, N and O. The first three correspond to days with low demand where no multi-trips were performed whereas the last two had total demand greater than the capacity of all vehicles, so multiple trips were indispensable to serve all customers. Our algorithm was run 30 times per instance with 5 minutes per run. Tables 6 and 7 show the results of the best solution obtained for each instance. Table 6 reports the total costs (TC) of the instances decomposed into the three cost categories defined in Section 3: the fixed vehicle cost,  $C_F$ , the store cost,  $C_C$ , and the distance cost,  $C_D$ . The table also displays the gap between both solutions. The store cost may seem independent of the solution since all customers must be served. In fact, this is not necessarily true in the solution presented by the company in which it was possible that a customer was served using two different trips, incurring a double cost per customer visited. Table 7 presents the routing indicators that the company uses for managerial reasons: the total number of routes employed (#nRoutes), the total distance traveled in kilometers (#nKm), and the total vehicles' idle capacity in boxes (#nE). For the multi-trip



**Table 6**  
Comparison of results I: GILS-VND vs. Company .

| Instance    | GILS-VND<br>C <sub>F</sub> (€) | Company<br>C <sub>F</sub> (€) | GILS-VND<br>C <sub>C</sub> (€) | Company<br>C <sub>C</sub> (€) | GILS-VND<br>C <sub>D</sub> (€) | Company<br>C <sub>D</sub> (€) | GILS-VND<br>TC(€) | Company<br>TC(€) | TC<br>Gap (%) |
|-------------|--------------------------------|-------------------------------|--------------------------------|-------------------------------|--------------------------------|-------------------------------|-------------------|------------------|---------------|
| Single trip |                                |                               |                                |                               |                                |                               |                   |                  |               |
| K           | <b>16,498</b>                  | 17,223                        | <b>2944</b>                    | <b>2944</b>                   | <b>11,065</b>                  | 12,305                        | <b>30,507</b>     | 32,472           | <b>-6.05</b>  |
| L           | <b>8064</b>                    | 9117                          | <b>2528</b>                    | <b>2528</b>                   | <b>5735</b>                    | 7353                          | <b>16,327</b>     | 18,997           | <b>-14.05</b> |
| M           | <b>9174</b>                    | 10,066                        | <b>2536</b>                    | <b>2536</b>                   | <b>6308</b>                    | 7665                          | <b>18,018</b>     | 20,266           | <b>-11.09</b> |
| Multi-trip  |                                |                               |                                |                               |                                |                               |                   |                  |               |
| N           | <b>25,815</b>                  | 29,543                        | <b>2992</b>                    | 3032                          | <b>16,716</b>                  | 19,075                        | <b>45,523</b>     | 51,650           | <b>-11.79</b> |
| O           | <b>22,943</b>                  | 25,931                        | <b>3008</b>                    | 3072                          | <b>14,895</b>                  | 16,761                        | <b>40,846</b>     | 45,764           | <b>-10.62</b> |

**Table 7**  
Comparison of results II: GILS-VND vs. Company .

| Instance    | GILS-VND<br>#nRoutes | Company<br>#nRoutes | GILS-VND<br>#nKm | Company<br>#nKm | GILS-VND<br>#nE | Company<br>#nE |
|-------------|----------------------|---------------------|------------------|-----------------|-----------------|----------------|
| SINGLE TRIP |                      |                     |                  |                 |                 |                |
| K           | <b>134</b>           | 134                 | <b>33,403</b>    | 36,733          | <b>1871</b>     | 4533           |
| L           | <b>66</b>            | 71                  | <b>17,601</b>    | 22,185          | <b>894</b>      | 4325           |
| M           | <b>75</b>            | 80                  | <b>19,334</b>    | 23,351          | <b>1003</b>     | 3658           |
| MULTI-TRIP  |                      |                     |                  |                 |                 |                |
| N           | <b>203/34</b>        | 234/86              | <b>50,244</b>    | 57,388          | <b>5420</b>     | 19,164         |
| O           | <b>181/12</b>        | 205/56              | <b>44,639</b>    | 50,423          | <b>3350</b>     | 14,667         |

**Table 8**  
Statistical results for the new set of instances: GILS-VND vs. Company .

| Instance | Company | GILS-VND |         |           |         |
|----------|---------|----------|---------|-----------|---------|
|          |         | Best     | Average | Std. dev. | Gap (%) |
| K        | 32,472  | 30,507   | 30,692  | 70        | -5.48   |
| L        | 18,997  | 16,327   | 16,427  | 42        | -13.53  |
| M        | 20,266  | 18,017   | 18,146  | 43        | -10.46  |
| N        | 51,609  | 45,523   | 45,813  | 100       | -11.23  |
| O        | 45,764  | 40,846   | 40,995  | 66        | -10.42  |

instances, the second number in the “#nRoutes” column represents the number of vehicles that performed two trips.

The GILS-VND algorithm was able to obtain cheaper solutions in all instances. The numbers reported belong to the best solution obtained by our algorithm in all the runs. These solutions represent savings on the operational costs of up to € 6,127 (e.g., Instance N), reducing fixed vehicle costs by € 3,728 (i.e., using less vehicles) and traveling shorter distances (i.e., 7144 less kilometers in one day). Considering that each instance corresponds to the distribution of a single day, the potential annual savings are considerable. In addition, our solutions are better with respect to all other routing indicators: less trips needed, less distance traveled, and vehicles with less residual capacity.

Next, we provide some statistical results on the total cost obtained by the algorithm for all 30 runs performed for the new set of five instances. Table 8 shows these figures. The last column displays the percentage gap between the average algorithm solution and the company solution, calculated as:

$$\text{gap}_i = \frac{\overline{TC}_i^{\text{GILS-VND}} - TC_i^{\text{COMPANY}}}{TC_i^{\text{COMPANY}}} \quad (3)$$

where  $\overline{TC}_i^{\text{GILS-VND}}$  is the average algorithm solution and  $TC_i^{\text{COMPANY}}$  is the company solution for instance  $i$ . In the worst case, the average cost is almost 6% better than the company solution.

## 6. Conclusions and extensions

Real vehicle routing problems present a variety of constraints that are sometimes disregarded in model formulations. These realistic constraints may have a significant impact on the solution

implementation. This study analyzed a Heterogeneous Fleet Multi-trip VRP (HFMVRP) faced by a distribution company in Europe that serves around 400 customers. This real VRP variant considers a fleet of heterogeneous vehicles (i.e., vehicles with different capacities and costs) with the possibility of performing multiple trips or being unable to serve particular customers (for maneuverability reasons, for example). The objective function included the company's set of costs: a fixed cost per vehicle used, a variable vehicle cost per distance traveled and a fixed cost per customer visited. Due to the difficulty of the problem, we proposed a heuristic algorithm, the GILS-VND, that combines an ILS, a GRASP, and a VND. The algorithm uses the power of the GRASP to build a feasible initial solution, and then within the ILS structure, it uses the VND as local search combined with the Refine method based on several levels of perturbation. With the use of smart mechanisms that discard solutions based on previous values stored in ADSs, it was possible to enhance the solution reevaluation time in up to 98.90% for one of the real instances studied in this paper.

To test the performance of our algorithm, we experimented with a set of real instances provided by the company. These instances correspond to 15 business days with all customers' demands. First, our algorithm was tested against the Rand-MER algorithm by [Caceres-Cruz et al. \(2014\)](#) in order to compare the performance of our approach with one already validated in the literature. The computational results revealed that the GILS-VND algorithm was able to obtain more economical solutions. Furthermore, the comparison with the company solutions was also favorable and the GILS-VND algorithm led to significant cost savings (estimate yearly savings are over € 70,000) and better routing indicators: the total number of routes employed, the total distance traveled and the vehicles' idle capacity. Besides minimizing costs, the company is also concerned about using the least number of routes with full truckload vehicles. Another benefit of the algorithm is its speed and reliability. It was able to find good solutions with low variability in reduced time. This is particularly interesting since routing decisions must be made daily after receiving all customer demands in less than 30 minutes. In addition, the algorithm calibration is relatively simple and requires no complex fine-tuning processes. Overall, the method proposed is a powerful tool that can support distribution planners in their decision making.

As future extensions for this work, we could consider including **time windows** in the deliveries. Due to traffic constraints, for instance, it is possible that certain customers cannot be served during

some business hours. Algorithmically, new neighborhood structures related to the consecutive customers' relocation can be also incorporated. Finally, we also propose the implementation of a parallel version of the GILS-VND algorithm to benefit from the multi-core technology present in current machines.

## Acknowledgment

The authors are indebted to the two anonymous reviewers for their constructive suggestions that have helped us improve the original manuscript. This work has been partially supported by CNPq (grants 552289/2011-6 and 306458/2010-1), FAPEMIG (grants PPM CEX 497-13, APQ-04611-10), CAPES and Science Without Borders (grant 202380/2012-2 and 202381/2012-9), the Spanish Ministry of Economy and Competitiveness (TRA2013-48180-C3-P) and the Ibero-American Programme for Science, Technology and Development (CYTED2010-511RT0419)

## References

- Aiex, R. M., Resende, M. G. C., & Ribeiro, C. C. (2002). Probability distribution of solution time in GRASP: An experimental investigation. *Journal of Heuristics*, 8(3), 343–373.
- Aiex, R. M., Resende, M. G. C., & Ribeiro, C. C. (2007). TTTplots: A perl program to create time-to-target plots. *Optimization Letters*, 1(4), 355–366.
- Amorim, P., Parragh, S., Sperandio, F., & Almada-Lobo, B. (2014). A rich vehicle routing problem dealing with perishable food: A case study. *TOP*, 22(2).
- de Armas, J., & Melián-Batista, B. (2015). Variable neighborhood search for a dynamic rich vehicle routing problem with time windows. *Computers & Industrial Engineering*, 85, 120–131.
- Baldacci, R., Battarra, M., & Vigo, D. (2008). Routing a heterogeneous fleet of vehicles. In B. Golden, S. Raghavan, & E. Wasil (Eds.), *The vehicle routing problem: Latest advances and new challenges*: 43 (pp. 3–27). Springer.
- Baldacci, R., & Mingozzi, A. (2009). A unified exact method for solving different classes of vehicle routing problems. *Mathematical Programming*, 120(2), 347–380.
- Belfiore, P., Yoshizaki, Y., & Tsugunobu, H. (2009). Scatter search for a real-life heterogeneous fleet vehicle routing problem with time windows and split deliveries in Brazil. *European Journal of Operational Research*, 199(3), 750–758.
- Brandão, J. (2009). A deterministic tabu search algorithm for the fleet size and mix vehicle routing problem. *European Journal of Operational Research*, 195(3), 716–728.
- Caceres-Cruz, J., Grasas, A., Ramalhinho, H., & Juan, A. A. (2014). A savings-based randomized heuristic for the heterogeneous fixed fleet vehicle routing problem with multi-trips. *Journal of Applied Operational Research*, 6(2), 69–81.
- Choi, E., & Tcha, D.-W. (2007). A column generation approach to the heterogeneous fleet vehicle routing problem. *Computers & Operations Research*, 34(7), 2080–2095.
- Clarke, G., & Wright, J. W. (1964). Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, 12(4), 568–581.
- Coelho, I. M., Munhoz, P. L. A., Haddad, M. N., Coelho, V. N., Silva, M. M., Souza, M. J. F., & Ochi, L. S. (2011). A computational framework for combinatorial optimization problems. In *VII ALIO/EURO workshop on applied combinatorial optimization*: 3 (pp. 51–54). Porto.
- Coelho, V. N., Guimaraes, F. G., Reis, A. J. R., Coelho, B. N., Coelho, I. M., & Souza, M. J. F. (2014). A general variable neighborhood search heuristic for short term load forecasting in smart grids environment. In *Proceedings of the Power systems conference (PSC)*, 2014 Clemson University (pp. 1–8).
- Coelho, V. N., Souza, M. J. F., Coelho, I. M., Guimaraes, F. G., Lust, T., & Cruz, R. C. (2012). Multi-objective approaches for the open-pit mining operational planning problem. *Electronic Notes in Discrete Mathematics*, 39, 233–240.
- Dantzig, G. B., & Ramser, J. H. (1959). The truck dispatching problem. *Management Science*, 6(1), 80–91.
- Dayarian, I., Crainic, T. G., Gendreau, M., & Rei, W. (2015). A column generation approach for a multi-attribute vehicle routing problem. *European Journal of Operational Research*, 241(3), 888–906.
- Feo, T. A., & Resende, M. G. C. (1995). Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 6(6), 109–133.
- Feo, T. A., & Smith, M. G. C. R. S. H. (1994). A greedy randomized adaptive search procedure for maximum independent set. *Operations Research*, 42(5), 860–878.
- Gendreau, M., Laporte, G., Musaraganyi, C., & Taillard, E. D. (1999). A tabu search heuristic for the heterogeneous fleet vehicle routing problem. *Computers & Operations Research*, 26(12), 1153–1173.
- Golden, B., Assad, A., Levy, L., & Gheysens, F. (1984). The fleet size and mix vehicle routing problem. *Computers & Operations Research*, 11(1), 49–66.
- Imran, A., Salhi, S., & Wassen, N. A. (2009). A variable neighborhood-based heuristic for the heterogeneous fleet vehicle routing problem. *European Journal of Operational Research*, 197(2), 509–518.
- Jiang, J., Ng, K. M., Poh, K. L., & Teo, K. M. (2014). Vehicle routing problem with a heterogeneous fleet and time windows. *Expert Systems with Applications*, 41(8), 3748–3760.
- Kritikos, M. N., & Ioannou, G. (2013). The heterogeneous fleet vehicle routing problem with overloads and time windows. *International Journal of Production Economics*, 144(1), 68–75.
- Laporte, G. (2009). Fifty Years of Vehicle Routing. *Transportation Science*, 43(4), 408–416.
- Leung, S. C., Zhang, Z., Zhang, D., Hua, X., & Lim, M. K. (2013). A meta-heuristic algorithm for heterogeneous fleet vehicle routing problems with two-dimensional loading constraints. *European Journal of Operational Research*, 225(2), 199–210.
- Li, F., Golden, B., & Wasil, E. (2007). A record-to-record travel algorithm for solving the heterogeneous fleet vehicle routing problem. *Computers & Operations Research*, 34(9), 2734–2742.
- Liu, S., Huang, W., & Ma, H. (2009). An effective genetic algorithm for the fleet size and mix vehicle routing problems. *Transportation Research Part E: Logistics and Transportation Review*, 45(3), 434–445.
- Lourenço, H. R., Martin, O. C., & Stützle, T. (2003). Iterated local search. In F. Glover, & G. Kochenberger (Eds.), *Handbook of metaheuristics* (pp. 321–353). Boston: Kluwer Academic Publishers.
- Mladenović, N., & Hansen, P. (1997). Variable neighborhood search. *Computers & Operations Research*, 24(11), 1097–1100.
- Ochi, L. S., Vianna, D. S., Drummond, L. M. A., & Victor, A. O. (1998). An evolutionary hybrid metaheuristic for solving the vehicle routing problem with heterogeneous fleet. In *Proceedings of the first European workshop, EuroGP'98*: 1391 (pp. 187–195). Paris, France.
- Or, I. (1976). *Traveling salesman-type combinatorial problems and their relation to the logistics of blood banking*. USA: Northwestern University Ph.D. thesis.
- Penna, P. H. V., Subramanian, A., & Ochi, L. S. (2013). An iterated local search heuristic for the heterogeneous fleet vehicle routing problem. *Journal of Heuristics*, 19(2), 201–232.
- Prins, C. (2002). Efficient heuristics for the heterogeneous fleet multitrail VRP with application to a large-scale real case. *Journal of Mathematical Modelling and Algorithms*, 1(2), 135–150.
- Prins, C. (2009). Two memetic algorithms for heterogeneous fleet vehicle routing problems. *Engineering Applications of Artificial Intelligence*, 22(6), 916–928.
- Resende, M. G. C., & Ribeiro, C. C. (2010). Greedy randomized adaptive search procedures: Advances, hybridizations, and applications. In M. Gendreau, & J. Potvin (Eds.), *Handbook of metaheuristics* (pp. 283–319). New York: Springer.
- Ribeiro, C. C., & Resende, M. G. C. (2012). Path-relinking intensification methods for stochastic local search algorithms. *Journal of Heuristics*, 18(2), 193–214.
- Ribeiro, G., Desaulniers, G., Desrosiers, J., Vidal, T., & Vieira, B. (2014). Efficient heuristics for the workover rig routing problem with a heterogeneous fleet and a finite horizon. *Journal of Heuristics*, 20(6), 677–708.
- Souza, M., Coelho, I., Ribas, S., Santos, H., & Merschmann, L. (2010). A hybrid heuristic algorithm for the open-pit-mining operational planning problem. *European Journal of Operational Research*, 207(2), 1041–1051.
- Stützle, T. (2006). Iterated local search for the quadratic assignment problem. *European Journal of Operational Research*, 174(3), 1519–1539.
- Subramanian, A., Uchoa, E., & Ochi, L. S. (2013). A hybrid algorithm for a class of vehicle routing problems. *Computers & Operations Research*, 40(10), 2519–2531.
- Tütüncü, G. Y. (2010). An interactive gramps algorithm for the heterogeneous fixed fleet vehicle routing problem with and without backhauls. *European Journal of Operational Research*, 201(2), 593–600.
- Vidal, T., Crainic, T. G., Gendreau, M., & Prins, C. (2013). Heuristics for multi-attribute vehicle routing problems: A survey and synthesis. *European Journal of Operational Research*, 231(1), 1–21.