An ILS-based Algorithm to Solve a Large-scale Real Heterogeneous Fleet VRP with Multi-trips and Docking Constraints

V.N. Coelho[a,**], Alex Grasas[b,*], Helena Ramalhinho[b,*], I.M. Coelho[c,*], M.J.F. Souza[d,*]

[a] *Department of Control and Automation Engineering, Federal University of Ouro Preto, Ouro Preto, MG, 35400-000, Brazil*
[b] *Department of Economics and Business, Universitat Pompeu Fabra, Barcelona, 08005, Spain*
[c] *Institute of Computing, Fluminense Federal University, Niterói, 24210-240, Brazil*
[d] *Department of Computer Science, Federal University of Ouro Preto, Ouro Preto, MG, 35400-000, Brazil*

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 goods. 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. Given the combinatorial nature and the size of the problem, which discard the use of efficient exact methods for its resolution, a novel heuristic algorithm is proposed. The proposed algorithm, called `GILS-VND`, combines Iterated Local Search (`ILS`), Greedy Randomized Adaptive Search Procedure (`GRASP`) and Variable Neighborhood Descent (`VND`) procedures. Our method obtains better solutions than other approaches found in the related literature, and improves the solutions used by the company leading to

*Corresponding author
**Principal corresponding author
Email addresses: `vncoelho@ufmg.br` (V.N. Coelho), `alex.grasas@upf.edu` (Alex Grasas), `helena.ramalhinho@upf.edu` (Helena Ramalhinho), `imcoelho@ic.uff.br` (I.M. Coelho), `marcone@iceb.ufop.br` (M.J.F. Souza)

significant savings in transportation costs.

Keywords: vehicle routing, heterogeneous fleet, multiple trips, docking constraints, iterated local search, OptFrame framework.

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 necessity of finding cost-effective routes that keep high customer satisfaction. The classical routing problem, first proposed by Dantzig and Ramser [1] and known as Capacitated VRP, has the objective of minimizing the total distance traveled by a homogeneous fleet of vehicles to serve all customers. Although this problem has been studied for more than five decades [2], real applications remain a huge challenge. They feature a variety of operational restrictions and rules that must be considered in any solution implementation. These additional considerations can affect, for example, customers, depots, and vehicles, complicate the problem and have a significant impact on the solution.

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. Vehicles' fixed and variable costs.

This problem is a variant of the Heterogeneous Fleet Multitrip Vehicle Routing Problem (HFMVRP) ([3]). In this variant, docking constraints have been added and the goal is to minimize: 1) a fixed cost of using a vehicle, 2) a fixed cost per customer served, and 3) 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, the total number of routes employed, the total distance traveled and the vehicles' idle capacity. Their purpose, apart from saving costs, is to have the least number of routes with full truckload vehicles.

The HFMVRP belongs to the class $\mathcal{NP}$-Hard, 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) an Iterated Local Search (`ILS`) [4, 5, 6], 2) a Greedy Randomized Adaptive Search Procedure (`GRASP`) [7, 8], and 3) a Variable Neighborhood Descent (`VND`) [9]. We test our algorithm using real instances provided by the company. The algorithm proved to be fast and reliable, and, in all cases, the solutions obtained were better in all dimensions than those obtained and implemented by the company.

The remainder of this paper is organized as follows. Section 2 reviews the literature on heterogeneous VRP. Section 3 defines the HFMVRP. Section 4 describes the algorithm used to solve this problem. Section 5 presents the computational experiments, and finally Section 6 draws the final considerations.

2. Heterogeneous VRPs

The VRP with heterogeneous fleet (HVRP) is gaining attention from researchers in the past years due to its applicability in real cases. The HVRPs can be divided according to vehicle availability (limited or unlimited) and vehicle costs (fixed or variable) [10]. 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 et al. [11]. The authors propose 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 [12]. The latter outperforms the former. They also develop a mathematical formulation for the variant with dependent costs, and obtain the first lower bounds for the VRP with unlimited fixed fleet.

Gendreau et al. [13] suggest an algorithm based on Tabu Search (TS) with a tour construction phase and an improvement phase that considers variable costs. They, however, assume Euclidean problems only, where nodes are located in the same plane. The number of vehicles is limited by an approximation of the fleet size made when the algorithm starts its execution. The VRP with Mix Fleet discussed by the authors mingle investment costs in the medium term with short-term operating costs that fluctuate according to the specific customers attended per day.

Choi and Tcha [14] obtain lower bounds for all variants of the unlimited fleet problem using a column generation approach based on the set covering problem. Baldacci and Mingozzi [15] propose a variant based on a set partitioning problem that uses 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 [16] follows the basic ideas of Gendreau et al. [13], using a deterministic TS algorithm for the Fleet Size and Mix VRP, that consists in determining the number of vehicles of each type, and the routes for each vehicle.

Among the heuristic approaches presented in the literature, noteworthy are those based on Evolutionary Algorithms. Ochi et al. [17] develop an algorithm that combines a Genetic Algorithm (GA) with Scatter Search [18] to solve the variant of the problem with limited fleet and fixed costs. Liu et al. [19] propose a hybrid GA with a hybrid local search procedure for the Fleet Size and Mix VRP. Moscato [20] tackles the variant with fixed and vehicle-dependent costs with a Memetic Algorithm (MA). Prins [21] also presents two MA. The first approach uses a chromosome encoded as a giant tour and a split procedure that performs the optimal distribution of vehicles and routes. The second algorithm uses distance calculation strategies in order to diversify the search in the solution space. More recently, Baldaccci et al.[22], Penna et al. [10], Martinez and Amaya [23] and Pillac et al. [24] compile different studies on the HVRP variants mentioned above. Vidal et al. [25] introduce a new local search operator based on the combination of standard VRP moves and swaps between trips, applied to Multi-Attribute VRP, and find many best known solutions. Wang et al. [26] solve a VRP variant with multiple trips and time windows, in which vehicles are allowed to perform multiple trips during a scheduling period and each customer must be served

within a given time interval.

Prins [3] also studies a HFMVRP applied to a large-scale real case (a French furniture manufacturer with 775 stores), that bears some similarities to the variant studied in this work. His objective function, however, is based on total distance traveled while our objective includes fixed and variable costs. He develops an algorithm based on TS with a simple bi-objective approach: minimizing the total duration of all trips and the number of vehicles. His results outperform the solutions used by the company.

Finally, [27] present a randomized heuristic based on the well-known CWS heuristic for a simpler version of the problem studied in this paper. As in Prins [3], the authors aim only at minimizing distances without considering docking constraints. They test their algorithm with the same data instances we use in this paper, provided by a European distribution company.

3. Problem Definition

The HFMVRP described in this paper can be defined over an undirected graph $G = (V, E)$, where $V = \{0, 1, ..., n\}$ 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 its nonnegative demand d_i. Each edge $(i, j) \in E$ has an associated nonnegative cost or distance c_{ij}. The fleet T is composed of m different types of vehicles, i.e., $T = \{1, ..., m\}$. For each $t \in T$ there are m_t available vehicles with capacity q_t (in boxes), 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 $S = \{(r, t) \mid t \in T,\ r = (v_0, v_1, ..., v_{n(r)+1})\}$ be a set of valid routes, where $n(r)$ denotes the number of customers visited in route r and t is the vehicle type associated to 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's total demand is: $Q_r^t = \sum_{i=1}^{n(r)} d_{v_i}$;
- The route's total cost is: $C_r^t = cf_t + n(r) \times cc_t + (\sum_{i=0}^{n(r)} c_{v_i,v_{i+1}}) \times cd_t$;
- The route's empty space is: $E_r^t = q_t - Q_r^t$.

This set of elements characterizes the HFMVRP. The problem seeks to build a set S^* that minimizes the total cost function given by:

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

Note that the cost of a route, C_r^t, is composed of three cost terms. For convenience (see Section 5.3), we group the cost 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.
- $C_D = \sum_{(r,t) \in S} (\sum_{i=0}^{n(r)} c_{v_i, v_{i+1}}) \times cd_t$ 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. All vehicles can perform one or two routes (single- and multiple-trip vehicles, respectively).

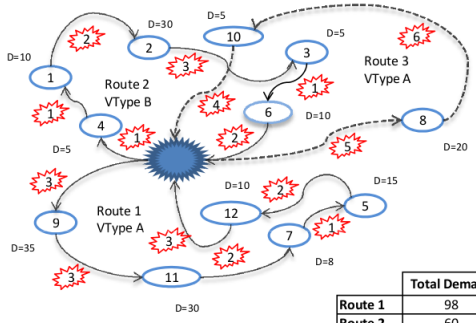
Figure 1 displays a solution example for a HFMVRP with 12 customers and 2 types of vehicles with a single vehicle per type. The "Parameters" table shows the values for all relevant parameters. The "Compatibility" table indicates the values of $comp(t, i)$, for $t \in \{\text{A}, \text{B}\}$ and $i \in V \setminus \{0\}$. The "Solution" table presents the resulting routes, with the total demand served, distance traveled, total cost and empty space. Note that the vehicle type A performs two routes, 1 and 3. For this example, $C_F = 50$, $C_C = 17$, $C_D = 68$, and the total cost is $TC = 135$.

Parameters

t	q_t	m_t	*Multi-Trip*	cf_t	cc_t	cd_t
A	100	1	YES	€ 15	€ 1	€ 2
B	80	1	YES	€ 20	€ 2	€ 1

Compatibility

	Vehicle Type t	
Costumer	A	B
1	1	1
2	1	1
3	1	1
4	1	1
5	1	1
6	1	1
7	1	1
8	1	0
9	1	1
10	1	0
11	1	1
12	1	1



Solution

	Total Demand	Vehicle Type	Total Distance	C_r^t	E_r^t
Route 1	98	A	14 KM	€ 48	2
Route 2	60	B	10 KM	€ 40	20
Route 3	25	A	15 KM	€ 47	75

Figure 1: HFMVRP example

4. Methodology

4.1. The GILS-VND Algorithm

The algorithm proposed in this paper, dubbed `GILS-VND`, combines an `ILS`, a `GRASP`, and a `VND`. The pseudocode is outlined in Algorithm 1. In this algorithm, the inputs *GRASPmax* and *IterMax* represent the number of iterations in which the GRASP construction phase is applied and the maximum number of iterations performed at a given level of perturbation, respectively (γ is explained below). The function $f(\cdot)$ is the objective function (Equation 1) defined in Section 3.

In Line 1, the initial solution is generated by applying the `GRASP` construction phase. This procedure is called BuildInitialSavingsSolution and uses the Clarke and Wright Savings algorithm adapted to the HFMVRP for each type of vehicle $t \in T$. At each iteration of the `GRASP` procedure, the order of vehicle types is chosen randomly. A restricted candidate list is created by ordering the routes regarding to its empty space, that is, routes with total demand closer to the capacity of the vehicle type t 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 γ sets the size of this list, following the reasoning of the `GRASP` procedure. The m_t best routes are added to the current solution. If customers remain still unassigned, a new Savings procedure is performed using the vehicles allowed to do multiple trips.

The local search (Line 2 of Algorithm 1) is carried out by the `VND` procedure (presented in Algorithm 2), using the neighborhood structures described

Algorithm 1: `GILS-VND`

Input: *GRASPmax*, *IterMax*, γ, Function $f(\cdot)$
Output: Solution s

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

in Section 4.2. Line 3 initializes p, a variable that regulates the "power" of diversification, and Lines 4–18 perform the main ILS loop as long as the stopping condition is not satisfied. Line 19 returns the final solution.

Algorithm 2 has the role of performing the local search: Lines 16 and 22 trigger the setLocalOptimum procedure, which sets neighborhoods as "local-optimum"; this mark is used by the Auxiliary Data Structures, described in Section 4.3.

Within the ILS in Algorithm 1, Line 7 calls the Refine procedure presented in Algorithm 3. In Line 2 of this algorithm, the SelectNeighborhood procedure selects randomly one of the neighborhood structures r^{pert}, described in Section 4.2. Then, Line 3 performs a shake in the current solution according to this selected neighborhood structure. In this proposed strategy, the Refine procedure has several levels of perturbation. 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.

Algorithm 2: `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 ← s_0
2  k^Inter ← 1
3  while k^Inter ≤ |r^Inter| do
4      Find the best neighbor s' ∈ N^(k^Inter)(s)
5      if f(s') < f(s) then
6          s ← s'
7          k^Inter ← 1
8          k^Intra ← 1
9          while k^Intra ≤ |r^Intra| do
10             Find the best neighbor s' ∈ N^(k^Intra)(s)
11             if f(s') < f(s) then
12                 s ← s'
13                 k^Intra ← 1
14             end
15             else
16                 setLocalOptimum(s, k)
17                 k^Intra ← k^Intra + 1
18             end
19         end
20     end
21     else
22         setLocalOptimum(s, k)
23         k^Inter ← k^Inter + 1;
24     end
25 end
26 return s
```

Algorithm 3: *Refine*

Input: r^{pert} perturbation neighborhoods in random order
Input: Initial solution s, Level p and Function $f(\cdot)$
Output: Solution s

```
for i ← 1 To p + 2 do
    k ← SelectNeighborhood(r^pert)
    s' ← Shake(s, k)
end
s ← VND(s', f)
return s
```

4.2. Neighborhood structures

Five different neighborhood structures were 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 neighborhood structure that consists in removing two non-adjacent arcs and inserting two new arcs, so that a new route is formed. Figure 2 exemplifies the movement: edges $(4, 6)$ and $(5, 8)$ 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. For a symmetric problem, the total distance among these customers remains unaffected.

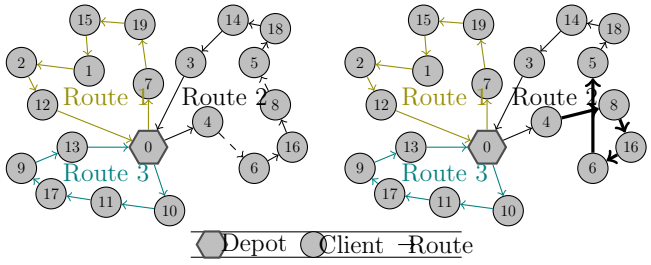


Figure 2: Example of *2-opt* Move

***Or-optk* move.** An *Or-optk* move is an intra-route neighborhood structure 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 [28], in which the removal involves up to three consecutive customers only. Figure 3 illustrates the movement with $k = 1$, where customer 5 is 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 [29].

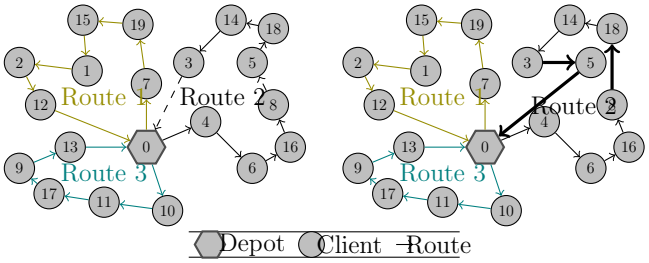


Figure 3: Example of *Or-optk* Move

***Exchange* move.** An *Exchange* move is an intra-route neighborhood structure that consists in exchanging two customers in the same route. Figure 4 shows an *Exchange* of costumers 5 and 18 in Route 2.

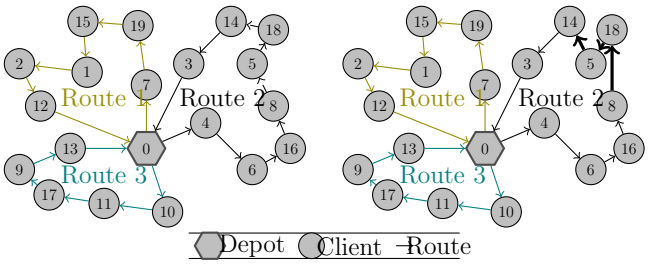


Figure 4: Example of *Exchange* Move

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

***Swap*(1,1) move.** A *Swap*(1,1) move is an inter-route neighborhood structure that exchanges two costumers from different routes.

All neighborhood structures above are used as a perturbation strategy. The application of these moves occurs randomly with no improvement verification in the objective function. This mechanism is key to diversify and

explore the solution space (exploration-exploitation). After applying a given move, the NeighborhoodStatus[j][i] vector status (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) have been used. A brief description is given below. $\#nRoutes$ indicates the total number of routes and $\#nNeighborhoods$ the number of neighborhood structures (see Section 4.2). For $i \in \{1, \ldots, \#nRoutes\}$ and $j \in \{1, \ldots, \#nNeighborhoods\}$ the following data structures are used:

- SumDemand[i]: it stores the sum of all customer demands assigned to route i, e.g., SumDemand[12] = 230 implies that all customers in route 12 have a total demand of 230.
- MinDemand[i]: it stores the minimum demand among all customers in route i, e.g., MinDemand[10] = 30 implies that the lowest demand among all customers in route 10 is 30.
- 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. 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 3), some "local-optimum" markers are removed from the routes that have been changed in that disruption.

Table 1 displays the representation and the ADSs of the HFMVRP solution provided in Figure 1. Note that route 2 is out of local-optimum, so moves related to this route should be verified.

5. Computational experiments

The `GILS-VND` algorithm was implemented in C++ with assistance from framework OptFrame 1.5 (Available at http://sourceforge.net/projects/optframe/) [30, 31]. This framework has been successfully applied in the literature (see

Table 1: HFMVRP Solution

Solution Representation	
Route (r,t)	**Costumers 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)		
		$i = (1, 2, 3)$
SumDemand[i]		(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)

Coelho et al. [32]). The tests were carried out on a Pentium Core 2 Quad (Q6600), 2.4 GHZ with 8GB 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.

To test the algorithm performance, 14 real instances from the distribution company were used, which correspond to 14 different days. Ten of them were already presented in the literature by Caceres et al. [27]. The other four instances are introduced in this paper. Since Caceres et al. [27] compared his results to other literature benchmark multi-trip instances, we avoid to present similiar results in this paper. So, the main focuses here is on solving the real c These instances have up to 382 customers and are divided in single- and multi-trip instances (HFMVRP and HFMVRP_MT, respectively). In the second group, the total demand is greater than the capacity of all vehicles, so multiple trips are indispensable to serve all customers. The customer demands are different in each case, and it is possible to have a customer with 0 demand in a particular instance. Table 2 shows the composition of the fleet by vehicle type, with its capacity, the possibility of performing multiple trips, and the corresponding costs (in €).

Table 2: Company's fleet composition

Veh. type (t)	Cap. (q_t)	Avail. (m_t)	MT	Costs (in €) 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

After some preliminary experiments with different parameter values, the input parameters adopted were: $\gamma = 0.4$, $GRASPmax = 100$ and $IterMax = 1000$. 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 computes the results with all costs involved.

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 3 and 4 exhibit a typical indicators output from `checkModule` of the OptFrame framework. The first column in Table 3 indicates the OptFrame component. All five neighborhood structures are implemented in OptFrame core as sequential neighborhoods. In this application, only a few basic method implementations were needed. The "Optimized" neighborhood structures have an efficient reimplementation 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 3, respectively.

Tests 1 and 2 display the computational time to build an initial solution and the ADS, respectively. 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 0.0074ms. 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 of changing to the selected neighbor on the evaluation function. 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 0.0016ms.

On the other hand, Table 4 shows the average number of solutions generated by each neighborhood structure in 30 tests. The "Valid Neighborhood moves" column indicates the average number of moves that lead to other feasible solutions; columns "Standard" and "Optimized" indicate the average number of moves generated by the standard OptFrame implementation and the implementation using the ADS, respectively. Note the percentage reduction of ineffective moves in the "Improvement" column calculated using Equation 2. The results indicate that, from the use of some standard neighborhood structures provided by OptFrame and through the implementation of an efficient ADS, it was possible to find the same number of valid solutions with a reduced number of moves: 98.90% and 76.60% less moves for the neighborhood structures Shift(1,0) and Swap(1,1), respectively. Using ADS

Table 3: `CheckModule` Output - Computational times

Component	#Tests	average(ms)
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	1,144	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)	6,493	0.0921
Shift(1,0)-Optimized	6,493	0.0016
Swap(1,1)	16,701	0.0917
Swap(1,1)-Optimized	16,701	0.0016

Table 4: `CheckModule` Output - Efficiency of the Neighborhood Structures

		Average number of moves from neighborhood in 30 tests		
Neighborhood	Valid neigh. 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

in the intra-route neighborhoods does not reduce the number of moves, since the vehicle loads do not change with these moves.

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

5.2. Time-to-Target plot results

In the second experiment, time-to-target plots (TTTplots) [33] 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 have been also used in Ribeiro and Resende [34] as a way to characterize the running times of stochastic algorithms for combinatorial optimization.

Aiex et al. [35] describe a Perl program to create TTTplots for measured times that are assumed to fit a shifted exponential distribution, closely following Aiex et al. [36]. Such plots are very useful to compare different algorithms or strategies for solving a given problem and have been widely used as a tool for algorithm design and comparison.

A set of 100 executions applying the `GILS-VND` algorithm were made to solve HFMVRP_1 instance, with a target equal to €32,472.37 (the company cost for this instance). The performance ended once the algorithm found the target value. Figure 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

First, the `GILS-VND` algorithm was run 10 times with different seeds for the set of instances introduced by Caceres et al. [27]. It should be noticed that the maximum computational time here was set to 5 minutes instead of 10 minutes as in Table 5 our results are compared to the ones obtained with the Rand-MER algorithm by Caceres et al. [27].

The table shows the best and the average total distance-based cost (in km) for each algorithm. Our GILS-VND algorithm finds better solutions in all instances reducing up to 9% the toal distance traveled by all vehicles. The differences between both algorithms in the average results are smaller, although GILS-VND beats Rand-MER in 6 out of 10 instances.

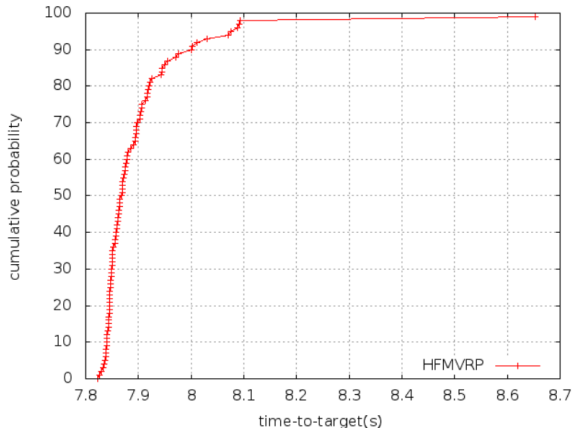


Figure 5: Time-to-Target Plot - HFMVRP_1

Table 5: Results for real multi-trip instances: `GILS-VND` vs. `Rand-MER`

		Rand-MER		`GILS-VND`			
Instance	#nStores	best (1)	average (2)	best (3)	average (4)	gap (%) (3-1)	gap (%) (4-2)
A	372	39534	39841	38995	39572	-1.36	-0.68
B	366	41072	41399	40670	41243	-0.98	-0.38
C	371	49669	50082	46116	49639	-7.15	-0.89
D	364	31378	31543	31300	31600	-0.25	0.18
E	372	45485	45836	45300	46206	-0.41	0.80
F	373	45275	45681	42100	44942	-7.01	-1.64
G	372	45165	45493	44983	45883	-0.40	0.85
H	374	44386	44909	44230	45114	-0.35	0.46
I	370	49053	49354	47345	48292	-3.48	-2.20
J	372	38973	39252	35366	38074	-9.25	-3.10

The last experiment compares the solutions obtained by the GILS-VND algorithm and those used by the company, in four new real instances (K, L, M, N) plus the instance H. The GILS-VND 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) for five instances decomposed into three cost categories: 1) the fixed vehicle cost, C_F, 2) the store cost, C_C, and 3) the distance cost, C_D. The table also displays the gap between both solutions. The store cost may seem independent of the solution employed 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 instances, the second number in the "$\#nRoutes$" column represents the number of vehicles that performed two trips.

Our GILS-VND algorithm was able to obtain better solutions in all instances. These solutions represent savings on the operational costs of up to €6,127 (e.g., in HFMVRP_MT1). Considering that each instance corresponds to the distribution planning of a single day, the potential annual savings are considerable. Our solutions also improve all other routing indicators, that is, less trips were needed, less distance was traveled, and vehicles traveled with less empty space. These results consider only the best solution obtained by our algorithm in all the runs. Next, we provide some statistical results on the total cost obtained by the algorithm for all 30 runs performed for the new set of four instances. Table 8 shows these figures. The last column displays the percentage gap between the average algorithm solution and the company solution, calculated as:

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

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

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

Instance	`GILS-VND` C_F(€)	Company C_F(€)	`GILS-VND` C_C(€)	Company C_C(€)	`GILS-VND` C_D(€)	Company C_D(€)	`GILS-VND` TC(€)	Company TC(€)	TC Gap (%)
SINGLE TRIP									
K	**16,498**	17,223	**2,944**	**2,944**	**11,065**	12,305	**30,507**	32,472	**-6.05**
L	**8,064**	9,117	**2,528**	**2,528**	**5,735**	7,353	**16,327**	18,997	**-14.05**
M	**9,174**	10,066	**2,536**	**2,536**	**6,308**	7,665	**18,018**	20,266	**-11.09**
MULTITRIP									
N	**25,815**	29,543	**2,992**	3,032	**16,716**	19,075	**45,523**	51,650	**-11.79**
H	**22,943**	25,931	**3,008**	3,072	**14,895**	16,761	**40,846**	45,764	**-10.62**

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

Instance	`GILS-VND` $\#nRoutes$	Company $\#nRoutes$	`GILS-VND` $\#nKm$	Company $\#nKm$	`GILS-VND` $\#nE$	Company $\#nE$
SINGLE TRIP						
K	**134**	134	**33,403**	36,733	**1,871**	4,533
L	**66**	71	**17,601**	22,185	**894**	4,325
M	**75**	80	**19,334**	23,351	**1,003**	3,658
MULTITRIP						
N	**203/34**	234/86	**50,244**	57,388	**5,420**	19,164
H	**181/12**	205/56	**44,639**	50,423	**3,350**	14,667

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

		`GILS-VND`			
Instance	company	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

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 Multitrip 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 served. 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, then within the `ILS` structure, it uses the `VND` as local search combined with the Refine method based on several levels of perturbation. The algorithm was implemented adapting the OptFrame components (which are publicly available). The quality and efficiency verification of the neighborhood structures showed that it was possible to enhance the efficiency up to 98.91%.

To test the performance of our algorithm, we experimented with a set of real instances provided by the company. These instances correspond to 14 business days with all customers demands. First, our algorithm was compared to the Rand-MER algorithm ([27]) to see that it could obtain better solutions in all instances reported. Computational results pointed that `GILS-VND` algorithm could obtain better quality solution in all instances presented by the literature. Furthermore, the computational results obtained by the `GILS-VND` algorithm represent significant cost savings (estimate yearly savings are over €$70,000$) but also outperform the other routing indicators used by the company: 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 is able to find good quality 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 V. N. Coelho, M. J. F. Souza, I. M. Coelho and R. C. Cruz thank 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), for supporting the development of this work.

References

[1] G. B. Dantzig, J. H. Ramser, The truck dispatching problem, Management Science 6 (1) (1959) 80–91.

[2] G. Laporte, Fifty Years of Vehicle Routing, Transportation Science 43 (4) (2009) 408–416.

[3] C. Prins, Efficient heuristics for the heterogeneous fleet multitrip VRP with application to a large-scale real case, Journal of Mathematical Modelling and Algorithms 1 (2) (2002) 135–150.

[4] H. R. Lourenço, O. C. Martin, T. Stützle, Iterated local search, in: F. Glover, G. Kochenberger (Eds.), Handbook of Metaheuristics, Kluwer Academic Publishers, Boston, 2003, pp. 321 – 353.

[5] T. Stützle, Iterated local search for the quadratic assignment problem, European Journal of Operational Research 174 (3) (2006) 1519–1539.

[6] M. J. F. Souza, M. T. Mine, M. de Souza Alves Silva, L. S. Ochi, A. Subramanian, A hybrid heuristic, based on Iterated Local Search and GENIUS, for the Vehicle Routing Problem with Simultaneous Pickup and Delivery, International Journal of Logistics Systems and Management 10 (2) (2011) 142–157.

[7] T. A. Feo, M. G. C. Resende, Greedy randomized adaptive search procedures, Journal of Global Optimization 6 (6) (1995) 109–133.

[8] M. G. C. Resende, C. C. Ribeiro, Greedy randomized adaptive search procedures: Advances, hybridizations, and applications, in: M. Gendreau, J. Potvin (Eds.), Handbook of Metaheuristics, 2nd Edition, Springer, New York, 2010, pp. 283–319.

[9] N. Mladenović, P. Hansen, Variable neighborhood search, Computers & Operations Research 24 (11) (1997) 1097 – 1100.

[10] P. H. V. Penna, A. Subramanian, L. S. Ochi, An iterated local search heuristic for the heterogeneous fleet vehicle routing problem, Journal of Heuristics 19 (2) (2013) 201–232.

[11] B. Golden, A. Assad, L. Levy, F. Gheysens, The fleet size and mix vehicle routing problem, Computers & Operations Research 11 (1) (1984) 49 – 66. doi:http://dx.doi.org/10.1016/0305-0548(84)90007-8.

[12] G. Clarke, J. W. Wright, Scheduling of Vehicles from a Central Depot to a Number of Delivery Points, Operations Research 12 (4) (1964) 568–581.

[13] M. Gendreau, G. Laporte, C. Musaraganyi, E. D. Taillard, A tabu search heuristic for the heterogeneous fleet vehicle routing problem, Computers & Operations Research 26 (12) (1999) 1153 – 1173.

[14] E. Choi, D.-W. Tcha, A column generation approach to the heterogeneous fleet vehicle routing problem, Computers & Operations Research 34 (7) (2007) 2080–2095.

[15] R. Baldacci, A. Mingozzi, A unified exact method for solving different classes of vehicle routing problems, Mathematical Programming 120 (2) (2009) 347–380.

[16] J. Brandão, A deterministic tabu search algorithm for the fleet size and mix vehicle routing problem, European Journal of Operational Research 195 (3) (2009) 716–728.

[17] L. S. Ochi, D. S. Vianna, L. M. de A. Drummond, A. O. Victor, An evolutionary hybrid metaheuristic for solving the vehicle routing problem with heterogeneous fleet, in: Proceedings of the First European Workshop, EuroGP'98, no. 1391, Paris, France, 1998, pp. 187–195.

[18] F. Glover, M. Laguna, R. Martí, Scatter search, in: A. Ghosh, S. Tsutsui (Eds.), Advances in Evolutionary Computing: Theory and Applications, Springer-Verlag, 2003, pp. 519–537.

[19] S. Liu, W. Huang, H. Ma, An effective genetic algorithm for the fleet size and mix vehicle routing problems, Transportation Research Part E: Logistics and Transportation Review 45 (3) (2009) 434 – 445.

[20] P. Moscato, A gentle introduction to memetic algorithms, in: F. Glover, G. Kochenberger (Eds.), Handbook of Metaheuristics, Kluwer Academic Publishers, Boston, 2003, pp. 105–144.

[21] C. Prins, Two memetic algorithms for heterogeneous fleet vehicle routing problems, Engineering Applications of Artificial Intelligence 22 (6) (2009) 916–928.

[22] R. Baldaccci, M. Battarra, D. Vigo, Routing a heterogeneous fleet of vehicles, in: B. Golden, S. Raghavan, E. Wasil (Eds.), The Vehicle Routing Problem: Latest Advances and New Challenges, Vol. 43, Springer, 2008, pp. 3–27.

[23] L. Martinez, C. Amaya, A vehicle routing problem with multi-trips and time windows, Journal of the Operational Research Society 64 (11) (2013) 1630–1643.

[24] V. Pillac, M. Gendreau, C. Guéret, A. L. Medaglia, A review of dynamic vehicle routing problems, European Journal of Operational Research 225 (1) (2013) 1 – 11.

[25] T. Vidal, T. G. Crainic, M. Gendreau, C. Prins, Heuristics for multi-attribute vehicle routing problems: A survey and synthesis, European Journal of Operational Research 231 (1) (2013) 1 – 21. doi:http://dx.doi.org/10.1016/j.ejor.2013.02.053.

[26] Z. Wang, W. Liang, X. Hu, Metaheuristic based on a pool of routes for the VRPMTW, Journal of the Operational Research Society 65 (1) (2014) 37–48. URL http://dx.doi.org/10.1057/jors.2013.4

[27] J. Caceres-Cruz, A. Grasas, H. Ramalhinho, A. A. Juan, A savings-based randomized heuristic for the heterogeneous fixed fleet vehicle routing problem with multi-trips, Journal of Applied Operational Research 6 (2) (2014) 69 – 81.

[28] I. Or, Traveling salesman-type combinational problems and their relation to the logistics of blood banking, Ph.D. thesis, Northwestern University, USA (1976).

[29] A. Subramanian, E. Uchoa, L. S. Ochi, A hybrid algorithm for a class of vehicle routing problems, Computers & Operations Research 40 (10) (2013) 2519 – 2531. doi:http://dx.doi.org/10.1016/j.cor.2013.01.013.

[30] I. M. Coelho, P. L. A. Munhoz, M. N. Haddad, V. N. Coelho, M. M. Silva, M. J. F. Souza, L. S. Ochi, A computational framework for combinatorial optimization problems, in: VII ALIO/EURO Workshop on Applied Combinatorial Optimization, no. 3, Porto, 2011, pp. 51–54.

[31] I. M. Coelho, S. Ribas, M. H. P. Perche, P. L. A. Munhoz, M. F. Souza, L. S. Ochi, Optframe: a computational framework for combinatorial optimization problems, in: XLII Brazilian Symposium of Operational Research, no. 2, Bento Gonçalves/RS, Brazil, 2010, pp. 1887 – 1898.

[32] V. N. Coelho, M. J. F. Souza, I. M. Coelho, F. G. Guimaraes, T. Lust, R. C. Cruz, Multi-objective approaches for the open-pit mining operational planning problem, Electronic Notes in Discrete Mathematics 39 (2012) 233 – 240.

[33] T. A. Feo, M. G. C. Resende, S. H. Smith, A greedy randomized adaptive search procedure for maximum independent set, Operations Research 42 (5) (1994) 860–878.

[34] C. C. Ribeiro, M. G. C. Resende, Path-relinking intensification methods for stochastic local search algorithms, Journal of Heuristics 18 (2) (2012) 193–214.

[35] R. M. Aiex, M. G. C. Resende, C. C. Ribeiro, TTTplots: a perl program to create time-to-target plots, Optimization Letters 1 (4) (2007) 355–366.

[36] R. M. Aiex, M. G. C. Resende, C. C. Ribeiro, Probability distribution of solution time in GRASP: An experimental investigation, Journal of Heuristics 8 (3) (2002) 343–373.