A Hybrid Algorithm for the Heterogenous Fleet Vehicle Routing Problem

Anand Subramanian[a,b], Puca Huachi Vaz Penna[b], Eduardo Uchoa[c], Luiz Satoru Ochi[b]

[a] *Universidade Federal da Paraíba, Departamento de Engenharia de Produção, Centro de Tecnologia, Bloco G, Cidade Universitária, João Pessoa-PB, 58051-970, Brazil*
[b] *Universidade Federal Fluminense - Instituto de Computação, Rua Passo da Pátria 156, Bloco E - 3º andar, São Domingos, Niterói-RJ, 24210-240, Brazil*
[c] *Universidade Federal Fluminense - Departamento de Engenharia de Produção, Rua Passo da Pátria 156, Bloco E - 4º andar, São Domingos, Niterói-RJ, 24210-240, Brazil*

Abstract

This paper deals with the Heterogeneous Fleet Vehicle Routing Problem (HFVRP). The HFVRP generalizes the classical Capacitated Vehicle Routing Problem by considering the existence of different vehicle types, with distinct capacities and costs. The objective is to determine the best fleet composition as well as the set of routes that minimize the total costs. The proposed hybrid algorithm is composed by an Iterated Local Search (ILS) based heuristic and a Set Partitioning (SP) formulation. The SP model is solved by means of a Mixed Integer Programming solver that interactively calls the ILS heuristic during its execution. The developed algorithm was tested in benchmark instances with up to 360 customers. The results obtained are quite competitive with those found in the literature and new improved solutions are reported.

Key words: Routing, Heterogeneous Fleet, Matheuristics, Iterated Local Search, Set Partitioning

*Corresponding author: Tel. +55 21 2629-5665; Fax +55 21 2629-5666.
Email addresses: anand@ct.ufpb.br (Anand Subramanian), ppenna@ic.uff.br (Puca Huachi Vaz Penna), uchoa@producao.uff.br (Eduardo Uchoa), satoru@ic.uff.br (Luiz Satoru Ochi)

1. Introduction

This paper deals with the Heterogeneous Fleet Vehicle Routing Problem (HFVRP), which can be defined as follows. Let $G = (V, A)$ be a directed graph where $V = \{0, 1, \ldots, n\}$ is a set composed by $n + 1$ vertices and $A = \{(i, j) : i, j \in V, i \neq j\}$ is the set of arcs. The vertex 0 denotes the depot, where the vehicle fleet is located, while the set $V' = V \setminus \{0\}$ is composed by the remaining vertices that represents the n customers. Each customer $i \in V'$ has a non-negative demand q_i. The fleet is composed by m different types of vehicles, with $M = \{1, \ldots, m\}$. For each $u \in M$, there are m_u available vehicles, each with a capacity Q_u. Every vehicle type is also associated with a fixed cost denoted by f_u. Finally, for each arc $(i, j) \in A$ there are associated costs $c_{ij}^u = d_{ij} r_u$, where d_{ij} is the distance between the vertices (i, j) and r_u is a type-variable travel cost per distance unit, of a vehicle of type u. The objective is to determine the best fleet composition as well as the set of routes that minimize the sum of fixed and travel costs in such a way that: (i) every route starts and ends at the depot and is associated to a vehicle type; (ii) each customer belongs to exactly one route; (iii) the vehicle's capacity is not exceeded. The HFVRP is $\mathcal{NP}$-hard since it includes the classical VRP as a special case when all vehicles are identical.

The HFVRP is a very important problem, since fleets are likely to be heterogeneous in most practical situations. According to Hoff et al. (2010), even when the acquired fleet is homogeneous, it can become heterogeneous over the time when vehicles with different characteristics are incorporated. Moreover, insurance, maintenance and operating costs may have different values based to the level of depreciation or usage time of the fleet.

We consider the cases where the fleet is limited (Heterogeneous Vehicle Routing Problem – HVRP) as well as the cases where the fleet is unlimited (Fleet Size and Mix – FSM). More specifically, we tackle the following variants:

- HVRPFV, limited fleet, with fixed and variable costs;

- HVRPV, limited fleet, with variable costs but without fixed costs, i.e., $f_u = 0, \forall u \in M$;
- FMSFV, unlimited fleet, i.e.,$m_u = +\infty, \forall u \in M$, with fixed and variable costs;
- FSMF, unlimited fleet, with fixed costs but without variable costs, i.e., $r_u = 1, \forall u \in M$;
- FMSV, unlimited fleet, with variable costs but without fixed costs.

In this work, we propose a hybrid algorithm, that is composed by an Iterated Local Search (ILS) based heuristic and a Set Partitioning (SP) formulation. The SP model is built using routes generated by ILS and it is solved by means of a Mixed Integer Programming (MIP) solver that interactively calls the ILS heuristic during its execution. This strategy differs from other approaches that also create solutions out of routes such as those of Rochat & Taillard (1995) and Tarantilis & Kiranoudis (2002).

The remainder of this paper is organized as follows. Section 2 reviews some works related to the HFVRP. Section 3 explains the proposed hybrid algorithm. Section 4 contains the results obtained and a comparison with those reported in the literature. Section 5 presents the concluding remarks of this work.

2. Related Works

Since its introduction by Golden et al. (1984), few authors have proposed exact methods for FSM variants. Yaman (2006) suggested valid inequalities and presented lower bounds for the FSMF. Choi & Tcha (2007) obtained lower bounds for all FSM variants by means of a column generation algorithm based on a set covering formulation. Baldacci et al. (2008) proposed some valid inequalities as well as a two-commodity MIP formulation for the

same variant. The HFVRP is considered to be much harder than corresponding problems with a homogeneous fleet. At that point, the instances proposed by Golden et al. (1984) with only 20 customers were not solved to optimality. Pessoa et al. (2009) (see also Pessoa et al., 2008) proposed a Branch-Cut-and-Price (BCP) algorithm over an extended formulation capable of solving instances with up to 75 customers. More recently, Baldacci & Mingozzi (2009) put forward a SP based algorithm that uses bounding procedures based on linear relaxation and lagrangian relaxation. That algorithm obtained even better results and could solve a few instances with 100 customers. Nevertheless, such exact algorithms can be very time-consuming and are not suitable for larger instances. On the other hand, there is a rich literature on heuristic methods for the HFVRP.

Many metaheuristic based approaches were proposed for the FSM over the years. Ochi et al. (1998a) proposed a hybrid evolutionary procedure that combines Scatter Search with Genetic Algorithm (GA) to solve the FSMF. A parallel implementation of the same algorithm was presented by Ochi et al. (1998b). Gendreau et al. (1999) developed a heuristic algorithm that combines Tabu Search (TS), adaptive memory and a GENIUS approach.

Renaud & Boctor (2002) proposed a sweep-based heuristic for the FSMF that employs traditional construction and improvement VRP procedures. Lee et al. (2008) proposed a hybrid algorithm that combines TS and SP. Brandão (2009) put forward a deterministic TS with different procedures for generating initial solutions. A hybrid GA that employs local search as a mutation approach was developed by Liu et al. (2009) to solve the FSMF and the FMSV. Two Memetic Algorithms were developed by Prins (2009) to solve all FSM variants and the HVRPV. Imran et al. (2009) developed a Variable Neighborhood Search (VNS) algorithm that makes use of classical algorithms for generating initial solutions. All FSM variants were considered by the authors. Finally, Penna et al. (2011) proposed an ILS based heuristic for solving the same FSM and HVRP variants considered in the present work.

The HVRP was proposed by Taillard (1999). The author developed an algorithm based on TS, adaptive memory and column generation which was also applied to solve the FSM. Prins (2002) dealt with the HVRP by developing an algorithm that extends a number of VRP classical heuristics followed by a local search procedure based on the Steepest Descent Local Search and TS. Tarantilis et al. (2003) solved the HVRPV by implementing a threshold accepting procedure where a worse solution is only accepted if it is within a given threshold. The same authors (Tarantilis et al., 2004) later presented another threshold accepting procedure to solve the same problem. Li et al. (2007) put forward a record-to-record travel algorithm for the HVRPV. Li et al. (2010) proposed a multi-start adaptive memory procedure combined with Path Relinking and a modified TS to solve the HVRPFV. More recently, Brandão (2011) proposed a TS algorithm for the HVRP which includes additional features such as strategic oscillation, shaking and frequency-based memory.

3. The ILS-RVND-SP Algorithm

The proposed hybrid algorithm, called ILS-RVND-SP, is composed by an ILS (Lourenço et al., 2003) heuristic, that uses a procedure based on the Variable Neighborhood Descent (Mladenovic & Hansen, 1997) with Random neighborhood ordering (RVND) in the local search phase, and a SP formulation.

Let $\mathcal{R}$ be the set of all possible routes of all vehicle types, $\mathcal{R}_i \subseteq \mathcal{R}$ be the subset of routes that contain customer $i \in V'$, and $\mathcal{R}_u \subseteq \mathcal{R}$ be the set of routes associated with vehicle type $u \in M$. Define y_j as the binary variable associated to a route $j \in \mathcal{R}$, and c_j as its cost. The SP formulation for the HVRP can be expressed as follows.

$$\text{Min} \sum_{j \in \mathcal{R}} c_j y_j \tag{1}$$

subject to

$$\sum_{j \in \mathcal{R}_i} y_j = 1 \qquad \forall i \in V' \tag{2}$$

$$\sum_{j \in \mathcal{R}_u} y_j \leq m_u \qquad \forall u \in M \tag{3}$$

$$y_j \in \{0, 1\}. \tag{4}$$

The objective function (1) minimizes the sum of the costs by choosing the best combination of routes. Constraints (2) state that a single route from the subset $\mathcal{R}_i$ visits costumer $i \in V'$. Constraints (3) are limits on the fleet composition. Constraints (4) define the domain of the decision variables. Since this complete formulation has an exponential number of variables, it can not be directly solved. Solving it by branch-and-price or related methods, as done in some proposed exact algorithms, is time-consuming and only practical up to a certain instance size. The ILS-RVND-SP algorithm actually solves a SP problem similar to (1–4), where $\mathcal{R}$ is restricted to a few thousands routes generated by the ILS-RVND heuristic.

In the case of FSM, we drop constraints (3) because there is no upper bound on the number of vehicles of each type. In addition, when the resolution of the restricted SP by a MIP solver exceeds the time limit imposed or the gap between the linear relaxation of the root node and the incumbent solution s^* is larger than a given limit (this usually happens when fixed costs are considered), the algorithm enforces the fleet composition to be equal to the one used by s^* . Let m_u^* be the number of vehicles of type u used in s^*. The vehicle fleet can be fixed by adding the following constraints:

$$\sum_{j \in \mathcal{R}_u} y_j = m_u^* \qquad \forall u \in M. \tag{5}$$

Of course, this limits the improvements that can be made by solving the SP problem but it makes the problem much more computationally tractable in

an acceptable time.

Alg. 1 describes the higher-level ILS-RVND-SP algorithm. At first, an empty pool of routes is initialized (line 2). Next, a solution s^* is generated using the ILS-RVND heuristic (see Subsection 3.1), which also fills the pool with the routes in every local optimal solution visited (line 3). The variable $Cutoff$ is initialized with the Upper Bound (UB) value associated to s^* (line 4). The SP model, given by expressions (1)-(4), is build using the pool of routes (line 5). The SP problem and s^* are given to a MIP solver (line 6) which calls the ILS-RVND heuristic whenever an incumbent solution is found (Procedure IncumbentCallback, lines 14-21). If the solution s^* is improved in the IncumbentCallback, the $Cutoff$ value is updated (line 19), but s^* is not given back to the solver since it may contain a route that does not belong to the set of routes $\mathcal{R}$ of the SP model. We assume that the MIP solver uses a Branch-and-bound or a Branch-and-cut solution procedure. The MIP solver stopping criteria are: (i) optimal solution found; (ii) LB $> Cutoff$; (iii) $RootGap > MaxRootGap$, where $RootGap$ is the gap between the LB and the UB after solving the root node and $MaxRootGap$ is the maximum $RootGap$ allowed; (iv) $Time > TimeMax$, where $Time$ is the execution time of the solver and $TimeMax$ is a time limit imposed for the solver. If the solver has been interrupt due to (iii) or (iv) and the fleet is unlimited, then the SP model is updated by adding constraints (5), $MaxRootGap$ is set to infinity and the solver is called again with the same stopping criteria.

3.1. The ILS-RVND heuristic

The ILS-RVND heuristic is based on the one developed by Penna et al. (2011) for the HFVRP and its steps are summarized in the Alg. 2. The heuristic executes $MaxIter$ iterations and it returns the best solution s^* among all iterations. (lines 2-26). The parameter $MaxIterILS$ represents the maximum number of consecutive perturbations allowed without improvements. If an starting solution s_0 is not provided, a constructive procedure is applied for generating an initial solution (line 4) and the value of

Algorithm 1 ILS-RVND-SP

1: **Procedure** ILS-RVND-SP($MaxIter$, $MaxTime$, $MaxRootGap$)
2: $RoutePool \leftarrow$ `NULL`
3: $s^* \leftarrow$ ILS-RVND($MaxIter$, `NULL`, $RoutePool$)
4: $Cutoff \leftarrow f(s^*)$
5: $SP\_model \leftarrow$ CreateSetPartitioningModel($RoutePool$)
6: $MIPSolver(SP\_Model, s^*, Cutoff, MaxRootGap, MaxTime, IncumbentCallback(s^*))$
7: **if** (($Time > MaxTime$ **or** $RootGap > MaxRootGap$) **and** (unlimited fleet)) **then**
8: Update SP_model {Fixing the fleet}
9: $MaxRootGap \leftarrow \infty$
10: MIPSolver($SP\_Model, s^*, Cutoff, MaxRootGap, MaxTime, IncumbentCallback(s^*)$)
11: **end if**
12: return s^*
13: **end** ILS-RVND-SP
14: **Procedure** IncumbentCallback(s^*)
15: $s \leftarrow$ Incumbent Solution
16: $s \leftarrow$ ILS-RVND(1, s, `NULL`)
17: **if** $f(s) < f(s^*)$ **then**
18: $s^* \leftarrow s$
19: $Cutoff \leftarrow f(s)$
20: **end if**
21: **end** IncumbentCallback

$MaxIterILS$ is set to $n + v$, where v is the number of vehicles of the generated solution (lines 3-5). This expression was empirically formulated according to preliminary experiments when it was observed that the appropriate number of perturbations was directly proportional to n and v. In contrast, if a solution s_0 is provided, then $MaxIterILS$ is set to 1000 (lines 6-9). We assume that s_0 is a relatively good solution and, in view of this, much more trials has to be given for the algorithm to possibly improve it. It is important to mention that we have dealt with instances with up to 360 customers and hence $n+v < 1000$. The main ILS loop (lines 11-20) aims to improve the generated initial solution using a RVND procedure (line 12) in the local search phase combined with a set of perturbation mechanisms (line 18). Notice that the perturbation is always performed on the best current solution (s') of a given iteration (acceptance criterion). The ILS-RVND original structure was slightly modified in order to store routes during its execution. Every time

a local search is performed, the pool of routes is updated by only adding routes that still have not been included in the pool (lines 13). This updating is ignored when ILS-RVND is called during the IncumbentCallback.

Algorithm 2 ILS-RVND

```
1: Procedure ILS-RVND(MaxIter, s_0, RoutePool)
2: for i ← 1, ..., MaxIter do
3:    if s_0 = NULL then
4:       s ← GenerateInitialSolution(v, seed)
5:       MaxIterILS ← n + v
6:    else
7:       s ← s_0
8:       MaxIterILS ← 1000
9:    end if
10:   iterILS ← 0
11:   while iterILS ≤ MaxIterILS do
12:      s' ← RVND(s)
13:      UpdateRoutePool(RoutePool, s')
14:      if f(s) < f(s') then
15:         s' ← s
16:         iterILS ← 0
17:      end if
18:      s ← Perturb(s', seed)
19:      iterILS ← iterILS + 1
20:   end while
21:   if f(s') < f* then
22:      s* ← s'
23:      f* ← f(s')
24:   end if
25: end for
26: return s*
27: end ILS-RVND
```

3.1.1. Constructive Procedure

The constructive procedure works as follows. For the HVRP, we first initialize empty routes associated to each available vehicle. For the FSM, we first initialize one empty route per vehicle type and whenever it is necessary (i.e., when it is no longer possible to add unrouted customers to the current partial solution), we add an empty route associated to a random vehicle type.

Let the Candidate List (CL) be initially composed by all customers. Each route is initially filled with a seed customer k, randomly selected from the CL. An insertion criterion and an insertion strategy is chosen at random. An initial solution is generated using the selected combination of criterion and strategy. If the fleet is unlimited (FSM), an empty route associated to each type of vehicle is added to the constructed solution s. These empty routes are necessary to allow a possible fleet resizing during the local search phase.

Two insertion criteria were adopted: the Modified Cheapest Feasible Insertion Criterion (MCFIC) and the Nearest Feasible Insertion Criterion. The first consists of a modification of the well-known Cheapest Insertion Criterion by allowing only feasible insertions and also by including an insertion incentive for those customers located far from the depot. The second consists of of an adaptation of the classical Nearest Insertion Criterion by only allowing feasible insertions.

Two insertion strategies were employed, specifically the Sequential Insertion Strategy (SIS) and the Parallel Insertion Strategy (PIS). In SIS, while there is at least one unrouted customer that can be added to the current partial solution, each route is filled with a customer selected using the correspondent insertion criterion, whereas in PIS all routes are considered while evaluating the least-cost insertion. We refer to Penna et al. (2011) for a more detailed description of the constructive procedure.

3.1.2. Local Search

The local search is performed by a VND (Mladenovic & Hansen, 1997) procedure which utilizes a random neighborhood ordering (RVND). Firstly, a Neighborhood List (NL) containing a predefined number of inter-route moves is initialized. While NL is not empty, a neighborhood $N^{(\eta)} \in$ NL is chosen at random and then the best admissible move is determined. In case of improvement, an intra-route local search is performed on the modified routes. For the FSM, the fleet is updated and the NL is populated with all the neighborhoods. Otherwise, $N^{(\eta)}$ is removed from the NL. The fleet

updating assures that there is exactly one empty vehicle of each type.

Let N' be a set composed by r' intra-route neighborhood structures. The intra-route local search is as follows. At first, a neighborhood list NL$'$ is initialized with all the intra-route neighborhood structures. Next, while NL$'$ is not empty a neighborhood $N'^{(\eta)} \in$ NL$'$ is randomly selected and a local search is exhaustively performed until no more improvements are found.

3.1.3. Inter-Route Neighborhood structures

Seven VRP neighborhood structures involving inter-route moves were employed and they are described next. The inter-route neighborhood structures are described next. **Shift(1,0)**, a customer k is transferred from a route r_1 to a route r_2. **Swap(1,1)**, permutation between a customer k from a route r_1 and a customer l, from a route r_2. **Shift(2,0)**, two adjacent customers, k and l, are transferred from a route r_1 to a route r_2. This move can also be seen as an arc transferring. In this case, the move examines the transferring of both arcs (k, l) and (l, k). **Swap(2,1)**, permutation of two adjacent customers, k and l, from a route r_1 by a customer k' from a route r_2. As in Shift(2,1), both arcs (k, l) and (l, k) are considered. **Swap(2,2)**, permutation between two adjacent customers, k and l, from a route r_1 by another two adjacent customers k' and l', belonging to a route r_2. All the four possible combinations of exchanging arcs (k, l) and (k', l') are considered. **Cross**, the arc between adjacent clients k and l, belonging to a route r_1, and the one between k' and l', from a route r_2, are both removed. Next, an arc is inserted connecting k and l' and another is inserted connecting k' and l. K**-Shift**, a subset of consecutive customers K is transferred from a route r_1 to the end of a route r_2. In this case, it is assumed that the variable and fixed costs of r_2 is smaller than those of r_1. It should be pointed out that the move is also taken into account when r_2 is an empty route.

The solution spaces of the seven neighborhoods are explored exhaustively, that is, all possible combinations are examined, and the best improvement strategy is considered. The computational complexity of each one of these

moves is $\mathcal{O}(n^2)$. Only feasible moves are admitted, i.e., those that do not violate the maximum load constraints. Therefore, every time an improvement occurs, the algorithm checks whether this new solution is feasible or not. This checking is trivial and it can be performed in a constant time by just verifying if the sum of the customers demands of a given route does not exceed the vehicle's capacity at the depot.

3.1.4. Intra-Route Neighborhood structures

Five well-known intra-route neighborhood structures were adopted. The set N' is composed by Or-opt, 2-opt and exchange moves. The computational complexity of these neighborhoods is $\mathcal{O}(\bar{n}^2)$, where $\bar{n}$ is the number of customers of the modified routes. Their description is as follows. **Reinsertion**, one customer is removed and inserted in another position of the route. **Or-opt2**, two adjacent customers are removed and inserted in another position of the route. **Or-opt3**, three adjacent customers are removed and inserted in another position of the route. **2-opt**, two nonadjacent arcs are deleted and another two are added in such a way that a new route is generated. **Exchange**, permutation between two customers.

3.2. Perturbation Mechanisms

A set P of three perturbation mechanisms were adopted. Whenever the `Perturb()` function is called, one of the moves described below is randomly selected. **Multiple-Swap(1,1)**, $P^{(1)}$, multiple Swap(1,1) moves are performed randomly. After some preliminary experiments, the number of successive moves was empirically set to $0.5v$. **Multiple-Shift(1,1)**, $P^{(2)}$, multiple Shift(1,1) moves are performed randomly. The Shift(1,1) consists in transferring a customer k from a route r_1 to a route r_2, whereas a customer l from r_2 is transferred to r_1. In this case, the number of moves is randomly selected from the interval $\{0.5v, 0.6v, \ldots, 1.4v, 1.5v\}$. **Split**, $P^{(3)}$, a route r is divided into smaller routes. Let $M' = \{2, \ldots, m\}$ be a subset of M composed by all vehicle types, except the one with the smallest capacity. Firstly, a route

$r \in s$ (let $s = s'$) associated with a vehicle $u \in M'$ is selected at random. Next, while r is not empty, the remaining customers of r are sequentially transferred to a new randomly selected route $r' \notin s$ associated with a vehicle $u' \in \{1, \ldots, u-1\}$ in such a way that the capacity of u' is not violated. The new generated routes are added to the solution s while the route r is removed from s. The procedure described is repeated multiple times where the number of repetitions is chosen at random from the interval $\{1, 2, ..., v\}$. This perturbation was applied only for the FSM, since it does not make sense for the HVRP. Only feasible perturbations moves are accepted.

4. Computational Results

The algorithm ILS-RVND-SP was coded in C++ (g++ 4.4.3) and executed in an Intel Core i7 Processor 2.93 GHz with 8 GB of RAM running Ubuntu Linux 10.04 (kernel version 2.6.32). The SP formulation was implemented using the solver CPLEX 12.2. The developed approach was tested in well-known instances, containing up to 100 customers, namely those proposed by Golden et al. (1984) and adapted by Taillard (1999) and Choi & Tcha (2007). Table 1 describes the characteristics of these instances. We also tested ILS-RVND-SP in the instances of Brandão (2011), containing up to 199 customers, and Li et al. (2007), containing up to 360 customers. Their description can be found in Tables 2 and 3, respectively.

The following parameters values were selected after some preliminary experiments: $MaxIter = 30$, $MaxTime = 30$ seconds, $MaxRootGap = 0.02$. For all five HFVRP variants, each instance was executed 10 times and the results are presented in Subsections 4.2-4.6. A comparison is performed with the best known algorithms reported in the literature.

In the tables presented hereafter, **Inst.** denotes the number of the test-problem, $\boldsymbol{n}$ is the number of customers, **BKS** represents the best known solution reported in the literature, **Best Sol.** and **Time** indicate, respectively, the best solution and the average computational time associated to

the correspondent work, **Avg. Sol.** represents the average solution of the 10 runs, **Gap** denotes the gap between the best solution found by ILS-RVND-SP and the best known solution, **Avg. Gap** corresponds to the gap between the average solution found by ILS-RVND-SP and the best known solution. **Scaled time** indicates the scaled time in seconds of each computer using the performances, in Mflop/s, of computers listed in Dongarra (2010) for our 2.93 GHz. The best solutions are highlighted in boldface and the solutions improved by the ILS-RVND-SP algorithm are underlined.

4.1. Evaluating the performance of each phase of ILS-RVND-SP

In this subsection we are interested in evaluating the performance of each phase of ILS-RVND-SP, i.e., ILS-RVND and SP. Table 4 illustrates the influence, in terms of computing time and solution cost, of both phases in the final solution on each set of instances. It can be observed that phase 2 is always capable of substantially improving the solutions found in the first phase. It is noteworthy to mention that the number of perturbations without improvements of phase 1 is considerably smaller from those adopted in Penna et al. (2011), leading to a faster procedure but less effective in terms of solution quality. Nevertheless, when including phase 2, ILS-RVND-SP not only finds better average solutions but still outperforms the ILS-RVND presented in Penna et al. (2011) in terms of computational time, as it will be shown in the following subsections.

4.2. HVRPFV

Baldacci & Mingozzi (2009), Li et al. (2010) and Penna et al. (2011) were, to our knowledge, the only authors that dealt with the HVRPFV instances considered in this work. By observing the results presented in Table 5, it can be noted that the ILS-RVND-SP was found capable to improve the result of one instance and to equal the BKS of the remaining ones. The average gap between the Avg. Sols. obtained by ILS-RVND-SP and the BKSs was 0.14%.

4.3. HVRPV

Tables 7-8 present a comparison between the results obtained by the ILS-RVND-SP and the best heuristics proposed in the literature, namely those of Taillard (1999), Li et al. (2007), Prins (2009) and Penna et al. (2011), in the set of instances of Taillard (1999). All proven optimal solutions were found by the proposed algorithm and in the only instance where the optimal solution is not known, the ILS-RVND-SP, as well as the algorithm of Li et al. (2007), Prins (2009) and Penna et al. (2011), failed to obtain the best solution reported by Taillard (1999). The average gap between the Avg. Sols. found by ILS-RVND-SP and the BKSs was 0.16% and the average computational time was 3.61 seconds. In the set of instances proposed by Brandão (2011), ILS-RVND-SP outperformed the TS algorithm of same author in terms of solution quality, with an average gap of 0.09%, as can be observed in Tables 9-10. Finally, in the large size concentric instances of Li et al. (2007), ILS-RVND-SP did not perform as good as the other approaches from the literature and the average gap was 2.33% (see Tables 11-12). Despite the poor performance of the proposed algorithm in 3 test-problems of this last particular benchmark, we strongly believe that instances with such geographical distribution are seldom found in practice.

4.4. FMSFV

In Tables 13-14 a comparison is performed between the results found by the ILS-RVND-SP and the best heuristics available in the literature, particularly the ones of Choi & Tcha (2007), Prins (2009), Imran et al. (2009) and Penna et al. (2011). The ILS-RVND-SP was found capable to improve one solution and to equal the result of the remaining ones, outperforming the other algorithms in terms of number of best solutions found. The average gap between the Avg. Sols. found by ILS-RVND-SP and the BKSs was 0.02%. Moreover, the average computational time was quite similar to the one reported by Prins (2009), i.e., between 6 and 7 seconds.

Table 1: HFVRP Instances

Inst.	n	A				B				C				D				E				F			
		Q_A	f_A	r_A	m_A	Q_B	f_B	r_B	m_B	Q_C	f_C	r_C	m_C	Q_D	f_D	r_D	m_D	Q_E	f_E	r_E	m_E	Q_F	f_F	r_F	m_F
3	20	20	20	1.0	20	30	35	1.1	20	40	50	1.2	20	70	120	1.7	20	120	225	2.5	20				
4	20	60	1000	1.0	20	80	1500	1.1	20	150	3000	1.4	20												
5	20	20	20	1.0	20	30	35	1.1	20	40	50	1.2	20	70	120	1.7	20	120	225	2.5	20				
6	20	60	1000	1.0	20	30	1500	1.1	20	150	3000	1.4	20												
13	50	20	20	1.0	4	30	35	1.1	2	40	50	1.2	4	70	120	1.7	4	120	225	2.5	2	200	400	3.2	1
14	50	120	1000	1.0	4	160	1500	1.1	2	300	3500	1.4	1												
15	50	50	100	1.0	4	100	250	1.6	3	160	450	2.0	2												
16	50	40	100	1.0	2	80	200	1.6	4	140	400	2.1	3												
17	75	50	25	1.0	4	120	80	1.2	4	200	150	1.5	2	350	320	1.8	1								
18	75	20	10	1.0	4	50	35	1.3	4	100	100	1.9	2	150	180	2.4	2	250	400	2.9	1	400	800	3.2	1
19	100	100	500	1.0	4	200	1200	1.4	3	300	2100	1.7	3												
20	100	60	100	1.0	6	140	300	1.7	4	200	500	2.0	3												

Table 2: HFVRP Instances of Brandão (2011)

Inst.	n	A			B			C			D			E			F		
		Q_A	v_A	n_A	Q_B	v_B	n_B	Q_C	v_C	n_C	Q_D	v_D	n_D	Q_E	v_E	n_E	Q_F	v_F	n_F
N1	150	50	1	5	100	1.5	4	150	1.9	4	200	2.2	3	250	2.6	2	350	3.2	1
N2	199	50	1	8	100	1.5	6	150	1.9	5	200	2.2	4	250	2.6	2			
N3	120	50	1	6	100	1.5	3	150	1.9	3	200	2.2	2						
N4	100	50	1	4	120	1.6	4	180	2.1	4	240	2.6	2						
N5	134	900	1	5	1500	1.5	3	2000	1.8	2	2500	2.2	1						

Table 3: HFVRP Instances of Li et al. (2007)

Inst.	n	A			B			C			D			E			F		
		Q_A	v_A	n_A	Q_B	v_B	n_B	Q_C	v_C	n_C	Q_D	v_D	n_D	Q_E	v_E	n_E	Q_F	v_F	n_F
H1	200	50	1	8	100	1.1	6	200	1.2	4	500	1.7	3	1000	2.5	1			
H2[a]	240	50	1	10	100	1.1	5	200	1.2	5	500	1.7	4	1000	2.5	1			
H3	280	50	1	10	100	1.1	5	200	1.2	5	500	1.7	4	1000	2.5	2			
H4	320	50	1	10	100	1.1	8	200	1.2	5	500	1.7	2	1000	2.5	2	1500	3	1
H5[a]	360	50	1	10	100	1.2	8	200	1.5	5	500	1.8	1	1500	2.5	2	2000	3	1

[a]: Using the values presented in Brandão (2011) (see Brandão (2011), p. 146 for more details).

4.5. FSMF

Tables 15-16 illustrate the results obtained by the ILS-RVND-SP for the FSMF. These results are compared with those of Choi & Tcha (2007), Brandão (2009), Prins (2009), Liu et al. (2009) and Penna et al. (2011). It can be seen that the proposed algorithm equaled the results of all instances, with the exception of instance 20, where a new improved solution was found. Once again the ILS-RVND-SP outperformed the algorithms proposed in the literature in terms of best solutions obtained. The average gap between the Avg. Sols. found by ILS-RVND-SP and the BKSs was 0.08%. Furthermore, it can be seen that the average computational time of our algorithm was smaller than those of the literature.

4.6. FMSV

The best results obtained in the literature for the FMSV using heuristic approaches were reported by Choi & Tcha (2007), Brandão (2009), Prins (2009), Imran et al. (2009) and Penna et al. (2011). These results along with those found by the ILS-RVND-SP are presented in Tables 17–18. In this variant, the optimal solutions of all instances were proven in the literature. From Table 17, it can be observed that the ILS-RVND-SP was capable of finding all optimal solutions and the average gap between the Avg. Sols. produced by the ILS-RVND-SP and the BKSs was 0.06%. One can also verify that our algorithm presented the best performance in terms of best solutions and average computational time. Brandão (2011) presented results

Table 4: Performance evaluation of each phase of ILS-RVND-SP

Variant (Benchmark set)	Phase 1 (ILS-RVND)		Phase 2 (SP)		Avg. Number of Routes (columns)
	Avg. Gap (%)	Time (s)	Avg. Gap (%)	Time (s)	
HVRPFV (Taillard, 1999)	0.86	2.38	0.17	5.35	4031
HVRPV (Taillard, 1999)	1.09	2.42	0.18	1.61	4110
HVRPV (Brandão, 2011)	0.89	20.09	0.05	33.50	15079
HVRPV (Li et al., 2007)	2.37	247.68	2.15	55.09	61345
FSMFV (Taillard, 1999)	1.02	1.73	0.01	5.83	2190
FSMF (Golden et al., 1984)	1.44	2.18	0.11	6.91	3338
FSMV (Taillard, 1999)	0.85	2.15	0.12	1.17	3596
FMSV (Brandão, 2011)	2.63	23.26	0.15	17.45	17942
Average	1.39	37.74	0.37	15.86	13954

for the FSMV by running the TS algorithm proposed in Brandão (2009) in the instances proposed by the same author. We compare such results with those found by ILS-RVND-SP in Tables 19-20, where it can be seen that ILS-RVND-SP was capable to improve the result of 2 instances and to equal the solution cost of the remaining ones.

5. Concluding Remarks

This article dealt with Heterogeneous Fleet Vehicle Routing Problem (HFVRP). This kind of problem often arises in practical applications and one can affirm that this model is more realistic than the classical homogeneous Vehicle Routing Problem. Five HFVRP variants involving limited and unlimited fleet with fixed and/or variable costs were considered. These variants were solved by a hybrid algorithm based on the Iterated Local (ILS) Search metaheuristic, that uses Variable Neighborhood Descent with random neighborhood ordering (RVND) in the local search phase, combined with a Set Partitioning Formulation.

The proposed hybrid algorithm (ILS-RVND-SP) was tested in 67 benchmark instances with up to 360 customers and it was found capable to obtain 8 new improved solutions, to equal the result of 54 instances and failed to obtain the best known solution of only 5 instances.

Table 5: Results for HVRPFV instances

Inst.	n	BKS	MAMP Li et al.[1] Best Sol.	MAMP Li et al.[1] Time[a] (s)	ILS-RVND Penna et al.[2] Best Sol.	ILS-RVND Penna et al.[2] Time[b] (s)	ILS-RVND-SP Best Sol.	ILS-RVND-SP Gap (%)	ILS-RVND-SP Avg. Sol.[b]	ILS-RVND-SP Time[a] (s)	ILS-RVND-SP Gap[a] (%)
13	50	3185.09[a]	**3185.09**	110	**3185.09**	19.84	**3185.09**	0.00	3186.32	1.99	0.04
14	50	10107.53[a]	**10107.53**	34	**10107.53**	11.28	**10107.53**	0.00	10110.61	1.29	0.03
15	50	3065.29[a]	**3065.29**	46	**3065.29**	12.48	**3065.29**	0.00	3065.29	1.77	0.00
16	50	3265.41[a]	**3265.41**	99	**3265.41**	12.22	**3265.41**	0.00	3273.15	1.67	0.24
17	75	2076.96[a]	**2076.96**	148	**2076.96**	29.59	**2076.96**	0.00	2081.55	5.95	0.22
18	75	3743.58[a]	**3743.58**	119	**3743.58**	36.38	**3743.58**	0.00	3758.83	16.47	0.41
19	100	10420.34	**10420.34**	287	**10420.34**	73.66	**10420.34**	0.00	10421.05	15.80	0.01
20	100	4788.49	4832.17	200	**4788.49**	68.46	**4761.26**	-0.57	4822.16	16.87	0.55

[a]: Optimality proved; [a]: Average of 10 runs; [b]: Average of 30 runs; [1]: Intel 2.2 GHz (1917 Mflop/s); [2]: Intel i7 2.93 GHz (5839 Mflop/s)

Table 6: Summary of results for HVRPFV

Method	Best Run Gap (%)	Best Run BKS Found	Best Run BKS Improved	Average[1] Gap (%)	Average[1] Scaled Time (s)
MAMP (Li et al., 2010)	0.11	7	0	0.22	43.25
ILS-RVND (Penna et al., 2011)	0.00	7	0	0.29	32.89
ILS-RVND-SP	-0.07	7	1	0.17	7.73

[1]: Average of 10 runs for Li et al. (2010) and ILS-RVND-SP and of 30 runs for Penna et al. (2011)

Table 7: Results for HVRPV instances

			HCG Taillard[1]		BATA Tarantilis et al[2]		HRTR Li et al[3]		SMA-D2 Prins[4]		ILS-RVND Penna et al.[5]		ILS-RVND-SP				
Inst.	n	BKS	Best Sol.	Time[b] (s)	Best Sol.	Time (s)	Best Sol.	Time (s)	Best Sol.	Time (s)	Best Sol.	Time[c] (s)	Best Sol.	Gap (%)	Avg. Sol.	Time[d] (s)	Gap[d] (%)
13	50	1517.84[a]	1518.05	473	1519.96	843	**1517.84**	358	**1517.84**	33.2	**1517.84**	19.29	**1517.84**	0.00	1517.84	1.33	0.00
14	50	607.53[a]	615.64	575	611.39	387	**607.53**	141	**607.53**	37.6	**607.53**	11.20	**607.53**	0.00	608.74	1.09	0.20
15	50	1015.29[a]	1016.86	335	**1015.29**	368	**1015.29**	166	**1015.29**	6.6	**1015.29**	12.56	**1015.29**	0.00	1015.29	2.13	0.00
16	50	1144.94[a]	1154.05	350	1145.52	341	**1144.94**	188	**1144.94**	7.5	**1144.94**	12.29	**1144.94**	0.00	1145.23	1.41	0.03
17	75	1061.96[a]	1071.79	2245	1071.01	363	**1061.96**	216	1065.85	81.5	**1061.96**	29.92	**1061.96**	0.00	1064.67	4.22	0.26
18	75	1823.58[a]	1870.16	2876	1846.35	971	**1823.58**	366	**1823.58**	190.6	**1823.58**	38.34	**1823.58**	0.00	1831.48	4.06	0.43
19	100	1117.51	**1117.51**	5833	1123.83	428	1120.34	404	1120.34	177.8	1120.34	67.72	1120.34	0.25	1121.11	9.12	0.32
20	100	1534.17[a]	1559.77	3402	1556.35	1156	**1534.17**	447	**1534.17**	223.3	**1534.17**	63.77	**1534.17**	0.00	1536.89	8.89	0.18

[a]: Optimality proved; [b]: Average time of 5 runs; [c] : Average of 30 runs; [d] : Average of 10 runs; [1]: Sun Sparc 10 workstation 50 MHz (27 Mflop/s);
[2]: Pentium II 400 MHz (262 Mflop/s); [3]: AMD Athlon 1.0 GHz (1168 Mflop/s); [4]: Pentium IV M 1.8 GHz (1564 Mflop/s); [5]: Intel i7 2.93 GHz (5839 Mflop/s).

Table 8: Summary of results for HVRPV

Method	Best Run			Average[1]	
	Gap (%)	BKS Found	BKS Improved	Gap (%)	Scaled Time (s)
HCG (Taillard, 1999)	0.93	1	0	2.50	9.30
BATA (Tarantilis et al., 2004)	0.62	1	0	–	27.24[2]
HRTR (Li et al., 2007)	0.03	7	0	–	57.16[4]
SMA-D2 (Prins, 2009)	0.08	6	0	–	25.38[3]
ILS-RVND (Penna et al., 2011)	0.03	7	0	0.22	31.89
ILS-RVND-SP	0.03	7	0	0.18	4.03

[1]: Average of 5 runs for Taillard (1999), of 30 runs for Penna et al. (2011) and of 10 runs for ILS-RVND-SP; [2]: Single Run; [3]: Best Run; [4]: Determistic Algorithm

Table 9: Results for HVRPV on the instances of Brandão (2011)

Inst.	n	BKS	TSA Brandão		ILS-RVND-SP				
			Best Sol.	Time[1] (s)	Best Sol.	Gap (%)	Avg. Sol.[a]	Time[a] (s)	Gap[a] (%)
N1	150	2243.76	**2243.76**	–	**2235.87**	-0.35	2244.31	51.50	0.02
N2	199	2874.13	**2874.13**	–	**2864.83**	-0.32	2906.24	102.77	1.12
N3	120	2386.90	**2386.90**	–	**2378.99**	-0.33	2382.10	51.71	-0.20
N4	100	1839.22	**1839.22**	–	**1839.22**	0.00	1839.22	9.64	0.00
N5	134	2062.48	**2062.48**	–	**2047.81**	-0.71	2047.81	52.33	-0.71

[a]: Average of 10 runs; [1]: Pentium IV 2.6 GHz (2266 Mflop/s)

Table 10: Summary of results for HVRPV on the instances of Brandão (2011)

Method	Best Run			Average[1]	
	Gap (%)	BKS Found	BKS Improved	Gap (%)	Scaled Time
TSA (Brandão, 2011)	0.00	5	0	–	–
ILS-RVND-SP	-0.34	1	4	0.05	53.59

[1]: Average of 10 runs for ILS-RVND-SP.

Table 11: Results for HVRPV on the instances of Li et al. (2007)

Inst.	n	BKS	HRTR Li et al.		TSA Brandão		ILS-RVND-SP				
			Best Sol.	Time[1] (s)	Best Sol.	Time[2] (s)	Best Sol.	Gap (%)	Avg. Sol.[b]	Time[b] (s)	Gap[b] (%)
H1	200	12050.08	12067.65	687.82	**12050.08**	1395	**12050.08**	0.00	12052.69	72.10	0.02
H2	240	10208.32[a]	10234.40	995.27	10226.17	3650	10329.15	1.18	10436.20	176.43	2.23
H3	280	16223.39[a]	16231.80	1437.56	16230.21	2822	16282.41	0.36	16526.89	259.61	1.87
H4	320	17458.65	17576.10	2256.35	**17458.65**	8734	17743.68	1.63	18022.37	384.52	3.23
H5	360	23166.56[a]	–	–	23220.72	13321	23493.87	1.41	23948.97	621.17	3.38

[a]: Found by Brandão (2011) using TSA with a different calibration; [b]: Average of 10 runs;

[1]: AMD Athlon 1.0 GHz (1168 Mflop/s); [2]: Pentium IV 2.6 GHz (2266 Mflop/s)

Table 12: Summary of results for HVRPV on the instances of Li et al. (2007)

Method	Best Run			Average[1]	
	Gap (%)	BKS Found	BKS Improved	Gap (%)	Scaled Time
HRTR (Li et al., 2007)	0.28[a]	0	0	–	346.22[2]
TSA (Brandão, 2011)	0.09 (0.05)[a]	2	0	–	1246.28[2]
ILS-RVND-SP	0.92 (0.80)[a]	1	0	2.15 (1.84)[a]	302.77

[1]: Average of 10 runs for ILS-RVND-SP; [2]: Determistic Algorithm; [a]: Values in instances H1-H4

Table 13: Results for FMSFV instances

Inst.	n	BKS	CG Choi and Tcha[1]		SMA-U1 Prins[2]		VNS1 Imran et al[3]		ILS-RVND Penna et al[4]		ILS-RVND-SP				
			Best Sol.	Time (s)	Best Sol.	Time (s)	Best Sol.	Time (s)	Best Sol.	Time[b] (s)	Best Sol.	Gap (%)	Avg. Sol.	Time[c] (s)	Gap[c] (%)
3	20	1144.22[a]	**1144.22**	0.25	**1144.22**	0.01	**1144.22**	19	**1144.22**	4.05	**1144.22**	0.00	1144.22	0.34	0.00
4	20	6437.33[a]	**6437.33**	0.45	**6437.33**	0.07	**6437.33**	17	**6437.33**	3.03	**6437.33**	0.00	6437.33	0.31	0.00
5	20	1322.26[a]	**1322.26**	0.19	**1322.26**	0.02	**1322.26**	24	**1322.26**	4.85	**1322.26**	0.00	1322.26	0.28	0.00
6	20	6516.47[a]	**6516.47**	0.41	**6516.47**	0.07	**6516.47**	21	**6516.47**	3.01	**6516.47**	0.00	6516.47	0.32	0.00
13	50	2964.65[a]	**2964.65**	3.95	**2964.65**	0.32	**2964.65**	328	**2964.65**	27.44	**2964.65**	0.00	2964.65	1.70	0.00
14	50	9126.90[a]	**9126.90**	51.70	**9126.90**	8.90	**9126.90**	250	**9126.90**	11.66	**9126.90**	0.00	9126.90	1.53	0.00
15	50	2634.96[a]	**2634.96**	4.36	2635.21	1.04	**2634.96**	275	**2634.96**	13.83	**2634.96**	0.00	2634.96	1.34	0.00
16	50	3168.92[a]	**3168.92**	5.98	3169.14	13.05	3168.95	313	**3168.92**	18.20	**3168.92**	0.00	3168.92	6.72	0.00
17	75	2004.48[a]	2023.61	68.11	**2004.48**	23.92	**2004.48**	641	**2004.48**	43.68	**2004.48**	0.00	2007.12	6.96	0.13
18	75	3147.99[a]	**3147.99**	18.78	3153.16	24.85	3153.67	835	3149.63	47.80	**3147.99**	0.00	3148.91	4.21	0.03
19	100	8661.81[a]	8664.29	905.20	8664.67	163.25	8666.57	1411	**8661.81**	59.13	**8661.81**	0.00	8662.89	29.86	0.01
20	100	4153.02	4154.49	53.02	4154.49	41.25	4164.85	1460	**4153.02**	59.07	**4153.02**	0.00	4153.12	37.21	0.00

[a]: Optimality proved; [b]: Average of 30 runs; [c]: Average of 10 runs; [1]: Pentium IV 2.6 GHz (2266 Mflop/s); [2]: Pentium IV M 1.8 GHz (1564 Mflop/s); [3]: Pentium M 1.7 GHz (1477 Mflop/s); [4]: Intel i7 2.93 GHz (5839 Mflop/s).

Table 14: Summary of results for FMSFV

Method	Best Run			Average[1]	
	Gap (%)	BKS Found	BKS Improved	Gap (%)	Scaled Time (s)
CG (Choi & Tcha, 2007)	0.08	9	0	0.11	42.82
SMA-U1 (Prins, 2009)	0.02	7	0	–	6.86
VNS1 (Imran et al., 2009)	0.04	8	0	–	117.92[2]
ILS-RVND (Penna et al., 2011)	0.01	11	0	0.09	24.64
ILS-RVND-SP	0.00	12	0	0.01	7.56

[1]: Average of 5 runs for Choi & Tcha (2007) and Prins (2009), of 30 runs for Penna et al. (2011) and of 10 runs for ILS-RVND-SP; [2]: Total Time.

Table 15: Results for FSMF instances

Inst.	n	BKS	CG Choi and Tcha[1]		TSA1 Brandão[2]		SMA-D1 Prins[3]		GA Liu et al[4]		ILS-RVND Penna et al.[5]		ILS-RVND-SP				
			Best Sol.	Time (s)	Best Sol.	Time (s)	Best Sol.	Time (s)	Best Sol.	Time[b] (s)	Best Sol.	Time[b] (s)	Best Sol.	Gap (%)	Avg. Sol.[c]	Time[c] (s)	Gap[c] (%)
3	20	961.03[a]	**961.03**	0	**961.03**	21	**961.03**	0.04	**961.03**	21	**961.03**	4.91	**961.03**	0.00	961.03	0.28	0.00
4	20	6437.33[a]	**6437.33**	1	**6437.33**	22	**6437.33**	0.03	**6437.33**	18	**6437.33**	3.16	**6437.33**	0.00	6437.33	0.25	0.00
5	20	1007.05[a]	**1007.05**	1	**1007.05**	20	**1007.05**	0.09	**1007.05**	13	**1007.05**	5.88	**1007.05**	0.00	1008.76	0.25	0.17
6	20	6516.47[a]	**6516.47**	0	**6516.47**	25	**6516.47**	0.08	**6516.47**	22	**6516.47**	3.07	**6516.47**	0.00	6516.47	0.20	0.00
13	50	2406.36[a]	**2406.36**	10	**2406.36**	145	**2406.36**	17.12	**2406.36**	91	2408.41	30.29	**2406.36**	0.00	2411.31	1.96	0.21
14	50	9119.03[a]	**9119.03**	51	**9119.03**	220	**9119.03**	19.66	**9119.03**	42	**9119.03**	11.89	**9119.03**	0.00	9119.03	1.64	0.00
15	50	2586.37[a]	**2586.37**	10	2586.84	110	**2586.37**	25.1	**2586.37**	48	**2586.37**	20.24	**2586.37**	0.00	2586.37	6.02	0.00
16	50	2720.43[a]	**2720.43**	11	2728.14	111	2729.08	16.37	2724.22	107	**2724.22**	20.67	**2720.43**	0.00	2724.55	3.85	0.15
17	75	1734.53[a]	1744.83	207	1736.09	322	1746.09	52.22	**1734.53**	109	**1734.53**	52.49	**1734.53**	0.00	1744.23	11.61	0.56
18	75	2369.65[a]	2371.49	70	2376.89	267	**2369.65**	36.92	**2369.65**	197	2371.48	55.35	**2369.65**	0.00	2373.79	11.83	0.17
19	100	8661.81[a]	8664.29	1179	8667.26	438	8665.12	169.93	8662.94	778	8662.86	63.92	**8661.81**	0.00	8662.54	25.15	0.01
20	100	4037.90	4039.49	264	4048.09	601	4044.78	172.73	4038.46	1004	**4037.90**	93.88	**4032.81**	-0.13	4038.63	46.06	0.02

[a]: Optimality proved; [b]: Average of 30 runs; [c]: Average of 10 runs; [1]: Pentium IV 2.6 GHz (2266 Mflop/s); [2]: Pentium M 1.4 GHz (1564 Mflop/s);
[3]: Pentium IV M 1.8 GHz (1564 Mflop/s); [4]: Pentium IV 3.0 GHz (3181 Mflop/s); [5]: Intel i7 2.93 GHz (5839 Mflop/s).

Table 16: Summary of results for FSMF

Method	Best Run			Average[1]	
	Gap (%)	BKS Found	BKS Improved	Gap (%)	Scaled Time (s)
CG (Choi & Tcha, 2007)	0.06	8	0	0.17	58.36
TSA1 (Brandão, 2009)	0.08	6	0	–	39.95[2]
SMA-D1 (Prins, 2009)	0.10	8	0	–	10.92
GA (Liu et al., 2009)	0.01	10	0	0.19	107.96
ILS-RVND (Penna et al., 2011)	0.01	9	0	0.23	30.48
ILS-RVND-SP	-0.01	11	1	0.11	9.09

[1]: Average of 5 runs for Choi & Tcha (2007) and Prins (2009) and of 10 runs for Liu et al. (2009) and ILS-RVND-SP; [2]: Determistic Algorithm.

Table 17: Results for FMSV instances

Inst.	n	BKS	CG Choi and Tcha[1]		TSA2 Brandão[2]		SMA-U2 Prins[3]		VNS2 Imran et al[4]		ILS-RVND Penna et al.[5]		ILS-RVND-SP				
			Best Sol.	Time (s)	Best Sol.	Time (s)	Best Sol.	Time (s)	Best Sol.	Time[b] (s)	Best Sol.	Time[c] (s)	Best Sol.	Gap (%)	Avg. Sol.[d]	Time[d] (s)	Gap[d] (%)
3	20	623.22[a]	623.22	0.19	–	–	–	–	–	–	**623.22**	4.58	**623.22**	0.00	623.22	0.25	0.00
4	20	387.18[a]	**387.18**	0.44	–	–	–	–	–	–	**387.18**	2.85	**387.18**	0.00	387.34	0.23	0.04
5	20	742.87[a]	**742.87**	0.23	–	–	–	–	–	–	**742.87**	5.53	**742.87**	0.00	742.87	0.22	0.00
6	20	415.03[a]	**415.03**	0.92	–	–	–	–	–	–	**415.03**	3.37	**415.03**	0.00	415.03	0.18	0.00
13	50	1491.86[a]	**1491.86**	4.11	**1491.86**	101	**1491.86**	3.45	**1491.86**	310	**1491.86**	31.62	**1491.86**	0.00	1492.01	1.91	0.01
14	50	603.21[a]	**603.21**	20.41	**603.21**	135	**603.21**	0.86	**603.21**	161	**603.21**	14.66	**603.21**	0.00	605.00	1.61	0.30
15	50	999.82[a]	**999.82**	4.61	**999.82**	137	**999.82**	9.14	**999.82**	218	**999.82**	15.33	**999.82**	0.00	1001.03	1.47	0.12
16	50	1131.00[a]	**1131.00**	3.36	**1131.00**	95	**1131.00**	13.00	**1131.00**	239	**1131.00**	17.77	**1131.00**	0.00	1131.85	1.44	0.07
17	75	1038.60[a]	**1038.60**	69.38	**1038.60**	312	**1038.60**	9.53	**1038.60**	509	**1038.60**	49.18	**1038.60**	0.00	1042.48	6.39	0.37
18	75	1800.80[a]	1801.40	48.06	1801.40	269	**1800.80**	18.92	**1800.80**	606	**1800.80**	53.88	**1800.80**	0.00	1802.89	4.75	0.12
19	100	1105.44[a]	**1105.44**	182.86	**1105.44**	839	**1105.44**	52.31	**1105.44**	1058	**1105.44**	77.84	**1105.44**	0.00	1106.71	10.62	0.11
20	100	1530.43[a]	**1530.43**	98.14	1531.83	469	1535.12	104.41	1533.24	1147	1530.52	88.02	**1530.43**	0.00	1534.23	10.88	0.25

[a]: Optimality proved; [b]: Total time of 10 runs; [c]: Average of 30 runs; [d]: Average of 10 runs; [1]: Pentium IV 2.6 GHz (2266 Mflop/s);

[2]: Pentium M 1.4 GHz (1564 Mflop/s); [3]: Pentium IV M 1.8 GHz (1564 Mflop/s); [4]: Pentium M 1.7 GHz (1477 Mflop/s); [5]: Intel i7 2.93 GHz (5839 Mflop/s).

Table 18: Summary of results for FMSV

Method	Best Run			Average[1]	
	Gap (%)	BKS Found	BKS Improved	Gap (%)	Scaled Time (s)
CG (Choi & Tcha, 2007)	0.00	11	0	0.12	21.05
TSA1 (Brandão, 2009)	0.02	6	0	–	61.36[2]
SMA-D1 (Prins, 2009)	0.04	7	0	–	8.46[a]
VNS1 (Imran et al., 2009)	0.02	7	0	–	134.32
ILS-RVND (Penna et al., 2011)	0.00 (0.00)[a]	11 (7)[a]	0	0.17 (0.26)[a]	30.38 (43.54)[a]
ILS-RVND-SP	0.00 (0.00)[a]	12 (8)[a]	0	0.12 (0.09)[a]	3.33 (4.29)[a]

[1]: Average of 5 runs for Choi & Tcha (2007) and Prins (2009), 10 runs for Liu et al. (2009) and ILS-RVND-SP; [2]: Determistic Algorithm; [a]: Values in instances 13-20

Table 19: Results for FMSV on the instances of Brandão (2011)

			TSA Brandão		ILS-RVND-SP				
Inst.	n	BKS	Best Sol.	Time[1] (s)	Best Sol.	Gap (%)	Avg. Sol.[a]	Time[a] (s)	Gap[a] (%)
N1	150	2220.01	**2220.01**	–	**2212.77**	-0.33	2219.66	39.60	-0.02
N2	199	2827.76	**2827.76**	–	**2823.75**	-0.14	2844.96	106.97	0.61
N3	120	2234.57	**2234.57**	–	**2234.57**	0.00	2234.85	19.27	0.01
N4	100	1822.78	**1822.78**	–	**1822.78**	0.00	1823.07	8.38	0.02
N5	134	2016.79	**2016.79**	–	**2016.79**	0.00	2019.26	29.35	0.12

[a]: Average of 10 runs; [1]: Pentium IV 2.6 GHz (2266 Mflop/s)

Table 20: Summary of results for FMSV on the instances of Brandão (2011)

Method	Best Run			Average[1]	
	Gap	BKS Found	BKS Improved	Gap	Scaled Time
TSA (Brandão, 2009)[a]	0.00	5	0	–	–
ILS-RVND-SP	-0.09	3	2	0.15	40.71

[1]: Average of 10 runs for ILS-RVND-SP; [a]: Presented in Brandão (2011) using TSA version of Brandão (2009)

A. New best solutions

A.1. HVRPFV

Instance 20: 12 routes, cost 4761.26

(A): 0 18 83 8 45 17 84 60 0; (A): 0 74 22 41 15 43 57 2 0; (A): 0 91 44 38 14 42 0; (A): 0 92 37 100 98 99 96 6 0; (A): 0 70 78 34 29 24 25 55 54 0; (B): 0 12 80 68 79 3 77 76 28 0; (B): 0 52 7 48 19 11 62 88 31 69 0; (B): 0 94 95 97 87 13 58 53 0; (B): 0 10 32 90 63 64 49 36 47 46 82 0; (C): 0 89 5 61 86 16 85 93 59 0; (C): 0 26 4 39 67 23 56 75 72 73 21 40 0; (C): 0 50 33 81 51 9 35 71 65 66 20 30 1 27 0

A.2. HVRPV

Instance N1: 17 routes, cost 2235.87

(A): 0 42 142 43 15 41 145 0; (A): 0 105 53 0; (A): 0 147 89 0; (A):0 55 25 67 56 73 0; (B): 0 58 2 115 57 144 87 137 0; (B):0 97 100 119 14 38 140 44 91 0; (B):0 18 114 8 45 125 83 60 0; (B):0 46 124 47 36 143 49 64 7 0; (C):0 50 102 33 81 120 9 103 51 0; (C):0 1 122 20 128 66 71 65 136 35 135 34 78 0; (C):0 28 138 12 150 80 68 116 76 111 0; (C):0 109 54 130 134 24 29 121 129 79 3 77 0; (D):0 127 88 148 62 11 107 19 123 48 82 106 52 0; (D):0 6 61 16 141 86 113 17 84 5 118 0; (D):0 146 31 10 108 126 63 90 32 131 30 70 101 69 132 27 0; (E):0 40 21 72 74 22 133 75 23 39 139 4 110 149 26 0; (E):0 13 117 95 92 37 98 85 93 59 104 99 96 94 112 0;

Instance N2: 24 routes, cost 2864.83

(A): 0 112 0; (A): 0 58 152 0; (A): 0 132 1 176 0; (A): 0 138 154 0; (A): 0 156 147 0; (A): 0 6 91 140 38 43 15 57 0; (B): 0 167 127 190 162 27 0; (B): 0 98 16 86 113 17 84 60 0; (B): 0 121 29 24 163 134 54 195 0; (B): 0 126 63 181 64 49 143 36 46 0; (B): 0 94 95 97 117 13 0; (B): 0 26 149 180 105 53 0; (C): 0 21 72 74 75 23 186 56 197 198 0; (C): 0 18 114 8 174 45 125 199 83 166 0; (C): 0 153 82 124 47 168 48 7 194 106 0; (C): 0 12 109 177 150 80 68 116 184 28 0; (C): 0 40 73 171 133 22 41 145 115 178 2 0; (D): 0 122 20 188 66 65 136 35 135 71 161 103 51 0; (D): 0 69 101 70 30 128 160 131 32 90 108 189 10 31 0; (D): 0 110 155 4 139 187 39 67 170 25 55 165 130 179 0; (D): 0 52 182 123 19 107 175 11 159 62 148 88 146 0; (E): 0 89 118 5 173 61 85 93 59 104 99 96 183 0; (E): 0 137 87 144 172 42 142 14 192 119 44 141 191 193 100 37 92 151 0; (F): 0 76 196 77 158 3 79 129 169 78 34 164 120 9 81 185 33 157 102 50 111 0;

Instance N3: 13 routes, cost 2378.99

(A): 0 120 119 82 0; (A): 0 105 106 107 103 104 102 0; (A): 0 67 70 69 0; (A): 0 87 86 111 88 0; (A): 0 84 113 83 117 112 0; (B): 0 95 96 94 97 115 110 98 116 99 0; (B): 0 92 89 91 90 114 108 118 18 85 0; (B): 0 21 26 29 32 35 36 34 33 30 27 31 28 23 20 0; (C): 0 73 71 74 72 75 78 80 79 77 76 68 101 0; (C): 0 81 2 1 3 4 11 15 14 13 9 10 5 0; (C): 0 6 7 8 12 16 22 24 25 19 17 109 0; (D): 0 53 55 58 56 60 63 66 64 62 61 65 59 57 54 52 100 0; (D): 0 40 43 45 48 51 50 49 46 47 44 41 42 39 38 37 93 0;

Instance N5: 11 routes, cost 2047.81

(A):0 80 33 0; (A):0 20 83 85 84 86 87 89 90 25 0; (A):0 77 64 63 79 67 70 69 68 133 78 0; (A):0 29 93 94 45 43 44 40 3 41 42 2 4 5 6 7 8 9 10 12 11 14 88 15 13 16 92 28 27 0; (A):0 66 71 118 46 82 0; (B):0 72 47 75 1 62 52 51 50 49 48 34 32 134 76 74 73 0; (B):0 17 131 114 115 119 130 65 19 0; (B):0 91 21 26 30 31 59 23 24 22 0; (C):0 60 58 57 105 97 96 38 39 95 37 98 100 99 36 35 101 104 102 53 103 56 55 54 61 0; (C):0 18 117 116 106 107 108 109 120 121 122 0; (D):0 81 112 125 111 110 123 124 126 127 128 129 113 132 0;

A.3. FSMF

Instance 20: 19 routes, cost 4032.81

(A): 0 68 80 54 0 (A): 0 59 97 95 0 (A): 0 41 22 75 74 21 0 (A): 0 26 72 73 40 0 (A): 0 50 33 81 51 0 (A): 0 85 100 92 0 (A): 0 77 3 79 1 0 (A): 0 96 93 94 0 (A): 0 48 46 8 83 60 0 (A): 0 52 7 62 31 0 (A): 0 99 5 84 17 45 0 (A): 0 89 6 13 58 0 (A): 0 69 10 11 19 88 0 (A): 0 27 76 28 53 0 (A): 0 87 42 43 15 57 2 0 (B): 0 18 82 47 36 49 64 63 90 32 70 0 (B): 0 61 16 86 38 14 44 91 98 37 0 (B): 0 30 20 66 65 71 35 9 34 78 29 0 (B): 0 12 24 55 25 39 67 23 56 4 0

A.4. FSMV

Instance N1: 17 routes, cost 2212.77

(A): 0 112 0; (A): 0 138 149 26 0; (A): 0 53 105 0; (A): 0 57 15 43 38 140 91 6 0; (A): 0 121 29 24 25 55 130 0; (B): 0 73 133 22 41 145 115 2 58 0; (B): 0 18 114 8 45 125 83 60 0; (B): 0 7 64 49 143 36 47 124 46 0; (C): 0 69 122 20 66 65 136 35 135 71 103 51 1 0; (D): 0 77 3 79 129 78 34 120 9 81 33 102 50 0; (D):

0 40 21 72 74 75 56 23 67 39 139 4 110 0; (D): 0 146 31 10 108 126 63 90 32 131 128 30 70 101 132 27 0; (D): 0 13 117 97 100 141 44 119 14 142 42 144 87 137 0; (D): 0 28 12 109 54 134 80 150 68 116 76 111 0; (D): 0 52 106 82 48 123 19 107 11 62 148 88 127 0; (D): 0 89 118 5 84 17 113 86 16 61 99 104 0; (D): 0 94 95 92 37 98 85 93 59 96 147 0;

Instance N2 : 18 routes, cost 2823.75

(A): 0 183 13 0; (A): 0 117 91 140 38 43 15 57 0; (A): 0 152 58 0; (A): 0 112 156 0; (B): 0 126 63 181 64 49 143 36 46 0; (B): 0 121 29 24 163 134 54 195 0; (C): 0 106 194 7 48 168 47 124 82 153 0; (C): 0 111 50 102 3 158 77 196 76 0; (C): 0 132 69 162 31 190 127 167 27 0; (D): 0 94 95 92 151 98 85 93 59 104 99 96 6 0; (D): 0 89 166 60 84 17 113 86 141 16 61 173 5 0; (D): 0 146 88 148 62 159 11 175 107 19 123 182 52 0; (D): 0 176 1 122 30 128 160 131 32 90 108 10 189 0; (D): 0 137 2 178 115 145 41 22 133 74 171 73 180 105 0; (D): 0 138 154 12 109 177 150 80 68 116 184 28 0; (D): 0 179 130 165 55 25 170 67 39 187 139 155 4 110 0; (D): 0 185 79 129 169 78 34 164 120 9 81 33 157 0; (D): 0 18 114 8 174 45 125 199 83 118 147 0; (D): 0 26 149 198 197 56 186 23 75 72 21 40 53 0; (D): 0 101 70 20 188 66 65 136 35 135 71 161 103 51 0; (D): 0 97 37 100 193 191 44 119 192 14 142 42 172 144 87 0;

References

Baldacci, R., Christofides, N., & Mingozzi, A. (2008). An exact algorithm for the vehicle routing problem based on the set partitioning formulation with additional cuts. *Mathematical Programming*, *115*, 351–385.

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

Brandão, J. (2011). A tabu search algorithm for the heterogeneous fixed fleet vehicle routing problem. *Computers & Operations Research*, *38*, 140–151.

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

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

Dongarra, J. J. (2010). *Performance of various computers using standard linear equations software*. Technical Report CS-89-85 Computer Science Department, University of Tennessee.

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

Golden, B. L., Assad, A. A., Levy, L., & Gheysens, F. G. (1984). The feet size and mix vehicle routing problem. *Computers & Operations Research*, *11*, 49–66.

Hoff, A., Andersson, H., Christiansen, M., Hasle, G., & Løkketangen, A. (2010). Industrial aspects and literature survey: Fleet composition and routing. *Computers & Operations Research*, *37*, 2041–2061.

Imran, A., Salhi, S., & Wassan, N. A. (2009). A variable neighborhood-based heuristic for the heterogeneous fleet vehicle routing problem. *European Journal of Operational Research*, *197*, 509–518.

Lee, Y., Kim, J., Kang, K., & Kim, K. (2008). A heuristic for vehicle fleet mix problem using tabu search and set partitioning. *Journal of the Operational Research Society*, *59*, 833–841.

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*, 2734–2742.

Li, X., Tian, P., & Aneja, Y. (2010). An adaptive memory programming metaheuristic for the heterogeneous fixed fleet vehicle routing problem.

Transportation Research Part E: Logistics and Transportation Review, *46*, 1111 – 1127.

Liu, S., Huang, W., & Ma, H. (2009). An effective genetic algorithm for the fleet size and mix vehicle routing problems. *Transportation Research Part E*, *45*, 434–445.

Lourenço, H. R., Martin, O. C., & Stützle, T. (2003). Handbook of metaheuristics. chapter Iterated Local Search. (pp. 321–353). Kluwer Academic Publishers.

Mladenovic, N., & Hansen, P. (1997). Variable neighborhood search. *Computers & Operations Research*, *24*, 1097–1100.

Ochi, L., Vianna, D., Drummond, L. M. A., & Victor, A. (1998a). An evolutionary hybrid metaheuristic for solving the vehicle routing problem with heterogeneous fleet. *Lecture Notes in Computer Science*, *1391*, 187–195.

Ochi, L., Vianna, D., Drummond, L. M. A., & Victor, A. (1998b). A parallel evolutionary algorithm for the vehicle routing problem with heterogeneous fleet. *Future Generation Computer Systems*, *14*, 285–292.

Penna, P. H. V., Subramanian, A., & Ochi, L. S. (2011). An iterated local search heuristic for the heterogeneous fleet vehicle routing problem. *Journal of Heuristics*, . To appear.

Pessoa, A., Uchoa, E., & de Aragão, M. P. (2008). The vehicle routing problem: Latest advances and new challenges. chapter Robust Branch-and-Cut-and-Price Algorithm for Vehicle Routing Problems. (pp. 297–325). Springer.

Pessoa, A., Uchoa, E., & de Aragão, M. P. (2009). A robust branch-cut-and-price algorithm for the heterogeneous fleet vehicle routing problem. *Networks*, *54*, 167–177.

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

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

Renaud, J., & Boctor, F. (2002). A sweep-based algorithm for the fleet size and mix vehicle routing problem. *European Journal of Operational Research*, *140*, 618–628.

Rochat, Y., & Taillard, R. D. (1995). Probabilistic diversification and intensification in local search for vehicle routing. *Journal of Heuristics*, *1*, 147–167.

Taillard, E. D. (1999). A heuristic column generation method for heterogeneous fleet. *RAIRO (Recherche opeH rationnelle)*, *33*, 1–14.

Tarantilis, C. D., & Kiranoudis, C. (2002). Boneroute: An adaptive memory-based method for effective fleet management. *Annals of Operations Research*, *115*, 227–241.

Tarantilis, C. D., Kiranoudis, C., & Vassiliadis, V. (2003). A list based threshold accepting metaheuristic for the heterogeneous fixed fleet vehicle routing problem. *Journal of the Operational Research Society*, *54*, 65–71.

Tarantilis, C. D., Kiranoudis, C. T., & Vassiliadis, V. S. (2004). A threshold accepting metaheuristic for the heterogeneous fixed fleet vehicle routing problem. *European Journal of Operational Research*, *152*, 148–158.

Yaman, H. (2006). Formulations and valid inequalities for the heterogeneous vehicle routing problem. *Mathematical Programming*, *106*, 3650–390.