International Conference
on Industrial Engineering and Systems Management
IESM 2011
May 25 - May 27
METZ - FRANCE

A Hybrid Algorithm for the Fleet Size and Mix Vehicle Routing Problem ⋆

Anand SUBRAMANIAN [a], Puca Huachi Vaz PENNA [b], Eduardo UCHOA [c], Luiz Satoru OCHI [a]

[a] *Universidade Federal Fluminense*
Instituto de Computação
Rua Passo da Pátria 156 - Bloco E - 3º andar
São Domingos, Niterói - RJ - Brasil - CEP: 24210-240

[b] *Universidade Federal Fluminense*
Instituto do Noroeste Flumiense de Educação Superior
Rua Passo da Pátria 156 - Bloco E - 3º andar
São Domingos, Niterói - RJ - Brasil - CEP: 24210-240

[c] *Universidade Federal Fluminense*
Departamento de Engenharia de Produção
Rua Passo da Pátria 156 - Bloco E - 4º andar
São Domingos, Niterói - RJ - Brasil - CEP: 24210-240

Abstract

This paper deals with the Fleet Size and Mix (FSM) Vehicle Routing Problem. The FSM generalizes the classical Vehicle Routing Problem (VRP) by allowing the existence of an unlimited heterogeneous fleet of vehicles. Each type of vehicle is associated with a distance-dependent and/or fixed cost. The objective is to determine the best fleet composition as well as the set of routes that minimize the travel 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 (MIP) solver that interactively calls the ILS heuristic during its execution. The developed algorithm was tested on benchmark instances involving up to 100 customers. The results obtained are highly competitive since the hybrid approach was always capable to find or improve the best known solutions reported in the literature.

Key words: Fleet Size and Mix, Vehicle Routing Problem, Matheuristics, Iterated Local Search, Set Partitioning

1 Introduction

The Vehicle Routing Problem (VRP) is one of the most studied problems in the fields of Operations Research and Combinatorial Optimization. The importance of this problem can be verified by observing the huge number of works that had been proposed over the last 50 years. Motivated by applications that

⋆ This paper was not presented to another journal. Corresponding author Puca Huachi V. Penna. Tel. +55 21 2629-5665. Fax +55 21 2629-5666.

Email addresses: anand@ic.uff.br (Anand SUBRAMANIAN), ppenna@ic.uff.br (Puca Huachi Vaz PENNA), uchoa@producao.uff.br (Eduardo UCHOA), satoru@ic.uff.br (Luiz Satoru OCHI).

arise in real-life, a large variety of VRPs were suggested in the literature. These variants include some particularities such as time windows, multiple depots, route duration, mixed vehicle fleet, etc.

This paper deals with the Fleet Size and Mix (FSM) Vehicle Routing Problem, which can be define 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 unlimited fleet is composed by m different types of vehicles, with $M = \{1, \ldots, m\}$, each with a capacity Q_u. Every vehicle is associated with a fixed cost denoted by f_u. Finally, for each arc $(i, j) \in A$ there is an associated cost $c^u_{ij} = d_{ij} r_u$, where d_{ij} is the distance between the vertices (i, j) and r_u is a dependent (variable) cost per distance unit, of a vehicle u. The objective is to determine the best fleet composition as well as the set of routes that minimize the travel costs in such a way that: (i) every route starts and ends at the depot; (ii) all the demands are satisfied; (iii) vehicle capacities are not exceeded; (iv) a customer is visited by only a single vehicle; (v) the sum of costs is minimized. The FSM is $\mathcal{NP}$-hard since it can be reduced to the classical VRP when all vehicles are identical.

In practical situations, a fleet of vehicles is less likely to be homogeneous [10]. Normally, either a acquired fleet is heterogeneous itself or it becomes heterogeneous over the time when vehicles with different characteristics are incorporated into the homogeneous fleet. In addition, insurance, maintenance and operating costs usually have distinct values according to the level of depreciation or usage time of the fleet. Moreover, from both tactical and operational point of view, a mixed vehicle fleet also increases the flexibility in terms of distribution planning.

In this work, we propose a hybrid algorithm, that is composed of an Iterated Local Search (ILS) based heuristic and a Set Partitioning (SP) formulation, to solve the following FSM variants: (i) FSM with fixed and dependent costs (FSMFD); (ii) FSM with fixed costs (FSMF); and (iii) FSM with dependent costs (FSMD). The SP model is solved by means of a Mixed Integer Programming (MIP) solver that interactively calls the ILS heuristic during its execution. The three main FSM variants were tackled and the results obtained were compared with other solution approaches found in the literature.

The remainder of this paper is organized as follows. Section 2 reviews some works related to the FSM. 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

The FSM was introduced by Golden et al. [9] where the authors developed two heuristic algorithms as solution approaches. The first one is based on the savings algorithm [5] whereas the second one is based on a giant tour scheme. Another variant, called Heterogeneous VRP (HVRP), was later proposed by Taillard [30] at it consists in optimizing the use of an available fixed fleet. The author developed an algorithm based on adaptive memory procedure (AMP), Tabu Search (TS) and column generation which was also applied to solve the FSM.

Some exact approaches were developed for the FSM. Yaman et al. [31] suggested valid inequalities and presented lower bounds for the FSMF. Choi and Tcha [4] obtained lower bounds for all FSM variants by means of a column generation algorithm based on a set covering formulation. Baldacci et al. [5] proposed some valid inequalities as well as a two-commodity Mixed Integer Programming (MIP) formulation for the same variant. Pessoa et al. [24] proposed a Branch-Cut-and-Price (BCP) algorithm also capable of solving all FSM variants. The same authors also employed a BCP algorithm over an extended formulation to solve the FSM and other VRPs such as the Open VRP and the Assymetric VRP (Pessoa et al. [23]). More recently, Baldacci and Mingozzi [2] put forward a set-partitioning based algorithm, also capable of solving the HVRP, that uses bounding procedures based on linear relaxation and lagrangean relaxation.

Some authors implemented heuristic procedures based on Evolutionary Algorithms. Ochi et al. [19] developed a hybrid evolutionary heuristic that combines a Genetic Algorithm (GA) [11] with Scatter Search [8] to solve the FSMF. A parallel version, based on the island model, of the same algorithm was presented by Ochi et al. [20]. A hybrid GA, whose main components were taken from Prins [25], was proposed by

Liu et al. [15] to solve the FSMF and the FSMD. A Memetic Algorithm (MA) [18] was proposed by Lima et al. [14] for solving FSMF. Two heuristic procedure based on the same metaheuristic were developed by Prins [26] to solve all FSM variants and the HVRP.

Renaud and Boctor [27] proposed a sweep-based heuristic for the FSMF that integrates classical construction and improvement VRP approaches. Imran et al. [12] developed a Variable Neighborhood Search (VNS) [17] algorithm that makes use of a procedure based on Dijkstra's and sweep algorithms for generating an initial solution and several neighborhood structures in the local search phase. The authors considered all FSM variants.

A couple of TS heuristics were proposed to solve the FSMF and the FSMD. Gendreau et al. [7] suggested a TS algorithm that incorporates a GENIUS approach and an adaptive memory procedure. Lee et al. [13] developed an algorithm that combines TS with a SP approach. More recently, Brandão [3] proposed a deterministic TS that makes use of different procedures a for generating initial solutions.

The reader is referred to the recent survey performed by Baldacci et al. [1] for further details concerning the solutions approaches developed for the FSM.

3 The ILS-RVND-SP Algorithm

The proposed hybrid algorithm, called ILS-RVND-SP, is composed of an ILS [16] heuristic, that uses a procedure based on the Variable Neighborhood Descent [17] with Random neighborhood ordering (RVND) in the local search phase, and a SP formulation.

Let $\mathcal{R}$ be the set the routes, $\mathcal{R}_i \subseteq \mathcal{R}$ be the subset of routes that contain customer $i \in V'$, y_j be the variables associated to route $j \in \mathcal{R}$, and c_j bet the cost of each route $j \in \mathcal{R}$. The SP formulation for the FSM can be expressed as follows.

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

subject to

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

$$y_j \in \{0, 1\} \tag{3}$$

The objective functions (1) minimizes the sum of the costs by choosing the best combination of routes. Constraints (2) state that only one route from the subset $\mathcal{R}_i$ can visit costumer $i \in V'$. Constraints (3) define the domain of the decision variables.

The SP resolution tends to be harder for those cases where the fixed costs are considered. As an alternative, one can specify the vehicle fleet to be used by including additional constraints to the original formulation in order to facilitate the problem resolution. Of course, this will possibly imply in less effective solutions but it makes the problem computationally more tractable in an acceptable time. Let $\mathcal{R}_k \subseteq \mathcal{R}$ be the subset of routes associated to the vehicle $k \in M$ and m_k^* be the number of vehicles of type k presented in a given solution s^*. Constraints (4) impose that the solution of the SP model must have the same fleet composition of the solution s^*.

$$\sum_{j \in \mathcal{R}_k} y_j = m_k^* \qquad \forall k \in M \tag{4}$$

Alg. 1 describes how the ILS-RVND-SP algorithm operates. 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 in turn also updates the pool of routes (line 3). The variable *Cutoff* is initialized with the Upper Bound (UB)

value associated to the solution s^* (line 4). The SP model, given by expressions (1)-(3), is built using the pool of routes (line 5). The SP problem is solved using 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. In this case, 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), then the SP model is updated by adding constraints (4), $MaxRootGap$ is set to infinity and the solver is called again with the same stopping criteria.

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** $RootGap > MaxRootGap$) **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

3.1 The ILS-RVND heuristic

The ILS-RVND heuristic is based on the one developed by Subramanian et al. [28] for the VRP with Simultaneous Pickup and Delivery 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 $MaxIteriLS$ 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 the we have dealt with instances with up to 100 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). 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 the 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') {The call to this function is ignored in the IncumbentCallback}
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 operates as follows. Let the Candidate List (CL) be initially composed by all customers Each route is 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. 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, namely the Modified Cheapest Feasible Insertion Criterion (MCFIC) and the Nearest Feasible Insertion Criterion (NFIC) were adopted. The insertion criterion is randomly selected at each iteration.

The cost of inserting an unrouted customer $k \in$ CL in a given route using the MCFIC is expressed in Eq. 5, where function $g(k)$ represents the insertion cost. The value of $g(k)$ is computed by the sum of two parcels. The first computes the insertion cost of the client k between every pair of adjacent customers i and j while the second corresponds to a surcharge used to avoid late insertions of clients located far away from the depot. The cost back and forth from the depot is weighted by a factor γ, which is chosen from the interval $\{0.00, 0.05, \ldots, 1, 70\}$. This range was calibrated in [28].

$$g(k) = \left(c^u_{ik} + c^u_{kj} - c^u_{ij}\right) - \gamma\left(c^u_{0k} + c^u_{k0}\right) \tag{5}$$

The NFIC directly computes the distance between a customer $k \in$ CL and every customer i that has been already included into the partial solution, as can be observed by function g in Eq. 6. It is assumed that the insertion of k is always performed after i.

$$g(k) = c^u_{ik} \tag{6}$$

In both criteria, the insertion associated with the least-cost is taken into account, i.e. $\min\{g(k)|k \in \text{CL}\}$.

Two insertion strategies were employed, specifically the Sequential Insertion Strategy (SIS) and the Parallel Insertion Strategy (PIS).

The SIS works as follows. Firstly, one vehicle of each type is considered. While the CL is not empty and there is at least one customer $k \in$ CL that can be added to the current partial solution without violating any constraint, each route is filled with a customer selected using the correspondent insertion criterion. If the solution s is still incomplete then a new vehicle, chosen at random from the available types, is added.

The PIS differs from the SIS because all routes are considered while evaluating the least-cost insertion. While the CL is not empty and there is at least one customer $k \in$ CL that can be included in s, the insertions are evaluated using the selected insertion criterion and the customer associated with the least-cost insertion is then included in the correspondent route. The remain of the procedure is likewise the SIS.

3.1.2 Local Search

The local search is performed by a VND [17] procedure which utilizes a random neighborhood ordering (RVND). Let $N = \{N^{(1)}, \ldots, N^{(r)}\}$ be the set of neighborhood structures. Whenever a given neighborhood of the set N fails to improve the incumbent solution, the RVND randomly chooses another neighborhood from the same set to continue the search throughout the solution space. In this case, N is composed only by inter-route neighborhood structures.

The RVND is as follows. Firstly, a Neighborhood List (NL) containing e 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, 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. Five of them are based on the λ-interchanges scheme [22], which consists of exchanging up to λ customers between two routes. To limit the number of possibilities we have considered $\lambda = 2$. Another one is based on the Cross-exchange operator [29], which consists of exchanging two segments of different routes. Finally, a new neighborhood structure called, K-Shift, which consists of transferring a set of consecutive customers from a route to another one, was implemented. 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 when the same is leaving (or arriving on - it depends on the case) the depot.

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 linking 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 dependent 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.

3.1.4 Intra-Route Neighborhood structures

Five well-known intra-route neighborhood structures were adopted. The set N' is composed by Or-opt [21], 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. **Or-opt1**, 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 in sequence randomly. After some preliminary experiments, the number of succesive moves was empirically set to $0.5v$. **Multiple-Shift(1,1)**, $P^{(2)}$, multiple Shift(1,1) moves are performed in sequence 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 $\{0.5v, 0.6v, \ldots, 1.4v, 1.5v\}$. This perturbation is more "aggressive" than the previous one since it admits a larger number of moves. **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. There is a limit of $2 \times v$ trials to obtain a feasible solution when applying one of the above perturbations. In preliminary tests, $P^{(1)}$ and $P^{(2)}$ were always capable to produce feasible solutions using less than $2 \times v$ attempts. The same did not happen for $P^{(3)}$. Therefore, when the number of trials is achieved the algorithm randomly changes the perturbation mechanism to $P^{(1)}$ or $P^{(2)}$.

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 (5839 Mflops/s) with 8 GB of RAM memory running Ubuntu Linux 10.04 (kernel version 2.6.32). The SP formulation was implemented using the solver CPLEX 12.1. The developed approach we tested on well-known instances, namely those proposed by Golden et al. [9] and adapted by Taillard [30] and Choi and Tcha [4].

The following parameters values were selected after some preliminary experiments: $MaxIter = 30$, $MaxTime = 60$ seconds, $MaxRootGap = 2.00\%$. For all three FSM variants, each instance was executed 50 times and the results are presented in Subsection 4.1-4.3. 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 50 runs, **Gap** denotes the gap between the best solution found by the ILS-RVND-SP and the best known solution **Avg. gap/time** corresponds to the average gap between the best solution of the corresponding approach and the BKS followed by the average time, in seconds, of the best run. **Avg. scaled time** indicates the average scaled time of each

Table 1
Results for FSMFD instances

Inst.			**CG** [4]		**SMA-U1** [26]		**VNS1** [12]		**ILS-RVND-SP**					
#	n	BKS	Best Sol.	Time[1]	Best Sol.	Time[2]	Best Sol.	Time[3]	Best Sol.	Time	Gap	Sol.[b]	Time[b]	Gap[b]
3	20	1144.22[a]	**1144.22**	0.25	**1144.22**	0.01	**1144.22**	19.00	**1144.22**	0.17	0.00%	**1144.22**	0.30	0.00%
4	20	6437.33[a]	**6437.33**	0.45	**6437.33**	0.07	**6437.33**	17.00	**6437.33**	0.13	0.00%	**6437.33**	0.30	0.00%
5	20	1322.26[a]	**1322.26**	0.19	**1322.26**	0.02	**1322.26**	24.00	**1322.26**	0.23	0.00%	**1322.26**	0.30	0.00%
6	20	6516.47[a]	**6516.47**	0.41	**6516.47**	0.07	**6516.47**	21.00	**6516.47**	0.13	0.00%	**6516.47**	0.32	0.00%
13	50	2964.65[a]	**2964.65**	3.95	**2964.65**	0.32	**2964.65**	328.00	**2964.64**	1.71	0.00%	**2964.64**	1.93	0.00%
14	50	9126.90[a]	**9126.90**	51.70	**9126.90**	8.90	**9126.90**	250.00	**9126.90**	1.18	0.00%	**9126.90**	1.81	0.00%
15	50	2634.96[a]	**2634.96**	4.36	2635.21	1.04	**2634.96**	275.00	**2634.96**	1.35	0.00%	**2634.96**	1.61	0.00%
16	50	3168.92[a]	**3168.92**	5.98	3169.14	13.05	3168.95	313.00	**3168.92**	3.85	0.00%	**3168.92**	6.85	0.00%
17	75	2004.48[a]	2023.61	68.11	**2004.48**	23.92	**2004.48**	641.00	**2004.48**	4.05	0.00%	2006.73	7.21	0.11%
18	75	3147.99[a]	**3147.99**	18.78	3153.16	24.85	3153.67	835.00	**3147.99**	3.77	0.00%	3148.94	4.59	0.03%
19	100	8661.81[a]	8664.29	905.20	8664.67	163.25	8666.57	1411.00	**8661.81**	18.39	0.00%	8662.42	37.53	0.01%
20	100	4153.11[c]	4154.49	53.02	4154.49	41.25	4164.85	1460.00	**4153.02**	67.59	0.00%	4153.46	69.20	0.01%
Avg. gap/time			0.08%	92.70	0.02%	23.06	0.04%	466.17	0.00%	8.55		0.01%	11.00	
Avg. scaled time				35.98		6.18		117.92		8.55			11.00	

[a]: Optimality proved; [b]: Average of 50 runs (Solution, Time, Gap); [c]: First found by Prins [26]
[1]: Pentium IV 2.6GHz (2266 Mflop/s); [2]: Pentium IV M 1.8 GHz (1564 Mflop/s); [3]: Pentium IV M 1.7 GHz (1477 Mflop/s).

computer using the performances, in Mflop/s, of computers listed in Dongarra [6] 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 FSMFD

In Table 1 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 and Tcha [4], Prins [26] and Imran et al. [12]. The ILS-RVND-SP was found capable to improve 1 solution and to equal the result of the remaining ones. The average gap between the Avg. Sols. found by ILS-RVND-SP and the BKSs was 0.01%.

4.2 FSMF

Table 2 illustrates the results obtained by the ILS-RVND-SP for the FSMF. These results are compared with those of Brandão [3], Prins [26] and Liu et al. [15]. 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. The average gap between the Avg. Sols. found by ILS-RVND-SP and the BKSs was 0.10%.

4.3 FSMD

The best results obtained in the literature for the FSMD using heuristic approaches were reported by Choi and Tcha [4], Brandão [3] and Prins [26]. These results along with those found by the ILS-RVND-SP are presented in Table 3. In this variant the optimal solutions of all instances were proven in the literature. From Table 3 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 0.08%.

5 Concluding Remarks

This article dealt with Fleet Size and Mix (FSM) Vehicle Routing Problem. 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. Three FSM variants involving fixed and/or dependent costs were considered. These variants were solved by a hybrid algorithm based on the Iterated Local (ILS) Search

Table 2
Results for FSMF instances

Inst.			**TSA2** [3]		**SMA-D1** [26]		**GA** [15]		**ILS-RVND-SP**					
#	n	BKS	Best Sol.	Time[1]	Best Sol.	Time[2]	Best Sol.	Time[3,c]	Best Sol.	Time	Gap	Sol.[b]	Time[b]	Gap[b]
3	20	961.03[a]	**961.03**	21.00	**961.03**	0.04	**961.03**	0.00	**961.03**	0.21	0.00%	**961.03**	0.33	0.00%
4	20	6437.33[a]	**6437.33**	22.00	**6437.33**	0.03	**6437.33**	0.00	**6437.33**	0.17	0.00%	6438.05	0.28	0.01%
5	20	1007.05[a]	**1007.05**	20.00	**1007.05**	0.09	**1007.05**	2.00	**1007.05**	0.21	0.00%	1009.65	0.34	0.26%
6	20	6516.47[a]	**6516.47**	25.00	**6516.47**	0.08	**6516.47**	0.00	**6516.47**	0.16	0.00%	6516.81	0.28	0.01%
13	50	2406.36[a]	**2406.36**	145.00	**2406.36**	17.12	**2406.36**	91.00	**2406.36**	1.51	0.00%	2412.31	1.87	0.25%
14	50	9119.03[a]	**9119.03**	220.00	**9119.03**	19.66	**9119.03**	42.00	**9119.03**	1.54	0.00%	9119.06	1.98	0.00%
15	50	2586.37[a]	2586.84	110.00	**2586.37**	25.10	**2586.37**	48.00	**2586.37**	4.71	0.00%	**2586.37**	6.06	0.00%
16	50	2720.43[a]	2728.14	111.00	2729.08	16.37	2724.22	107.00	**2720.43**	2.07	0.00%	2723.79	4.15	0.12%
17	75	1734.53[a]	1736.09	322.00	1746.09	52.22	**1734.53**	109.00	**1734.53**	9.28	0.00%	1743.23	15.35	0.50%
18	75	2369.65[a]	2376.89	267.00	**2369.65**	36.92	**2369.65**	197.00	**2369.65**	6.25	0.00%	2371.97	12.47	0.10%
19	100	8661.81[a]	8667.26	438.00	8665.12	169.93	8662.94	778.00	**8661.81**	17.63	0.00%	8662.32	36.94	0.01%
20	100	4038.46	4048.09	601.00	4044.78	172.73	**4038.46**	1004.00	**4029.74**	59.77	-0.22%	**4038.22**	74.67	-0.01%
Avg. gap/time			0.03%	191.83	0.02%	42.52	0.01%	198.17	-0.02%	8.63		0.10%	12.63	
Avg. scaled time				39.95		11.39		107.96		8.63			12.63	

[a]: Optimality proved; [b]: Average of 50 runs (Solution, Time, Gap); [c]: Average time of 10 runs.
[1]: Pentium M 1.4 GHz (1216 Mflop/s); [2]: Pentium IV M 1.8 GHz (1564 Mflop/s) [3]: Pentium 4 3.0 GHz (3181 Mflop/s).

Table 3
Results for FSMD instances

Inst.			**CG** [4]		**TSA2** [3]		**SMA-U2** [26]		**ILS-RVND-SP**					
#	n	BKS	Best Sol.	Time[1]	Best Sol.	Time[2]	Best Sol.	Time[3]	Best Sol.	Time	Gap	Sol.[b]	Time[b]	Gap[b]
3	20	623.22[a]	**623.22**	0.21	–	–	–	–	**623.22**	0.15	0.00%	623.66	0.22	0.07%
4	20	387.18[a]	**387.18**	0.48	–	–	–	–	**387.18**	0.12	0.00%	387.55	0.18	0.10%
5	20	742.87[a]	**742.87**	0.32	–	–	–	–	**742.87**	0.18	0.00%	742.87	0.25	0.00%
6	20	415.03[a]	**415.03**	0.88	–	–	–	–	**415.03**	0.10	0.00%	415.18	0.15	0.04%
13	50	1491.86[a]	**1491.86**	3.98	**1491.86**	101.00	**1491.86**	3.45	**1491.86**	1.54	0.00%	1491.99	1.78	0.01%
14	50	603.21[a]	**603.21**	37.32	**603.21**	135.00	**603.21**	0.86	**603.21**	0.91	0.00%	603.32	1.19	0.02%
15	50	999.82[a]	**999.82**	5.96	**999.82**	137.00	**999.82**	9.14	**999.82**	1.06	0.00%	1000.53	1.65	0.07%
16	50	1131.00[a]	**1131.00**	6.41	**1131.00**	95.00	**1131.00**	13.00	**1131.00**	1.09	0.00%	1131.51	1.49	0.05%
17	75	1038.60[a]	**1038.60**	102.97	**1038.60**	312.00	**1038.60**	9.53	**1038.60**	3.53	0.00%	1039.44	5.68	0.08%
18	75	1800.80[a]	1801.40	80.96	**1800.80**	269.00	**1800.80**	18.92	**1800.80**	4.13	0.00%	1804.18	4.80	0.19%
19	100	1105.44[a]	**1105.44**	299.38	**1105.44**	839.00	**1105.44**	52.31	**1105.44**	7.20	0.00%	1105.89	8.89	0.04%
20	100	1530.43[a]	**1530.43**	112.00	**1530.43**	469.00	1533.29	104.41	**1530.43**	7.88	0.00%	1534.64	9.95	0.28%
Avg. gap/time			0.00%	54.24	0.00%	294.63	0.02%	26.45	0.00%	2.32		0.08%	3.02	
Avg. scaled time				21.05		61.36		7.09		2.32(3.42[c])			3.02(4.43[c])	

[a]: Optimality proved; [b]: Average of 50 runs (Solution, Time, Gap); [c]: Average Time in instances 13-20
[1]: Pentium IV 2.6GHz (2266 Mflop/s); [2]: Pentium M 1.4 GHz (1216 Mflop/s); [3]: Pentium IV M 1.8 GHz (1564 Mflop/s).

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 on 36 benchmark instances with up to 100 customers and it was found capable to obtain 2 new improved solutions and to equal the result of the remaining 34 instances. In addition, the average gap between the average solutions and best known solutions was only 0.06%. This fact clearly illustrates the robustness in terms of solution quality of the hybrid approach.

References

[1] R. Baldacci, M. Battarra, and D. Vigo. *The Vehicle Routing Problem: Latest Advances and New Challenges*, chapter Routing a Heterogeneous Fleet of Vehicles, pages 11–35. Springer, 2008.

[2] R. Baldacci and A. Mingozzi. A unified exact method for solving different classes of vehicle routing problems. *Math. Program.*, 120:347–380, 2009.

[3] J. Brandão. A deterministic tabu search algorithm for the fleet size and mix vehicle routing problem. *Eur. J. of Oper. Res.*, 195:716–728, 2009.

[4] E. Choi and D.-W. Tcha. A column generation approach to the heterogeneous fleet vehicle routing problem. *Comput. & Oper. Res.*, 34:2080–2095, 2007.

[5] G. Clarke and J. W. Wright. Scheduling of vehicles from a central depot to a number of delivery points. *Oper. Res.*, 12:568–581, 1964.

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

[7] M. Gendreau, G. Laporte, C. Musaraganyi, and E. D. Taillard. A tabu search heuristic for the heterogeneous fleet vehicle routing problem. *Comput. and Oper. Res.*, 26:1153–1173, 1999.

[8] F. Glover, M. Laguna, and R. Marti. *Handbook of Metaheuristics*, chapter Scatter Search and Path Relinking: Advances and Appl., pages 1–36. Kluwer Academic Publishers., 2003.

[9] B. L. Golden, A. A. Assad, L. Levy, and F. G. Gheysens. The feet size and mix vehicle routing problem. *Comput. & Oper. Res.*, 11:49–66, 1984.

[10] A. Hoff, H. Andersson, M. Christiansen, G. Hasle, and A. Løkketangen. Industrial aspects and literature survey: Fleet composition and routing. *Comput. & Oper. Res.*, 2010.

[11] J. H. Holland. *Adaptation in Natural and Artificial Systems.* University of Michigan Press, 1975.

[12] A. Imran, S. Salhi, and N. A. Wassan. A variable neighborhood-based heuristic for the heterogeneous fleet vehicle routing problem. *Eur. J. of Oper. Res.*, 197:509–518, 2009.

[13] Y.H. Lee, J.I. Kim, K.H. Kang, and K.H. Kim. A heuristic for vehicle fleet mix problem using tabu search and set partitioning. *J. of the Oper. Res. Soc.*, 59:833–841, 2008.

[14] C. M. R. R Lima, M. C. Goldbarg, and E. F. G. Goldbarg. A memetic algorithm for the heterogeneous fleet vehicle routing problem. *Electron. Notes in Discret. Math.*, 18:171–176, 2004.

[15] S. Liu, W. Huang, and H. Ma. An effective genetic algorithm for the fleet size and mix vehicle routing problems. *Transp. Res. Part E*, 45:434–445, 2009.

[16] H. R. Lourenço, O. C. Martin, and T Stützle. *Handbook of Metaheuristics*, chapter Iterated Local Search, pages 321–353. Kluwer Academic Publishers., 2003.

[17] N. Mladenovic and P. Hansen. Variable neighborhood search. *Comput. & Oper. Res.*, 24:1097–1100, 1997.

[18] P. Moscato and C. Cotta. *Handbook of Metaheuristics*, chapter A Gentle Introduction to Memetic Algorithm., pages 105–144. Kluwer Academic Publishers., 2003.

[19] L.S. Ochi, D.S. Vianna, L. M. A. Drummond, and A.O. Victor. An evolutionary hybrid metaheuristic for solving the vehicle routing problem with heterogeneous fleet. *Lect. notes in Comput. Sci.*, 1391:187–195, 1998.

[20] L.S. Ochi, D.S. Vianna, L. M. A. Drummond, and A.O. Victor. A parallel evolutionary algorithm for the vehicle routing problem with heterogeneous fleet. *Futur. Gener. Comput. Syst.*, 14:285–292, 1998.

[21] I. Or. *Traveling salesman-type combinational problems and their relation to the logistics of blood banking.* Phd thesis, Northwestern University, USA, 1976.

[22] I. H. Osman. Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem. *Ann. of Oper. Res.*, 41(1-4):421–451, 1993.

[23] A. Pessoa, E. Uchoa, and M. P. Aragão. *The Vehicle Routing Problem: Latest Advances and New Challenges*, chapter Robust Branch-and-Cut-and-Price Algorithm. for Vehicle Routing Problems, pages 297–325. Springer, 2008.

[24] A. Pessoa, E. Uchoa, and M. P. Aragão. A robust branch-cut-and-price algorithm for the heterogeneous fleet vehicle routing problem. *Netw.*, 54(4):167–177, 2009.

[25] C. Prins. A simple and effective evolutionary algorithm for the vehicle routing problem. *Comput. & Oper. Res.*, 31:1985–2002, 2004.

[26] C. Prins. Two memetic algorithms for heterogeneous fleet vehicle routing problems. *Eng. Appl. of Artif. Intell.*, 22(6):916–928, 2009.

[27] J. Renaud and F.F. Boctor. A sweep-based algorithm for the fleet size and mix vehicle routing problem. *Eur. J. of Oper. Res.*, 140:618–628, 2002.

[28] A. Subramanian, L.M.A. Drummond, C. Bentes, L.S. Ochi, and R. Farias. A parallel heuristic for the vehicle routing problem with simultaneous pickup and delivery. *Comput. & Oper. Res.*, 37(11):1899 – 1911, 2010.

[29] E. Taillard, P. Badeau, M. Gendreau, F. Guertin, and Potvin J. Y. A tabu search heuristic for the vehicle routing problem with soft time windows. *Transp. Sci.*, 31:170–186, 1997.

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

[31] H. Yaman. Formulations and valid inequalities for the heterogeneous vehicle routing problem. *Math. Program.*, 106:3650–390, 2006.