AN ITERATED LOCAL SEARCH HEURISTIC FOR OPEN VEHICLE ROUTING PROBLEMS

Puca Huachi Vaz Penna
Instituto do Noroeste Fluminense de Educação Superior - Universidade Federal Fluminense
Rua João Jazbik, s/no., Aeroporto, 28470-000, Santo Antônio de Pádua, RJ
Instituto de Computação - Universidade Federal Fluminense
Rua Passo da Pátria 156, Bloco E - 3ºandar, São Domingos, 24210-240, Niterói, RJ
ppenna@ic.uff.br
Anand Subramanian
Departamento de Engenharia de Produção - Universidade Federal da Paraíba
Centro de Tecnologia, Bloco G, Cidade Universitária, 58051-970, João Pessoa, PB
anand@ct.ufpb.br
Luiz Satoru Ochi
Instituto de Computação - Universidade Federal Fluminense
Rua Passo da Pátria 156, Bloco E - 3ºandar, São Domingos, 24210-240, Niterói, RJ
satoru@ic.uff.br

ABSTRACT

This paper deals with the Open Vehicle Routing Problem (OVRP) with homogeneous and heterogeneous fleet. The objective is to determine the set of routes that minimize the total costs. When the fleet is homogeneous, it is commonly assumed that the number of vehicles must be minimized. The proposed algorithm is based on the Iterated Local Search (ILS) metaheuristic which uses a Variable Neighborhood Descent procedure, with random neighborhood ordering (RVND), in the local search phase. The developed algorithm was tested in benchmark instances with up to 480 customers. The results obtained are quite competitive with those found in the literature.
KEYWORDS. Open Vehicle Routing Problem, Heterogeneous Fleet, Metaheuristic, Iterated Local Search.

Main areas: MH - Metaheuristics, CO - Combinatorial Optimization.

RESUMO

Este trabalho trata o Problema de Roteamento de Veículos Aberto (PRVA) com frota homogênea e heterogênea. O objetivo é determinar um conjunto de rotas que minimizem o custo total. Quando a frota é homogênga, geralmente assume-se que o número de veículos deve ser minimizado. O algoritmo proposto é baseado na metaheurística Iterated Local Search (ILS), que faz uso do método Variable Neighborhood Descent com uma ordem de vizinhança aleatória (RVND), na fase de busca local. O algoritmo desenvolvido foi testado em instâncias com até 480 clientes. Os resultados obtidos são bastante competitivos se comparados com os encontrados na literatura.
PALAVRAS-CHAVE. Problema de Roteamento de Veículos Aberto, Frota Heterogênea, Metaheurística, Iterated Local Search.

Áreas Principais: MH - Metaheurísticas, OC - Otimização Combinatória.

1 Introduction

The Vehicle Routing Problem (VRP) is one of the best known problems in the field of Operations Research. Inspired by real world applications, several variants were proposed over the years. In this work, our interest relies on the Open Vehicle Routing Problem (OVRP) and the Heterogeneous Fixed Fleet Open Vehicle Routing Problem (HFFOVRP). The OVRP is a special case of the Asymmetric Capacitated Vehicle Routing Problem (ACVRP) where the vehicles need not to return to the depot after visiting the last customer of a given route. Any OVRP instance can be converted to an ACVRP instance by simply setting $c_{i0} = 0, \forall i \in V$. Most authors also state that the number of vehicles must be minimized. The HFFOVRP generalizes the OVRP by allowing vehicles with different capacities, instead of a homogeneous fleet.

Applications of the OVRPs may arise when a company chooses to hire a vehicle fleet to be in charge of the delivery services and, due to logistic reasons, these vehicles are not forced to return to the company's depot. In this case, the distribution costs are generally proportional to the lenght of the routes and/or to the number of vehicles used by the outsourced company.

Schrage (1981) was the first to address the problem by describing certain characteristics of some real-life VRPs. One example mentioned by the author is the air express courier, in which aircrafts depart from a depot city, deliver their cargo to a set of customers geographically spread and then collect the cargo from the same set of customers by retracing their routes back to the depot. Bodin *et al.* (1983) have presented a case study of this type of application at the FedEx Express company. Besides capacity constraints, other restrictions such as time windows and routes duration were also considered.

The objective of this work is to present a heuristic algorithm, based on the Iterated Local Search (ILS) metaheuristic and on the Randomized Variable Neighborhood Descent (Subramanian *et al.*, 2010). The proposed solution approach is an extension of the one proposed by Penna *et al.* (2011) for the Heterogeneous Fleet Vehicle Routing Problem (HFVRP).

The remainder of this paper is organized as follows. Section 2 reviews some works related to the OVRP and HFFOVRP. Section 3 explains the proposed hybrid heuristic. 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

After the works of Schrage (1981) and Bodin *et al.* (1983), the OVRP literature remained practically unchanged for nearly two decades until it was revisited by Sariklis e Powell (2000). The authors proposed a cluster-first, route second approach (Bodin e Golden, 1981) where the first phase consists in grouping the customers according to the capacity constraints, while the second phase consists of a Minimum Spanning Tree (MST) heuristic that incorporates a penalty procedure.

Letchford *et al.* (2007) presented an Integer Linear Programming formulation, a set of valid inequalities, as well as a Branch-and-Cut algorithm that is mainly based on the one described in Lysgaard *et al.* (2004). Their procedure is capable of solving to optimality several small and medium-sized instances. This work, along with the one of Pessoa *et al.* (2008), are to date the only exact approaches that dealt with the OVRP.

A considerable number of OVRP heuristic algorithms have been published since 2004. Some of these are based on the Tabu Search (TS) metaheuristic. Brandão (2004) proposed a TS heuristic that makes use of a nearest neighborhood heuristic and a K-tree based procedure for generating initial solutions, whereas the local search is performed by shift and swap moves. Tarantilis *et al.* (2004b) suggested a heuristic that interactively combines the TS and Adaptive Memory Procedure (AMP) methods. Fu *et al.* (2005, 2006) developed a TS algorithm that employs a farthest-first heuristic for constructing an initial solution while shift, swap and 2-opt moves are used in the local search phase. Derigs e Reuter (2009) proposed a Attribute Based Hill Climber procedure which is similar to the one presented in Derigs e Kaiser (2007) for the CVRP.

OVRP algorithms based on other local search metaheuristics were also proposed. Tarantilis *et al*. (2004a) presented a threshold accepting approach (Dueck e Scheuer, 1990) that consists of an adaptation of the Simulated Annealing (SA) procedure in which a worse solution is only accepted if it is within a given threshold. The same authors (Tarantilis *et al*., 2005) also proposed another threshold accepting procedure that is integrated in a single-parameter metaheuristic. Li *et al*. (2007b) put forward a record-to-record (Dueck, 1993) travel algorithm that, also as the threshold method, consists of a deterministic variant of the SA. Fleszar *et al*. (2009) presented a VNS heuristic whose neighborhood operators are composed by exchanging segments between two routes and reversing segments of a single route. Zachariadis e Kiranoudis (2010) developed a local search metaheuristic that explores wide neighborhoods by only evaluating parts of a current solution that have been modified by a previous move.

Differently from pure local search approches, there are few works containing applications of Evolutionary Strategies (ES) to the OVRP. Li e Tian (2006) presented an Ant Colony (AC) algorithm combined with local search followed by a post-optimization procedure applied to the best solution obtained. A similar approach was later developed by Li *et al*. (2009) where a TS procedure is incorporated into the AC framework. Repoussis *et al*. (2010) suggested a heuristic based on ES in which offspring individuals (solutions) are generated through mutation operators and these intermediate solutions are improved by a procedure based on Guided Local Search (GLS) and TS.

To our knowledge, the HFFOVRP was proposed by Li *et al*. (2012). According to the authors, the HFFOVRP is more realistic in real situations than OVRP. To solve the problem they developed an algorithm based on multi-start AMP with a modified TS used as an improvement procedure. In the modified TS procedure infeasible solutions, that violate the vehicle capacity constraint, are allowed.

As mentioned before, most works on OVRP has aimed at minimizing the number of vehicles. However, this objective was not considered in the HFFOVRP proposed in Li *et al*. (2012).

3 The MILS-RVND Algorithm

This section describes the proposed heuristic algorithm, called MILS-RVND (see Alg. 1). Let v be the number of vehicles (or routes), where its value is defined on line 3. The multi-start heuristic executes $MaxIter$ iterations (lines 5-22), where at each iteration a solution is built using a constructive procedure (line 6). The ILS procedure (lines 9-17) aims at improving this initial solution by means of a combination between local search (RVND, line 10) and perturbation (line 15). With respect to the acceptance criterion, it can be observed that algorithm only perturbs the best current solution (s') of a particular iteration. The maximum number of consecutive perturbations allowed without improvements is denote by the parameter $MaxIterILS$.

The next subsections provide a detailed explanation of the main components of the MILS-RVND heuristic.

3.1 Estimating the Number of Vehicles

For the OVRP, use is made of a lower bound on the number of vehicles (v_{min}) which is computed dividing the sum of the customers demands by the capacity of the vehicle. As for the HFFOVRP, the maximum number of vehicles of each type is initially considered.

3.2 Constructive Procedure

The initial solutions are built using the following constructive procedure. Firstly, a random seed customer k from a Candidate List (CL) is randomly selected to be inserted in a route. This step is repeated until $v-1$ vehicles are filled with a single customer. Next, the algorithm generates an initial solution by randomly

Algorithm 1 MILS-RVND

```
Procedure MILS-RVND(MaxIter, MaxIterILS, v)
LoadData( );
v ← EstimateTheNumberOfVehicles();
f* ← ∞;
for i ← 1, . . . , MaxIter do
    s ← GenerateInitialSolution(v, MaxIter, seed);
    s' ← s;
    iterILS ← 0;
    while iterILS ≤ MaxIterILS do
        s ← RVND(s);
        if f(s) < f(s') {or v of s < v of s' (considered only when v must be minimized)} then
            s' ← s;
            iterILS ← 0;
        end if
        s ← Perturb(s', seed);
        iterILS ← iterILS + 1;
    end while
    if f(s') < f* then
        s* ← s';
        f* ← f(s');
    end if
end for
return s*;
end MILS-RVND.
```

selecting an insertion criterion and an insertion strategy. Two insertion criteria were considered: 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 taking into account an insertion incentive for those customers located far from the depot. Two insertion strategies were adopted, namely 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. In PIS, all routes are considered while evaluating the least-cost insertion.

3.3 Local Search

The local search is performed using a RVND procedure whose description can be found in Alg. 2. Firstly, a Neighborhood List (NL) is initialized with a set of inter-route neighborhood structures (line 3). In the main loop (lines 4-13), a neighborhood $N^{(\eta)} \in$ NL is randomly selected (line 5) and then the best admissible move is determined (line 6). In case of improvement, the algorithm performs an intra-route local search in the modifed routes and NL is populated with all the neighborhoods (lines 7-10). Otherwise, $N^{(\eta)}$ is removed from the NL (line 12). A set of Auxiliary Data Structures (ADSs) is updated (see Penna *et al.*, 2011) at the beginning of the procedure (line 2) and whenever a neighborhood search is performed (line 13). Finally, a procedure that tries to empty a route is applied (line 14).

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

Algorithm 2 RVND

Procedure RVND(s)
Update ADSs;
Initialize the inter-route Neighborhood List (NL);
while NL $\neq 0$ **do**
  Choose a neighborhood $N^{(\eta)} \in$ NL at random;
  Find the best neighbor s' of $s \in N^{(\eta)}$;
  if $f(s') < f(s)$ **then**
    $s \leftarrow s'$;
    $s \leftarrow$ IntraRouteSearch(s);
    Update NL; {NL in populated with all inter-route neighborhood structures}
  else
    Remove $N^{(\eta)}$ from the NL;
  end if
  Update ADSs;
end while
TryToEmptyRoute(s); {considered only on the Homogeneous fleet OVRP}
return s;
end RVND.

3.3.1 Inter-Route Neighborhood structures

The following six inter-route neighborhood structures were considered in the local search. **Shift(1,0)**, a customer k is moved 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)**, an arc (k, l) is transferred from a route r_1 to a route r_2. The move also examines the transferring of the arc (l, k). **Swap(2,1)**, permutation of an arc $(k, l$ from a route r_1 by a customer k' from a route r_2. As in Shift(2,1), arc (l, k) is also considered. **Swap(2,2)**, permutation between two an arc (k, l), from a route r_1 by another one (k', 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.

It is important to emphasize that all possible combinations of the moves mentioned above are examined but only feasible moves are considered.

3.3.2 Intra-Route Neighborhood structures

Five 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.

3.3.3 Trying to Empty a Route

As stated by most authors, minimizing the number of vehicles is the primary goal in the OVRP. Hence a greedy randomized procedure was developed for dealing with this issue, as can be observed in Alg. 3. The idea is to make use of the residual capacity and residual duration of the routes of a given solution s by means of local search, with a view of decreasing the number of routes of s. The procedure starts by storing a backup of the solution s in s' (line 2). Let Route List (RL) be the list composed by the routes of s (line 3). While |RL| is greater than 1 (lines 4-11), an attempt to empty a route is performed. A route r is selected to be removed from RL (lines 6-7) according to one of the following criteria: (i) route with maximum load; (ii) route with maximum duration;

(iii) random selection. The route selection criterion is chosen at random (line 5). Next, while it is still possible to move a customer from any route $r' \in$ RL to r or it is still possible to exchange a customer from any route $r' \in$ RL with another one in r in such a way that the load of r is increased, a local search is performed between the route r and those in RL by the neighborhood structures Shift(1,0), Shift(2,0) and Swap(1,1) (lines 9-11). The best admissible move is considered for each of these three neighborhoods. Moreover, in the case of Shift(1,0) and Shift(2,0), a move is immediately accepted if a route $r' \in$ RL becomes empty, whereas in the case of Swap(1,1), a move is only accepted if the vehicle load of r is increased. An intra-route local search is performed in every modified route using 2-opt and exchange neighborhood structures. If the procedure is not capable of emptying a route then the current solution is restored (lines 12-13).

Algorithm 3 TryEmptyRoute

```
1:  Procedure TryEmptyRoute(s)
2:  s' ← s
3:  Initialize Route List (RL) with the routes of s
4:  while |RL| > 1 do
5:      Choose a route selection criterion at random;
6:      Choose a route r ∈ RL according to the selected criterion;
7:      Remove r from RL;
8:      while it is still possible to move a customer from any route r' ∈ RL to r or it is still possible to exchange a customer from any route
        r' ∈ RL with another one in r in such a way that the load of r is increased do
9:          s ← Shift(1,0);
10:         s ← Shift(2,0);
11:         s ← Swap(1,1);
12:     end while
13: end while
14: if number of routes of s is equal to the number of routes of s' then
15:     s ← s';
16: end if
17: return s;
18: end TryEmptyRoute.
```

3.4 Perturbation Mechanisms

A set P of two perturbation mechanisms were considered. Whenever the `Perturb()` function is called, one of the following moves is randomly selected. Only feasible perturbation moves are accepted.

Multiple-Swap(1,1) – $P^{(1)}$: Multiple random Swap(1,1) moves are performed in sequence.
Multiple-Shift(1,1) – $P^{(2)}$: Multiple random Shift(1,1) moves are performed in sequence. 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.

4 Computational Results

The algorithm MILS-RVND was coded in C++ (g++ 4.4.3) and, for the OVRP, the tests were executed in an Intel® Core^{TM} 2 Quad with 2.4 GHz and 4 GB of RAM running under Linux 64 bits (kernel 2.6.27-16). As for the HFFOVRP the tests were executed in an Intel® Core^{TM} i7 with 2.93 GHz and 8 GB of RAM running under Linux 64 bits (kernel 2.6.32-22). Only a single thread was used in the experiments.

In the tables presented hereafter, **Instance** denotes the number of the test-problem, ***n*** is the number of customers, **BKS** represents the best known solution reported in the literature, **Best Sol.**, **Avg. Sol.** and **Time(s)** indicate, respectively, the best solution, the average solution and the average computational time in seconds associated to the correspondent work, **Gap** denotes the gap between the best solution found by MILS-RVND and the best known solution, **Avg. Gap** corresponds to the gap between the average solution found by MILS-RVND and the best known solution. The best solutions are highlighted in boldface and the solutions improved by MILS-RVND are underlined. The approximate speed, in Mflop/s, of the machines used by other authors is also reported considering the factors suggested by the benchmarks of Dongarra (2010), when solving solving a system of equations of order 1000.

4.1 Parameter Tuning

It has been empirically observed that the suitable values of $MaxIterILS$ depends on the size of the instances, more precisely, on the number of customers and vehicles. For the sake of simplicity, we have chosen to use a simple linear expression for computing the value of $MaxIterILS$ according to n and v, as shown in Eq. 1.

$$MaxIterILS = n + \beta \times v \tag{1}$$

The parameter β in Eq. 1 corresponds to a non-negative integer constant that indicates the level of influence of the number of vehicles v in the value of $MaxIterILS$.

After some preliminary tests we decided to adopt the following values: $MaxIter = 50$ and $\beta = 5$.

4.2 OVRP

To examine the behavior of the MILS-RVND algorithm when applied to solve the OVRP, use was made of two well-know benchmark datasets from the literature. The first set contains the instances of Christofides *et al*. (1979) without and with route durations constraints (see Brandão, 2004, for more details), as well as another two generated by Fisher (1994) were also considered. The second set corresponds to the large-sized instances suggested by Li *et al*. (2007b) involving 200-480 customers.

Table 1 presents the results found by MILS-RVND in the first set of instances and a comparison with those pointed out by Pisinger e Røpke (2007) (ALNS 50K), Fleszar *et al*. (2009), Repoussis *et al*. (2010) and Zachariadis e Kiranoudis (2010). Regarding those of Christofides *et al*. (1979) and Fisher (1994), MILS-RVND was capable to obtain the BKS in 11 cases and to improve another 2 solutions, but it failed to find 3 BKSs. Furthermore, MILS-RVND also failed to always obtain solutions with the minimum number of vehicles on instances C7 and C9. Although the MILS-RVND was capable of producing competitive results, its performance with respect to the minimization of the number of vehicles was slightly worse when compared to the other algorithms, namely on those instances that include route duration constraints. The average gap between the Avg. Sols. obtained by MILS-RVND and the BKSs, disregarding those two instances where the average number of vehicles found by the proposed algorithm was larger than those associated to the BKSs, was 0.62%.

Table 2 shows the results obtained in the second set of instances. It can be observed that MILS-RVND improved 7 results and the average gap between the Avg. Sols produced by MILS-RVND and the BKSs was 0.19%. Also, the developed algorithm was successful to generate feasible solutions using v_{min} vehicles in all instances of this group.

Tabela 1. Results found for the instances of Christofides *et al*. (1979) and Fisher (1994)

Instance	n	v_{min}	BKS	*vbest*	Pisinger and Røpke (2007)			Fleszar *et al.* (2009)			Repoussis *et al.* (2010)			Zachariadis and Kiranoudis (2010)			MILS-RVND					
					Best Sol.	v	Time[1] (s)	Best Sol.	v	Time[2] (s)	Best Sol.	v	Time[3] (s)	Best Sol.	v	Time[4] (s)	Best Sol.	v	Avg. Sol.	Gap (%)	Avg. Gap (%)	Time (s)
C1	50	5	^{a}**416.06**	5	**416.06**	5	23	**416.06**	5	1.0	**416.06**	5	98	**416.06**	5	25	**416.06**	5	416.09	0.00	0.01	2.62
F11	71	4	^{a}**177.00**	4	**177.00**	4	104	178.66	4	6.2	**177.00**	4	264	**177.00**	4	93	**177.00**	4	177.12	0.00	0.07	6.30
C2	75	10	^{a}**567.14**	10	**567.14**	10	53	**567.14**	10	2.3	**567.14**	10	143	**567.14**	10	68	**567.14**	10	568.54	0.00	0.25	6.55
C3	100	8	^{a}**639.74**	8	641.76	8	128	641.40	8	9.5	**639.74**	8	330	**639.74**	8	103	**639.74**	8	640.49	0.00	0.12	13.98
C12	100	10	^{a}**534.24**	10	**534.24**	10	118	534.40	10	6.7	**534.24**	10	363	**534.24**	10	39	**534.24**	10	**534.24**	0.00	0.00	8.85
C11	120	7	**682.12**	7	**682.12**	7	141	**682.12**	7	10.7	**682.12**	7	318	**682.12**	7	85	**682.12**	7	682.21	0.00	0.01	27.79
F12	134	7	**769.55**	7	770.17	7	359	769.66	7	75.4	**769.55**	7	753	**769.55**	7	30	**769.55**	7	770.69	0.00	0.15	41.18
C4	150	12	**733.13**	12	733.13	12	279	737.82	12	45.4	**733.13**	12	613	**733.13**	12	190	**733.13**	12	733.43	0.00	0.04	39.71
C5	199	16	**893.39**	16	896.08	16	237	905.96	16	17.1	894.11	16	1272	**893.39**	16	355	898.08	16	927.31	0.52	3.80	101.86
C6	50	5	**412.96**	6	**412.96**	6	31	**412.96**	6	75.8	**412.96**	6	215	-	-	-	**412.96**	6	**412.96**	0.00	0.00	1.64
C7	75	10	**583.19**	10	**583.19**	10	33	596.47	10	22.3	584.15	10	367	-	-	-	584.15	10	588.07*	0.16		10.48
C8	100	8	**644.63**	9	645.16	9	114	**644.63**	9	587.6	**644.63**	9	510	-	-	-	**644.63**	9	645.67	0.00	0.16	9.48
C14	100	10	**591.87**	11	**591.87**	11	75	**591.87**	11	389	**591.87**	11	411	-	-	-	**591.87**	11	**591.87**	0.00	0.00	9.76
C13	120	7	**904.04**	11	909.80	11	116	**904.94**	11	1820.1	910.26	11	890	-	-	-	**899.16**	11	928.00	-0.54	2.65	19.50
C9	150	12	**757.84**	13	**757.84**	13	185	760.06	13	1094.1	764.56	13	933	-	-	-	759.36	13	759.27*	0.20		33.57
C10	199	16	**875.67**	17	**875.67**	17	224	**875.67**	17	1252.4	888.46	17	1678	-	-	-	**875.24**	17	887.92	-0.05	1.40	58.64
Average																				0.02	0.62	24.50

^{a}: Optimality proved.
^{b}: Average of 10 runs considering the following instances: C1, F11, C2, C3, C12, C11, F12, C4 and C5.
[1]: Average of 10 runs on a Pentium IV 3.0 GHz (3181 Mflop/s).
[2]: Best run on a Pentium M 2.0 GHz (1738 Mflop/s).
[3]: Best run on a Scaled to a Pentium II 400 MHz (262 Mflop/s).
[4]: Average of 10 runs on a T5500 1.66 GHz (2791 Mflop/s).

Tabela 2. Results found for the instances of Li *et al*. (2007a)

Instance	n	v_{min}	BKS	*vbest*	Repoussis *et al*. (2010)			Zachariadis and Kiranoudis (2010)			MILS-RVND					
					Best Sol.	*v*	Time[1] (s)	Best Sol.	*v*	Time[2] (s)	Best Sol.	*v*	Avg. Sol.	Gap (%)	Avg. Gap (%)	Time (s)
O1	200	5	**6018.52**	5	**6018.52**	5	452	**6018.52**	5	612	**6018.52**	5	**6018.52**	0.00	0.00	160.27
O2	240	9	**4557.38**	9	4583.7	9	613	**4557.38**	9	774	**4547.67**	9	4567.72	-0.21	0.23	172.18
O3	280	7	**7731.00**	7	7733.77	7	736	**7731.00**	7	681	**7721.16**	7	7730.30	-0.13	-0.01	335.04
O4	320	10	**7253.20**	10	7271.24	10	833	**7253.20**	10	957	**7220.19**	10	7248.29	-0.46	-0.07	490.66
O5	360	8	**9193.15**	8	9254.15	8	1365	**9193.15**	8	1491	9225.78	8	9289.39	0.35	1.05	996.76
O6	400	9	**9793.72**	9	9821.09	9	1213	**9793.72**	9	1070	**9771.31**	9	9809.87	-0.23	0.16	1207.13
O7	440	10	**10347.70**	10	10363.4	10	1547	**10347.70**	10	1257	**10325.70**	10	10365.80	-0.21	0.17	1567.80
O8	480	10	**12415.36**	10	12428.2	10	1653	**12415.36**	10	1512	**12389.40**	10	12412.60	-0.21	-0.02	1816.19
Average														-0.14	0.19	843.25

[1]: Best run on a Scaled to a Pentium II 400 MHz (262 Mflop/s).
[2]: Average of 10 runs on a T5500 1.66 GHz (2791 Mflop/s).

4.3 HFFOVRP

In the case of the HFFOVRP, a comparison performed between MILS-RVND and the MAMP algorithm proposed by Li *et al*. (2012) was not possible, because the authors have not provided their instances. Therefore, we decided to adapt the well-known instances prosed by Taillard (1999) for the HFFVRP. This set is composed by 8 instances up to 100 customers with fixed and variable costs. The results found by MILS-RVND for the HFFOVRP were presented in Table 3

Tabela 3. Results found for the HFFOVRP

Instance	n	t	*v*	MILS-RVND				
				Best Sol.	*v*	Avg. Sol.	Avg. Gap (%)	Time (s)
13	50	6	17	**2588.65**	16	2591.04	0.09	4.59
14	50	3	7	**9961.81**	6	9966.91	0.05	4.23
15	50	3	9	**2731.46**	9	2731.63	0.01	4.16
16	50	3	9	**2929.78**	8	2958.32	0.97	4.16
17	75	4	11	**1792.20**	10	1798.55	0.35	12.47
18	75	6	14	**3228.14**	12	3235.93	0.24	12.95
19	100	3	10	**10179.70**	8	10187.99	0.08	39.51
20	100	3	13	**4344.55**	13	4349.33	0.11	29.17
Average							0.24	13.91

5 Concluding Remarks

This paper dealt with Open Vehicle Routing Problem (OVRP) and the Heterogeneous Fixed Fleet Open Vehicle Routing Problem (HFFOVRP). These problems often arises in distribution management and transportation. Both variants were solved by a multi-start algorithm based on the Iterated Local Search (ILS) metaheuristic, that uses a Variable Neighborhood Descent procedure, with random neighborhood ordering (RVND) in the local search phase.

The proposed algorithm (MILS-RVND) was tested on 24 benchmark instances with up to 480 customers for the OVRP and it was found capable to obtain 9 new improved solutions, to equal the result of 11 instances and failed to obtain the best known solution of only 4 instances. Finally, we proposed 8 new instances for the HFFOVRP which were adapted from well-known instances available in the literature. The average gap between the Avg. Sols. and the Best Sol. obtained by MILS-RVND was only 0.24%.

Referências

Bodin, L. e Golden, B. (1981), Classification in vehicle routing and scheduling. *Networks*, v. 11, p. 97–108.

Bodin, L., Golden, B., Assad, A. e Ball, M. (1983), Routing and scheduling of vehicles and crews - the state of the art. *Computers & Opertions Research*, v. 10, n. 2, p. 63–212.

Brandão, J. (2004), A tabu search algorithm for the open vehicle routing problem. *European Journal of Operational Research*, v. 157, n. 3, p. 552–564.

Christofides, N., Mingozzi, A. e Toth, P. *Combinatorial Optimization*, Capítulo The Vehicle Routing Problem, p. 315–338. Wiley, Chinchester, UK, 1979.

Derigs, U. e Reuter, K. (2009), A simple and efficient tabu search heuristic for solving the open vehicle routing problem. *Journal of the Operational Research Society*, v. 60, p. 1658–1669.

Derigs, U. e Kaiser, R. (2007), Applying the attribute based hill climber heuristic to the vehicle routing problem. *European Journal of Operational Research*, v. 177, n. 2, p. 719–732.

Dongarra, J. J. Performance of various computers using standard linear equations software. Relatório Técnico CS-89-85, Computer Science Department, University of Tennessee, 2010.

Dueck, G. (1993), New optimization heuristics: The great deluge algorithm and the record-to-record travel. *Journal of Computational Physics*, v. 104, n. 1, p. 86–92.

Dueck, G. e Scheuer, T. (1990), Threshold accepting: a general purpose optimization algorithm appearing superior to simulated annealing. *Journal of Computational Physics*, v. 90, n. 1, p. 161–175.

Fisher, M. (1994), Optimal solutions of vehicle routing problems using minimum K-trees. *Operations Research*, v. 42, p. 626–642.

Fleszar, K., Osman, I. H. e Hindi, K. S. (2009), A variable neighbourhood search algorithm for the open vehicle routing problem. *European Journal of Operational Research*, v. 195, n. 3, p. 803–809.

Fu, Z., Eglese, R. e Li, L. Y. O. (2005), A new tabu search heuristic for the open vehicle routing problem. *Journal of the Operational Research Society*, v. 56, p. 267–274.

Fu, Z., Eglese, R. e Li, L. Y. O. (2006), A new tabu search heuristic for the open vehicle routing problem. *Journal of the Operational Research Society*, v. 57. Corrigendum.

Letchford, A., Lysgaard, J. e Eglese, R. W. (2007), A branch-and-cut algorithm for the capacitated open vehicle routing problem. *Journal of the Operational Research Societ*, v. 58, n. 12, p. 1642–1651.

Li, F., Golden, B. e Wasil, E. (2007a), A record-to-record travel algorithm for solving the heterogeneous fleet vehicle routing problem. *Computers and Operations Research*, v. 34, p. 2734–2742.

Li, F., Golden, B. e Wasil, E. (2007b), The open vehicle routing problem: Algorithms, large-scale test problems, and computational results. *Computers & Opertions Research*, v. 34, n. 10, p. 2918–2930.

Li, X.-Y. e Tian, P. An ant colony system for the open vehicle routing problem. *Proceedings of the 5th International Workshop on Ant Colony Optimization and Swarm Intelligence, ANTS 2006. Lecture Notes in Computer Science*, volume 4150, p. 356–363. Springer-Verlag, 2006.

Li, X.-Y., Tian, P. e Leung, S. C. H. (2009), An ant colony optimization metaheuristic hybridized with tabu search for open vehicle routing problems. *Journal of the Operational Research Society*, v. 60, p. 1012–1025.

Li, X., Leung, S. C. e Tian, P. (2012), A multistart adaptive memory-based tabu search algorithm for the heterogeneous fixed fleet open vehicle routing problem. *Expert Systems with Applications*, v. 39, p. 365–374.

Lysgaard, J., Letchford, A. N. e Eglese, R. W. (2004), A new branch-and-cut algorithm for the capacitated vehicle routing problem. *Mathematical Programming*, v. 100, p. 423–445.

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

Pessoa, A., de Aragão, M. P. e Uchoa, E. *The Vehicle Routing Problem: Latest Advances and New Challenges*, Capítulo Robust Branch-and-Cut-and-Price Algorithm. for Vehicle Routing Problems, p. 297–325. Springer, 2008.

Pisinger, D. e Røpke, S. (2007), A general heuristic for vehicle routing problems. *Computers & Operations Research*, v. 34, n. 8, p. 2403–2435.

Repoussis, P. P., Tarantilis, C. D., Bräysy, O. e Ioannou, G. (2010), A hybrid evolution strategy for the open vehicle routing problem. *Computers & Opertions Research*, v. 37, n. 3, p. 443–455.

Sariklis, D. e Powell, S. (2000), A heuristic method for the open vehicle routing problem. *Journal of the Operational Research Society*, v. 51, n. 5, p. 564–573.

Schrage, L. (1981), A generalized assignment heuristic for vehicle routing. *Networks*, v. 11, p. 229–232.

Subramanian, A., Drummond, L., Bentes, C., Ochi, L. e Farias, R. (2010), A parallel heuristic for the vehicle routing problem with simultaneous pickup and delivery. *Computers & Operations Research*, v. 37, n. 11, p. 1899 – 1911.

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

Tarantilis, C., Ioannou, G., Kiranoudis, C. e Prastacos, G. (2004a), A threshold accepting approach to the open vehicle routing problem. *RAIRO (Recherche Opérationnelle)*, v. 38, p. 345–360.

Tarantilis, C., Ioannou, G., Kiranoudis, C. e Prastacos, G. (2005), Solving the open vehicle routeing problem via a single parameter metaheuristic algorithm. *Journal of the Operational Research Society*, v. 56, p. 588–596.

Tarantilis, C. D., Diakoulaki, D. e Kiranoudis, C. T. (2004b), Combination of geographical information system and efficient routing algorithms for real life distribution operations. *European Journal of Operational Research*, v. 2, n. 152, p. 437–453.

Zachariadis, E. E. e Kiranoudis, C. T. (2010), An open vehicle routing problem metaheuristic for examining wide solution neighborhoods. *Computers & Opertions Research*, v. 37, n. 4, p. 712–723.