A Memetic Algorithm for the Car Renter Salesman Problem

Marco Goldbarg[1], Paulo Asconavieta[2] and Elizabeth Goldbarg[1]
[1]*Universidade Federal do Rio Grande do Norte*
[2]*Instituto Federal de Educação, Ciência e Tecnologia Sul-rio-grandense*
Brazil

1. Introduction

The Traveling Salesman Problem (TSP) is a classic Combinatorial Optimization problem. Given a graph $G=(N,M)$, where $N=\{1,...,n\}$ is the set of nodes and $M=\{1,...,m\}$ is the set of edges, and costs, c_{ij}, associated with each edge connecting vertices i and j, the problem consists in finding the minimum length Hamiltonian cycle. The TSP is NP-hard (Garey & Johnson, 1979) and one of the combinatorial optimization problems more intensively investigated. The size of the larger non trivial TSP instance solved by an exact method evolved from 318 cities in the 80's (Crowder & Padberg, 1980), to 7397 cities in the 90's (Applegate *et al.*, 1994) and 24978 cities in 2004. The best mark was reached in 2006 with the solution of an instance with 85900 cities (Applegate *et al.*, 2006). The TSP has several important practical applications and a number of variants (Gutin & Punnen, 2002). Some of these variants are classic such as the Peripatetic Salesman (Krarup, 1975) and the M-tour TSP (Russel, 1977), other variants are more recent such as the Colorful TSP (Xiong *et al.*, 2007) and the Robust TSP (Montemanni *et al.*, 2007), among others.
A new TSP variant is introduced in this chapter named The Car Renter Salesman Problem (CaRS). It models important applications in tourism and transportation areas and represents a complex variant that challenges the state of the art. In this paper the new problem and some variations are presented, its complexity is analyzed and some related problems are briefly overviewed. A memetic algorithm is proposed for the problem and it is compared to a hybrid GRASP/VND algorithm.
CaRS Problem is introduced in Section 2, where several conditions under which this variant can be presented are introduced. Section 3 presents two metaheuristic methods for the investigated problem. In order to compare the performance of the proposed approaches, a set of instances introduced for the new problem, named CaRSLib. This set contains Euclidean and non-Euclidean symmetric instances with number of cities ranging from 14 to 300 and number of cars between 2 and 5. A set of 40 instances is used in the computational experiments. The heuristics proposed in Section 3 establish the first upper limits for solutions of CaRSLib of instances. The results of computational experiments comparing the performance of the proposed approaches are presented in Section 4. Statistical tests are applied to support conclusions on the behavior of the proposed algorithms. According to

those results, the Memetic Algorithm is pointed as the best approach for CaRS considering the tested cases. Final conclusions and future directions for the research of algorithms for CaRS and its variations are addressed in Section 5.

2. The car renter salesman problem

2.1 The rental car industry

Today over 90 significant economic size car rental companies exist in the world market (Car, 2008). The importance of the car rental business can be measured both by the enterprise turnover as by the size of the companies that provide the service. For example, Hertz is a company in the car rental segment with wide accessibility of providing the services at approximately 8,000 locations in approximately 145 countries (Hertz, 2009). The Enterprise has more than 878,000 vehicles in its rental and leasing fleet and operates across 6,900 local markets (Enterprise, 2009). Avis operates in more than 3,800 locations all over Europe, Africa, the Middle East and Asia. In December 2007, the company operated an average fleet of 118,000 vehicles (Avis, 2009). Avis Budget Group Inc. earned $ 5.1 billion dollars in 2009 (Avis, 2010). In 2009, the Enterprise Holdings Inc. which owns today the National Car Rental, Alamo Rent A Car and WeCar earned about 12.1 billion dollars (Conrad & Perlut 2006; Wikipedia, 2010). These numbers represent only part of the market that also has other major car rental networks such as Dollar and Hertz. The world market in 2012 is estimated at 52.6 billion dollars (Car Rental, 2008).
Besides being itself a major business, spending on car rentals may represent a significant portion of the activities involving tourism and business. Currently the rental options are becoming increasingly diversified with the expansion of the companies, justifying the search for rent schemes that minimize the total cost of this form of transport.

2.2 Models of combinatorial optimization in the car rental industry

Among the various logistical problems of this branch of activity, the literature describes specific studies of combinatorial optimization in the Fleet Assignment Problem (Lia &Tao, 2010), the Strategic Fleet Planning and Tactical Fleet Planning (Pachon *et al.*, 2003), the Demand Forecast (Edelstein & Melnyk, 1977) and the car fleet management problem with maintenance constraints (Hertz *et al.* 2009). Logistic problems that occur in the car rental industry are reviewed by Yang *et al.* (2008). These studies focus on the viewpoint of the car rental industry, however, the customer's point of view has not yet been the subject of published research.

2.3 The car renter salesman problem

In general, under the viewpoint of a user of rented cars, the goal is to minimize the costs to move from a starting point to a destination. On the other hand, when someone rents a car, it is assumed that it meets the requirements of comfort and safety. During the travel, in addition to the costs of renting the car, at least the costs of fuel and the payment of fees to travel on the road should be considered. Let $G=(N, M, W)$ be a graph where $N=\{1,...,n\}$ represents the set of vertices, $M=\{1,...,m\}$ is the set of edges and $W=\{1,...,w\}$ is the set of distances between the vertices or the length of the edges of the set M. The problem described in this paper has the following features:

1. Several types of cars are available for rent, each of them has own characteristics, that is, specific operational costs. These costs include fuel consumption, fees that have to be paid to travel on roads and the value of the rent. The fees that have to be paid to travel on the roads may depend on the type of the car and on the specific roads chosen for the route. The value of the rent can also be associated with a cost per kilometer. Thus, without loss of generality, these costs can be considered as a function of each car on a value associated to the edges (*i,j*) of graph *G*. The operational cost of a given car *k* to traverse an edge (*i,j*) is denoted by c_{ij}^{k}.
2. A car rented in a given company can only be returned in a city where there is an agency of the same company. It is therefore not allowed to rent a car of a given company to travel on a certain segment of the route, if that car cannot be returned on the last city of the segment – there is not an agency of this company in the last city of the segment.
3. Whenever it is possible to rent a car in a city *i* and return it in city *j*, $i \neq j$, there is an extra for returning the car to its home city. The variable d_{ij}^{k} represents the expense to return car *k* to city *j* when it was rented in city *i*, $i \neq j$.
4. The tour begins and ends in the city where the first car is rented, the city that is the basis for the CaRS.
5. The return cost is null in case the tour is completed with a single car which is delivered in the same town it was rented. This case corresponds to the classic TSP considering the cost of other conditions associated only with the selected car.
6. Cars with the same characteristics rented in a single rental car company can be hired under different costs, depending on the city they where rented or on the contract negotiation. Therefore, without loss of generality, the designation of rent can be efficiently controlled by decisions related to cars, not considering the companies. The set $K=\{1,...,k\}$, $|K|=k$ is the set of different cars that can be in the solution.
7. The costs of returning the rented car may be strictly associated with the path between the city where the car is delivered and the city where the car was rented or these costs can be a result of independent calculation.

The objective of the proposed problem it to find the hamiltonian cycle that, starting on an initial vertex previously known, minimize the sum of total operating costs of cars in the tour. The total operating costs are composed of a parcel that unifies the rent and other expenses in a value associated to the edges, and a parcel associated to return the car to a city that is not its basis, calculated for each car and for each pair of cities origin/return in the cycle. The CaRS cycle may also be understood as obtained by the union of up to *t* Hamiltonian paths developed on up to *t* disjoint subsets of vertices of *G*. Each of the paths is accomplished with a different car or a car different from those used for the neighboring paths in the cycle. Therefore the cities that compose the cycle can be grouped into up to *t* different subsets of vertices of *G* that are covered by cars at least distinct from each other in the neighboring paths in the cycle.

Figure 1 illustrates, in a complete graph with six vertices, a typical instance of CaRS. In the example there are three different rental cars. Figures 1(a), (b) and (c) show the accounting of the costs involved in the displacement of each type of car. Note that, unlike the classical traveling salesman cycle, the solution of CaRS depends on the city chosen to be the starting point of the tour, the basis of the salesman. This fact is due to the rate of return is linked both to the starting city and the direction of devolution. In the example this city is represented by vertex F.

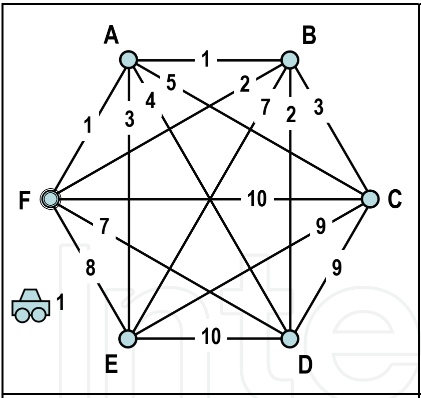
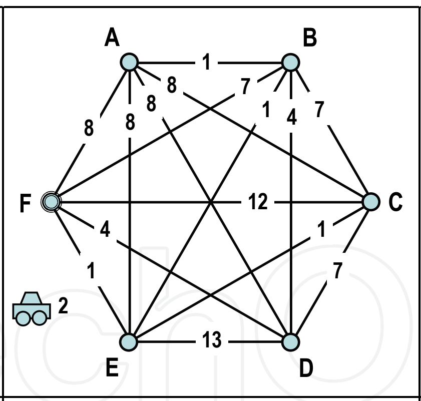
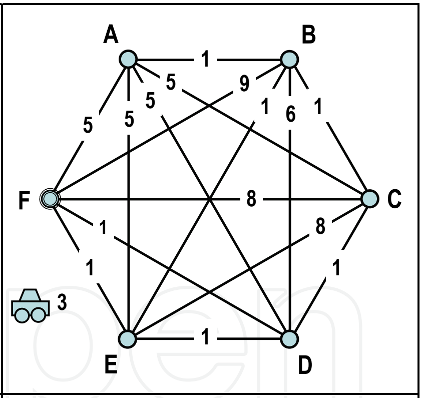
		
(a) Costs of car A	(b) Costs of car B	(c) Costs of car C

Fig. 1. Costs associated to each car

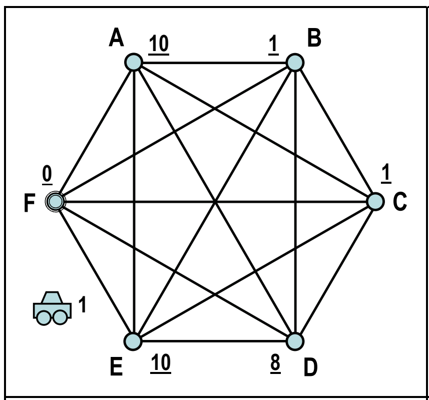
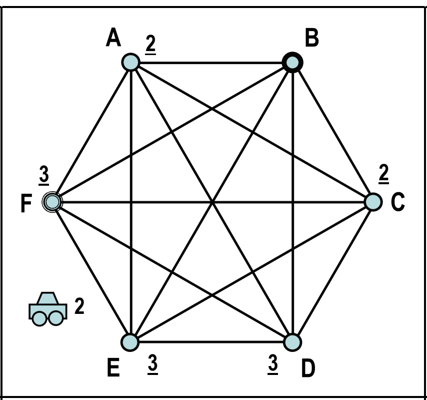
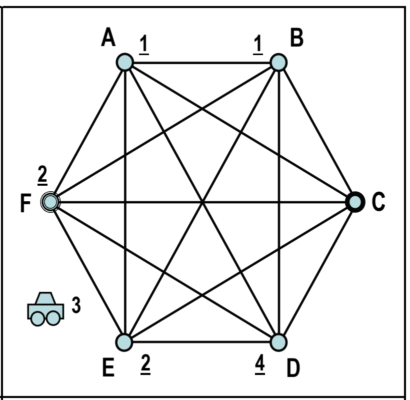
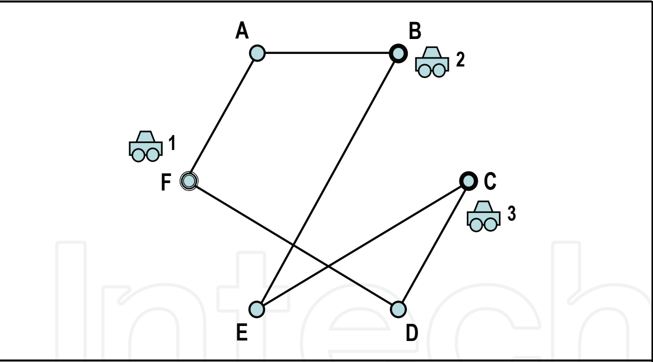
		
(a) Costs to return car 1 when rented on city F	(b) Costs to return car 2 when rented on city B	(b) Costs to return car 3 when rented on city C

Fig. 2. Return costs

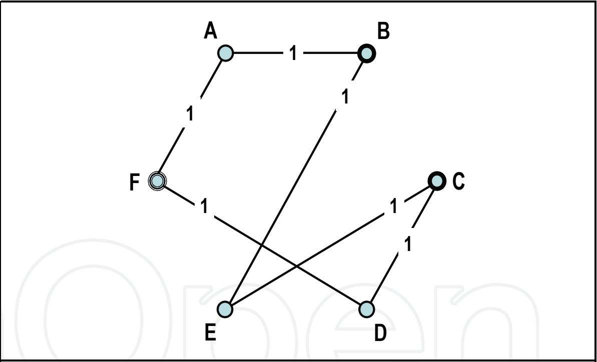
Figure 2 shows, for the example in Figure 1, some of the costs of returning the cars to their bases. Delivery costs appear as underlined numbers next to vertices. Figure 2(a) shows the graph of return of car 1 when rented on vertex F. Figure 2(b) shows the graph of return of car 2 when rented on vertex B and Figure 2(c) the return of car 3 when rented on vertex C. In the general case return costs are known of all cars when rented in any of the cities.

A solution of the problem exemplified in Figures 1 and 2 is exhibited in Figure 3. This solution considers a case where all available cars are rented and no car is rented more than once. The cost of the cycle, according to the solution shown in Figure 3, corresponds to the cost of path F-A-B for car 1, added to the cost of path B-E-C for car 2, added to the cost of path C-D-F of car 3, in a total of 6 unities. To this value it is necessary to add the cost of returning the car to their bases. For car 1, the cost of the return from B to F is one unity. For car 2, the cost of the return from C to B is two unities and, for car 3, the cost to return to C when the car is delivered in F is two unities. Thus, the cost of the final solution is 11 unities.

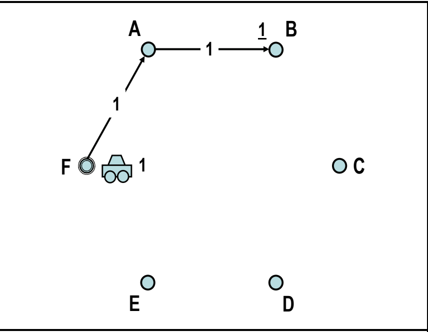
The CaRS Problem has several variants in accordance with the real conditions of the problem. The problem can be classified according to the availability of cars, the alternatives of return, the existence of symmetry of the cost matrix and the existent links between the cities, etc.



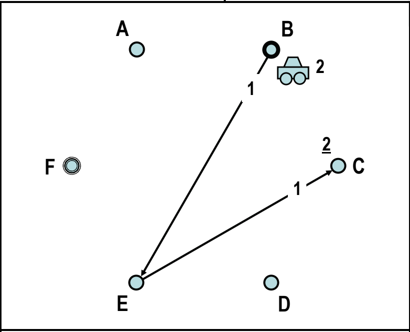
(a) Cars distributed in the route



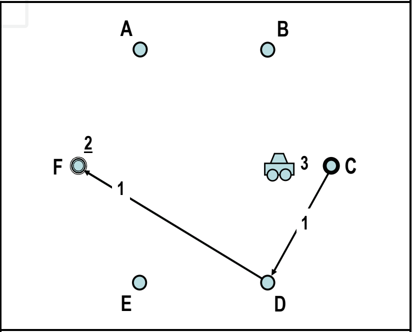
(b) Route costs



(a) Costs of the route of car 1



(b) Costs of the route of car 2



(b) Costs of the route of car 3

Fig. 3. Costs of the route of the exemplified problem

2.4 Variants of the car renter salesman problem

The Renter Salesman admits several specific situations being classified according to:

1. Availability of rental cars

Since there is no guarantee that the rental companies are present in all cities of the tour, it may not necessarily be assumed that any car can be rented in any city. The case in which all cars can be rented in all cities is called *total*. In any other case the problem is called *partial*. In this study, when no observation is made otherwise, the problem is considered total.

2. Alternatives to return the car to its basis

Since there is no guarantee that the rental companies operate services for receiving cars in all cities, it may not necessarily be assumed that any rental car can be returned in any city. To distinguish these different situations, the case in which all cities can receive all cars is called *unrestricted*. In any other situation the problem is called *restricted*. In this paper, when no observation is made otherwise, the problem is considered unrestricted.

3. The integrity of the contract

When the problem does not allow the same type of car is rented more than once on the tour, the problem is called *without repetition*, in this case $k \geq t$. The case without repetition where all cars have to be rented is called *exact*, in this case $k = t$. In any other situation the problem is called *with repetition*. In this paper, when no observation is made otherwise, the problem is considered without repetition.

4. Calculation of the costs of returning a rental car

The costs of returning the cars may be made of values independent of the topology or network restrictions. In this case the problem is called *free*. In the event that the cost of returning a car is calculated taking into account the route used by the car to return to its base, the problem is said to be *bonded*. In this paper, when no observation is made otherwise, the problem is considered free.

5. Symmetry of the distances between the cities

When $c_{ij}^{k} = c_{ji}^{k}$ for $1 \le i,j \le n$, $1 \le k \le ncars$, where *ncars* denotes the number of cars available in a given problem instance, the problem is said to be *symmetric*, otherwise the problem is said to be *asymmetric*.

6. Existence of links in the connection graph that models the problem

When the graph that models the problem is complete, the problem is called *complete*, otherwise the problem is called *incomplete*.

2.5 The difficulty of solving CaRS

The problem basically consists in determining a Hamiltonian cycle in a graph G by composition of paths developed on the vertices of G. Let $T=\{1,\ldots,t\}$ denote the set of indices of up to t subgraphs H_r of G, $r \in T$. Calling $V(H_r)$ the vertices of H_r, the subgraphs H_r of CaRS have the following properties.

$$\bigcup_{r=2}^{t} V(H_r) = N \tag{1}$$

$$\left| V(H_s) \cap V(H_r) \le 1 \right| \qquad \forall r,s \tag{2}$$

Constraint (1) determines that the union of all paths visits all vertices of G. Constraints (2) implies that two different subgraphs never have more than one vertex in common, a condition to prevent formation of subcycles. Note that the constraints (1) and (2) are not sufficient to guarantee the cycle of the CaRS. It is also necessary that the t subgraphs considered three to three, four to four, and so on, until t-1 to t-1, do not have more than one vertex in common.

Once this problem deals with Hamiltonian paths, each path done by a car in one subgraph H_r visits all vertices of H_r. The path of subgraph H_r has to be assigned to a car different from the cars assigned to neighbor paths during the construction of an Hamiltonian cycle in G. The costs of the edges of each subgraph correspond to the operation costs of the car traversing H_r. Furthermore, when $t \ge 2$ the total cost considers the return cost of each car rented in city i and returned in city j, $i \ne j$. Hamiltonian cycle and Hamiltonian path problems are well known NP-complete problems (Garey & Johnson, 1979). Due to what was previously exposed, the difficulty of solving CaRS is at least the same as the TSP. Nevertheless, although some solutions of the TSP are also solutions of CaRS, the latter has a number of feasible solutions greater than the former and incorporates all the requirements of the TSP, like other several classes of vehicle routing problems which are known to be more difficult than the TSP (Ralphs *et* al., 2003).

The Traveling Salesman Problem is a particular case of CaRS in the situation where there is only one vehicle available for rent. Note that the solution space of CaRS is exponentially greater than the solution space of the Traveling Salesman Problem. Considering $G = (N,M)$ a complete graph and that CaRS is total, unrestricted and without repetition, any permutation of the vertices of G is a feasible solution for the rental car problem considering only one of the k possible cars. Once there are k cars available for rent, there are $k.n!$ different feasible configurations that meet the condition of use of only one different car in CaRS. From $k \ge 2$

each hamiltonian cycle of the car renter salesman can be partitioned in up to k-subpaths in sequence and to each possibility of composition of such partitions a distinct cost for the route can be assigned. The possibility of this single value is guaranteed due to the composition of the costs related to the return fee of the rented cars with the costs of the trajectories of independent costs of each car. The number of possible partitions associated to each hamiltonian cycle in G is equal to the number of different ways the set of n cities of the cycle can be divided in s groups of cities that are visited by s different cars, $k \geq s \geq 2$. The number that counts this process of division of a set in disjoint sets that go from 2 to k is the Stirling number of the second type. In this way, the number of configurations of the space of solutions of CaRS is at least $O(2^n)$ greater than the space of solutions of the Traveling Salesman Problem, since that for $k = 2$ the associated number of Stirling is $O(2^n)$ (Amdeberhan *et al*. 2008). For a general case of CaRS the dimension of the space of solutions is still greater once there is not a theoretical limit for the number of different cars to be considered in the problem.

3. Metaheuristic algorithms

This section presents two heuristics for the investigated problem. The first one is a Greedy Randomized Adaptive Search Procedure (GRASP) (Feo & Resende, 1995) combined with Variable Neighborhood Descent (VND) (Mladenović & Hansen, 1997) in the local search phase. The second heuristics is a Memetic Algorithm (Moscato, 1989).

3.1 GRASP with VND

This algorithm has a pre-processing phase where *nCar* optimal TSP solutions are obtained with the Concorde TSP Solver (Applegate *et al*., 2001), one for each available car, where *nCar* is the number of cars available for the instance being considered. The constructive phase of the GRASP hybridized with VND, named GVND, starts with a random selected car at the home city. A path is built each iteration between two cities: a known origin and a destination city randomly chosen among the cities yet not considered by the algorithm. A Restricted Candidate List (RCL), with size α, is built with the cities that have the cheapest return rates for the car being considered in the current iteration. The destination city is selected at random, based on a roulette wheel, from the RCL and a path between the origin and the destination city is built. Except for the last iteration, the path is built based on the tours built in the pre-processing phase. In the first iteration the path between the origin and destination cities is obtained in the optimal solution correspondent to the first selected car. The path of the *i-th* car, $1 < i < nCar$, is also obtained from the optimal solution that corresponds to the *i-th* car, but in this case a procedure to remove cities already considered in paths constructed in previous iterations may be necessary. Suppose that city b, between cities a and c in the path built in the *i-th* iteration, is already in a path built in iteration j, $j < i$. Then the procedure removes city b from the *i-th* path and includes a link between cities a and c. The initial starting city is the origin of the first iteration. The origin city of iteration i, $i > 1$, is the destination city of iteration i-1. The destination city of the last iteration is the initial starting city. In the last iteration, the nearest neighbor heuristic is used to build the path of the last considered car.
In the local search phase a VND metaheuristic was used to explore the search space of three neighborhood structures named *InvertSol*, *Insert&Saving* and *Shift*. *InvertSol* is a simple low

time consuming heuristics that reverses the sequences of cities of an input solution. With the reversal, though the same cars traverse the same sets of cities, the cost of rent and fees for returning the car to where they were rented change. The *Insert&Saving* procedure searches for a car insertion in a given solution that yields a decreasing of its cost. Let *s* be a solution in which there is at least one car that is not assigned to a path in *s*. *Insert&Saving* method randomly chooses a not assigned car and searches the best position to insert it in *s*. The procedure verifies the cost of the insertion of the new car in every point of *s*. If any of these insertions produces a solution with a cost lower than the cost of *s*, the new solution is set as the current solution in the local search. The procedure continues until all non-assigned cars have been considered for insertion. In the third procedure, *Shift*, a neighboring solution *s'* of a solution *s* is generated by exchanging two cities within the path of one car of *s*. The whole neighborhood of each solution is searched in the local search procedures.

3.2 Memetic algorithm

Algorithm 1 shows the pseudo-code of the Memetic Algorithm (MA) developed for CaRS. The input parameters are: number of generations (*nOffspring*), population size (*sizePop*), recombination rate (*txCros*) (the number of individuals that reproduce in each generation), mutation rate (*txMuta*) and renewal rate of the population (*txRenw*).

Algorithm 1 – Main Procedure of Memetic for CaRS

```
1.  main(nameInstance,sizePop,nOffspring,txCros,txMuta,txRenw)
2.    instanceRead(nameInstance)
3.    Pop[] ← generateInitPop(sizePop)
4.    VNDlocalSearchPhase(Pop)
5.    for i of 1 to nOffspring do
6.      for j of 1 to sizePop*txCros do
7.        dad,mom ←  parentsSelection()
8.        son1, son2 ← Crossover(dad,mom)
9.        son1, son2 ←  carsMutation(son1,son2,txMuta)
10.       VNDlocalSearchPhase(son1,son2)
11.       if son1,son2 < Pop[dad],Pop[Mom]
12.         Pop[dad] ← son1, Pop[mom] ← son2
13.   generateNewIndividuals(sizePop*txRenw)
14. return(Pop[0])
```

Chromosomes are represented in 2-dimensional arrays with *n* elements as illustrated in figure 4 for an instance with *n* = 11 and 5 cars. The second row corresponds to the sequence of cities visited in the tour. The elements in the first row correspond to cars. Let car *c* be assigned to the cities in the second row corresponding to indices i_1 to i_m, that is, car *c* goes from city i_1 to city i_m, then the elements of the first row corresponding to indices i_1 to i_{m-1} are equal to *c*. The last city visited by a car is not assigned to that car in the chromosome, since the car is returned on that city and another car is rented there to continue the tour. The starting city (city 0) is not represented in the chromosome as the final destination. In figure 4 four of the five available cars are used. The tour begins at city 0 with car 2 which passes through cities 6, 4, 3, 10 and is delivered in city 7 where car 1 is rented. Car 1 proceeds to city 9 passing through city 1. Car 5 is rented in city 9, passes through cities 2 and 5 and is delivered in city 8 where the last car, 4, is rented. Car 4 is delivered in the starting city. The fitness of each chromosome is given by the inverse of the objective function, which means that the lower the value of the objective function the fittest the chromosome is.

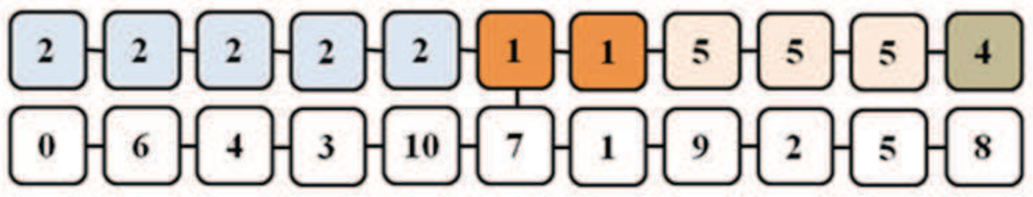


Fig. 4. Chromosome

The initial population is generated with a version of the nearest neighbor heuristics adapted for CaRS in procedure *generateInitPop*( ) which receives the size of the population as input parameter. Let *nCar* be the number of available cars of a given instance. The algorithm randomly selects a car *c* and a destination city *j* for $c, j \neq 0$. Then a path between cities 0 and *j* is built with the nearest neighbor heuristic. City *j* is set as the new origin and a new car and a new destination city are randomly selected. The procedure continues until all cities are added to the tour or until there is only one car available. In the latter case, the last available car is assigned to a path built with the same heuristics between the previous destination city and the starting city, closing the tour. In step 4, each individual of the initial population is subjected to the same VND procedure used in GVND.

Parents for recombination are selected with the roulette wheel method. A multi-point recombination operation adapted for CaRS is used to generate two children. The recombination operator is illustrated in Figure 5, considering an instance with n = 11 and 3 cars. Two parent chromosomes, A and B, generate offspring C and D. In Figure 5 a 2-point operator is used. The first and third parts of chromosomes A and B are inherited by chromosomes C and D, respectively. A restoration procedure may be necessary to restore feasibility regarding the routes and car assignments. For example, after recombination the route of chromosome C is [0 3 1 8 10 1 9 4 5 10 6] which is not feasible since cities 1 and 10 appear twice each and cities 2 and 7 are missing. Thus the route of chromosome C is replaced by [0 3 1 8 10 * 9 4 5 * 6] with asterisks replacing the second time cities 1 and 10 appears. Each asterisk is then replaced at random by cities 2 and 7. The row corresponding to the car assignment of chromosome C after recombination is [1 1 1 1 2 3 3 2 2 2 2 3] which is not feasible for the problem considered in this paper, since each car can be rented only once. Thus, the car assignment of chromosome C is replaced by [1 1 1 1 2 3 3 * * * * 3] and each asterisk is replaced by car 3. Chromosome D is analyzed similarly.

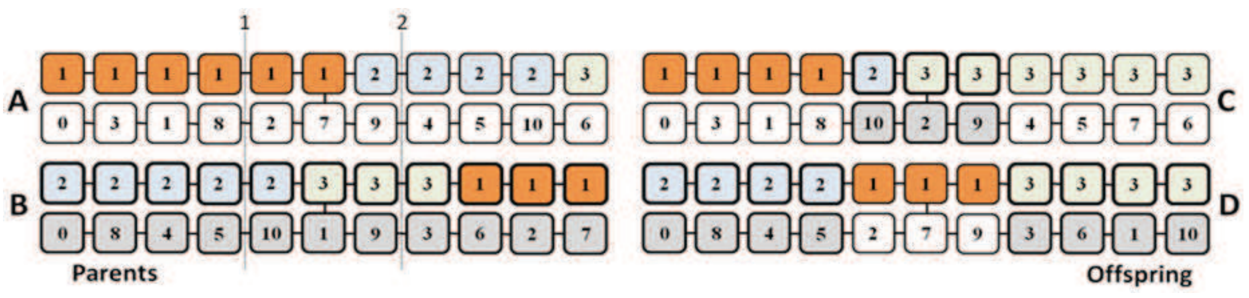


Fig. 5. Recombination operator

The solutions resulting from the recombination are subjected to mutation. The mutation operator verifies which vehicles are not in the solution represented in the chromosome. Each of these vehicles is inserted in the chromosome in a given segment defined in advance. The maximum size of each segment is defined as the mutation rate. The mutation operator is illustrated in figure 6 considering an instance with *n*= 11, 5 cars, and the mutation rate sets the size of the segments to 3. Cars 3 and 4 are not used in the solution shown in chromosome A of figure 6. The mutation operator inserts the missing cars in the solution, resulting on chromosome B. Cities 10 and 5 are chosen at random to be the starting cities of cars 3 and 4, respectively. Therefore, vehicle 5 is replaced by vehicle 3 in cities 10, 7 and 1 and car 4 replaces car 1 in cities 5 and 8, since these are the last two cities of the tour. The sequence of cars is the resulting chromosome is [2 2 2 5 3 3 3 5 1 4 4]. A repairing function has to be utilized to restore the feasibility of the resultant chromosome, since car 5 appears in two different paths. The second time car 5 appears in the solution is removed, resulting on [2 2 2 5 3 3 3 * 1 4 4]. The asterisks are replaced by the car that appears in the city immediately before to the ones which the asterisks are assigned. In this example, the asterisks are replaced by car 3.

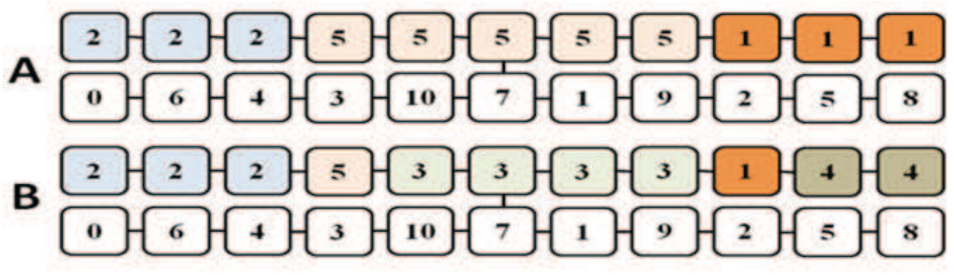


Fig. 6. Mutation operator

After crossover and mutation, the offspring is subjected to the VND method presented at GVND. The resulting solutions are compared with their parents and the best two individuals survive. Finally, part of the current population is replaced by new solutions generated with the constructive method used to create the initial population. The number of new individuals created with that procedure is given by the renewal rate and the individuals chosen to be replaced are those with the worst values of fitness. This renewal process promotes diversification and prevents premature convergence.

4. Computational experiments

This section presents the comparison between the performance of the GRASP/VND and the Memetic Algorithm, called GVND and MA, respectively. Since the problem introduced in this paper is new, a library of instances, named CaRSLib, was created with the purpose of testing the proposed algorithms. These instances have the following features: *total* (all cars can be rented in all cities), *unrestricted* (all cars can be delivered in any city), *without repetition* (each type of car can be rented at most once), *free* (the return costs are not correlated with instance topology), *symmetric* (costs to go from city *i* to city *j* and vice-versa are equal) and *complete* (the graph that models the instance is complete). The set consists of Euclidean and non-Euclidean instances. For each set, three groups of instances were created, the first is based on real maps, the second was formed with randomly generated data and the third one is based on the TSPLIB instances. The dataset, the description of each group of instances and file formats are available at http://www.dimap.ufrn.br/lae/en/projects/CaRS.php.

An exact backtracking algorithm was developed and adapted to CaRS. This method enumerates all possible configurations utilizing permutations of the available cars for each instance. The results were used to evaluate the solutions generated by the metaheuristic algorithms on the instances solved by the exact method. The algorithm was implemented in C++ and executed on an Intel Core Duo 1.67 GHz, 2GB RAM running Linux. The algorithm solved eighteen Euclidean and non Euclidean instances with *n* between 10 and 16 and 2 cars. Table 1 presents the results for the backtracking algorithm (column *Backtrack*) and the best results obtained by GVND and MA. Columns *gap* show the deviation of the best solution (*Best*) found by the metaheuristic algorithm from the optimum (#*Opt*).
Table 1 shows that the performance of the memetic algorithm is satisfactory for small instances, obtaining the optimum for all investigated instances. The success rate of GVND is 78% and the maximal deviation from the best solution is 3%.

INSTANCE			BACKTRACK		GVND			MA		
Name	City	Car	T(s)	#Opt	T(s)	Best	GAP	T(s)	Best	GAP
Mauritania10e	10	2	1	540	1	540	0.00	1	540	0.00
Mauritania10n	10	2	1	571	1	571	0.00	1	571	0.00
Colombia11e	11	2	19	620	1	620	0.00	1	620	0.00
Colombia11n	11	2	14	639	1	640	0.00	1	639	0.00
Angola12e	12	2	266	719	1	719	0.00	1	719	0.00
Angola12n	12	2	144	656	1	656	0.00	1	656	0.00
Peru13e	13	2	1953	672	1	672	0.00	1	672	0.00
Peru13n	13	2	1847	693	1	693	0.00	1	693	0.00
Libia14e	14	2	31273	730	1	730	0.00	1	730	0.00
Libia14n	14	2	28331	760	1	764	0.01	1	760	0.00
BrazilRJe	14	2	44104	294	1	297	0.01	1	294	0.00
BrazilRJn	14	2	35263	167	1	170	0.02	1	167	0.00
Congo15e	15	2	455788	756	1	756	0.00	1	756	0.00
Congo15n	15	2	412212	886	1	886	0.00	1	886	0.00
Argentina16e	16	2	7603200	955	1	955	0.00	1	955	0.00
Argentina16n	16	2	7612310	894	1	894	0.00	1	894	0.00
BrasilRN16e	16	2	7609203	375	1	375	0.00	1	375	0.00
BrasilRN16n	16	2	7613217	188	1	194	0.03	1	188	0.00

Table 1. Results of Backtrack, GVND, MA

The results shown in the following tables for GVND and MA were obtained on a PC Intel Xeon QuadCore W3520 2.8 GHz, 8G of RAM running Scientific Linux 5.5 64bits. The results refer to 40 CaRS instances, 20 of them are Euclidean and 20 are non Euclidean. Each group of instances is formed with 10 instances based on real maps, 5 random instances and 5 instances based on TSPLIB (Reinelt, 1995). Thirty independent executions of each algorithm were performed for each instance. Two groups of experiments were performed with fixed processing times. In the first group the average processing times spent by GVND to find its

best solution for each instance was given to both algorithms. In the second group the average processing times of the MA for each instance was fixed for both algorithms.
Preliminary tests to tune the parameters of the proposed algorithms were executed on a set of 20 CaRSLib instances, with number of cities ranging from 14 to 300 and 2 to 5 vehicles. Twenty independent executions were performed for each instance. Two groups of tests were performed with the GVND. First, the algorithm was executed without the VND method and then the VND was included. The parameter α was set to 0.25. The maximum number of re-starts was set to 300. An additional stopping criterion fixed 90 re-starts without improvement of the best current solution. The parameters chosen for the Memetic Algorithms are: *nOffspring* = 20, *sizePop* = 30, *txCros* = 0.60, *txMuta* = 0.40 and *txRenw* = 0.15. The stopping criterion was *maxGen* = 0.30*nOffspring* generations without improvement of the best current solution.

INSTANCE				GVND				MA				T(s)	p-value
Name	City	Car	#Best	Avg	SD	Best	Freq	Avg	SD	Best	Freq		
BrasilRJ14e	14	2	294	297	0	297	0	**294**	0	**294**	29	1	0
BrasilRN16e	16	2	375	**375**	0	**375**	30	**375**	2	**375**	29	1	0.85
BrasilPR25e	25	3	510	**510**	0	**510**	29	515	5	**510**	16	2	1
BrasilAM26e	26	3	467	495	1	495	0	**485**	9	**469**	0	3	0
BrasilMG30e	30	4	563	603	2	595	0	**599**	7	**575**	0	5	0
BrasilSP32e	32	4	611	633	5	626	0	**621**	4	**611**	2	8	0
BrasilRS32e	32	4	510	537	9	529	0	**522**	6	**510**	1	8	0
BrasilCO40e	40	5	779	**807**	2	805	0	822	10	**779**	1	18	1
BrasilNO45e	45	5	886	1008	0	1008	0	**978**	34	**886**	1	23	0
BrasilNE50e	50	5	822	963	5	940	0	**954**	27	**822**	1	43	0.08
Betim100e	100	3	1401	1723	8	1708	0	**1692**	89	**1410**	0	78	0.99
Vitoria100e	100	5	1598	**1802**	75	1642	0	1891	87	**1598**	1	155	1
PortoVelho200e	200	3	2827	**3142**	29	3041	0	3149	129	**2827**	1	466	0.98
Cuiaba200e	200	3	3052	**3379**	88	3212	0	3414	80	**3217**	0	686	1
Belem300e	300	4	4031	4635	121	4563	0	**4425**	76	**4031**	1	1804	0
berlin52eA	52	3	8948	**9020**	35	8991	0	9081	72	**8948**	4	20	1
eil76eB	76	4	1940	2228	42	2158	0	**2077**	43	**1940**	1	87	0
rat99eB	99	5	3339	**3439**	42	3351	0	3513	75	**3365**	0	194	1
rd100eB	100	4	9951	**10107**	81	**9951**	1	10364	172	10054	0	103	1
st70eB	70	4	2037	2201	44	2085	0	**2151**	46	**2042**	0	77	0

Table 2. Results with time determined by GVND for Euclidean instances

With the aim of comparing the algorithms on a fair basis, the same maximum processing time is given for both algorithms. These processing times were obtained in preliminary experiments with the stop conditions afore mentioned. The results are reported in Tables 2-5. These tables show the name of the instance (*Name*), the number of cities (*City*), the number of available cars (*Car*), the best solution found (#*Best*), the average (*Avg*) solution, the best solution obtained by one of the tested algorithms (*Best*), the standard deviation (*SD*), the number of times (*Freq*) the best known solution, reported in column #*Best*, was found by each algorithm, the maximum processing time in seconds (*T*) and the p-value obtained in the statistical U-test (Conover, 2001). Considering level of significance 0.05, values less than 0.05 in the last column indicate that the performance of MA is significantly better than the performance of GVND and values greater than 0.95 indicate that GVND produced better results than MA.

Tables 2 and 3 refer to Euclidean instances and show the results with the maximum processing time fixed by the stop conditions of GVND and MA, respectively. Similarly, Tables 4 and 5 show results for non Euclidean instances with the maximum processing time fixed by the stop conditions of GVND and MA, respectively.

INSTANCE				MA				GVND				T(s)	p-value
Name	City	Car	#Best	Avg	SD	Best	Freq	Avg	SD	Best	Freq		
BrasilRJ14e	14	2	294	**294**	1	**294**	25	297	0	297	0	1	0
BrasilRN16e	16	2	375	376	4	**375**	27	**375**	0	**375**	30	1	0.96
BrasilPR25e	25	3	510	515	5	**510**	17	**510**	0	**510**	30	2	1
BrasilAM26e	26	3	467	**481**	10	**467**	3	495	0	495	0	4	0
BrasilMG30e	30	4	563	**596**	10	**563**	1	602	2	595	0	6	0
BrasilSP32e	32	4	611	**624**	5	**615**	0	632	4	626	0	8	0
BrasilRS32e	32	4	510	**523**	7	**512**	0	536	9	529	0	8	0
BrasilCO40e	40	5	779	824	7	**801**	0	**806**	2	806	0	17	1
BrasilNO45e	45	5	886	**993**	27	**897**	0	1008	0	1008	0	25	0
BrasilNE50e	50	5	822	963	2	953	0	**962**	11	**908**	0	31	0.43
Betim100e	100	3	1401	**1642**	110	**1401**	1	1720	7	1708	0	128	0.30
Vitoria100e	100	5	1598	1922	29	1814	0	**1859**	77	**1676**	0	98	1
PortoVelho200e	200	3	2827	3134	117	**2871**	0	**3128**	22	3041	0	766	0.61
Cuiaba200e	200	4	3052	3415	96	**3052**	1	**3365**	56	3334	0	701	1
Belem300e	300	4	4031	**4434**	30	**4282**	0	4621	125	4563	0	2016	0
berlin52eA	52	3	8948	9094	65	**8948**	4	**9013**	24	8991	0	27	1
eil76eB	76	4	1940	**2069**	43	**1986**	0	2226	56	2129	0	61	0
rat99eB	99	5	3339	3525	71	**3339**	1	**3468**	54	3348	0	128	1
rd100eB	100	4	9951	10385	209	9994	0	**10055**	54	**9951**	1	161	1
st70eB	70	4	2037	**2158**	67	**2037**	1	2212	31	2137	0	54	0

Table 3. Results with time determined by Memetic Algorithm for Euclidean instances

Provided that the same computational effort (processing time) is fixed, throughout this section a statistical test for proportions comparison is applied. The test proposed by Taillard *et al.* (2008) compare success rates between two methods. In this paper, given two methods *A* and *B*, success of method *A* is stated when *A* achieves a better result than *B* for the same problem instance. The values for this test presented here were calculated with the tool available at http://qualopt.eivd.ch/stats/?page=stats with the one-tailed Taillard Test.

Column *p-value* of Table 2 shows that MA and GVND outperforms one another on 9 instances each, considering level of significance 0.05. With the processing time of MA fixed for both algorithms, column *p-value* of Table 3 shows that MA presents the best performance on 9 instances and the GVND on 8 instances. These results show that the algorithms present similar performance concerning the number of instances each of them is significantly better than the other. If hitting the lowest value solution is set as a target for the algorithms, then Table 2 and 3 show that MA hits this target 19 and 17 times, respectively, whilst GVND hits the target 3 and 5 times. For these results, the test to compare success rates between two methods (Taillard *et al.*, 2008) shows that the confidence level of the hypothesis that MA has success rate higher than GVND is 1 and 0.999968 when the maximum processing times are fixed by the GVND and the MA, respectively.

INSTANCE				GVND				MA				T(s)	p-value
Name	City	Car	#Best	Avg	SD	Best	Freq	Avg	SD	Best	Freq		
BrasilRJ14n	14	2	167	171	0	171	0	**167**	0	**167**	4	1	0
BrasilRN16n	16	2	190	203	0	203	0	**194**	2	**192**	0	1	0
BrasilPR25n	25	3	235	311	9	305	0	**255**	7	**239**	0	5	0
BrasilAM26n	26	3	204	242	6	239	0	**213**	4	**206**	0	5	0
BrasilMG30n	30	4	279	375	11	352	0	**330**	14	**298**	0	11	0
BrasilSP32n	32	4	285	336	16	298	0	**295**	5	**285**	1	12	0
BrasilRS32n	32	4	297	372	15	344	0	**337**	14	**297**	1	15	0
BrasilCO40n	40	5	655	826	42	755	0	**718**	37	**655**	1	39	0
BrasilNO45n	45	5	664	889	42	770	0	**753**	39	**664**	1	55	0
BrasilNE50n	50	5	707	1044	60	874	0	**844**	43	**761**	0	81	0
Londrina100n	100	3	1450	1783	80	1629	0	**1564**	51	**1450**	1	192	0
Osasco100n	100	4	1150	2000	60	1910	0	**1443**	109	**1265**	0	191	0
Aracaju200n	200	3	2467	3686	212	3223	0	**2802**	136	**2588**	0	903	0
Teresina200n	200	5	2192	3793	144	3261	0	**2480**	143	**2192**	1	1407	0
Curitiba300n	300	5	3676	6125	202	5680	0	**4081**	202	**3749**	0	3388	0
berlin52nA	52	3	1480	1777	82	1661	0	**1640**	51	**1543**	0	41	0
ch130n	130	5	2487	4706	307	3855	0	**2940**	245	**2487**	1	478	0
d198n	198	4	4807	7138	333	6529	0	**5332**	269	**4807**	1	1330	0
kroB150n	150	3	3824	5368	434	4414	0	**4312**	194	**3824**	1	464	0
rd100nB	100	4	1890	2953	169	2623	0	**2274**	118	**2083**	0	205	0

Table 4. Results with time determined by GRASP/VND for non Euclidean instances

Column *Freq* of Table 2 shows that, on average, 15% and 10% of the best solutions generated by one of the tested algorithms on the two experiments with fixed processing times are found by MA and GVND, respectively. Data presented in column *Freq* of Table 3 show that, on average, 13.5% and 10% of the best solutions are found by MA and GVND, respectively.

INSTANCE				MA				GVND				T(s)	p-value
Name	City	Car	#Best	Avg	SD	Best	Freq	Avg	SD	Best	Freq		
BrasilRJ14n	14	2	167	**167**	0	**167**	2	171	0	171	0	1	0
BrasilRN16n	16	2	190	**195**	3	**190**	1	203	0	203	0	1	0
BrasilPR25n	25	3	235	**256**	10	**235**	1	316	12	305	0	4	0
BrasilAM26n	26	3	204	**212**	4	**204**	1	242	5	239	0	5	0
BrasilMG30n	30	4	279	**328**	15	**279**	1	378	14	352	0	8	0
BrasilSP32n	32	4	285	**296**	7	**287**	0	331	14	300	0	13	0
BrasilRS32n	32	4	297	**340**	16	**304**	0	378	18	344	0	9	0
BrasilCO40n	40	5	655	**743**	33	**668**	0	839	41	710	0	20	0
BrasilNO45n	45	5	664	**764**	39	**667**	0	919	43	814	0	32	0
BrasilNE50n	50	5	707	**861**	61	**707**	1	1068	57	924	0	46	0
Londrina100n	100	3	1450	**1592**	50	**1471**	0	1767	85	1629	0	146	0
Osasco100n	100	4	1150	**1442**	139	**1150**	1	2046	80	1817	0	125	0
Aracaju200n	200	3	2467	**2744**	135	**2467**	1	3594	236	3106	0	922	0
Teresina200n	200	5	2192	**2551**	182	**2233**	0	3866	126	3611	0	836	0
Curitiba300n	300	5	3676	**4076**	216	**3676**	1	6050	160	5647	0	2384	0
berlin52nA	52	3	1480	**1642**	78	**1480**	1	1748	63	1661	0	38	0
ch130n	130	5	2487	**3020**	228	**2493**	0	4863	345	3813	0	237	0
d198n	198	4	4807	**5449**	318	**4887**	0	7407	378	6250	0	823	0
kroB150n	150	3	3824	**4259**	208	**3845**	0	5313	272	4871	0	418	0
rd100nB	100	4	1890	**2271**	143	**1890**	1	2962	180	2685	0	140	0

Table 5. Results with time determined by Memetic Algorithm for non Euclidean instances

The analysis of columns *p-value* of Tables 4 and 5 shows MA outperforms GVND on all non Euclidean instances, regardless the maximum processing time fixed for the experiments. All best results are found by MA. On average, MA finds approximately 2% of the best solutions on the 30 executions of each one of the 40 tested instances.

5. Conclusion

This chapter presented the Car Renter Salesman Problem (CARS), a new generalization of the classic Traveling Salesman Problem. An experimental investigation was carried out to compare two metaheuristic approaches proposed for this new problem: GRASP (Greedy Randomized Search Procedure) hybridized with VND (Variable Neighborhood Descent) and Memetic Algorithms. The algorithms were applied to 40 Euclidean and non Euclidean instances of the CaRSLib benchmark which is proposed for this problem. An exact procedure established the optimal solutions of 4 from the 40 instances, whilst the proposed heuristics established the first upper limits for the remaining 36 instances. Statistical tests are applied to the results generated by the proposed algorithms in order to support conclusions on their behaviors concerning quality of solution.

To establish a fair basis of comparison for the proposed algorithms, the effect of the computational effort demanded by each algorithm is neutralized by comparing the performance of each algorithm according to fixed processing times. These execution times are established in accordance to the requirements of each algorithm for its best performance. Therefore, the proposed algorithms are tested twice, first with the processing times fixed by the best performance of one algorithm and then with the processing times fixed by the best performance of the other. Thus, a superior qualitative behavior can be considered conclusive when it holds for both processing time conditions.

The results of the computational experiments showed that for Euclidean instances the proposed algorithms present similar behavior with some advantage for MA concerning the number of best solutions found. For the set of non Euclidean instances, MA outperformed GVND on the whole set regardless the maximum processing time fixed for both algorithms. The MA also presented the best solution values for all non Euclidean instances.

This chapter presented also other six variants for the introduced problem, opening up the topic for future research.

6. References

Amdeberhan, T.; Manna, D. & Moll, V. H. (2008). The 2-adic Valuation of Stirling Numbers, *Experiment. Math.* 17 (1), pp. 69-82

Applegate, D.L.; Bixby, R.; Chvátal, V. & Cook, W. (1994). Finding cuts in the TSP: a preliminary report distributed at The Mathematical Programming Symposium, Ann Arbor, Michigan

Applegate, D.L.; Bixby, R.; Chvátal, V. & Cook, W. (2001). TSP cuts which do not conform to the template paradigm, In : *Computational Combinatorial Optimization*, M. Junger and D. Naddef (editors), pp. 261-303

Applegate, D.L. Bixby, R.; Chvátal, V. & Cook, W. (2006). The Traveling Salesman Problem: A Computational Study, Princeton University Press

Avis (2009). Avis Europe plc. Financial and Strategic Analysis Review, *Global Markets Direct*, 17, available at http://www.researchandmarkets.com/reports/690999

Avis (2010). Avis Budget Rental Car Funding, available at http://www.dbrs.com/research/232065/avis-budget-rental-car-funding-aesop-llc-series-2010-2-3/rating-report-series-2010-2-3.pdf

Car Rental (2008). Web Page, available at http://thrifty4.com/article.cfm/id/284920

Car (2008). Car Rental Business, *Global Strategic Business Report Global Industry Analysts*, Inc., 278, available at http://www.researchandmarkets.com/reports/338373/

Conover, W. J. (2001). Practical Nonparametric Statistics, John Wiley & Sons, 3rd Ed.

Conrad, C. & Perlut, A. (2006). Enterprise Rent-A-Car Hits New Billion-Dollar Revenue Mark for 3rd Consecutive Year, *Enterprise rent-a-car*, available at: http//www.enterpriseholdings.com/NewsReleases/Enterprise_FYO6_Sept06.pdf

Crowder, H. & Padberg, M. W. (1980). Solving large scale symmetric traveling salesman problems to optimality, *Management Science* 26, pp. 495-509

Edelstein, M. & Melnyk, M. (1977). The pool control system, *Interfaces* 8(1), pp. 21-36

Enterprise (2009). Enterprise Rent-A-Car Company - Strategic Analysis Review, *Global Markets Direct* 9, available at http://www.researchandmarkets.com/reports/690685/

Feo, T.A. & Resende, M.G.C. (1995). Greedy randomized adaptive search procedures, *Journal of Global Optimization* 6, pp. 109-133

Garey, M. R. & Johnson, D. S. (1979). Computers and Intractability : A Guide to the Theory of NP-completeness. W.H. Freeman and Company, New York

Gutin, G. & Punnen, A. P. (2002). The Traveling Salesman Problem and Its Variations, Series: *Combinatorial Optimization* 12, Springer

Hertz (2009). The Hertz Corporation - Strategic Analysis Review, *Global Markets Direct* 12, available at http://www.researchandmarkets.com/reports/690555/

Hertz, A.; Schindl, D. & Zufferey, N. (2009). A solution method for a car fleet management problem with maintenance constraints, *Journal of Heuristics* 5, pp. 425–450

Krarup, J. (1975). The peripatetic salesman and some related unsolved problems, In: *Combinatorial Programming Methods and Applications*, Reidel, Dordrecht, pp. 173–178

Lia, Z. & Tao, F. (2010). On determining optimal fleet size and vehicle transfer policy for a car rental company, *Computers & Operations Research* 37, pp. 341-350.

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

Montemanni, R.; Barta, J.; Mastrolilli, M. & Gambardella, L. M. (2007). The robust traveling salesman problem with interval data, *Transportation Science* 41(3), pp. 366–381

Moscato, P. (1989). On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts: Towards Memetic Algorithm. *Caltech Concurrent Computation Program*. California Institute of Technology, USA

Pachon. J. E.; Iakovou, B.; Ip, C. & Aboudi, R. (2003). A synthesis of tactical fleet planning models for the car rental industry, *IIE Transactions* 35(9), pp. 907-916

Ralphs, T. K.; Kopman, L.; Pulleyblank, W. R. & Trotter, L. E. (2003). On the capacitated vehicle routing problem, *Mathematical Programming*, Ser. B 94, pp. 343–359

Reinelt, G. (1995). TSPLIB95, available at http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95

Russel, R. (1977). An effective heuristic for the m-tour traveling salesman problem with some side conditions, *Operations Research* 25, pp. 517-524

Taillard, E.; Waelti, P. & Zuber, J. (2008). Few statistical tests for proportions comparison, *European Journal of Operational Research*, vol. 185, pp. 1336–1350

Wikipedia (2010). Enterprise Rent-a-Car page, available at http://en.wikipedia.org/wiki/Enterprise_Rent-a-Car

Xiong, Y.; Golden, B. & Wasil, E. (2007). The Colorful Traveling Salesman Problem, Extending the Horizons: Advances in Computing, Optimization, and Decision Technologies, *Operations Research/Computer Science Interfaces Series* 37, Springer US, pp. 115-123

Yang, Y.; Jin, W. & Hao, X. (2008). Car Rental Logistics Problem: A Review of Literature, *IEEE International Conference on Service Operations and Logistics, and Informatics.* IEEE/SOLI 2008. 2, pp. 2815–2819