Hybrid Metaheuristics Applied to the Vehicle Routing Problem With Simultaneous Pickup and Delivery

Dilson Lucas Pereira[1] **, Geraldo Robson Mateus**[1]

[1]Departamento de Ciência da Computação
Universidade Federal de Minas Gerais

{dilson,mateus}@dcc.ufmg.br

Abstract. *This work addresses the Vehicle Routing Problem with Simultaneous Pickup and Delivery, where routes must be created to satisfy the pickup and delivery requests of a set of customers. Each customer must be served once and by only one route, the load it receives is brought from a central depot, to where the picked-up load is also taken. The capacity of the used vehicles must not be violated at any point of the routes. We propose some heuristics based on ideas of many metaheuristics, specially ILS, GRASP, and VND. We compare the developed heuristics with the best algorithms of the literature and obtain competitive results.*
keywords: *Vehicle Routing Problem, Pickup and Delivery Problem, Heuristics.*

1. Introduction

The main reasons to study Vehicle Routing Problems are: The high costs of transportation, which also affect product costs. The environmental concern, which has lead to an increase of effort to protect the environment, both by industry and legislative bodies. The problem difficulty itself, vehicle routing problems are NP-Hard, as they are generalizations of the Traveling Salesman Problem (Parragh et al., 2008; Lenstra and Kan, 1981).

This work addresses the Vehicle Routing Problem with Simultaneous Pickup and Delivery. We develop three heuristics based on many metaheuristics. These heuristics are composed of two phases. In the first phase, a good initial solution is found with the Greedy Randomized Adaptive Search Procedure (GRASP). In the second phase, this solution is used as a starting point to an Iterated Local Search (ILS). Local search is performed with the Variable Neighborhood Descent (VND). These three heuristics differ by the acceptance criterion on ILS, the first uses a simple acceptance criterion, the second uses a Simulated Annealing (SA) criterion and the third employs ideas of the Guided Local Search (GLS).

The text is organized in the following way. Section 2 introduces the addressed problem. Section 3 introduces the developed heuristics and briefly discusses the metaheuristics used. Results are presented in Section 4. Finally, Conclusion is given in Section 5.

2. The Vehicle Routing Problem with Simultaneous Pickup and Delivery

The Vehicle Routing Problem with Simultaneous Pickup and Delivery (VRPSPD) is defined over a complete graph $G = (V, E)$. There are n_c customers, represented by vertices 1 to $n_c = |V| - 1$, a depot, represented by vertex 0, and a set of vehicles with load capacity Q. Each customer i is associated to a delivery quantity d_i and a pickup quantity p_i. A cost c_{ij} is associated to each edge $e = (i, j) \in E$. We denote as route a sequence of customers $r = \{i_0, i_1, \ldots, i_{n_r-1}\}$ that starts and ends at the depot (i.e. $i_0 = i_{n_r-1} = 0$), where $n_r = |r|$, and uses only edges of E (i.e. $(i_k, i_{k+1}) \in E$, $0 \leq k < n_r - 1$). A feasible route is a route r such that the maximum load of the vehicle is not violated (i.e. $\sum_{0<k<n_r-1} d_{i_k} \leq Q$, $\sum_{0<k<n_r-1} p_{i_k} \leq Q$, and $\sum_{0<j\leq k} p_{i_j} + \sum_{k<j<n_r-1} d_{i_j} \leq Q$, $0 < k < n_r - 1$), and no customer is visited more than once (i.e. $i_j \neq i_k$, $0 < j, k < n_r - 1$, $j \neq k$). The cost c_r of a route is given by the sum of the edge costs connecting its customers (i.e. $c_r = \sum_{j<n_r-1} c_{i_j i_{j+1}}$). The problem consists on finding a set R of feasible routes of minimum cost, such that each customer is assigned to exactly one route (i.e. $i \neq j$; $i, j \neq 0$; $i \in r'$; $j \in r''$; $r', r'' \in R$; $r' \neq r''$).

The Traveling Salesman Problem with Simultaneous Deliveries and Pickups denotes the situation in which there is only one vehicle (see (Gendreau et al., 1999)).

If the pickup amount on each customer is less than or equal the delivery amount, the problem is reduced to the Vehicle Routing Problem, which is NP-hard, what suggests that so is the VRPSPD (Dell'amico and Righini, 2005).

We now introduce an interesting property related to the VRPSPD:

Property 1. Let I and I' be instances of the VRPSPD defined over the same graph, edge costs, and vehicle capacity, but with pickup and delivery quantities on each customer exchanged ($d'_i = p_i, p'_i = d_i$). Given a feasible solution s to I, the solution s', with the routes of s traversed in opposite direction, is feasible to I'.

Proof. Suppose a feasible route r in s with n_r edges and n_r customers (including the depot). Lets consider $n_r - 1$ situations in this route, where situation k corresponds to the load of the vehicle right after serving customer i_{k-1}, $0 < k < n_r - 1$. Since the route is feasible, all these situations are feasible. In r, the vehicle leaves the depot with load D_r (situation 1) and returns with load P_r (situation $n_r - 1$). In the route r' in s', corresponding to r in reverse direction, the vehicle will leave with load P_r (situation $n_r - 1$) since it will have to deliver the entire pickup load of r. When the vehicle gets to its first customer, which is element $n_r - 2$ in r, it will deliver what was picked-up , and pick-up what was deliverd in r, thus, returning to situation $n_r - 2$. When it proceeds to element $(n_r - 3)$, the same thing will happen, and it will get to situation $n_r - 3$. The vehicle will continue to traverse the customers in reverse order, undoing operations that were done in r, repeating situations from $n_r - 1$ to 1, which were all feasible. Thus, we conclude that r' is feasible. □

This result shows that to any solution to I there is an equivalent solution, with the same cost, but in reverse direction, to I'. In the instances of Salhi and Nagy (1999), widely adopted in the literature, instances of type X and Y express this situation, they are identical but with pickups and deliveries exchanged in each customer. However, most heuristics in the literature produce a result to instance of type X (Y), and has trouble finding the same result in Y (X), what suggests that if we have a problem to be solved, our heuristic may be able to find a better result if we invert the demands of pickup and delivery of the problem.

3. Developed Heuristics

In this section we introduce the developed heuristics. The main component of these heuristics is the ILS method. GRASP is used to generate a starting point to the search. VND is used as local search strategy. Ideas of SA and GLS are used to define criteria for accepting candidate solutions. A brief introduction to each of these methods is given as needed.

3.1. GRASP

The Greedy Randomized Adaptive Search Procedure (Feo and Resende, 1995) is a metaheuristic that can be seen as a solution space sampling technique. Each iteration of GRASP consists of a construction phase, where a randomized solution is built, and a local search phase, where the initial solution is improved. This procedure is repeated until certain stopping criterion is met and then the best obtained solution is returned.

3.1.1. Initial Solution

An initial solution of good quality can accelerate the local search process. A *Cluster-First Route-Second* heuristic, proposed in Assis (2007), based on Kruskal's Algorithm, is used to generate initial solutions. This algorithm was chosen due to its simplicity, computational speed and quality of solutions.

Initially, the customers are clustered and then, routes are generated for each cluster. Two conditions can be used during the cluster construction.

1. Cluster customers while the route which visits all delivery customers before the pickup customers in cluster g is feasible. Let M_g^1 be the maximum load in this route.

2. Cluster customers while the route which visits all pickup customers before the delivery customers in cluster g is feasible. Let M_g^2 be the maximum load in this route.

If clusters are created using condition 1, the customers in each resulting cluster must be routed using an algorithm for the TSPSDP. If condition 2 is used, as M_g^2 is the maximum possible load in any routing over the customers of g, as, by the construction process, this load is feasible, any algorithm for the TSP can be used.

The heuristic is based on Kruskal's algorithm for Minimum Spanning Trees. Initially, each customer represents a cluster. Then, the edges of the graph are analyzed in non decreasing order and if the customers of the current edge belong to different clusters and the condition being used is satisfied, the clusters are merged. We modify the algorithm slightly to generate randomized solutions, before the clustering phase, the edge weights are modified with a parameter $\alpha \in [0, 1]$. Algorithm 1 describes the process, $c_{\max}$ and $c_{\min}$ denote the biggest and smallest edge weight, respectively, and M_g^{con} denotes the maximum load on a route over the cluster g according to the condition being used ($con \in \{1, 2\}$).

Algorithm 1 DKRUSKAL

Input: Problem Instance
Output: Set R of feasible routes

for all $(i, j) \in E$ **do**
  $a =$ random number in the interval $[-\alpha(c_{\max} - c_{\min}), \alpha(c_{\max} - c_{\min})]$
  $\hat{c}_{ij} = \max(0, c_{ij} + a)$
end for
Create a new set E' with the edges in E ordered in non decreasing order according to the modified costs $\hat{c}_{ij}$
Create a set $G = \emptyset$ of clusters
For each customer $i \in V$, create a cluster g_i containing it and add g_i to G
for each $(i, j) \in E'$ (in sequence) **do**
  if $g_i \neq g_j$ and $M_{g_i \cup g_j}^{con} \leq Q$ **then**
    $G = G \setminus \{g_i, g_j\}$
    Unite g_i and g_j and add the resulting cluster to G
  end if
end for
Create a set $R = \emptyset$ of routes
for each $g \in G$ **do**
  Create a feasible route r with the customer of g
  $R = R \cup \{r\}$
end for
return R

Our implementation uses condition 2 and the Nearest Neighbor Heuristic for the TSP for the creation of routes for each cluster.

3.1.2. Local Search

The Variable Neighborhood Descent (Hansen and Mladenovic, 2003) is used for local search on GRASP. VND is a metaheuristic in which the basic idea consists on the com-

bination of local search (*Descents*) in many neighborhoods, based on the fact that a local minimum in a neighborhood is not necessarily a local minimum in another neighborhood. Hence, VND iterates through a list of neighborhoods, whenever a better solution is found, it is taken as current solution and the method goes back to the first neighborhood. At the end of the procedure, VND reaches a solution that is a local minimum on all neighborhoods used, and consequently, with good chances of being a good solution.

VND is a variant of the strategy *Variable Neighborhood Search* (VNS), which the basic idea the systematic change of neighborhoods, and from which many other meta-heuristics are derived.

The following neighborhoods are used in VND. They are used in the same order they are presented below.

1. *Route Inversion*. Consists of solutions in which customers are served in reverse order. This neighborhood is used only in the attempt to reduce the maximum load of the route.
2. *Or-Opt*. A sequence of customers in some route is moved to another position of the same route. First, a local minimum is obtained with a sequence of three customers, then with a sequence of two customers, and finally with only one customer.
3. *2-Opt*. This neighborhood consists of solutions obtained when two non adjacent edges of some route are removed and the route is reconnected by the addition of other two edges.
4. *3-Opt*. Consists of solutions obtained by the removal of three non adjacent edges of some route and replacement of these edges by other three edges such that the route is reconnected.
5. *Crossover*. Consists of solutions obtained by the removal of an edge (i, j) from a route r_1 and of an edge (k, l) from another route r_2, followed by the insertion of the edges (i, l) and (k, j). That is, r_1 and r_2 are replaced by r'_1 and r'_2, where r'_1 is formed by the first part of r_1 (from 0 until vertex i) and by the second part of r_2 (from l until the depot), and r'_2 is formed by the first part of r_2 (from 0 until vertex k) and by the second part of r_1 (from j until the depot).
6. *Shift*(n). This neighborhood consists of solutions obtained when a sequence of n customers of a route is transferred to another route. First, a local minimum is obtained with a sequence of three customers, then with a sequence of two customers, and finally with only one customer.
7. *Swap*(n, m). It is comprised by solutions obtained by the removal of a sequence s_1 of n customers and of a sequence s_2 of size m from two different routes r_1 and r_2, respectively, followed by the insertion of s_1 into some position of r_2 and insertion of s_2 into some position of r_1. Initially, a local minimum in the neighborhood with $n = 2$ and $m = 2$ is achieved, next, local search is performed with $n = 2$ and $m = 1$, and finally, with $n = 1$ and $m = 1$.

At each step of the local search in one of these neighborhoods, the best neighbor becomes the current solution until a local minimum is found.

Basically, this sequence was determined by two criteria. The first is to use intra-route neighborhoods first and inter-route neighborhoods later. Intra-route neighborhoods are less computationally expensive and reorganize the routes after inter-route changes. The second criterion is to use the neighborhoods that work over larger sequences of customers

first, since after a local search has been done in neighborhoods that work over smaller sequences it can be harder to find good neighbors in neighborhoods defined over larger sequences.

3.2. ILS

The basic idea of *Iterated Local Search* (ILS) (Lourenço et al., 2003) is to do a stochastic search in the space of local minima. ILS maintains a current solution and at each step, it finds a new solution by the perturbation of the current solution and local search on the perturbed solution. The new solution may or not be accepted as current solution. The ideal would be this process to generate a biased search through the search space. However, very strong perturbations of the current solution will cause the solution space to be explored in a random way, leading to a random restart algorithm, while weak perturbations can make the search go back to the previous solution and thus only a few new solutions will be explored.

Our starting point to ILS is generated with the GRASP implementation discussed in Section 3.1. The local search function defines the search space to be explored and hence it is crucial to the quality of the solutions. We use as local search function the VND implementation discussed in Section 3.1.2.

The acceptance criterion and perturbations guide the search through the search space. The most common approach would be not to take into account any information about the search history in this process and to accept only solutions of better quality. Nevertheless, more complete mechanisms can lead to more robust search procedures. We use three kinds of perturbation, with a strategy to avoid the method to be trapped into specific solutions, these perturbations are discussed in Section 3.2.1. Three different strategies to exchange current solutions are tested, these strategies are discussed in Section 3.2.2.

3.2.1. Perturbation

For perturbations, random neighbors are alternately selected on the following neighborhoods:

- *H*(p). This neighborhood consists of solutions in which up p to customers of the current solution are in different positions and was proposed in (Oliveira et al., 2006). In perturbation, p elements are randomly removed from the solution and reinserted into some feasible position. In the case it is not possible to reinsert any customer, a new route is created to serve it. The same customer can be considered more than once.
- *Relocate*(p). It is characterized by the solutions in which up to p elements, in each route, are in different positions of the same route. In perturbation, on each route, for p times, a customer is randomly removed and inserted into another feasible position of the same route (if there is one). The same customer can be considered more than once.
- *Swap*(p). Swap$(1, 1)$ neighborhood, discussed in section 3.1.2. We make p random movements in this neighborhood.

The use of random movements in perturbations avoids cycling. Defining a parameter p neither too big nor too small depends on the problem instance. To soften this problem, we used a function $\omega(n)$, which returns an integer i in the interval $[1, n]$ with

probability $(n-i+1)^2/\sum_{j=1}^{n} j$. That is, the probability of choosing i decreases quadratically as its value increases. Intervals $[p_{\min}, p_{\max}]$ were defined for each kind of perturbation. At the perturbation phase, a number p is drawn from this interval as follows, $p = (p_{\min} - 1 + \omega(p_{\max} + 1))$. These intervals are established using percentages of customers on the instance, for example, given an instance of n_c customers, we can use $p_{\min} = 0,1 \times n_c$ and $p_{\min} = 0,5 \times n_c$. With this strategy, different sizes of perturbation will be used, keeping the search from being trapped into a solution for too long and reducing the need to optimize the parameter p according to the instance being solved.

3.2.2. Acceptance Criteria

We tested three kinds of acceptance criteria:

1. *Accept if it is better*. The new solution is accepted in case it has total cost smaller than the current solution.
2. *Accept according to SA*. Simulated Annealing (Kirkpatrick et al., 1983; Henderson and Jacobson, 2003; Fleischer, 1995) is a metaheuristic inspired in cooling processes of certain materials. In SA, the candidate solution s' replaces the current solution s if it has smaller cost, or with probability $\exp^{-\delta/t}$, otherwise, where $\delta = f(s') - f(s)$. Hence, the solution acceptance depends on its quality and on the current temperature t. By accepting worsening solutions, SA escapes local minima. During the algorithm, the temperature is decreased, making it harder to accept worsening solutions.
   In our implementation, to obtain t_0, a solution is generated with $p = p_{\max}$ for each perturbation and t_0 is calculated such that the worst of these solutions has at least 95% of chances of being accepted. The temperature is updated at each ILS iteration, $t = t \times \gamma$, where γ is an algorithm parameter. Different values of γ were tested e are discussed in Section 4.
3. *Accept according to a strategy based on GLS*. Guided Local Search (Voudouris and Tsang, 2003) is a metaheuristic that sits on top of local search algorithms guiding them through the search space. Features of solutions of the problem are selected. Solutions have their cost modified by the penalty of having certain features. When facing a local minimum, new features are penalized. The objective function $f(s)$ is replaced by $h(s) = f(s) + \lambda \sum(p_i I_i(s))$, where p_i is the penalty of feature i, $I_i(s)$ is a binary variable that takes value 1 in case the solution possess feature i and value 0 otherwise, and λ is an algorithm parameter. Initially, all penalties are set to 0. In face of a local minimum s^*, a value $u_i(s^*) = I_i(s^*)c_i/(1+p_i)$, which defines the utility of penalizing every feature i is calculated. The feature with the biggest value u has its penalty increased by one unit. This mechanism penalizes features with high cost in the current solution that were not much penalized during the execution of the algorithm.
   Our idea is to use GLS to guide ILS. The strategy consists on replacing the current solution s by the candidate solution s' if $h(s') \leq h(s)$. That is, in case s' is a solution with good cost and attractive features (not penalized to much). When s' is accepted as current solution, the penalties are updated. Local search and perturbation are done using the simple cost function. With this strategy we try to introduce some memory in the walk of ILS trough the solution space.
   Different values λ of were tested in Section 4.

4. Computational Results

Summarizing what has been discussed in the previous sections: three methods were implemented, all of them use GRASP/VND to find an initial solution, used as a starting point for an ILS/VND search. The three methods differ on the acceptance criterion for the exchange of the current solution during the ILS/VND search. The first method accept only solutions with smaller total distance (we will refer to this as method 1) , the second accepts according to SA (method 2), and the third accepts in case the penalized cost of the new solution is smaller than the penalized cost of the current solution according to the method GLS (method 3).

The instances proposed by Salhi and Nagy (1999) and by Dethloff (2001) are widely adopted in the literature and were used in our computational experiments. The instances of Salhi and Nagy are generated from the instances of Christofides et al. (1979), which comprise 14 problems containing from 50 to 199 customers. Two instances, called X and Y, are generated from each instance of Christofides. For each customer i, with delivery request $\hat{d}_i$, in an instance of Christofides, a value $r_i = \min\{x_i/y_i, y_i/x_i\}$ is calculated, where x_i and y_i are the coordinates of the customer, then, a delivery request $d_i = r_i\hat{d}_i$ and pickup request $p_i = (1 - r_i)\hat{d}_i$ are generated. This is the instance of the X type. The instance of the Y type is generated by shifting the pickup and delivery requests (Property 1 shows that X and Y are equivalent). Some of these instances possess distance constraints, these instances were not used.

The instances of Dethloff comprise 40 problems of 50 customers each, and are generated stochastically. In instances of the SCA type, the coordinates of the customers are uniformly distributed in the interval $[0, 100]$. In instances of the CON type, half of the customers are distributed as in SCA and the other half is distributed in the interval $[100/3, 200/3]$, producing a more urban configuration, according to the author. The delivery request d_i of each customer i is uniformly distributed in the interval $[0, 100]$ and the corresponding pickup request p_i is given by $(0.5 + r_i)/d_i$, where r_i is uniformly distributed in the interval $[0, 1]$.The vehicle capacity is given by $Q = \sum_{i \in V, i \neq 0} d_i/\mu$, where μ has value 3 or 8 (this value appears after the letters in the name of the instance).

At each execution of the algorithm based on Kruskal's, a randomization parameter $\alpha = \omega(10)/30$ is used, where $\omega(n)$ returns a number between 1 and n, as stated in Section 3.2.1. Therefore, values $\{0,0333; 0,0666; 0,1; \ldots; 0,3\}$ are used, with smaller probability to bigger numbers (quadratically decreases as its index increases).

We used perturbation sizes in the intervals $[0,1 \times n_c; 0,3 \times n_c]$ for H, $[0,1 \times n_c/n_r; 0,5 \times n_c/n_r]$ for Relocate, and $[0,05 \times n_c; 0,15 \times n_c]$ for Swap. Where n_c is the number of customers and n_r is the number of routes in the solution. Values are drawn from these intervals as described in Section 3.2.1.

For the choice of parameter γ, used in the update of the SA temperature, preliminary tests were performed with values between $0,9$ and $0,99$. To choose parameter λ, used as to weight penalties in the GLS method, preliminary tests were performed with values in the set $\{0,5; 1,0; 1,5; \ldots; 5\}$. The programs were ran 10 times with each parameter and instance. Values $\gamma = 0,92$ and $\lambda = 1,5$ obtained the best results and are used in the experiments that followed.

Each method was executed 100 times with each instance. At each execution, 100

GRASP/VND iterations and 500 ILS/VND iterations were performed.

Tables 1 and 2 presents our best results and the current best results in the literature, for the instances of Salhi and Nagy (1999) and instances of Dethloff (2001), respectively, considering the total distance. For each of our implementations, the total distance (column TD), number of vehicles in the solution (column NV), and gap to the literature best are presented (column Gap). The gap was calculated as $(TD - TD_{best}) * 100/TD_{best}$, where TD is the obtained result and TD_{best} is the best known result. For each instance of Salhi and Nagy, we present the best result between the instance of type X and the instance of type Y, since by property 1, given a result for one of them, there will be an equivalent result to the other.

	Lit.		Met. 1			Met. 2			Met. 3		
Inst.	TD	NV	TD	NV	Gap	TD	NV	Gap	TD	NV	Gap
cmt1X/Y	458,96	3	466,77	3	1,7	466,77	3	1,7	466,77	3	1,7
cmt2X/Y	663,25	6	684,21	6	3,16	684,21	6	3,16	684,21	6	3,16
cmt3X/Y	721,27	5	721,27	5	0	721,27	5	0	721,27	5	0
cmt4X/Y	852,35	7	852,46	7	0,01	852,46	7	0,01	852,83	7	0,05
cmt5X/Y	1030,55	10	**1030,27**	10	-0,02	1032,59	10	0,19	**1029,8**	10	-0,07
cmt11X/Y	830,39	4	836,22	4	0,7	840,77	4	1,25	837,07	4	0,8
cmt12X/Y	644,7	5	662,22	5	2,71	662,22	5	2,71	662,22	5	2,71
Média	743,06	5,71	750,48	5,71	1,18	751,47	5,71	1,28	750,59	5,71	1,19

Table 1. Results for the instances of Salhi and Nagy Salhi and Nagy (1999).

To our knowledge, the best results for the instances of Salhi and Nagy (1999) are found in the works of:

- Wassan et al. (2008): cmt1, cmt2, cmt6, and cmt7.
- Zachariadis et al. (2009): cmt3 and cmt5.
- Subramanian et al. (2009): cmt5.

The best results for the instances of Dethloff (2001) are found in the works of:

- Subramanian et al. (2009): Obtains the best result in all instances, except CON8-9.
- Montané and Galvão (2006): SCA3-1, SCA3-2, SCA3-3, SCA3-4, SCA3-5, SCA3-6, SCA3-7, SCA3-8, SCA3-9, SCA8-3, CON3-1, CON3-3, CON3-5, CON3-8, CON8-1, CON8-3, CON8-4, and CON8-6.
- Ropke and Pisinger (2006): SCA3-1, SCA3-2, SCA3-3, SCA3-4, SCA3-5, SCA3-6, SCA3-8, SCA3-9, SCA8-2, SCA8-4, SCA8-5, SCA8-8, SCA8-9, CON3-0, CON3-1, CON3-3, CON3-4 ,CON3-5, CON3-6, CON3-7, CON3-8, CON3-9, CON8-0, CON8-1 , CON8-3, and CON8-4.
- Zachariadis et al. (2009): SCA3-1, SCA3-2, SCA3-3, SCA3-4, SCA3-5, SCA3-6, SCA3-7, SCA3-8, SCA3-9, SCA8-0, SCA8-2, SCA8-3, SCA8-4, SCA8-5, SCA8-6, SCA8-8, SCA8-9, CON3-0, CON3-1, CON3-3, CON3-5, CON3-7, CON3-8, CON8-0, CON8-1, CON8-3, CON8-4, CON8-6, CON8-7, and CON8-8.
- Chen et al. (2007): CON8-9.

Analyzing table 1, we see that the proposed methods performed similar to each other. The average gap to the literature best was close to 1%. The third method found a new best result for instance cmt5X/Y (method 1 also found a result better than the previous known one, but it is a little worse than the one found by method 3).

By analyzing table 2, it can be seen that the proposed methods obtained good results. Almost all of the best know results were achieved. Method 1 didn't find the best result in 3 of the 40 instances (SCA8-2, SCA8-7, and CON8-9). Method 2 didn't find the best result in only 1 instance (CON8-8). Method 3 didn't find the best result in 2 instances (SCA8-2 e CON8-9). Average gap was very small, close to 0,01%.

Inst.	Lit.		Met. 1			Met. 2			Met. 3		
	TD	NV	TD	NV	Gap	TD	NV	Gap	TD	NV	Gap
SCA3-0	635,62	4	635,62	4	0	635,62	4	0	635,62	4	0
SCA3-1	697,84	4	697,83	4	0	697,83	4	0	697,83	4	0
SCA3-2	659,34	4	659,33	4	0	659,33	4	0	659,33	4	0
SCA3-3	680,04	4	680,04	4	0	680,04	4	0	680,04	4	0
SCA3-4	690,5	4	690,5	4	0	690,5	4	0	690,5	4	0
SCA3-5	659,9	4	659,9	4	0	659,9	4	0	659,9	4	0
SCA3-6	651,09	4	651,08	4	0	651,08	4	0	651,08	4	0
SCA3-7	659,17	4	659,16	4	0	659,16	4	0	659,16	4	0
SCA3-8	719,47	4	719,47	4	0	719,47	4	0	719,47	4	0
SCA3-9	681	4	680,99	4	0	680,99	4	0	680,99	4	0
SCA8-0	961,5	9	961,49	9	0	961,49	9	0	961,49	9	0
SCA8-1	1049,65	9	1049,65	9	0	1049,65	9	0	1049,65	9	0
SCA8-2	1039,64	9	1039,71	9	0	1039,64	9	0	1044,48	9	0,46
SCA8-3	983,34	9	983,33	9	0	983,33	9	0	983,33	9	0
SCA8-4	1065,49	9	1065,49	9	0	1065,49	9	0	1065,49	9	0
SCA8-5	1027,08	9	1027,08	9	0	1027,08	9	0	1027,08	9	0
SCA8-6	971,82	9	971,82	9	0	971,82	9	0	971,82	9	0
SCA8-7	1051,28	9	1053,84	9	0,24	1051,28	9	0	1051,28	9	0
SCA8-8	1071,18	9	1071,18	9	0	1071,18	9	0	1071,18	9	0
SCA8-9	1060,5	9	1060,5	9	0	1060,5	9	0	1060,5	9	0
CON3-0	616,52	4	616,52	4	0	616,52	4	0	616,52	4	0
CON3-1	554,47	4	554,47	4	0	554,47	4	0	554,47	4	0
CON3-2	518	4	518	4	0	518	4	0	518	4	0
CON3-3	591,19	4	591,18	4	0	591,18	4	0	591,18	4	0
CON3-4	588,79	4	588,79	4	0	588,79	4	0	588,79	4	0
CON3-5	563,7	4	563,69	4	0	563,69	4	0	563,69	4	0
CON3-6	499,05	4	499,05	4	0	499,05	4	0	499,05	4	0
CON3-7	576,48	4	576,48	4	0	576,48	4	0	576,48	4	0
CON3-8	523,05	4	523,05	4	0	523,05	4	0	523,05	4	0
CON3-9	578,24	4	578,24	4	0	578,24	4	0	578,24	4	0
CON8-0	857,17	9	857,17	9	0	857,17	9	0	857,17	9	0
CON8-1	740,85	9	740,85	9	0	740,85	9	0	740,85	9	0
CON8-2	712,89	9	712,88	9	0	712,88	9	0	712,88	9	0
CON8-3	811,07	10	811,06	10	0	811,06	10	0	811,06	10	0
CON8-4	772,25	9	772,25	9	0	772,25	9	0	772,25	9	0
CON8-5	754,88	9	754,88	9	0	754,88	9	0	754,88	9	0
CON8-6	678,92	9	678,92	9	0	678,92	9	0	678,92	9	0
CON8-7	811,96	9	811,95	9	0	811,95	9	0	811,95	9	0
CON8-8	767,53	9	767,52	9	0	767,52	9	0	767,52	9	0
CON8-9	806,72	-	809	9	0,28	809	9	0,28	809	9	0,28
Média	758,47	6,46	758,59	6,52	0,01	758,53	6,52	0	758,65	6,52	0,01

Table 2. Results for the instances of Dethloff Dethloff (2001).

5. Conclusion

In this work, we proposed three heuristics based on metaheuristics to the Vehicle Routing Problem with Simultaneous Pickup and Delivery. The three methods are ILS/VND searches, with GRASP/VND to generate the starting point. The difference between them is the criterion to accept new solutions. The first has a simple criterion, accept if the new solution is better. The second accepts according to SA. The third does it according to a strategy similar to the GLS method. Several of the best literature results were reached by the best results of the heuristics. Considering average solutions, the gaps in relation to the best results of the literature were small. On average, the three methods performed similarly. We associate this fact to the quality of the VND local search. However, the GLS strategy led the third method to improve the literature's best result in one of the Salhi and Nagy instances.

References

L. P. Assis. Algoritmos para o problema de roteamento de veículos com coleta e entrega simultâneas. Master's thesis, Universidade Federal de Minas Gerais, 2007.

P. Chen, H. Huang, and X. Dong. An ant colony system based heuristic algorithm for the vehicle routing problem with simultaneous delivery and pickup. In *2nd IEEE Conference on Industrial Electronics and Applications*, 2007.

N. Christofides, A. Mingozzi, and P. Toth. *Combinatorial Optimization*, chapter The Vehicle Routing Problem, pages 315–338. Wiley, 1979.

M. Dell'amico and G. Righini. A branch-and-price approach to the vehicle routing problem with simultaneous distribution and collection. *Transportation Science*, 40(2):235–247, 2005.

J. Dethloff. Vehicle routing and reverse logistics: The vehicle routing problem with simultaneous delivery and pick-up. *OR Spectrum*, 23(1):79–96, 2001.

T. A. Feo and M. G. C. Resende. Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 6:109–134, 1995.

M. Fleischer. Simulated annealing: past, present, and future. In *Winter Simulation Conference*, 1995.

M. Gendreau, G. Laporte, and D. Vigo. Heuristics for the traveling salesman problem with pickup and delivery. *Computers & Operations Research*, 26(7):699–714, 1999.

P. Hansen and N. Mladenovic. A tutorial on variable neighborhood search. Technical report, Les Cahiers du GERAD, HEC Montreal and GERAD, 2003.

D. Henderson and S. H. Jacobson. *Handbook of Metaheuristics*, chapter The Theory and Pratice of Simulated Annealing, pages 287–319. Kluwer Academic Publishers, 2003.

S. Kirkpatrick, C. D. Gelatt Jr., and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

J. K. Lenstra and A. H. G. Rinnooy Kan. Complexity of vehicle routing and scheduling problems. *Networks*, 11(2):221–227, 1981.

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

F. A. T. Montané and R. D. Galvão. A tabu search algorithm for the vehicle routing problem with simultaneous pick-up and delivery service. *Computers & Operations Research*, 33 (3):595–619, 2006.

H. C. B. Oliveira, G. C. Vasconcelos, and G. B. Alvarenga. Reducing traveled distance in the vehicle routing problem with time windows using a multi-start simulated annealing. In *International Joint Conference on Neural Networks*, 2006.

S. N. Parragh, K. F. Doerner, and R. F. Hartl. A survey on pickup and delivery problems. part I: Transportation between customers and depot. *Journal für Betriebswirtschaft*, 58 (1):21–51, 2008.

S. Ropke and D. Pisinger. An unified heuristic for a large class of vehicle routing problems with backhauls. *European Journal of Operational Research*, 171(6):750–775, 2006.

S. Salhi and G. Nagy. A cluster insertion heuristic for single and multiple depot vehicle routing problems with backhauling. *Journal of the Operational Research Society*, 50: 1034–1042, 1999.

A. Subramanian, L. S. Ochi, and L. A. F. Cabral. An efficient ILS heuristic for the vehicle routing problem with simultaneous pickup and delivery. Technical report, Universidade Federal Fluminense, Universidade Federal da Paraíba, 2009.

C. Voudouris and E. P. K. Tsang. *Handbook of Metaheuristics*, chapter Guided Local Search, pages 186–218. Kluwer Academic Publishers, 2003.

N. A. Wassan, A. H. Wassan, and G. Nagy. A reactive tabu search algorithm for the vehicle routing problem with simultaneous pickups and deliveries. *Journal of Combinatorial Optimization*, 4(4):368–386, 2008.

E. E. Zachariadis, C. D. Tarantilis, and C. T. Kiranoudis. A hybrid metaheuristic for the vehicle routing problem with simultaneous delivery and pick-up service. *Expert Systems with Applications*, 36(2):1070–1081, 2009.