Clustering-based Search in a Memetic Algorithm for the Vehicle Routing Problem with Time Windows

Daniel Bustos Coral
Institute of Mathematics and Computer Science
University of São Paulo
São Carlos, SP, Brazil
daniel.bc@usp.br

Maristela Oliveira Santos
Institute of Mathematics and Computer Science
University of São Paulo
São Carlos, SP, Brazil
mari@icmc.usp.br

Claudio Toledo
Institute of Mathematics and Computer Science
University of São Paulo
São Carlos, SP, Brazil
claudio@icmc.usp.br

Luis Fernando Niño
Department of Systems and Industrial Engineering
National University of Colombia
Bogotá, Colombia
lfninov@unal.edu.co

***Abstract*—This paper addresses the vehicle routing problem with time windows (VRPTW), aiming to minimize the total travel time. A simple memetic algorithm (MA) is proposed for solving this problem. At the beginning of the search, a clustering procedure is applied to customers' spatial information. The search procedure consists of relocating customers between close routes, seeking to minimize detour costs associated with the relocations. The information gathered by the clustering procedure is used to identify which routes lie close to each other. Computational experiments on the Solomon's benchmark set show the effectiveness of the proposed approach, which produces competitive solutions and outperforms four out of six solution approaches considered for comparison regarding the travel cost attained over all the instances of the benchmark set.**

***Index Terms*—VRPTW, memetic algorithm, clustering**

I. Introduction

The vehicle routing problem with time windows (VRPTW) consists of determining a minimum-cost routing plan for a fleet of vehicles to visit a set of geographically scattered customers requiring goods to be either delivered or picked up. The routes should be planned in such a way that the vehicle capacities are not exceeded and all customers are visited within their period of availability, called a *time window*.

As remarked by Desaulniers, Madsen, and Ropke [1], the VRPTW has drawn the attention of academics and practitioners because of its practical applications and its difficulty. Several exact and heuristic methods for the VRPTW have been proposed. Heuristic and metaheuristic approaches have been shown to be successful in tackling this problem, since these methods can find good-quality solutions in short times.

Several objectives can be considered in the VRPTW. Traditionally, the minimization of the travel distance (or time) has been the objective of exact methods, whereas the heuristic and metaheuristic approaches have adopted a hierarchical objective, consisting of minimizing the number of vehicles used as their primary objective, and minimizing the total travel distance as their second objective [1]. Some heuristics that do not follow this pattern and seek to minimize the total travel distance as their primary objective are studied in [2], [3], [4], and [5].

As pointed out by Alvarenga, Mateus, and Tomi [3], the selection of the problem's objective depends largely on the characteristics of the situation being considered. In some cases, the minimization of the total distance traveled by the vehicles is more suitable, for example when the quantity of products to be transported is smaller than the total load capacity of the fleet. If the load capacity of the fleet is just barely big enough to satisfy the customers' demand, the minimization of the number of vehicles would be the main objective.

In this paper, the VRPTW is addressed seeking to minimize the total travel time. To this aim, a simple yet effective memetic algorithm (MA) is developed. A clustering procedure is applied to customers' spatial information at the beginning of the MA execution. The crossover and local search operators take advantage of the information gathered by the clustering procedure. Both operators relocate customers between routes that pass close to each other, seeking to minimize the detour costs associated with the modifications. The clusters "guide" the search since they are used to identify which routes pass close to each other. The crossover operator used in this work was first introduced in [6]. This paper presents a local search operator that is used to improve the quality of the solutions obtained from the crossover process.

Experiments conducted on the Solomon's benchmark set [7] show a competitive performance of the proposed MA. The proposed MA outperforms four out of six solution approaches considered for comparison (when comparing the total cumulative travel distance/time), and it finds the best-known solutions for several instances.

The main contributions of this paper are (1) the introduction of a local search procedure that uses clusters of customers to guide the exploration of the search space; and (2) computational experiments that show that a search based solely on relocating customers between close routes can lead to high-quality solutions.

The remainder of this paper is organized as follows: Section II presents a definition of the problem. Section III presents the problem-solving methodology. Section IV reports the computational results. Conclusions are provided in Section V.

II. Problem Definition

The VRPTW consists of determining a set of routes for distributing goods from a single depot to a set of n geograph-

ically scattered customers. The problem can be defined on a complete directed graph $G = (V, E)$. The set of nodes is given by $V = \{0, 1, \ldots, n\}$. Node 0 represents the depot, whereas $N = \{1, \ldots, n\}$ represents the set of customer nodes. Each node $i \in N$ is associated with a demand $q_i > 0$, a service time $s_i > 0$, and a time window $[e_i, l_i]$. The depot has no demand or service time, i.e., $q_0 = 0$ and $s_0 = 0$. The depot's time window represents the planning horizon and is given by $[e_0, l_0]$, where e_0 and l_0 represent the earliest possible departure time from the depot and the latest possible arrival time at the depot, respectively. The set of edges is given by $E = \{(i, j) | i, j \in V, i \neq j\}$. Each edge (i, j) is associated with a distance $d_{ij} \geq 0$ and a travel time $t_{ij} \geq 0$. A fleet of M vehicles is available for serving the customers. The fleet is assumed to be homogeneous, since all vehicles have the same capacity Q, and they operate at the same cost.

A route is defined by a sequence of nodes $r^v = \langle r_0^v, r_1^v, \ldots, r_{n_v}^v, r_{n_v+1}^v \rangle$, where v is the vehicle responsible for traversing the route and r_k^v represents the kth node visited by vehicle v. It should be noted that a vehicle can perform only one route over the planning horizon. Given that every route starts and ends at the depot, $r_0^v = r_{n_v+1}^v = 0$. For $1 \leq k \leq n_v$, $r_k^v \in N$ and all r_k^v are distinct, since all customers are visited at most once.

The travel time of a route r^v is given by $f(r^v) = \sum_{k=0}^{n_v} t_{r_k^v, r_{k+1}^v}$. Note that the travel time of a route is different from the total route time, that is the sum of total travel time, total waiting time, and total service time.

The start of service time $T_{r_k^v}$ at node r_k^v is given by the following recursive equation [1]:

$$T_{r_k^v} = \begin{cases} e_0 & \text{if } k = 0 \\ \max\left\{e_{r_k^v}, T_{r_{k-1}^v} + s_{r_{k-1}^v} + t_{r_{k-1}^v, r_k^v}\right\} & \text{otherwise.} \end{cases} \quad (1)$$

If vehicle v arrives at node r_k^v before $e_{r_k^v}$, it should wait to start the service. Arriving after $l_{r_k^v}$ is not allowed. Formally, the route r^v is feasible regarding the time-window constraints if $T_{r_k^v} \leq l_{r_k^v}$ for $k = 1, \ldots, n_v + 1$. The vehicle-capacity constraint is satisfied if $\sum_{k=1}^{n_v} q_{r_k^v} \leq \mathrm{Q}$. If both time-window and vehicle-capacity constraints are satisfied, the route r^v is said to be feasible.

Violations of time-window and vehicle-capacity constraints must be calculated to penalize infeasible routes. For a route r^v, time-window violations are given by $f_{tw}(r^v) = \sum_{k=1}^{n_v+1} \max\left\{T_{r_k^v} - l_{r_k^v}, 0\right\}$. Vehicle-capacity violations are given by $f_q(r^v) = \max\left\{\sum_{k=1}^{n_v} q_{r_k^v} - \mathrm{Q}, 0\right\}$.

A solution for the VRPTW is defined as a set of m routes $S = \left\{r^1, \ldots, r^m\right\}$ where each customer is served exactly once and $m \leq \mathrm{M}$.

The objective function to be minimized is

$$F(S) = \sum_{v=1}^{m} \left[f(r^v) + \alpha_1 f_{tw}(r^v) + \alpha_2 f_q(r^v)\right]. \quad (2)$$

Function (2) calculates the total travel time taken by the vehicles for serving all customers; α_1 and α_2 are weights used to penalize the violations to time-window and vehicle-capacity constraints, respectively. A solution is said to be feasible if $f_{tw}(r^v) = f_q(r^v) = 0$, for $v = 1, \ldots, m$.

The objective of the problem is to find a feasible solution S^* that minimizes $F(S)$.

III. Memetic Algorithm

The proposed MA performs a clustering analysis phase where the customers are clustered based on their spatial distance. The clusters obtained from that process have two purposes: they are used to create some initial solutions, and more importantly, they serve as a guide for the search operators of the MA. The details of the clustering process and the search methodology are presented below.

A. Solution Representation

Given that a solution S for a VRPTW instance is a set of routes, a solution in the MA is represented as a set of lists, where each list represents a route. Customers are visited according to the order in which their identifiers appear within the list. Since every route starts and ends at the depot, the identifier of the depot, 0, is not considered in this representation.

B. Objective Function

The fitness of a solution is calculated using Function (2).

C. Generation of the Initial Population

The initial population of the MA is composed of two kinds of solutions. Solutions of *kind 1* are created by performing hierarchical agglomerative clustering (HAC) on the matrix of distances between customers. To create a solution, the dendrogram obtained from the clustering process is cut at one point, so that a clustering arrangement is obtained. The solution is put together by defining a route on each cluster and letting the visit order be random. Several clustering arrangements, with different numbers of clusters, are obtained by cutting the dendrogram at different points (see [8] for further details). The number of clusters created varies between two values: C_{min} and C_{max}. The aim of this approach is to define routes composed of close customers. These solutions do not necessarily satisfy the time-window and vehicle-capacity constraints.

In addition to being used to create solutions of *kind 1*, the clustering arrangements generated are saved in the set A, which is used afterward by the search operators of the MA.

Solutions of *kind 2* are created by forming routes with customers selected randomly, checking that the vehicle-capacity constraint is not violated. These solutions are created to increase the diversity of the initial population. Solutions of *kind 1* and *kind 2* are generated with probabilities ρ and $(1-\rho)$, respectively.

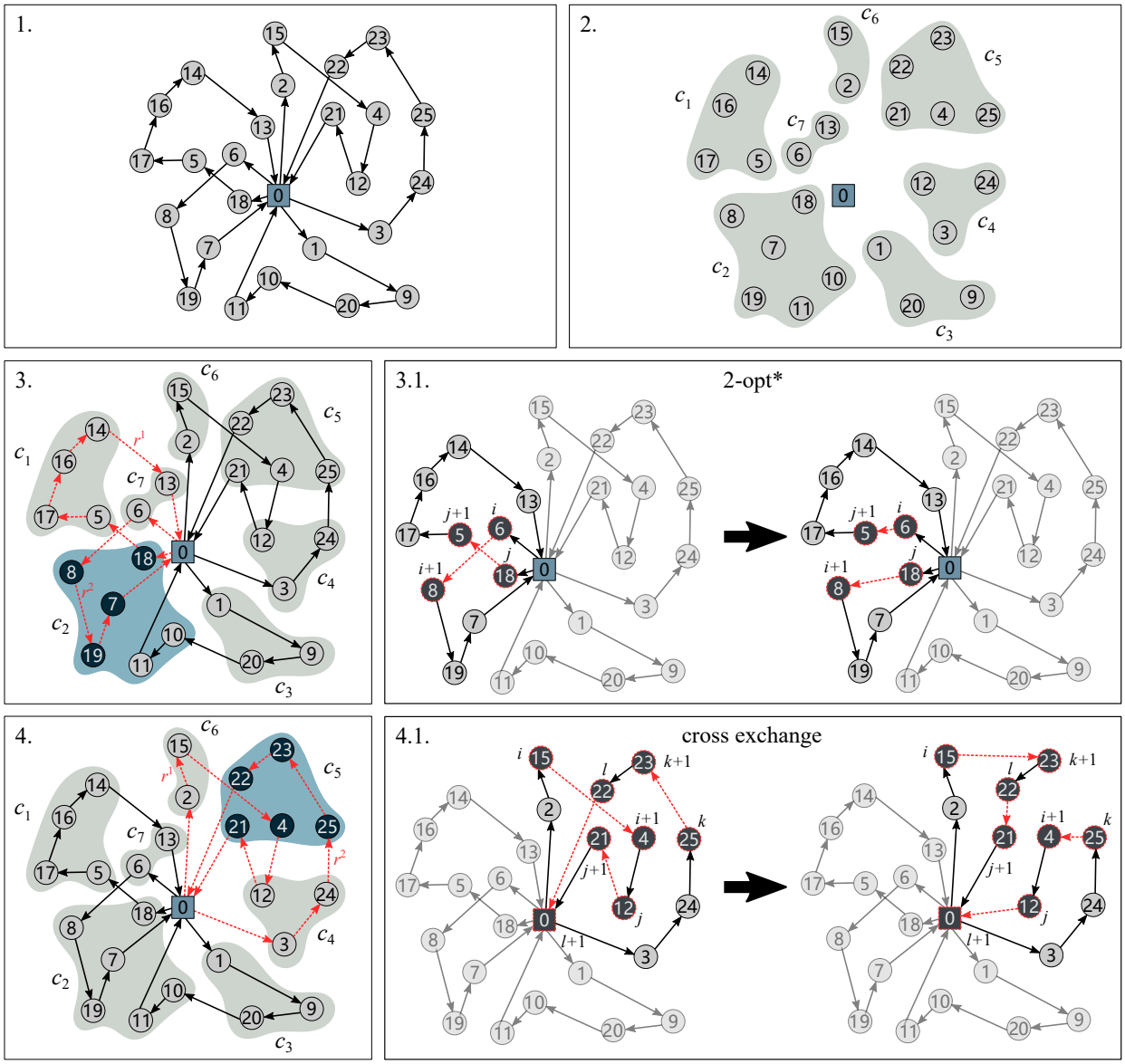

Figure 1. Example: Inter-route neighborhood search.

D. Crossover

The crossover operator is responsible for the exploration of the search space; it takes two solutions and combines the information encoded by them to create a new solution. The proposed MA uses the clustering-based best cost route crossover (CB-BCRC) [6], which is a modification of the best cost route crossover (BCRC) [9]. Both crossover operators follow a similar process to build a solution: Given two solutions S_1 and S_2, a route is selected randomly from S_1. The customers belonging to the selected route are removed and reinserted in S_2, considering that each missing customer must be reinserted at a feasible location that yields the minimum detour cost. Once the reinsertion process is completed, a new solution is obtained. In the case of the CB-BCRC, clustering arrangements are used to identify the most promising routes for reinserting the missing customers, thus avoiding performing the exhaustive search the BCRC does. In the proposed MA, the CB-BCRC uses the clustering arrangements contained in set A. Details on the CB-BCRC operation can be found in [6].

E. Local Search

The local search operator is responsible for performing an exploitative search that seeks to refine the solutions found by the crossover operator. Two kinds of neighborhoods are considered within the search: *inter-route* neighborhoods, where solutions are obtained by moving one or more customers between routes; and *intra-route* neighborhoods, where solutions are obtained by modifying a single route. The clustering arrangements saved in set A are used for guiding the search through the inter-route neighborhoods. Details on the local search process are given below.

Inter-route neighborhood search. Since the inter-route neighborhoods are composed of solutions obtained by moving customers between two or more routes, a question that must be addressed is which routes are good candidates for performing such moves.

The search procedure begins by identifying a set of routes passing close to each other; this is achieved using the clustering arrangements saved in set A. Once the candidate routes have been identified, traditional inter-route neighborhood moves are applied to modify them (reviews on traditional neighborhoods for the VRP can be found in [1] and [10]).

By only moving customers between routes passing close to each other, the search procedure seeks to minimize the detour costs associated with such moves, thus producing good-quality solutions. On the other hand, a blind exchange of customers between routes passing far apart from each other could lead to low-quality solutions, since the detour costs of the exchange could be high.

Figure 1 shows an example of how the routes are selected and modified. Let S be a solution (Figure 1.1) and let C be a clustering arrangement of customers containing seven clusters (Figure 1.2). The procedure begins by selecting a cluster and identifying the routes passing through it (a route is said to pass through a cluster if both have at least one common customer). The routes found are considered to visit a common region, and thus they pass close to each other. At least two different routes should be found in order to continue. From the routes identified, two are randomly selected. In the example, clusters c_2 and c_5 were chosen (Figure 1.3 and Figure 1.4, respectively), and the routes selected are shown by dashed lines. A new solution is obtained by applying traditional inter-route moves to the selected routes. In the example, the *2-opt** move is applied to the routes found in Figure 1.3.: the arcs $(i, i+1)$ and $(j, j+1)$ are replaced by the arcs $(i, j+1)$ and $(j, i+1)$ (Figure 1.3.1). Similarly, the *Cross Exchange* move is applied to the routes found in Figure 1.4.: the arcs $(i, i+1)$, $(j, j+1)$, $(k, k+1)$, and $(l, l+1)$ are replaced by the arcs $(i, k+1)$, $(l, j+1)$, $(k, i+1)$, and $(j, l+1)$ (Figure 1.4.1). In both cases, the arcs to be modified are selected randomly.

The inter-route neighborhood search consists of applying iteratively the process illustrated in Figure 1. The pseudocode for the clustering-based inter-route neighborhood search (CB-Inter-Route-NS) is presented in Figure 2. The CB-INTER-ROUTE-NS procedure takes as input a solution S, a set of clustering arrangements A, and a neighborhood function $\mathcal{N}$. The latter takes as input a solution S and the indexes of two of its routes, r^1 and r^2, and it creates a new solution by applying an inter-route move to a set of arcs randomly selected from r^1 and r^2. Since the inter-route moves can be applied only to solutions with two or more routes, line 1 checks whether S meets this requirement. If S has only one route, it is returned without changes (line 2). The incumbent is initialized in line 3. A clustering arrangement C is randomly selected from A in line 4. The loop on lines 5-12 performs the search process; basically, it executes the process illustrated in Figure 1 with each cluster $c \in C$. Line 6 finds the routes of the incumbent

```
CB-INTER-ROUTE-NS(S, A, N)
 1  if |S| < 2
 2      return S
 3  S_best = S
 4  C = RANDOM(A)
 5  for each cluster c ∈ C
 6      routes = FIND-ROUTES(S_best, c)
 7      if |routes| < 2
 8          continue
 9      r^1, r^2 = RANDOM(routes)
10      S' = N(S_best, r^1, r^2)
11      if F(S') < F(S_best)
12          S_best = S'
13  return S_best
```

Figure 2. Pseudocode for the clustering-based inter-route neighborhood search procedure.

passing through cluster c. If there are not at least two routes passing through c, the search continues with the next cluster (lines 7 and 8). Optionally, the routes found can be checked for ensuring they meet some criteria, for example having a minimum number of customers. From the routes identified, r^1 and r^2 are randomly selected in line 9. Line 10 creates a neighbor solution S' by applying the function $\mathcal{N}$ to the routes r^1 and r^2 from the incumbent. Lines 11 and 12 update the incumbent when a better solution is found. Finally, line 13 returns the best found solution.

Intra-route neighborhood search. Seeking to further improve the solutions obtained by the CB-INTER-ROUTE-NS procedure, an intra-route neighborhood search procedure is applied to one randomly selected route. In this case, a traditional simulated annealing (SA) heuristic is responsible for performing the search. The SA heuristic randomly applies the intra-route neighborhood moves *2-opt*, *or-opt*, and *swap* to perform the search (for further details on SA and the mentioned neighborhood moves, the reader is referred to [11] and [10], respectively).

Variable neighborhood search procedure. The inter-route and intra-route neighborhood searches are carried out within a variable neighborhood search (VNS) framework [12]. The pseudocode for the VNS procedure is presented in Figure 3. In line 1, the list of neighborhood functions $\mathcal{N}_l$ is initialized. In this case, the neighborhoods considered are *2-opt** and *Cross Exchange*. The incumbent is initialized in line 2. Line 3 sets the neighborhood function to start the search with. Lines 4-10 comprise the search procedure. The length of list $\mathcal{N}_l$ is denoted by $\text{len}(\mathcal{N}_l)$. The INTER-ROUTE-NS procedure in line 5 performs the inter-route neighborhood search. This procedure takes as input the incumbent, the set of clustering arrangements, a neighborhood function k, and a number of iterations I_{max}. It applies the procedure CB-INTER-ROUTE-NS to the incumbent during I_{max} iterations to obtain a new solution S'. This step can be considered a combination of the

```
VNS(S, A, I_max, γ, β)
 1  N_l = ⟨2-opt*, cross-exchange⟩
 2  S_best = S
 3  k = 1
 4  while k ≤ len(N_l)
 5      S' = INTER-ROUTE-NS(S_best, A, N_l[k], I_max)
 6      S' = INTRA-ROUTE-NS(S', γ, β)
 7      if F(S') < F(S_best)
 8          S_best = S'
 9          k = 1
10      else k = k + 1
11  return S_best
```

Figure 3. Pseudocode for the VNS procedure.

shaking and local search steps of the traditional VNS. The intra-route phase of the local search is applied in line 6. The INTRA-ROUTE-NS procedure takes as input a solution S', a temperature value γ, and a cooling rate value β. It applies the SA heuristic to one randomly chosen route from S'. If the updated S' turns out to be a better solution, it becomes the incumbent, and the search process is restarted with the first neighborhood function. If no better solution is found, the algorithm continues the search with the next neighborhood function (lines 7-10). Finally, line 11 returns the best found solution.

F. MA Main Procedure

The pseudocode for the proposed MA is presented in Figure 4. The MA procedure takes as input a VRPTW instance, X; the population size, $\mathcal{L}$; the maximum number of generations, G_{max}; the crossover rate, R_{rate}; the local search rate, L_{rate}; the parameters that control the local search: I_{max}, γ, and β; and the parameters used for creating the initial population of solutions: C_{min}, C_{max}, and ρ. The population of solutions P and a set of clustering arrangements A are created in line 1. The incumbent is initialized in line 2. The loop on lines 3-7 performs the search by updating the population of solutions and the incumbent at each iteration.

The way in which the solutions are updated is presented in Figure 5. The NEW-POP procedure seeks to replace each solution in P with a better one. The procedure begins with the creation of an empty population of solutions Q, which is filled by the loop on lines 2-13. This loop goes through each solution S_j in P and creates an updated version of it in the following manner: First, an additional solution S_o is taken at random from P (line 4). Both solutions S_j and S_o undergo crossover, and a child solution S_c is created (line 7). If the cost of S_c is more favorable than that of S_j, S_c is kept; otherwise S_j becomes S_c (lines 8 and 9). Next, the local search procedure is applied to S_c, and the solution obtained from this process takes the place of S_j in the next generation (lines 12 and 13). Finally, the updated population of solutions is returned in line 14.

```
MA(X, L, G_max, R_rate, L_rate, I_max, γ, β, C_min, C_max, ρ)
1  P, A = INITIAL-POP(X, L, C_min, C_max, ρ)
2  S_best = GET-BEST-SOLUTION(P)
3  for i = 1 to G_max
4      P = NEW-POP(P, A, L, R_rate, L_rate, I_max, γ, β)
5      S' = GET-BEST-SOLUTION(P)
6      if F(S') < F(S_best)
7          S_best = S'
8  return S_best
```

Figure 4. Pseudocode for the MA main procedure.

```
NEW-POP(P, A, L, R_rate, L_rate, I_max, γ, β)
 1  let Q[1..L] be a new array
 2  for j = 1 to L
 3      S_j = P[j]
 4      S_o = RANDOM(P)
 5      let S_c be a new solution
 6      if U(0,1) ≤ R_rate
 7          S_c = CROSSOVER(S_o, S_j, A)
 8          if F(S_c) > F(S_j)
 9              S_c = S_j
10      else S_c = S_j
11      if U(0,1) ≤ L_rate
12          S_c = VNS(S_c, A, I_max, γ, β)
13      Q[j] = S_c
14  return Q
```

Figure 5. Pseudocode for the creation of a new population of solutions.

It is necessary to note that as the search goes on, the MA is expected to deal with multiple locally minimal solutions that will survive several generations until better solutions replace them. For that reason, the intra-route local search operator is applied to only one route. Since a solution can remain for several successive generations, the intra-route search will eventually be performed on several routes of the solution.

IV. COMPUTATIONAL RESULTS

The proposed MA was implemented using the Kotlin programming language. All experiments were conducted on an Intel Core i7-4790U PC with 16 GB memory. The MA parameters were set to the following values: $\mathcal{L} = 100$, $G_{max} = 1000$, $R_{rate} = 0.9$, $L_{rate} = 0.03$, $I_{max} = 50$, $\gamma = 100000$, $\beta = 0.001$, $C_{min} = 2$, $C_{max} = 25$ (the fleet size in the considered instances), $\alpha_1 = 1000$, $\alpha_2 = 1000$, and $\rho = 0.6$. The parameter values were selected after preliminary testing, were a wide range of parameter settings was explored, seeking a balanced trade-off between fast execution times and solution quality. The hierarchical clustering was performed using the unweighted pair group method with arithmetic mean (UPGMA) algorithm.

The well-known Solomon's VRPTW benchmark set [7] was selected for testing the proposed MA. This benchmark

Table I
COMPARATIVE RESULTS: BEST SOLUTIONS FOR EACH DATA SET

Data Set	JU [2]	AL [3]	OL [4]	UR [5]	GN [13]	VI [14]	Proposed MA		
							Best	Average	Cost Gap (%)
C1	828.38	828.38	828.38	828.38	828.38	828.38	828.38	828.38	0.00
	10.00	10.00	10.00	10.00	10.00	-	10.00	10.00	
C2	589.86	589.86	589.86	589.86	589.86	589.86	589.86	589.86	0.00
	3.00	3.00	3.00	3.00	3.00	-	3.00	3.00	
R1	1179.95	1183.38	1186.94	1188.85	1187.32	1178.98	1184.95	1191.39	0.51
	13.25	13.25	13.33	13.5	13.08	-	13.42	13.54	
R2	878.41	899.90	878.79	885.78	897.95	877.2	878.74	881.44	0.18
	5.36	5.55	5.36	5.55	4.00	-	5.36	5.35	
RC1	1343.65	1341.67	1362.44	1360.96	1348.22	1338.18	1351.43	1364.25	0.99
	13.00	12.88	13.25	13.25	12.63	-	13.13	13.26	
RC2	1004.21	1015.90	1004.59	1013.29	1036.65	1003.95	1004.77	1008.01	0.08
	6.25	6.50	6.13	6.88	5.38	-	6.25	6.20	
CTC	54779.02	55134.27	55020.00	55137.04	55378.61	54708	54909.37	55144.87	0.37
CNV	486	489	488	498	459	-	489	491	
runs	100	3	15	30	30	82	10	10	

set includes 56 instances with 100 customers. Depending on the spatial distribution of the customers, time-window characteristics, and vehicle capacity, the instances are classified into classes C1, R1, RC1, C2, R2, and RC2. The customers are clustered in instances of type C and randomly located in instances of type R. In instances of type RC, there is a combination of clustered and randomly located customers. In instances of type 1, the customers have narrow time windows and the vehicle capacity is small, thus more vehicles are required for serving all customers. In instances of type 2, the customers have wider time windows and the vehicle capacity is larger, thus fewer vehicles are needed to serve all customers. The service time is the same for all customers; in instances of type C, the service time is equal to 90, whereas in instances of type R and RC, the service time is equal to 10.

Each instance was solved 10 times, using double-digit precision. The solutions obtained were compared with the results of other heuristics. Since the objective of the proposed MA is the minimization of total travel time, the heuristics selected for comparison were those whose main objective is to minimize the total travel distance or total travel time (travel times are equal to their corresponding distances in Solomon's instances). The heuristics selected were those of Jung and Moon [2] (JU), Alvarenga, Mateus, and Tomi [3] (AL), Oliveira and Vasconcelos [4] (OL), Ursani, Essam, Cornforth, *et al.* [5] (UR), Garcia-Najera and Bullinaria [13] (GN), and Vidal, Crainic, Gendreau, *et al.* [14] (VI). The heuristic of Garcia-Najera and Bullinaria [13] is indeed a multi-objective genetic algorithm; nevertheless, a comparison against this heuristic is fair, since the authors reported the solutions with the best travel distances that their algorithm found. It should also be noted that Vidal, Crainic, Gendreau, *et al.* [14] conducted experiments with two heuristics (HGA and MS-ILS) using different relaxation schemes, and reported the best solutions found during all experiments.

Table I summarizes the results for each Solomon data set following the format commonly adopted in the VRPTW literature. Considering the best solutions found by the heuristics after being run several times on each instance, two values are reported for each data set. The upper value indicates the average travel distance (or travel time), and the lower value indicates the average number of vehicles. The values CTC and CNV indicate the cumulative total travel distance (or travel time) and the cumulative number of vehicles, respectively (Vidal, Crainic, Gendreau, *et al.* [14] did not report the information related to the number of vehicles of their solutions). The number of runs is reported in the last row. For the proposed MA, the values corresponding to the average solution quality are reported as well. The cost gaps between the best results of the cited heuristics and the best results obtained by the proposed MA are reported in the last column of the table. Let x_{tt} be the best travel cost achieved by the cited heuristics and let x^*_{tt} be the best travel cost achieved by the proposed MA. The cost gap is given by $100 \cdot (x^*_{tt} - x_{tt})/x_{tt}$.

The heuristics of Vidal, Crainic, Gendreau, *et al.* [14] produced the best-known results for the Solomon's instances regarding distance minimization. These heuristics achieved the shortest travel distance on all data sets and the shortest cumulative travel distance. The overall performance of the proposed MA can be assessed by checking the CTC value corresponding to the best solutions. It is possible to see that the proposed MA outperformed four out of six cited heuristics, whereas the gap with respect to the results of Vidal, Crainic, Gendreau, *et al.* [14] was 0.37%. Analyzing the results category-wise, it can be seen that the proposed MA achieved the best solution quality on instances of type C. Remarkable results were obtained on instances RC2. In this case, the gap with respect to the best results was less than 0.1%. On the other data sets, the gaps were a bit greater; in any case, the gaps with respect to the best results were less than 1.0%. These results show that the proposed MA is able to produce high-quality solutions, which are competitive with those produced by the other heuristics.

The results obtained for all the instances are presented in Table II. Two values are reported for each solution: the total travel cost in terms of distance or time (TC), and the number of vehicles (NV). The best published solutions are listed with the references to the papers that recorded them. Most of the best-

Table II
RESULTS FOR EACH INSTANCE

Instance	Other Heuristics		Proposed MA					
	Best Solution		Best Solution		Average	St Dv.	CV	Avg. time (s)
	TC | NV	Ref.	TC | NV	Cost Gap (%)	TC | NV	TC | NV	TC | NV	
C101	828.94 | 10	[2]	**828.94 | 10**	**0.00**	828.94 | 10.00	0.00 | 0.00	0.00 | 0.00	19.82
C102	828.94 | 10	[2]	**828.94 | 10**	**0.00**	828.94 | 10.00	0.00 | 0.00	0.00 | 0.00	21.87
C103	828.06 | 10	[2]	**828.06 | 10**	**0.00**	828.06 | 10.00	0.00 | 0.00	0.00 | 0.00	23.85
C104	824.78 | 10	[2]	**824.78 | 10**	**0.00**	824.78 | 10.00	0.00 | 0.00	0.00 | 0.00	24.93
C105	828.94 | 10	[2]	**828.94 | 10**	**0.00**	828.94 | 10.00	0.00 | 0.00	0.00 | 0.00	20.88
C106	828.94 | 10	[2]	**828.94 | 10**	**0.00**	828.94 | 10.00	0.00 | 0.00	0.00 | 0.00	21.09
C107	828.94 | 10	[2]	**828.94 | 10**	**0.00**	828.94 | 10.00	0.00 | 0.00	0.00 | 0.00	21.26
C108	828.94 | 10	[2]	**828.94 | 10**	**0.00**	828.94 | 10.00	0.00 | 0.00	0.00 | 0.00	22.37
C109	828.94 | 10	[2]	**828.94 | 10**	**0.00**	828.94 | 10.00	0.00 | 0.00	0.00 | 0.00	23.64
C201	591.56 | 3	[2]	**591.56 | 3**	**0.00**	591.56 | 3.00	0.00 | 0.00	0.00 | 0.00	38.10
C202	591.56 | 3	[2]	**591.56 | 3**	**0.00**	591.56 | 3.00	0.00 | 0.00	0.00 | 0.00	39.14
C203	591.17 | 3	[2]	**591.17 | 3**	**0.00**	591.17 | 3.00	0.00 | 0.00	0.00 | 0.00	38.92
C204	590.6 | 3	[2]	**590.6 | 3**	**0.00**	590.60 | 3.00	0.00 | 0.00	0.00 | 0.00	38.13
C205	588.88 | 3	[2]	**588.88 | 3**	**0.00**	588.88 | 3.00	0.00 | 0.00	0.00 | 0.00	38.09
C206	588.49 | 3	[2]	**588.49 | 3**	**0.00**	588.49 | 3.00	0.00 | 0.00	0.00 | 0.00	38.31
C207	588.29 | 3	[2]	**588.29 | 3**	**0.00**	588.29 | 3.00	0.00 | 0.00	0.00 | 0.00	38.67
C208	588.32 | 3	[2]	**588.32 | 3**	**0.00**	588.32 | 3.00	0.00 | 0.00	0.00 | 0.00	37.78
R101	1642.88 | 20	[2]	**1642.88 | 20**	**0.00**	1645.51 | 20.00	3.02 | 0.00	0.18 | 0.00	30.04
R102	1472.62 | 18	[3]	1473.62 | 18	0.07	1478.39 | 18.00	2.67 | 0.00	0.18 | 0.00	28.65
R103	1213.62 | 14	[3]	1217.46 | 14	0.32	1223.28 | 14.90	2.42 | 0.32	0.20 | 2.12	28.03
R104	976.61 | 11	[2]	985.17 | 11	0.88	992.44 | 10.80	4.89 | 0.42	0.49 | 3.90	27.06
R105	1360.78 | 15	[2]	1365.66 | 15	0.36	1371.97 | 15.70	2.85 | 0.48	0.21 | 3.08	28.01
R106	1239.37 | -	[14]	1245.35 | 13	0.48	1250.83 | 13.10	4.57 | 0.32	0.37 | 2.41	28.09
R107	1072.12 | -	[14]	1081.7 | 12	0.89	1089.60 | 11.70	5.07 | 0.48	0.47 | 4.13	27.45
R108	938.2 | -	[14]	951.24 | 10	1.39	953.87 | 10.40	2.14 | 0.52	0.22 | 4.97	27.22
R109	1013.16 | 12	[15]	1155.46 | 13	14.05	1161.32 | 13.00	5.00 | 0.00	0.43 | 0.00	27.60
R110	1072.41 | 12	[2]	1082.47 | 12	0.94	1092.04 | 12.00	5.04 | 0.00	0.46 | 0.00	26.81
R111	1053.5 | 12	[2]	1055.6 | 12	0.20	1063.82 | 12.00	3.95 | 0.00	0.37 | 0.00	27.18
R112	953.63 | 10	[2]	962.82 | 11	0.96	973.55 | 10.90	5.27 | 0.32	0.54 | 2.90	26.46
R201	1147.8 | 8	[4]	1148.09 | 8	0.03	1149.68 | 8.30	1.17 | 0.48	0.10 | 5.82	31.07
R202	1034.35 | 8	[2]	1034.82 | 8	0.05	1040.16 | 6.90	2.85 | 0.74	0.27 | 10.69	32.72
R203	874.87 | 6	[2]	875.12 | 6	0.03	876.57 | 6.00	1.29 | 0.00	0.15 | 0.00	34.85
R204	735.8 | 5	[4]	736.27 | 5	0.06	737.61 | 4.60	1.38 | 0.52	0.19 | 11.23	37.31
R205	954.16 | 5	[4]	**954.16 | 5**	**0.00**	958.09 | 5.30	3.05 | 0.48	0.32 | 9.11	33.85
R206	879.89 | 5	[2]	**879.89 | 5**	**0.00**	883.06 | 5.00	2.75 | 0.00	0.31 | 0.00	35.94
R207	797.99 | 4	[4]	800.83 | 4	0.36	803.06 | 4.00	1.88 | 0.00	0.23 | 0.00	38.47
R208	705.33 | -	[14]	707.58 | 3	0.32	711.39 | 3.50	2.45 | 0.53	0.34 | 15.06	41.84
R209	859.39 | 5	[2]	**859.39 | 5**	**0.00**	861.02 | 5.00	1.90 | 0.00	0.22 | 0.00	33.85
R210	904.78 | -	[14]	910.83 | 6	0.67	913.54 | 6.20	1.99 | 0.63	0.22 | 10.20	33.38
R211	753.15 | -	[14]	759.14 | 4	0.80	761.62 | 4.00	2.15 | 0.00	0.28 | 0.00	35.12
RC101	1623.58 | 15	[16]	1646.53 | 16	1.41	1657.52 | 16.30	6.46 | 0.48	0.39 | 2.96	26.14
RC102	1461.23 | 14	[2]	1480.46 | 15	1.32	1487.73 | 15.00	6.06 | 0.00	0.41 | 0.00	27.36
RC103	1249.86 | 11	[17]	1279.15 | 12	2.34	1292.62 | 12.00	11.83 | 0.00	0.91 | 0.00	27.32
RC104	1135.48 | 10	[18]	1139.14 | 10	0.32	1154.03 | 10.40	6.49 | 0.52	0.56 | 4.97	26.82
RC105	1518.58 | 16	[2]	1542.37 | 16	1.57	1552.54 | 15.80	5.69 | 0.63	0.37 | 4.00	27.01
RC106	1376.26 | -	[14]	1383.85 | 13	0.55	1404.15 | 13.60	11.87 | 0.52	0.85 | 3.80	27.01
RC107	1211.11 | -	[14]	1216.65 | 12	0.46	1232.31 | 12.00	8.78 | 0.00	0.71 | 0.00	26.93
RC108	1117.53 | 11	[3]	1123.26 | 11	0.51	1133.14 | 11.00	4.33 | 0.00	0.38 | 0.00	26.52
RC201	1265.56 | 9	[2]	1269.07 | 9	0.28	1273.45 | 9.10	3.38 | 0.88	0.27 | 9.62	29.89
RC202	1095.64 | 8	[2]	**1095.64 | 8**	**0.00**	1098.73 | 7.90	2.23 | 0.32	0.20 | 4.00	31.62
RC203	926.82 | -	[14]	**926.82 | 5**	**0.00**	933.13 | 5.00	4.18 | 0.00	0.45 | 0.00	33.22
RC204	786.38 | 4	[4]	**786.38 | 4**	**0.00**	787.70 | 4.00	1.31 | 0.00	0.17 | 0.00	35.84
RC205	1157.55 | 7	[4]	**1157.55 | 7**	**0.00**	1158.98 | 7.00	1.52 | 0.00	0.13 | 0.00	29.28
RC206	1054.61 | 7	[2]	1057.65 | 7	0.29	1062.57 | 6.10	2.79 | 0.74	0.26 | 12.10	31.97
RC207	966.08 | 6	[2]	**966.08 | 6**	**0.00**	966.91 | 6.00	1.65 | 0.00	0.17 | 0.00	31.08
RC208	778.93 | -	[14]	**778.93 | 4**	**0.00**	782.65 | 4.50	2.24 | 0.53	0.29 | 11.71	32.61

known solutions have been found by some of the heuristics cited previously. The best solutions found by the proposed MA are reported along with the travel cost gaps attained with respect to the best published solutions. The average, standard deviation, and coefficient of variation of the TC and NV values gathered during 10 runs are reported to provide further details on the MA performance. The average running time (in seconds) for each instance is reported as well. The proposed MA found the best-known solution for 27 out of 56 instances (marked in bold), including all instances of type C and six out of eight instances of the data set RC2. For most of the other instances, the cost gaps were less than 1.0%. Overall, a solution with a cost gap less than 1.0% with respect to the best-known solution was found for 50 out of 56 instances. The values of the coefficients of variation associated with travel time were less than 1.0% for all instances, which suggests that the proposed MA performed consistently in all experiments. Statistical tests were carried out to confirm this fact. Since the results did not conform to normality (Shapiro-Wilk test p-value < 0.05), the Kruskal-Wallis test was employed to compare the

travel-time distributions obtained from all experiments. At a 95% confidence level, the Kruskal-Wallis test indicated that the travel-time distributions did not significantly differ across experiments (p-value $= 1.0 > 0.05$). Thus, it is possible to conclude that the proposed MA consistently found solutions of similar quality.

V. Conclusions

This paper proposed a simple yet effective MA for solving the VRPTW. A local search procedure, which consists of relocating customers between close routes was described. The search methodology consists of applying a clustering procedure to customers' spatial information, using the clustering arrangements obtained to identify which routes pass close to each other, and performing neighborhood moves between the routes identified. Computational experiments on the Solomon's benchmark set showed that the proposed approach can produce high-quality solutions, which are competitive with those produced by other heuristics. The proposed MA outperformed four out of six solution approaches considered for comparison (when assessing the results in terms of the travel cost obtained over all instances), and it found the best-known solutions for several instances.

Further research could assess the performance of the proposed MA in solving other variants of the VRP and related problems. The development of new clustering approaches to guide the search is another possible direction of further research.

Acknowledgments

The authors would like to thank CNPq, CAPES, and CeMEAI-CEPID (FAPESP No. 2013/07375-0) for the financial support received.

References

[1] G. Desaulniers, O. B. G. Madsen, and S. Ropke, "The Vehicle Routing Problem with Time Windows," in *Vehicle Routing: Problems, Methods, and Applications*, P. Toth and D. Vigo, Eds., 2nd, Philadelphia, PA: SIAM - Society for Industrial and Applied Mathematics, 2014, ch. 5, pp. 119–159.

[2] S. Jung and B.-R. Moon, "A Hybrid Genetic Algorithm for the Vehicle Routing Problem with Time Windows," in *Proceedings of the 4th Annual Conference on Genetic and Evolutionary Computation*, ser. GECCO'02, San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2002, pp. 1309–1316.

[3] G. Alvarenga, G. Mateus, and G. de Tomi, "A genetic and set partitioning two-phase approach for the vehicle routing problem with time windows," *Computers & Operations Research*, vol. 34, no. 6, pp. 1561–1584, 2007.

[4] H. C. de Oliveira and G. C. Vasconcelos, "A hybrid search method for the vehicle routing problem with time windows," *Annals of Operations Research*, vol. 180, no. 1, pp. 125–144, 2010.

[5] Z. Ursani, D. Essam, D. Cornforth, and R. Stocker, "Localized genetic algorithm for vehicle routing problem with time windows," *Applied Soft Computing*, vol. 11, no. 8, pp. 5375–5390, 2011.

[6] D. Bustos, M. O. Santos, and C. F. Toledo, "Clustering-Based Crossover in an Evolutionary Algorithm for the Vehicle Routing Problem with Time Windows," in *Proceedings of the 29th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2017)*, Boston, MA, 2017, (to appear).

[7] M. M. Solomon, "Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints," *Operations Research (INFORMS)*, vol. 35, no. 2, pp. 254–265, 1987.

[8] R. Xu and D. Wunsch, "Hierarchical Clustering," in *Clustering*, Hoboken, NJ: John Wiley & Sons, Inc., 2008, ch. 3, pp. 31–62.

[9] B. Ombuki, B. J. Ross, and F. Hanshar, "Multi-objective genetic algorithms for vehicle routing problem with time windows," *Applied Intelligence*, vol. 24, no. 1, pp. 17–30, 2006.

[10] B. Funke, T. Grünert, and S. Irnich, "Local Search for Vehicle Routing and Scheduling Problems: Review and Conceptual Integration," *Journal of Heuristics*, vol. 11, no. 4, pp. 267–306, 2005.

[11] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing.," *Science (New York, N.Y.)*, vol. 220, no. 4598, pp. 671–680, 1983.

[12] N. Mladenović and P. Hansen, "Variable neighborhood search," *Computers & Operations Research*, vol. 24, no. 11, pp. 1097–1100, 1997.

[13] A. Garcia-Najera and J. A. Bullinaria, "An improved multi-objective evolutionary algorithm for the vehicle routing problem with time windows," *Computers & Operations Research*, vol. 38, no. 1, pp. 287–300, 2011.

[14] T. Vidal, T. G. Crainic, M. Gendreau, and C. Prins, "Time-window relaxations in vehicle routing heuristics," *Journal of Heuristics*, vol. 21, no. 3, pp. 329–358, 2015.

[15] W.-C. Chiang and R. A. Russell, "A Reactive Tabu Search Metaheuristic for the Vehicle Routing Problem with Time Windows," *INFORMS Journal on Computing*, vol. 9, no. 4, pp. 417–430, 1997.

[16] Y. Rochat and É. Taillard, "Probabilistic diversification and intensification in local search for vehicle routing," *Journal of Heuristics*, vol. 1, no. 1, pp. 147–167, 1995.

[17] K. C. Tan, T. H. Lee, K. Ou, and L. H. Lee, "A messy genetic algorithm for the vehicle routing problem with time window constraints," in *Proceedings of the 2001 Congress on Evolutionary Computation (IEEE Cat. No.01TH8546)*, vol. 1, Seoul, Korea, 2001, pp. 679–686.

[18] J.-F. Cordeau, G. Laporte, and A. Mercier, "A unified tabu search heuristic for vehicle routing problems with time windows," *Journal of the Operational Research Society*, vol. 52, no. 8, pp. 928–936, 2001.