© International Journal of Combinatorial Optimization Problems and Informatics, Vol. 9, No. 1, Jan-April 2018, pp. 35-45. ISSN: 2007-1558.

# A Parallel Memetic Algorithm to Solve the Capacitated Vehicle Routing Problem with Time Windows

Oscar M. González, Carlos Segura, S. Ivvan Valdez Peña
A Research Center of Mathematics (CIMAT A.C.),
Computational Sciences Department
Jalisco S/N, Col. Valenciana CP: 36240 Guanajuato Gto., Mexico
*{oscar.gonzalez,carlos.segura,ivvan}@cimat.mx*

**Abstract.** Nowadays, smart cities management has become an important topic where algorithms are crucial to solving their problems. Some topics related to smart cities can be modeled as combinatorial optimization problems. A common challenge is to find the optimal route, or routes, to attend demands of customers of a store. This challenge is known as the Vehicle Routing Problem (VRP). Evolutionary Algorithms (EAs) have demonstrated a high capacity to approximate optimal solutions in a relatively small interval of time. In this article, we propose an EA to solve the Capacitated Vehicle Routing Problem with Time Windows (CVRPTW). Our proposal is a memetic algorithm that integrates simulated annealing and novel crossover operators. Although this kind of algorithm obtains higher quality solutions than simple EAs, they require a greater amount of time. Hence, the use of parallel computing is essential to get high-quality solutions in admissible times. Additionally, a novel crossover operator that enhances results of state-of-art schemes is proposed. The proposal is validated using Solomon's benchmark, by contrasting our results with the best solutions reported in the specialized literature. A new best-known solution for a commonly used instance could be generated.

**Keywords:** Capacitated Vehicle Routing Problem, memetic algorithm, CVRPTW, Parallel Computing, Solomon's benchmark.

## 1 Introduction

Combinatorial problems are commonly found in industrial, economic and scientific domains [5]. When the problem, in turn, has a relatively small number of variables, dynamic programming or other exact techniques [2] might be an adequate approach. However, when the number of variables is relatively large (hundreds or thousands), these approaches are not the best alternative, because they need too much time or they might be non-feasible for delivering a global optimum of the problem.

Another alternative is to approximate a global optimum by using heuristic and metaheuristic algorithms [7]. Although they might require a large computational time to attain high-quality solutions, they have the enormous advantage of being highly parallelizable. Additionally, the stopping criterion might be set to a specific amount of elapsed time. As expected, the quality of the obtained results depends on such a stopping criterion.

The Vehicle Routing Problem (VRP) term [6] is a general concept which is used to define a set of problems where there are a number of customers that must be visited by a set of vehicles. Generally, the problem is defined as a graph where nodes are the customers and edges are paths between customers. There is usually a special node which is the depot of the store. In the VRP, a set of routes must be designed where each route starts and ends in the depot and each customer is visited one time [13]. The aim of the optimization problem is to find a set of routes of minimum cost that visits all customers. In Fig. 1, an example of a Vehicle Routing Problem and a candidate solution is showed, where there are a depot, 4 routes, and 16 customers. Several modifications to the classic VRP have been proposed by adding constraints to the original problem. Capacitated Vehicle Routing Problem (CVRP) is the VRP with constraints of capacity. Specifically, each route has a maximum capacity to carry and each customer has a demand. The sum of the demands of the customers of each route cannot be higher than its capacity. Capacitated Vehicle Routing Problem with Time Windows (CVRPTW) [18] adds, for each customer, a time windows when this customer is available to be visited. The vising time of customer *j* is equal to the time associated to the customer *i* previously

Received Dec 2, 2017 / Accepted Jan 22, 2018

visited in its route plus the cost of travel from customer *i* to customer *j*. If the vehicle reaches a customer previously to the initial time of its time windows, it must wait there. Thus, in such cases, the visiting time of the customer is considered to be the initial time of its time windows.

In this paper, we propose a memetic algorithm to solve the CVRPTW. Specifically, it is a population-based evolutionary algorithm that is integrated with a local search procedure. Simulated annealing [11] is selected as a local search because it has attained promising results in the CVRPTW. Memetic algorithms are, usually, more computationally expensive and in consequence require more time than typical population-based Evolutionary Algorithms (EAs). In order to obtain high-quality solutions, in relatively short times, we take advantage of a distributed-memory parallelization.

## 2 Vehicle Routing Problem: Mathematical Definition

There are many variants of the VRP where the difference between each of them are the constraints and/or objectives in the optimization model. In this article, we address the Capacitated Vehicle Routing Problem with Time Windows (CVRPTW) introduced previously.

The mathematical definition of the objectives and constraints of the CVRPTW can be modeled as follows: let $G = (V, E)$ be a non-directed complete graph,

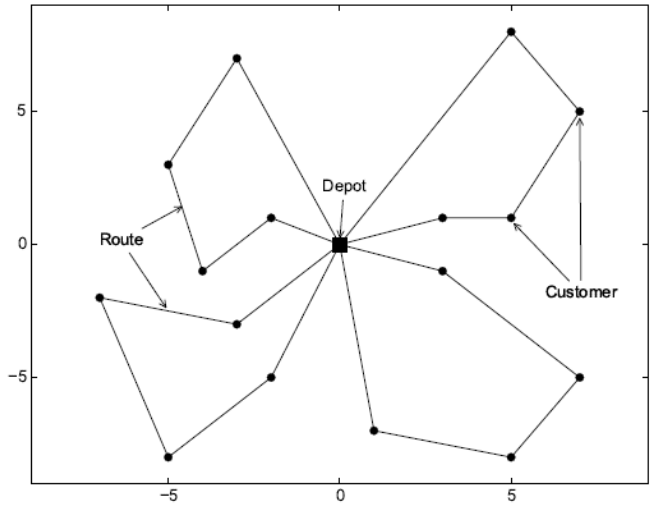


**Fig. 1.** Vehicle Routing Problem

where vertices $V = \{1, \ldots, N\}$ correspond to a depot and customers, and the edges $e \in E \{(i, j) : i, j \in V\}$ are paths among them.

A candidate solution is represented by a set of *K* routes that start and end at the depot. Note that, some of these routes might be empty, so *K* represents the maximum number of routes. Each route k is a simple cycle, so they can be established by indicating whether a vehicle travels from node *i* to node *j*. Thus, these routes can be mathematically defined with a set of decision variables $X_{ij}^{k}$, to set routes as follows:

$$X_{ij}^{k} = \begin{cases} 1 & \text{if vehicle } k \text{ travels from node } i \text{ to node } j \\ 0 & \text{otherwise} \end{cases}$$

The CVRPTW can be stated as follows:

$$\min TD = \sum_{k=1}^{K}\sum_{i=1}^{N}\sum_{j=1}^{N} X_{ij}^{k} C_{ij} \quad (1)$$

Subject to

$$X_{ii}^{k} = 0\,(\forall i \in \{1,\ldots,N\}, \forall k \in \{1,\ldots,K\}) \quad (2)$$

$$X_{ij}^{k} \in \{0,1\}(\forall i,j \in \{1,\ldots,N\}, \forall k \in \{1,\ldots,K\}) \quad (3)$$

$$\sum_{k=1}^{K}\sum_{i=1}^{N} X_{ij}^{k} = 1(\forall j \in \{2,\ldots,N\}) \quad (4)$$

$$\sum_{i=1}^{N}\sum_{j=2}^{N} X_{ij}^{k} d_j \leq Q^{k}(\forall k \in \{1,\ldots,K\}) \quad (5)$$

$$\sum_{k=1}^{K}\sum_{j=2}^{N} X_{1j}^{k} \leq K \quad (6)$$

$$\sum_{j=2}^{N} X_{1j}^{k} - \sum_{j=2}^{N} X_{j1}^{k} = 0(\forall k \in \{1,\ldots,K\}) \quad (7)$$

$$s_{ki} + C_{ij} - L(1 - X_{ij}^{k}) \leq s_{kj}(\forall i,j \in \{1,\ldots,N\}, \forall k \in \{1,\ldots,K\}) \quad (8)$$

$$a_j \leq s_{kj} \leq b_j(\forall i,j \in \{1,\ldots,N\}, \forall k \in \{1,...,K\}) \quad (9)$$

The objective function is defined in equation (1) where $C_{ij}$ is the cost for traveling from node i to node j (here $C_{ij}$ is considered as the distance or time required for traveling from node *i* to node *j*), *K* is the maximum number of vehicles that can be used, *N* is the number of customers plus the depot (the depot is tagged with number 1, and the customers are denoted as *2*, . . . , *N*). Equation (2) enforces that a vehicle cannot travel from a customer to the same customer. Equation (3) restricts the value of $X_{ij}^{k}$ to 0 or 1; when it is set to one it means that vehicle k travel from node *i* to node *j*, otherwise it is set to 0. Equation (4) ensures that a customer is visited only one time. Equation (5) guarantees that sum of customer capacities visited by a vehicle does not exceed the capacity of the vehicle ($Q^k = Q$ for all vehicles), where $Q^k$ is the loading capacity of vehicle k (in our case, the same capacity Q is considered for all vehicles) and $d_j$ is the demand at customer *j*. Equation (6) requires that a maximum of K routes start at a depot. Equation (7) guarantees all routes start and finish in the depot. Equation (8) means that if vehicle k is traveling from customer *i* to customer *j* we cannot arrive at customer *j* previously to $S_{ki} + C_{ij}$, where $S_{kj}$ is the visiting time of customer *j* by vehicle *k* and *L* is a large scalar $(L \geq \sum_{i=1}^{N}\sum_{j=1}^{N} C_{ij})$ . Finally, Equation (9) ensures that time windows are fulfilled, where $a_j$ is the earliest time for customer *j* to allow the service and $b_j$ is the latest time for customer j to allow the service.

## 3 Literature Review

In this section, we provide a brief review of different approaches, reported in specialized literature [7], to tackle the CVRPTW.

Several trajectory-based scheme have been proposed in literature. Among them, SA has provided really promising results. In [3] authors propose a SA algorithm with multiple temperatures. Additionally, they present a parallel version of their algorithm using OpenMP and MPI for solving the Solomon's benchmark. In the Solomon's benchmark, instances are categorized in several groups. The Cxxx instances have the customers clustered in geographic regions; in the instances Rxxx the positions of customers are randomized; finally, the instances RCxxx are a combination between Cxxx and Rxxx. The results reported in [3] are good with easy instances such as the Cxxx, but they do not get so high-quality results when dealing with harder instances such as the Rxxx and RCxxx.

A more complex memetic algorithm is used in [15] for approaching a multi- objective version of the CVRPTW to minimize the traveling distance and the number of routes at the same time. The authors also test their proposal in Solomon's benchmark. Anyway, using a multi-objective algorithm to address a single-objective problem is complex and several inconveniences might arise [16]. In any case, since they help to improve the diversity, in some other cases they are quite helpful. Authors found the best-known solutions in the easiest instances, but the performance deteriorates in more complex cases. A hybrid evolutionary algorithm is proposed in [12]. The authors combine an adaptive large neighborhood search (ALNS) [1] and a population-based search to construct their hybrid evolutionary algorithm. Since this algorithm is not applied with the classical Solomon's benchmark it is difficult to compare against this proposal. Finally, in [10] a framework that uses a kind of dynamic programming heuristic to solve the Traveling Salesman Problem (TSP) is introduced. Authors use this approach to approximate solutions to any kind of VRP. The proposal is tested on Solomon's benchmark and Goel's instances [8] that are a modification of Solomon's benchmark. They only report the mean fitness obtained when taking into account the whole benchmark set.

Another issue with VRP is the representation of candidate solutions. In [15], authors encode each route independently. Specifically, they save each route in a different array or list that is the sequence of customers visited in such a route. In [4] authors improve the Davis encoding system, where a chromosome represents a permutation of ***n*** customers to be partitioned into ***m*** vehicles. The evaluation function assumes that the first ***m*** customers of a chromosome are placed into the ***m*** vehicles. The remaining ***n-m*** customers are examined individually. As each new customer is selected a possible sub tour is evaluated for each vehicle and the best sub tour is selected. Probably, the main drawback of this last encoding is that not every possible candidate solution of the problem have a corresponding encoding. Thus, the optimal solution might not be represented in some cases. As a result, the same encoding as in [15] is used in our proposal.

An important operation of the memetic algorithm is the crossover operation. In [14], authors propose the Sequence-Based Crossover where each offspring is generated by merging two routes from two different parents. Specifically, two breakpoints are selected in such routes, and the last parts of the routes are attached to the first part of the routes. Additionally, authors propose the Route Based Crossover. This operator replaces a route in the first parent by choosing a random route from the second parent to generate the offspring. In [9], the proposed crossover operator combines iteratively various routes (R1) of parent solution P1 with a subset of customers, selected from the nearest neighbor routes from parent P2. Finally, with the aim of generating a feasible solution, a removal procedure and an insertion heuristic inspired by [17] coupled to a random customer acceptance procedure is applied. Since the SBX obtains really promising results, this operators and an extension of it, has been integrated in our proposal.

## 4 Memetic Algorithm

In this paper, a novel parallel memetic algorithm to address the CVRPTW is proposed. It is a population-based algorithm that uses crossover operators and local search to enhance offspring. Taking into account the performance of different state-of-art schemes, we decided to apply the Sequence-Based Crossover operator (SBX). In addition, we propose a novel variant of SBX that enhances its performance which is deeply explained in section 4.1. This novel operator is called as Single Breaking-point Sequence-Based Operator (SBSBX).

The proposed memetic algorithm applies simulated annealing (SA) as local search. Specifically, SA is applied on each offspring resulting from the crossover by setting its stopping criterion to 5 seconds. SA takes into account the definition of neighborhood propose in [3]. Additionally, the population initialization proposed in [3] is used by the proposed memetic algorithm.

Specifically, the initial routes are built by using three fast strategies. In the first one, the customers are assigned to routes randomly until all customer are assigned. In each step, the customer is chosen randomly. In the second one, the customers are inserted to routes according to their identifier, i.e. customer 1 is assigned first, then customer 2 is assigned, and so on. Finally, in the third proposal the customers are assigned to vehicles in ascending order of time windows. Customers with the smaller end time of its time window.

Note that in our scheme, a population of 50 individual is used. Taking into account that each local search requires a lot of computational resources when compared to the resources required by the remaining steps, a parallelization with MPI is used to parallelize the application of local search. Specifically, a master-slave scheme is used to distribute the application of the local search procedure. This process is detailed in Section 4.3. A general overview of our proposal is showed in algorithm 1.

In algorithm 1, step 1 initializes the population as previously described. Then, until a stopping criterion is reached (step 2), which in our case is set as the elapsed time, the steps 3 to 10 are executed. Step 3 executes step 4 and 5 to get a new population of offspring. Step 4 selects two parents randomly. Step 5 applies a crossover operator to generate offsprings. Step 7 applies local search. Since our proposal is based on a master-slave scheme, offspring are distributed to slaves and slaves run the local search on their corresponding offspring. In step 5 each offspring enhanced by threads is sent back to the master process. Step 9

**Algorithm 1:** general schema of memetic algorithm

```
   Data: size population, stop criteria, temperature, operator SBX used
   Result: the best individual found
 1 initialization of population;
 2 while stop criteria is not reached do
 3 |   while offspring generated is less to size population - 1 individuals do
 4 |   |   select two parents randomly;
 5 |   |   apply SBX or SBSBX operator;
 6 |   end
 7 |   distribute offspring to threads; apply simulated annealing to offspring of
   |    each thread;
 8 |   send offspring after the application of simulated annealing to master process;
 9 |   insert elite individual (best individual of previous population);
10 |   store new elite individual;
11 end
```

insert the elite individual of the previous generation in the new generation with the aim of preserving the best individual along each generation of the memetic algorithm. Finally, step 10 chooses the new elite individual by comparing new offspring with the previous elite individual.

### 4.1 Crossover Operator

Sequence-Based Crossover (SBX) [14] is used as crossover operator by our proposal. The procedure of SBX operator is illustrated in Fig. 2. Given two parents, $\mathbf{P}_1$ and $\mathbf{P}_2$, a new offspring is generated. From each parent, one route — $\mathbf{R}_1$ and $\mathbf{R}_2$ — is randomly selected. Then, a random customer is chosen from each route as breakpoint of the route. As results of this, two sub-routes $\mathbf{R}_1 = (\mathbf{R}_{11}, \mathbf{R}_{12})$ and $\mathbf{R}_2 = (\mathbf{R}_{21}, \mathbf{R}_{22})$ are created. A new route $\boldsymbol{R}_{new}$ is generated by merging the first part of route $\mathbf{R}_1$ and the second part of route $\mathbf{R}_2$ in $\boldsymbol{R}_{new} = (\mathbf{R}_{11}, \mathbf{R}_{22})$. The new offspring is generated by replacing $\mathbf{R}_1$ by $\boldsymbol{R}_{new}$ in $\mathbf{P}_1$.

Solution with unrouted (see the black diamond in Fig. 2) or duplicated customers (blue square in Fig. 2) might appear. Thus, we apply a repairing procedure in order to always generate valid offspring, i.e. offspring where each customer is visited one time. Three different steps are used to repair solutions:

- **Duplicated customer in $R_{new}$**: A duplicated customer is chosen randomly to remove it.
- **Duplicated customer in different routes**: Since we want to keep $R_{new}$, the duplicated customer that it is not in $R_{new}$ is removed.
- **Unrouted Customer:** This customer is inserted at the feasible insertion place that minimizes the route where it is inserted.

If a non-feasible offspring is generated, i.e. one that does not fulfill all the constraints, in [14] the process is repeated until a feasible offspring is generated.

**Algorithm 2:** Single Breaking-point Sequence-Based Crossover Operator

**Data:** $Parent_1$ selected, $Parent_2$ selected
**Result:** offspring generated
1 copy $parent_1$ to offspring;
2 select $Route_1$ and $customer_1$ in $parent_1$ ;
3 search for $customer_1$ in $parent_2$ and select his route as $Route_2$ ;
4 merge $Route_1$ and $Route_2$ in $Route_{new}$ with $customer_1$ as breakpoint ;
5 apply repairing phase to offspring generated ;

This step is avoided in our case, because we could verify that our SA usually converts any non-feasible solution into a feasible one in very few time.

By applying some distance metrics, we could verify that the classical SBX is quite disruptive, especially when the breaking points selected in the candidate solutions are different. As a result, we propose a new variant, the SBSBX which selects the same breaking points in both parents (see Algorithm 2). In this way, a less disruptive behavior is induced. In addition, SBSBX is less elitist than the original SBX with respect to unrouted customers. Specifically, SBSBX places unrouted customers randomly instead of choosing the best position. The route selected for placing unrouted customers can be any except $R_{new}$.

In algorithm 2, step 1 copies the $parent_1$ to offspring candidate. In step 2 a customer is selected and, in this case, its route is selected too. The customer selected of $parent_1$ is searched in $parent_2$ and its route is selected in step 3. $Route_{new}$ is generated by merging both routes selected and the customer as single breaking-point in step 4. Finally, in step 5 the repairing phase is applied to the offspring generated.

### 4.2 Local Search

The local search phase applied in our proposal is a Simulated Annealing [11] based on 10 different transformations to generate neighbors. In each iteration of SA a transformation is chosen with the probabilities given in Table 1. Each transformation works as follow:

- **Customer Random Reallocation**: A random customer is selected from a random route and is placed in an allocation of its original route chosen randomly.
- **Customer Best Reallocation**: A random customer is selected from a random route and is placed in the best position of its original route.
- **Customer Random Migration**: A random customer is selected from a random route. Then, this customer is placed in a different non-empty route in a random position.
- **Customer Best Migration**: A random customer is selected from a random route. Then, this customer is placed in a different non-empty route in the best position

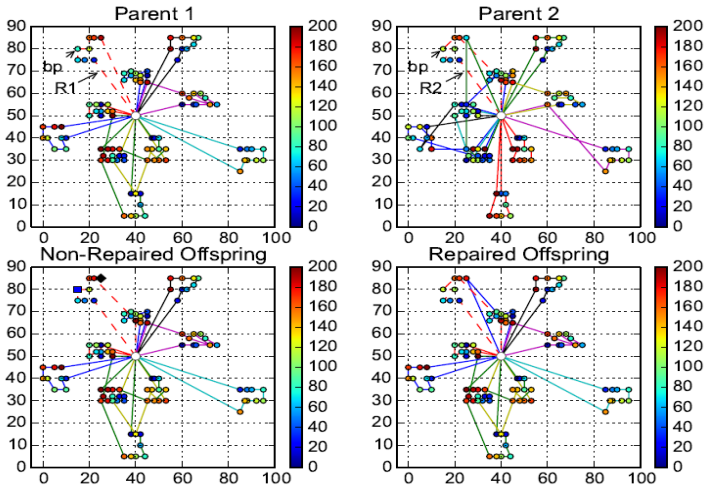


**Fig. 2.** Sequence-Based Crossover operator with repairing phase (bp=breakpoint).

- **Customers Random Exchange:** Two random customer $C_1$ and $C_2$ are selected from random routes $R_1$ and $R_2$ respectively and they are exchanged.
- **Customer Best Exchange:** Two random routes $R_1$ and $R_2$ are selected and the pair of customer $C_1$ and $C_2$ that provide the largest improvement ($C_1 \in R_1$, $C_2 \in R_2$) are exchanged.
- **Customer Exchange With coincident Time-Window:** A customer is selected from a random route. Then, it is exchanged with a customer with a similar time-window from a random non-empty route. In order, to select the most similar time-window, the upper bound of time-windows is taken into account.
- **Route Partition:** A random customer is selected from a random route. Then, this route is broken into two parts, the first part is from start of the route to this customer, the second part is from customer to end of the route. The first part is conserved in the original route and second part is assigned in a random way to empty routes. If there are not empty routes this operator does not do anything; in another cases, each customer is assigned one by one to any of this set of empty routes in a random way

**Table 1.** Transformation Operation Probability

| Transformation Operation | Probability |
|---|---|
| Customer Random Reallocation | 15% |
| Customer Best Reallocation | 5% |
| Customer Random Migration | 15% |
| Customer Best Migration | 5% |
| Customers Random Exchange | 10% |
| Customer Best Exchange | 5% |
| Customer Exchange With coincident Time-Window | 10% |
| Route Partition | 15% |
| New Route | 15% |
| Route Elimination | 5% |

- **New Route:** A random customer is selected from a random route. This customer is assigned to an empty route.
- **Route Elimination:** A route is randomly selected, Then, all customer from this route are assigned in the best position from other routes.

### 4.3 Parallelization

As it has been described, our proposal runs the local search procedure by setting its stopping criterion to 5 seconds. Since local search is the computationally more expensive part of our proposal, we take advantage of a distributed-memory parallelization based on a master-slave model in order to reduce the computation time. In other words, each slave process does the local search to a portion of the total population. In this way, all processes are working together in parallel to enhance the offspring. A master-slave scheme is proposed to do this work, where the master process applies the SBSBX operator and the slave processes are

responsible of performing the local search. Note that, since the master does not apply local search to any offspring, there might be a loss of performance. However, this overhead diminishes when using a larger amount of slaves.

# 5 Results

In this section, the results of our proposal are described. Since our proposal is a stochastic algorithm, in order to evaluate the performance we run 30 times our algorithm for each instance. A population of 50 individuals is selected and the comparative between SBX and our modification of SBX is presented in section 5.2. Results of our parallelization are presented in section 5.3.

## 5.1 Benchmark

In order to evaluate the performance of our proposal, we have used the well- known Solomon's benchmark. The reason is that this benchmark is the most

**Table 2.** SBX Results with temperature = 10

| Instance | BKS | Mean | Median | std | Best | Worst |
|---|---|---|---|---|---|---|
| C103 | **826.3** | 828.06 | 828.06 | 2.46e-07 | 828.06 | 828.06 |
| C108 | **827.3** | 828.94 | 828.94 | 3.59e-07 | 828.94 | 828.94 |
| R103 | **1208.7** | 1215.07 | 1215.15 | 0.82 | 1213.62 | 1216.72 |
| R108 | 947.55 | 941.23 | 939.61 | 3.17 | **938.20** | 949.14 |
| RC103 | **1258** | 1262.56 | 1262.98 | 0.72 | 1261.67 | 1264.59 |
| RC108 | **1114.2** | 1118.09 | 1117.83 | 0.61 | 1117.53 | 1119.25 |

widely used benchmark with CVRPTW. Solomon's benchmark has 3 different set of instances. Each kind distributes the customers in different ways, at it was previously described. In order to evaluate our proposal with a diverse set of instances, we have selected two instances of each kind.

Solomon's benchmark comprises instances with 25 customers, 50 customers, 100 customers and 200 customers. In our analyses, instances of 100 customers have been selected because they have larger search spaces than those with 25 and 50 customers and because not many results are available for the cases of 200 customers. Since we consider that proper conclusions can be drawn with 2 instances of each structure, C103, C108, R103, R108, RC103 and RC108 have been selected to show our results.

## 5.2 Sequential Scheme

In order to illustrate the effectiveness of the proposed memetic algorithm, we compared it with the best-known solutions. Table 2 and Table 3 show the best- known solutions, as well as the results of our algorithm using the original SBX operator, for two different values of the initial temperature. Table 4 and Table 5 show similar data for the SBSBX operator. The proposal using the initial temperature set to 10 attains much better results than with a temperature of 100. Moreover, SBSBX obtains better results than SBX in both cases, but with the temperature set to 10 the improvement is less noticeable than with the temperature set to 100. This means that SBSBX increases the robustness of the scheme, i.e. it's reduce the sensitivity to its correct parameterization.

Fig. 3 shows the evolution of the fitness attained by our proposal using two different temperatures. The temperature of 10 provided much better results. The reason is that this temperature induces a much more exploitative approach which is required to properly intensify in each region located by the memetic algorithm. The temperature of 100 provided worse results. The reason is that this temperature induces a much more explorative behavior but at the cost of reducing the intensification capabilities of the scheme. Analyses of diversity show that when using such a high temperature, the proposal does not converge, which is the reason of its inferior behavior.

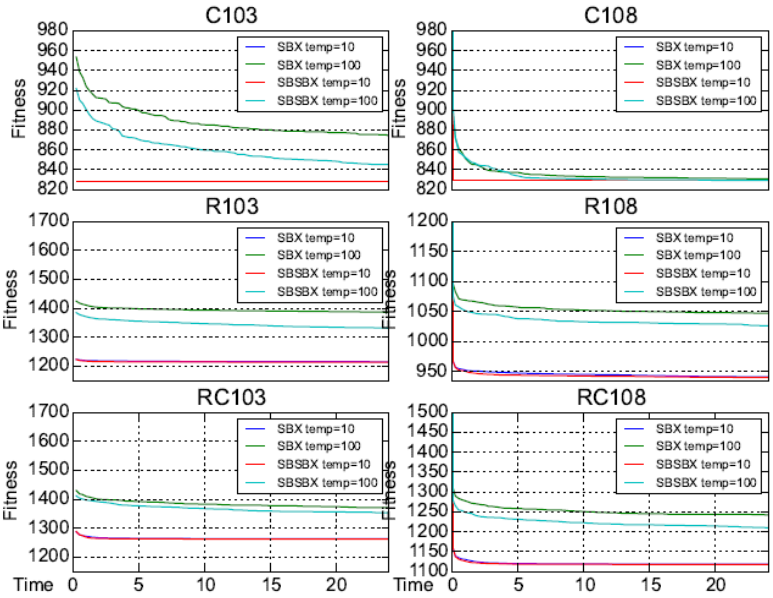


**Fig. 3.** Evolution of Fitness.

### 5.3 Parallel Scheme

The parallel variant of our proposal was tested with 3, 11, 26 and 51 processes. Note that one of them is the master, whereas the remaining ones are the slaves. In order to test its performance, the stopping criterion was established by taking into account the number of generations that are performed by the sequential scheme in 24 hours. Then, the time required to finish with the different number of slaves is calculated. Fig. 4 shows the ideal and observed speedup. Results show that our parallel version almost reaches the ideal speedup. Moreover, its performance does not decrease when more threads are used. The obtained speedup is practically the ideal speedup minus one. The reason is that in our proposal, the master does not perform any work during the local search phase. The speedup might probably be improved by adapting the master-slave model, so that the master works during the local search phase. In any case, this is left for future work.

**Table 3.** SBX Results with temperature = 100

| Instance | BKS | Mean | Median | std | Best | Worst |
|---|---|---|---|---|---|---|
| C103 | **826.3** | 874.25 | 875.80 | 16.44 | 847.30 | 903.00 |
| C108 | **827.3** | 830.98 | 829.92 | 2.33 | 828.94 | 837.18 |
| R103 | **1208.7** | 1386.11 | 1387.10 | 8.76 | 1366.30 | 1399.97 |
| R108 | **947.55** | 1047.29 | 1047.09 | 7.13 | 1025.85 | 1058.93 |
| RC103 | **1258** | 1370.73 | 1371.42 | 14.23 | 1332.19 | 1400.64 |
| RC108 | **1114.2** | 1242.57 | 1243.84 | 11.09 | 1205.49 | 1262.86 |

**Table 4.** SBSBX Results with temperature = 10

| Instance | BKS | Mean | Median | std | Best | Worst |
|---|---|---|---|---|---|---|
| C103 | **826.3** | 828.06 | 828.06 | 4.02e-07 | 828.06 | 845.04 |
| C108 | **827.3** | 828.94 | 828.94 | 8.95e-07 | 828.94 | 828.94 |
| R103 | **1208.7** | 1213.84 | 1213.73 | 0.51 | 1213.62 | 1216.05 |
| R108 | 947.55 | 940.22 | 939.06 | 2.63 | **938.20** | 947.03 |
| RC103 | **1258** | 1261.97 | 1262.02 | 0.26 | 1261.67 | 1262.98 |
| RC108 | **1114.2** | 1117.62 | 1117.52 | 0.36 | 1117.53 | 1119.25 |

## 6 Conclusions

The VRP is a very important NP-hard optimization problem that arises in the management of smart cities. Several variants of EAs have shown great promise in dealing with the VRP. However, when dealing with relatively large instances, they are not usually capable of obtaining optimal solutions. Thus, more efforts are required to improve their performance. This paper proposes a novel memetic algorithm to deal with a variant of the VRP, which is called the CVRPTW. Specifically, our proposal is a memetic algorithm that integrates a variant of SA with the recently proposed SBX crossover operator. In addition, a novel crossover operator (SBSBX) is proposed by extending the original SBX. This operator reduces the disruptive behavior that SBX shows in some particular cases. Since our sequential proposal requires large computational resources to attain high-quality results, a parallel variant based on the master-slave paradigm is also proposed.

Our experimental validation with a well-known set of benchmarks shows a really promising behavior. In several cases, results that are quite close to the best-known solutions are obtained. In addition, in the R108 instance, a new best-known solution could be attained, which is a remarkable achievement, especially when taking into account that this set of instances have been tackled with a large number of different techniques. The new SBSBX operator has been able to reduce the sensitivity of the memetic algorithm to its correct parameterization. Moreover, the parallelization shows an adequate performance, obtaining a speedup that is close to the optimal one, even when considering up to 51 slaves.

**Table 5.** SBSBX Results with temperature = 100

| Instance | BKS | Mean | Median | std | Best | Worst |
|---|---|---|---|---|---|---|
| C103 | **826.3** | 845.23 | 843.82 | 10.02 | 829.28 | 871.40 |
| C108 | **827.3** | 829.26 | 828.94 | 0.85 | 828.94 | 832.80 |
| R103 | **1208.7** | 1331.37 | 1332.13 | 14.67 | 1299.55 | 1361.81 |
| R108 | **947.55** | 1026.26 | 1026.58 | 8.01 | 1001.00 | 1042.01 |
| RC103 | **1258** | 1352.25 | 1351.70 | 11.51 | 1334.17 | 1372.20 |
| RC108 | **1114.2** | 1209.91 | 1210.25 | 11.12 | 1185.29 | 1234.61 |

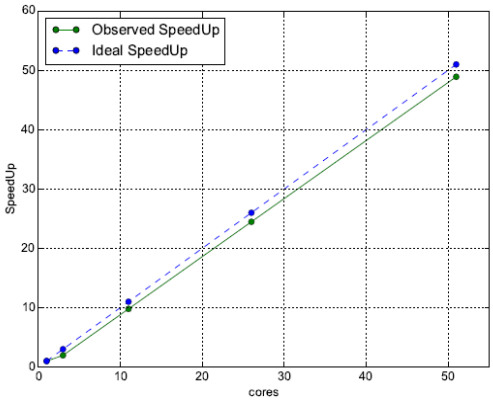


**Fig. 4.** Speedup.

Several lines of future work might be explored. First, in order to improve the parallelization, the model might be extended by including the master in the local-search phase. Second, more complex paradigms such as the island- based or cellular models might be used. Finally, in order to improve further the performance of the sequential approach, models that explicitly consider the diversity might be taken into account.

## References


1. Ahuja, R.K., Orlin, J.B., Sharma, D.: Very large-scale neighborhood search. Inter- national Transactions in Operational Research 7, 307–307 (2000)
2. Baldacci, R., Mingozzi, A., Robertic, R.: Recent exact algorithms for solving the vehicle routing problem under capacity and time window constraints. Operations Research 218, 1–6 (2012)
3. Baños, R., Ortega, J., Gil, C., de Toro, F., Montoya, M.G.: Analysis of openmp and mpi implementations of meta-heuristics for vehicle routing problems. Applied Soft Computing 43, 262–275 (2016)
4. Blanton, J., Wainwright, R.: Multiple vehicle routing with time and capacity constraints using genetic algorithms. Morgan Kaufmann Publishers. pp. 452–459 (1993)
5. Cook, W., Cunningham, W., Pulleyblank, Schrijver, A.: Combinatorial Optimization. John Wiley & Sons (1998)
6. Dantzig, G.B., Ramser, J.: The truck dispatching problem. Management Science 6, 90–91 (1959)
7. El-Sherbeny, N.A.: Vehicle routing with time windows: An overview of exact, heuristic and metaheuristic methods. Journal of King Saud University (Science) 1, 123–131 (2010)
8. Goel, A.: Vehicle scheduling and routing with drivers' working hours. Computers & Operations Research 43(1), 17–26 (2009)
9. Goldberg, D.: Genetic algorithms in search, optimization, and machine learning. New York: Addison Wesley (1989)
10. Gromicho, J., van Hoorn, J., Kok, A., Schutten, J.: Restricted dynamic programming: a flexible framework for solving realistic vrps. Computers & Operations Research (2011)
11. Khachaturyan, A., Semenovskaya, S., Vainshtein, B.: Statistical-thermodynamic approach to determination of structure amplitude phases. Sov.Phys. Crystallography 24, 519–524 (1979)
12. Çağrı Koç, Bektaş, T., Jabali, O., Laporte, G.: A hybrid evolutionary algorithm for heterogeneous fleet vehicle routing problems with time windows. Computers & Operations Research 64, 11–27 (2015)
13. Lee, L.H., Tan, K.C., Ou, K., YH, C.: Vehicle capacity planning system: a case study on vehicle routing problem with time windows. IEEE Trans Syst Man Cybern Part A: Syst Hum 33, 69–78 (2003)
14. Potvin, J.Y., Bengio, S.: The vehicle routing problem with time windows - part ii: Genetic search. Informs J Comput 8(2), 165–172 (1996)
15. Qi, Y., Hou, Z., Li, H., Huang, J., Li, X.: A decomposition based memetic algorithm for multi-objective vehicle routing problem with time windows. Computers & Operations Research 62, 61–77 (2015)
16. Segura, C., Coello, C.A.C., Miranda, G., León, C.: Using multi-objective evolutionary algorithms for single-objective constrained and unconstrained optimization. Annals of Operations Research 240(1), 217–250 (2016)
17. Solomon, M.M.: Algorithms for the vehicle routing and scheduling problems with time window constraints. Operations Research 35, 254–265 (1987)
18. Solomon, M., Desrosiers, J.: Time window constrained routing and scheduling problems. Transportation Science 22, 1–13 (1988)