Comparison of Similarity Measures for the Multi-Objective Vehicle Routing Problem with Time Windows

Abel Garcia-Najera
School of Computer Science
University of Birmingham
Birmingham B15 2TT, United Kingdom
A.G.Najera@cs.bham.ac.uk

John A. Bullinaria
School of Computer Science
University of Birmingham
Birmingham B15 2TT, United Kingdom
J.A.Bullinaria@cs.bham.ac.uk

ABSTRACT

The Vehicle Routing Problem can be seen as a fusion of two well known combinatorial problems, the Travelling Salesman Problem and Bin Packing Problem. It has several variants, the one with Time Windows being the case of study in this paper. Its main objective is to find the lowest-distance set of routes to deliver goods to customers, which have service time windows, using a fleet of identical vehicles with restricted capacity. We consider the simultaneous minimisation of the number of routes along with the total travel distance. Although previous research has considered evolutionary methods for solving this problem, none of them has concentrated on the similarity of solutions. We analyse here two methods to measure similarity, which are incorporated into an evolutionary algorithm to solve the multi-objective problem. We have applied this algorithm to a publicly available set of benchmark instances, and when these similarity measures are considered, our solutions are seen to be competitive or better than others previously published.

Categories and Subject Descriptors

I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search—*Heuristic methods.*

General Terms

Algorithms, performance, experimentation.

Keywords

Multi-objective optimisation, vehicle routing problem, similarity measures.

1. INTRODUCTION

Combinatorial optimisation problems can be found in many real world circumstances. Moreover, many of these problems have not only one, but several objectives to be optimised, which are frequently in conflict. So, instead of looking for a single permutation which gives the optimal solution, we search for arrangements to provide a set of solutions that allow *trade-offs* between such objectives.

There are many theoretical combinatorial problems that can be directly applied to real-life, such as the Travelling Salesman Problem, Bin Packing Problem, Job Shop Scheduling Problem, and Vehicle Routing Problem (VRP) [16]. These are of high importance for many industries; in particular, the VRP can be adopted in transportation logistics like post, parcel and distribution services.

The main objective of the VRP is to obtain the lowest-distance set of routes to deliver goods to customers, but we can also think about reducing the number of vehicles needed to do the deliveries. This means that we can consider the VRP as a multi-objective problem [8]. Moreover, the VRP has several variants of increased difficulty; in particular, the one with time windows (VRPTW), which has time as well as capacity constraints, forms the main problem to be studied in this paper.

Exact methods can be used to find optimal solutions for small instances of the VRPTW, but the computation time required increases considerably for larger sizes [5]. This is why we are interested in solving this problem by means of using heuristics, in particular, an evolutionary algorithm (EA).

There are many research studies that have used heuristics to solve the VRPTW as a single-objective problem. The recent surveys by Bräysy and Gendrau [1, 2] provide an excellent review of them. Lately, Homberger and Gehring [7] and LeBouthillier and Crainic [10] proposed hybrid methods, which combine an evolutionary algorithm and a tabu search procedure, to solve this problem, optimising, one at a time, the number of routes and the travel distance.

There have been also some multi-objective approaches, like that of Tan et al. [15], who used the *dominance rank* scheme to assign fitness to individuals. They designed a problem specific crossover operator called *route-exchange crossover* and used a multi-mode mutation which considered the swapping, splitting and merging of routes. They also used three local search heuristics which were applied every 50 generations. Ombuki et al. [12] also proposed the problem specific genetic operators *best cost route crossover* and *constrained route reversal mutation*, which is an adaptation of the widely used inversion method. None of these studies has considered to measure the similarity of solutions, even though it is known to be desirable to preserve the diversity of solutions in the evolutionary populations [19].

GECCO'09, July 8–12, Montréal Québec, Canada.

Well known methods like SPEA2 [20] and NSGA-II [4] do have diversity preservation tools, but their use would not be appropriate here because they require the definition of niche spaces, a task that would be problematic since most good solutions to the VRPTW reside in a very small portion of the discrete vehicle number dimension [12].

To measure the similarity of solutions, one must take into account the solution encoding, because different representations may require different methods. Besides, if we considered similarity in the objective space, we could be led to unreliable measures. Sörensen [14] proposed that the *Edit distance* could be applied to routing problems using a permutation-based encoding, and, more recently, Garcia-Najera and Bullinaria [6] proposed a straightforward method to measure similarity of solutions to the VRPTW, based on Jaccard's similarity coefficient, which is independent of the representation used.

The work presented here is concerned with the solution of the VRPTW as a multi-objective problem, using an EA which incorporates a similarity measure applied in the genotype space. We compare and analyse the results obtained when the measure used is the Edit distance [14] and Jaccard similarity [6]. We have tested this algorithm on publicly available benchmark instances, and when our results are compared with those from recent publications, our algorithm appears very competitive.

The remainder of this paper is organised as follows. First, in Section 2, we provide a brief description of what multi-objective optimisation problems are, and explain a couple of key performance metrics. Next, in Section 3, we introduce the VRPTW and the objective functions we aim to optimise. In Section 4 we review the similarity measures we use in this work. Our proposed EA for solving the VRPTW as a multi-objective problem is described in Section 5. In Section 6 we present the results from our work, as well as a comparison with some others already published. Finally, we give our conclusions in Section 7.

2. MULTI-OBJECTIVE OPTIMISATION

Any multi-objective optimisation problem can, without loss of generality, be defined as a minimisation problem of the form:

$$\text{minimise } \mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), ..., f_k(\mathbf{x})) \tag{1}$$

subject to constraints:

$$g_i(\mathbf{x}) \leq 0, \qquad \forall\, i = 1, 2, ..., m \tag{2}$$

$$h_j(\mathbf{x}) = 0, \qquad \forall\, j = 1, 2, ..., p \tag{3}$$

where $\mathbf{x} = (x_1, x_2, ..., x_n) \in \mathcal{X}$ is the vector of decision variables, $\mathcal{X}$ is the parameter space, and $f_i : \mathbb{R}^n \rightarrow \mathbb{R}$, for $i = 1, ..., k$, are the objective functions. The constraint functions $g_i, h_j : \mathbb{R}^n \rightarrow \mathbb{R}$ in (2) and (3) restrict $\mathbf{x}$ so that only feasible solutions are considered.

A decision vector $\mathbf{x} \in \mathcal{X}$ is said to *dominate* a decision vector $\mathbf{y} \in \mathcal{X}$ (written as $\mathbf{x} \prec \mathbf{y}$) if and only if $f_i(\mathbf{x}) \leq f_i(\mathbf{y})\ \forall\, i = 1, 2, ..., k$ and $\exists\, j \in \{1, 2, ..., k\} : f_j(\mathbf{x}) < f_j(\mathbf{y})$. Similarly, we say that a decision vector $\mathbf{x} \in \mathcal{X}$ is *non-dominated* if there is no decision vector $\mathbf{y} \in \mathcal{X}$ such that $\mathbf{y} \prec \mathbf{x}$. A decision vector $\mathbf{x} \in \mathcal{X}$ is said to be *Pareto optimal* if it is non-dominated. The *Pareto optimal set* is defined as $\mathcal{P}_s = \{\mathbf{x} \in \mathcal{X} \mid \mathbf{x} \text{ is Pareto optimal}\}$. Finally, the *Pareto front* is defined as $\mathcal{P}_f = \{\mathbf{f}(\mathbf{x}) \in \mathbb{R}^n \mid \mathbf{x} \in \mathcal{P}_s\}$.

The comparison of the performance between multi-objective optimisers is not easy, as their outcome, which is commonly called the *Pareto approximation*, is a set of solutions, in contrast with the single-objective case, where we can compare straightforwardly the best solution, or the average of them, from the methods studied. For this reason, the use of appropriate performance metrics is crucial. This field of research has been widely studied and remains a subject of investigation [18, 3, 21, 9]. We, in this work, will focus on only two of these metrics. The first is the one proposed by Zitzler et al. [18], that shows to what extent one set of solutions is covered by another, and which we will call *coverage*. The second is that from Deb and Jain [3], which evaluates the convergence of one set towards another, and we will call *convergence*. The following provides formal definitions of these metrics. For each of them, $\mathcal{R}$ is a reference set of solutions, and $\mathcal{A}$ a Pareto approximation.

2.1 Coverage

This performance metric measures the coverage of the reference set $\mathcal{R}$ by the approximation set $\mathcal{A}$. The function $\mathcal{M}_1$ maps the ordered pair $(\mathcal{A}, \mathcal{R})$ to the interval [0,1] [18]:

$$\mathcal{M}_1(\mathcal{A}, \mathcal{R}) = \frac{|\{\mathbf{r} \in \mathcal{R} : \exists\, \mathbf{a} \in \mathcal{A}, \mathbf{a} \preceq \mathbf{r}\}|}{|\mathcal{R}|} \tag{4}$$

The value $\mathcal{M}_1 = 1$ means that all solutions in $\mathcal{R}$ are covered by or equal to solutions in $\mathcal{A}$, while $\mathcal{M}_1 = 0$ indicates the opposite situation, i.e. in which none of the solutions in $\mathcal{R}$ are covered by $\mathcal{A}$.

2.2 Convergence

This metric measures the convergence of the approximation set $\mathcal{A}$ towards the reference set $\mathcal{R}$. To define this metric, we need first to calculate the smallest normalised Euclidean distance $d_{\mathbf{i}}$ from each point $\mathbf{i} \in \mathcal{A}$ to the reference set $\mathcal{R}$ as follows [3]:

$$d_{\mathbf{i}} = \min_{\mathbf{j} \in \mathcal{R}} \sqrt{\sum_{k=1}^{M} \left(\frac{f_k(\mathbf{i}) - f_k(\mathbf{j})}{f_k^{\max} - f_k^{\min}} \right)^2} \tag{5}$$

where $f_k^{\max}$ and $f_k^{\min}$ are the maximum and minimum function values of the k-th objective function in $\mathcal{R}$. Then we define the convergence $\mathcal{M}_2$, for the pair $(\mathcal{A}, \mathcal{R})$, as

$$\mathcal{M}_2(\mathcal{A}, \mathcal{R}) = \frac{1}{|\mathcal{A}|} \sum_{\mathbf{i} \in \mathcal{A}} d_{\mathbf{i}} \tag{6}$$

i.e. the average normalised distance for all points in $\mathcal{A}$.

3. THE VEHICLE ROUTING PROBLEM WITH TIME WINDOWS

The VRP is a combinatorial problem which can be seen as a fusion of the well known Travelling Salesman Problem and the Bin Packing Problem. The general VRP has several variants of increased difficulty; in particular, the one with time windows (VRPTW) has both capacity and time constraints. The VRPTW can be formally defined as follows. Given:

- a set $\mathcal{V} = \{v_1, \ldots, v_n\}$ of vertices, called customers, with known demands $d_i > 0$, $\forall\, i \in \{1, \ldots, n\}$,
- a special node v_0, called *depot*, with $d_0 = 0$,
- a symmetric distance c_{ij} between any pair of vertices, $\forall\, i, j \in \{0, \ldots, n\}, i < j$,

• a time window $[b_i, e_i]$ associated with each customer $v_i \in \mathcal{V}$ during which the customer has to be supplied,
• an unload time s_i associated with each customer $v_i \in \mathcal{V}$, and
• a fleet of identical vehicles with limited capacity $Q \geq \max\{d_i : i \in \{1, \ldots, n\}\}$,
we have to design a minimum-distance set of routes, so that each route begins and ends at the depot and each customer is serviced by exactly one vehicle.

Since every customer has a service time window, a solution becomes infeasible if customer v_i is supplied after e_i. Moreover, if a vehicle arrives at customer v_i before b_i, a waiting time has to be added to the travel time. A solution also becomes infeasible if the total load on any vehicle is greater than Q.

Let us denote as $r_k = \langle u_1^k, \ldots, u_{n_k}^k \rangle$ the k-th route that supplies n_k customers, with u_i^k the i-th customer to visit in the route. Then we can define the travel distance

$$C_k = c_{0u_1^k} + \sum_{i=1}^{n_k - 1} c_{u_i^k u_{i+1}^k} + c_{u_{n_k}^k 0} \tag{7}$$

associated with route r_k.

Now that we have defined the problem, we can identify the two objective functions that we concentrate on minimising in this study. Let $\mathcal{R} = \{r_1, \ldots, r_m\}$ be the set of designed routes. The first objective function is then

$$f_1(\mathcal{R}) = |\mathcal{R}| \tag{8}$$

i.e. the number of routes, and the second is

$$f_2(\mathcal{R}) = \sum_{k=1}^{|\mathcal{R}|} C_k \tag{9}$$

i.e. the total travel distance.

4. SIMILARITY MEASURES

This study will consider two different route similarity measures. The first is based on the Edit distance [14], which is defined as the minimal number of edit operations required to transform one string into another. The second is based on the Jaccard's similarity coefficient [6], which essentially measures the ratio between the common elements in two sets and the total number of elements.

4.1 Edit distance

The edit distance is based on Levenshtein distance [11], which was first introduced in the field of error correcting codes for dealing with binary strings. The edit distance between two binary strings is the minimal number of edit operations required to transform one of the strings into the other. These edit operations are defined as follows [14]:

Reversal: $0 \rightarrow 1$ or $1 \rightarrow 0$
Deletion: $0 \rightarrow \Lambda$ or $1 \rightarrow \Lambda$
Insertion: $\Lambda \rightarrow 0$ or $\Lambda \rightarrow 1$

where Λ is the null-character, specifying the absence of a character, i.e. $|\Lambda| = 0$.

Wagner and Fischer [17] extended the work of Levenshtein, first, by considering that strings are composed of any finite alphabet, and second, by widening the reversal operation, which became substitution when one character is converted into another. They also provided a dynamic programming algorithm to calculate the edit distance, which is $O(n^2)$, n being the length of the strings, and is publicly available from a number of world wide web sites.

4.2 Jaccard similarity coefficient

The Jaccard's similarity coefficient compares two sets A and B, dividing the cardinality of the intersection of the sets by the cardinality of their union, i.e. $J(A, B) = \frac{|A \cap B|}{|A \cup B|}$. It is easy to observe that if sets A and B do not share any element at all, $|A \cap B| = 0$, so $J(A, B) = 0$. On the other hand, if A and B contain the same elements, that is $A = B$, $|A \cap B| = |A \cup B|$, so $J(A, B) = 1$.

Now we can define the similarity between two solutions to the VRPTW, according to the Jaccard's similarity coefficient, simply as the ratio of the number of shared arcs to the total number of arcs used in both solutions [6].

Defining $y_{ijk} = 1$ if the arc (i, j) from vertex i to vertex j is used in any route in solution k, and 0 otherwise, the similarity ς_{pq} between solutions p and q can be written as

$$\varsigma_{pq} = \frac{\sum_{i=0}^{n} \sum_{j=0}^{n} y_{ijp} \cdot y_{ijq}}{\sum_{i=0}^{n} \sum_{j=0}^{n} \text{sign}\,(y_{ijp} + y_{ijq})} \tag{10}$$

where $y_{ijp} \cdot y_{ijq}$ will only equal 1 if the arc (i, j) is used in both solutions, while $\text{sign}\,(y_{ijp} + y_{ijq})$ will equal 1 if it is used in any of them. If solutions p and q are two completely different solutions with no arc in common, the numerator will equal 0, and therefore $\varsigma_{pq} = 0$. On the contrary, if they are the same solutions, the sum in the numerator will equal the sum in the denominator, and hence $\varsigma_{pq} = 1$.

The complexity of the algorithm designed by Garcia-Najera and Bullinaria [6] for computing the Jaccard's similarity measure is $O(n)$, where n is the number of customers in the instance.

5. EVOLUTIONARY ALGORITHM FOR SOLVING THE VRPTW

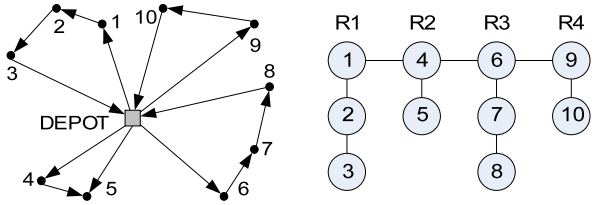
We present in this section our proposed EA for solving the VRPTW as a multi-objective problem. We give details about the encoding, the stages of processing, and how we incorporate the similarity measures described above.

5.1 Encoding of solutions

We are using a binary tree representation. In this encoding, a right child points to the following route in the solution, while the left represents the next customer to visit in a route. A solution for an example instance and its representation are shown in Figure 1. In this case, the assignment of customers to routes, and the sequence in which they will be visited within each route, is as follows: customers 1, 2 and 3 to the first route, customers 4 and 5 to the second, 6, 7 and 8 to the third, and 9 and 10 to the fourth.

5.2 Initial population

Our algorithm starts by building a set of feasible random solutions. Each of these solutions contains a set of randomly generated routes. Such routes are constructed in the following way: First, a customer is selected and placed as the first to be visited on that route. Then, a second customer is chosen and, if the capacity and time constraints are met, it is placed after the previous one. If any of the constraints are not met, a new route is created and this customer will be the first to be visited in the new route. This process is repeated until all customers are assigned.



(a) Solution (b) Encoding

Figure 1: The solution and its encoding for an example instance of the VRPTW.

5.3 Fitness assignment

When solving a single-objective problem using an EA, fitness is assigned to an individual according to its objective function evaluation. In the multi-objective case, this assignment cannot be done straightforwardly, due to there being not only one, but at least two objective functions. We have used in this work the non-dominance sort criteria [4] to assign fitness to solutions, where the population is divided into several non-dominated fronts and the depth specifies the fitness of the individuals belonging to them.

5.4 Selection of parents

The evolutionary process starts with the selection of two parents that are going to be submitted to the crossover process. We have used a binary tournament selection method for selecting individuals, but under two different criteria. The first parent will always be selected according to the fitness. For selecting the second parent, we will be comparing nine different methods, each of which could plausibly be the best way to proceed. The first of them is to select the second parent, like the first, according to the fitness. This method will be identified by the letter F. The other methods perform selection based on one of the two similarity measures, denoted J for Jaccard and E for Edit distance, which can each be used in four different ways: The value of the similarity measure can be computed to be the average similarity of each individual with respect to the entire population, or the similarity of each individual with regard to the first parent. For both of these, we can choose whether the more or the less similar individual is selected. We will use the labels -A, +A, -P and +P, for the less similar on average, the more similar on average, the less similar to the first parent, and the more similar to the first parent, respectively. This nomenclature will be used when presenting the results in Section 6, e.g. J-A refers to the algorithm that uses Jaccard's similarity and selects the second parent to be the less similar on average.

5.5 Crossover

The evolution proceeds with the crossover of the two selected parents. The crossover of two example parents is shown in Figure 2. Here, the algorithm aims at preserving routes from both parents. First, a random number of routes are chosen from the first parent and copied into the offspring. Next, all those routes from the second parent which are not in conflict with the customers already copied from the first, are replicated into the offspring. In our example, both routes on the left from the first parent were selected to be copied into the offspring, and we can only copy from

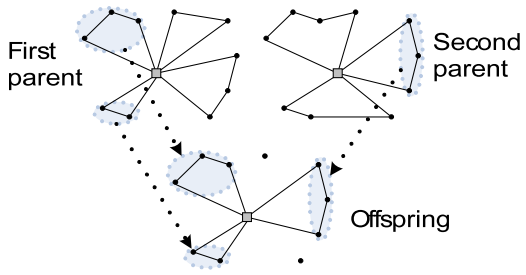
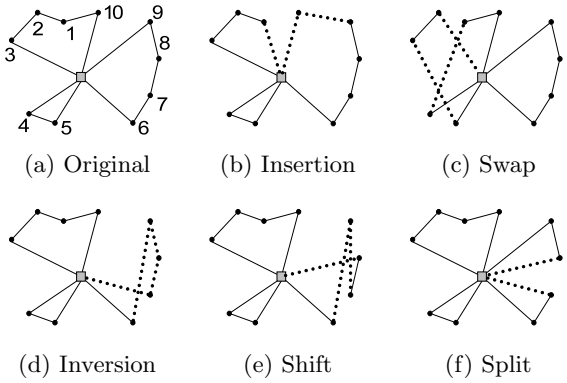


Figure 2: The crossover process.



(a) Original (b) Insertion (c) Swap

(d) Inversion (e) Shift (f) Split

Figure 3: The mutation operators.

the second parent the route on the right, as the other two contain customers already present in the offspring. If there remain unassigned customers, these are allocated, in the order they appear in the second parent, to the route where the lowest travel distance is achieved, like in the example given in Figure 3(a). If a solution would become infeasible after inserting such a customer, a new route is created.

5.6 Mutation

We have designed a composite mutation process with five possible operators, which can be categorised as *inter-* and *intra-route*. In the former, the algorithm will perform changes between two routes, thus modifying the assignment of customers to routes, and in the latter, the changes will be done within a route, hence affecting the travel sequence.

In the first category, we can identify two viable processes which are: (*i*) removing a sequence of customers from a route and *inserting* it into another, and (*ii*) *swapping* two sequences of customers from different routes. In the case of intra-route, we use three operations: (*i*) the *inversion* of the sequence of a sub-route, (*ii*) the *shift* of one customer, and (*iii*) *splitting* a route. Examples of these operations are shown graphically in Figure 3. The dotted lines in each figure represent the changes in the sequences. In Figure 3(b), customer 10 was removed from the route on the left and has been inserted in the route on the right. Figure 3(c) shows the swap of customer 4 with customers 2 and 3. In Figure 3(d) we have the inversion of the sequence of customers 7, 8 and 9. Figure 3(e) shows how customer 9 has been shifted between customers 6 and 7. Finally, in Figure 3(f), the route on the right has been split between customers 7 and 8.

Not all of the mutation operators are applied each time an offspring is mutated. First, the split operator is performed

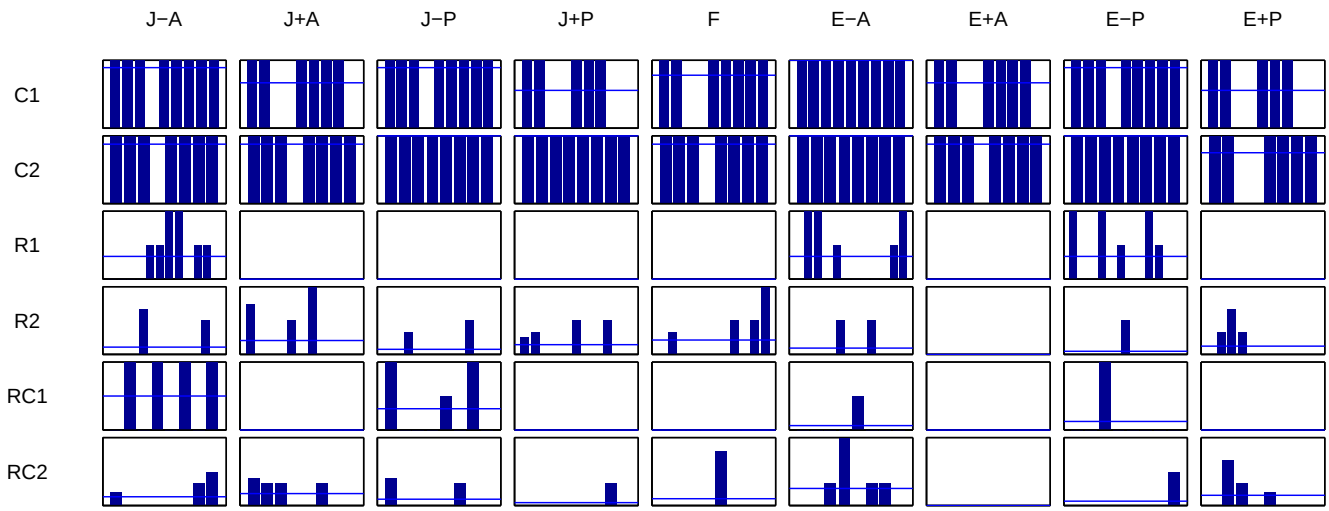


Figure 4: Results from the coverage metric. The bars represent the extent to which the composite reference set is covered by the overall Pareto approximation for each method.

with a probability equal to the inverse of the number of routes in the solution. Then, the solution is submitted to one of the inter-route operators. The decision of which to apply is random. Finally, one of the intra-route operators is applied to the solution.

5.7 Elitism

After the selection, crossover and mutation processes, the algorithm evaluates the objective functions for each solution in the offspring population, and combines both parent and offspring populations to assign fitness. Those solutions belonging to the best fitness fronts are taken to form the next generation. For the final such front, if that introduces conflict with the population size, similarity is computed for the solutions in it, and the least similar are taken.

6. EXPERIMENTS AND RESULTS

Our experimental set-up has two purposes. First, we are going to compare, by means of using the two performance metrics previously described, the results from our set of nine algorithms introduced in Section 5. Then we are going to compare the lowest distances found by our algorithms with those presented in recent publications, in order to see how well they perform compared with existing approaches.

To do this, we used the standard benchmark set[1] due to Solomon [13] that includes 56 instances of size $n = 100$. These instances are categorised as Clustered (C1, C2), Random (R1, R2), and mixed (RC1, RC2), and have been previously studied in detail. A recent analysis by Tan et al. [15] suggests that categories C1 and C2 have positively correlating objectives, which means that the travel distance of a solution increases with the number of routes. However, the majority of the instances in categories R1, R2, RC1 and RC2 have conflicting objectives.

We ran each of our algorithms 30 times for every instance and recorded the solutions in the Pareto approximation each time. Finally, we built an overall Pareto approximation for each instance and algorithm from the 30 approximation sets, in line with what has become the common practice for measuring performance [18, 3].

The parameters of our algorithm were set to values that have proved to work well in previous studies, namely: population size = 100, number of generations = 500, tournament size = 10, crossover rate = 0.9, and mutation rate = 0.1.

To compute the edit distance between two solutions, we first calculated the distance between each route in one of the solutions and all routes in the other. We then summed the smallest distance for every route in one of the solutions, and finally averaged it by the number of routes

6.1 Application of performance metrics

To evaluate the results using the two performance metrics, we constructed for each instance a composite reference set of solutions from the overall Pareto approximations, and took that to be $\mathcal{R}$ in equations (4) and (6). Then each overall Pareto approximation played the role of $\mathcal{A}$.

Results from the two performance metrics are shown in Figures 4 and 5. Each of these figures consists of six rows, which represent the six instance set categories, and nine columns, one for each combination of similarity measure and selection technique. For the box in a given row and column, we present a plot of bars, which correspond to the instances in that category. The height of the box is unitary. There is also a line across the bars located on the average value for that instance set category.

Figure 4 shows the the coverage metric results. This metric computes the extent to which the composite reference set is covered by the overall Pareto approximation, so the higher the bar, the larger the coverage of the reference set. On the contrary, if a given instance shows a small bar, it means that the coverage of the reference set is small.

For example, for categories C1 and C2, all the algorithms found the optimal solutions for almost all the instances, except E-A which found all of them. We can also observe that boxes for all the other categories in the E+A column do not show any bars, which means that solutions found with this method do not cover any solutions in the reference set. If we look at the boxes for categories R1 and RC1, we can see that

[1] `http://w.cba.neu.edu/~msolomon/home.htm`

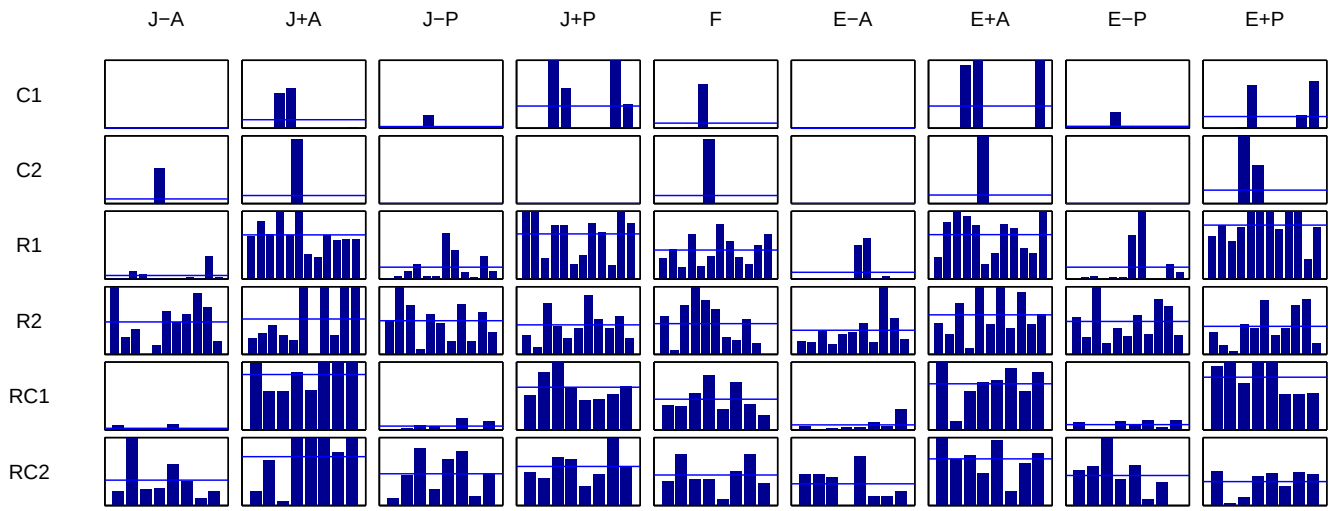


Figure 5: Results from the convergence metric. The bars represent the average normalised distance from all points in the Pareto approximation for each method to the composite reference set.

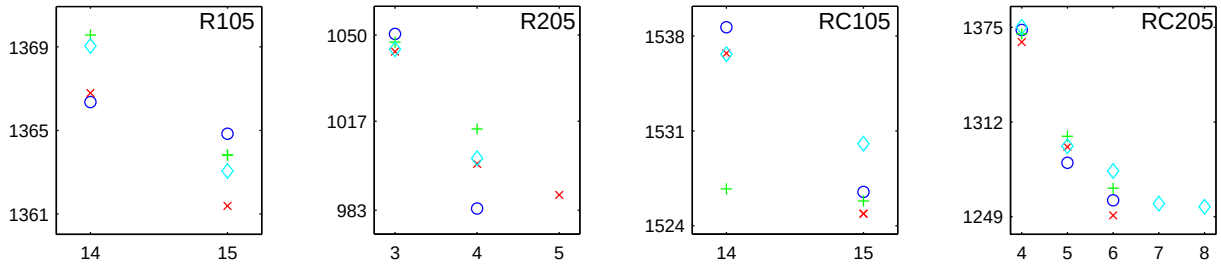


Figure 6: Pareto approximations for four instances in categories R and RC. Markers '○', '+', '×', and '◇' represent solutions found by techniques J-A, J-P, E-A and E-P respectively.

those in columns J-A, J-P, E-A and E-P are the only ones showing bars, which means that these are the only methods that are contributing solutions to the composite reference set. Additionally, we can see that for instances in categories R2 and RC2, solutions from all methods, except for those from E+A, cover to some extent the solutions in the reference set. Based on this visual analysis, we can say that the Pareto approximations from J-A, J-P, E-A and E-P have, on average, a better coverage of the reference sets than the other five combinations.

Figure 5 shows the convergence metric results. This metric measures the average distance from all points in the Pareto approximation to the reference set. This means that when solutions in the Pareto approximation are closer to the reference set, the bars are smaller. We have normalised the values so that we can make visible comparisons.

We can observe that solutions for instances in categories C1, C2, R1 and RC1, found by techniques J-A, J-P, E-A and E-P, are clearly, on average, closest to the reference set, while those from J+A, J+P, E+A and E+P are the most distant. This is consistent with the results obtained with the coverage metric, as these methods were the only ones contributing solutions to the composite reference set. We can also see that, for instances in category RC2, solutions in columns J+A and E+A are, on average, the most distant to the reference set. In the case of the category R2, there is no clear visible difference. Taking these observations into account, we can state that, on average, solutions found by J-A, J-P, E-A and E-P, are the closest to the reference set.

From Figures 4 and 5 together we can argue that methods J-A, J-P, E-A and E-P performed better that the others. Given these results, we now present in Figure 6 the Pareto approximations from these four methods for one typical instance in categories R1, R2, RC1 and RC2. This figure shows four boxes corresponding to instances R105, R205, RC105 and RC205, in which are displayed the solutions in the Pareto approximation found with the techniques specified. The horizontal axis correspond to the number of routes and the vertical axis to the travel distance.

Consider, for example, the plot for instance R105. In this we can observe that the solutions with the lowest distance were found by J-A (○) and E-A (×), for 14 and 15 routes respectively. This means that solutions in these approximation sets dominate the solutions found by J-P (+) and E-P (◇). We can see that solutions from the same techniques also dominate for instances R205 and RC205. In the case of instance RC105, the dominating solutions are those from J-P and E-A. These four methods had the same performance for the 17 instances in categories C1 and C2.

Unfortunately, due to space limitations, we cannot present the analysis for the remaining instances in categories R1, R2, RC1 and RC2, but solutions found by techniques J-A and

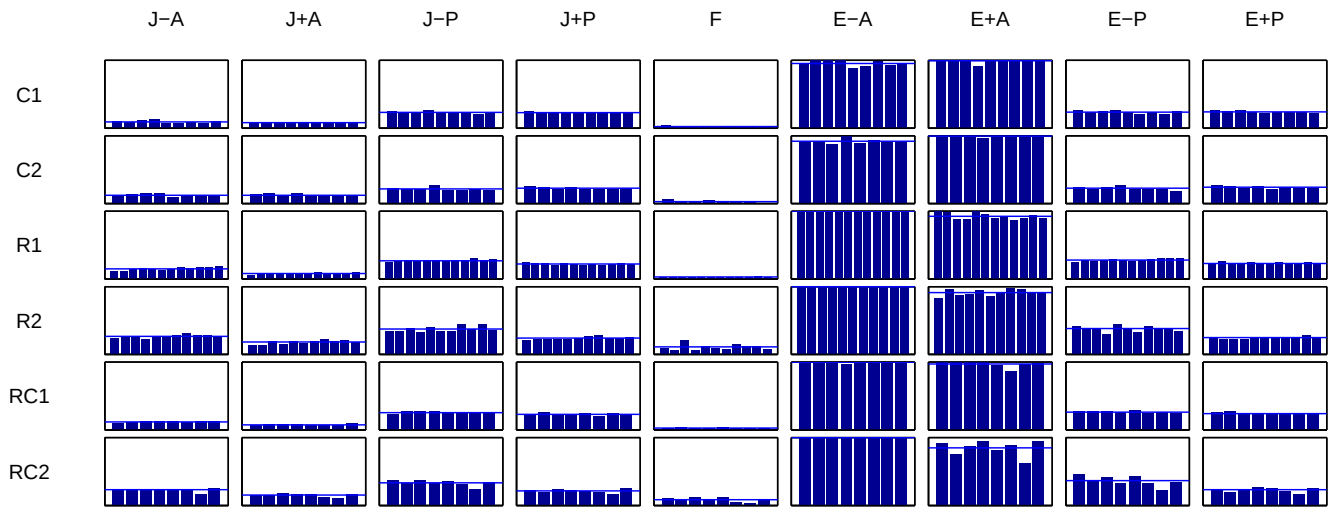


Figure 7: Normalised average execution times for runs of 500 generations.

E-A together dominate the others in 20 out of 39 of them. Based on these results, we can say that techniques J-A and E-A performed, on average, better than J-P and E-P.

Finally, it is relevant to present here the execution times for all our tested algorithms. Figure 7 shows the average execution time, for a run of 500 generations, for all instances and combinations of similarity measure and selection technique. It is divided in six rows, one for each instance set category, and nine columns, one for each selection method. For each box in a given row and column, there is a plot of bars representing the average execution time for each instance in that set. The execution time is normalised to the maximum execution time in all techniques.

It is evident that techniques E, using the edit distance, occupy, on average, more processing time, while the technique F, with no similarity measure, is the quickest. We can also see that techniques J-A and J+A executed faster than J-P and J+P. These behaviours are what would be expected, as we explained earlier that the complexity of the edit distance algorithm is quadratic, in contrast with the linear execution time of the Jaccard similarity measure.

6.2 Comparison with recent publications

Recently published research on the VRPTW has not presented Pareto approximations, even though it was considered as a multi-objective problem. Thus, we cannot compare our results with them using proper multi-objective methods. Instead, we have averaged the lowest distance in all iterations over the instances in each category, as in the literature this is the common way to present and compare results for this specific problem. Table 1 shows the results from our techniques J-A and E-A, and the results from four recent publications that minimise both objectives, number of routes and total distance, either one after another [7, 10] or simultaneously [15, 12]. We show for each algorithm and instance set the average number of routes (upper) and the average total distance (lower). The last column presents the total accumulated sum, indicating the total number of routes and the total distance for all 56 instances.

Analysing this table, we can observe that our methods J-A and E-A obtained the lowest distances for categories R1 and RC1, in spite of using a larger number of routes than the others. For categories R2 and RC2, they are the second best, and use fewer routes than the algorithm which found the lowest distances. In the case of the accumulated total distance, E-A found the lowest, while J-A found the third best, though they are using a larger number of routes than the most expensive solutions.

Table 1: Comparison of the best results, averaged for each category, with others previously published.

Alg.	R1	R2	RC1	RC2	Accum.
[10]	12.08	2.73	11.50	3.25	407.00
	1209.19	960.95	1386.38	1133.30	57412.37
[7]	11.91	2.73	11.50	3.25	405.00
	1212.73	955.03	1386.44	1108.52	57192.00
[15]	12.92	3.55	12.38	4.25	441.00
	1187.35	951.74	1355.37	1068.26	56290.48
[12]	13.17	4.55	13.00	5.63	471.00
	1204.48	893.03	1384.95	1025.31	55740.33
J-A	12.92	3.91	12.50	4.63	449.00
	1185.61	919.74	1343.89	1061.55	55766.89
E-A	12.75	4.18	12.50	5.13	454.00
	1186.70	915.16	1347.59	1054.22	55695.90

7. CONCLUSIONS

We have analysed the performance of our EA for solving the multi-objective VRPTW, which employs a similarity measure for controlling population diversity. We have tested in this algorithm nine different techniques for selecting the second parent for the crossover operation, any of which might reasonably be expected to work best. The first simply uses fitness (F), as for the first parent. The others base selection on one of two recently proposed methods to measure similarity of solutions for this problem: Jaccard similarity (J) and Edit distance (E). For each we can select the second parent to be the least (-A) or the most (+A) similar on average, or the least (-P) or the most (+P) similar to the first parent.

This comparison analysis was carried out using two standard multi-objective performance metrics available in the literature, which are coverage and convergence with respect to a composite reference set. The first of them indicates to what extent a set of solutions cover those in the reference set. The second metric measures the average distance between a set of solutions and the reference set. The results obtained using these metrics suggests that techniques J-A, J-P, E-A and E-P performed better than the others, as the solutions found by them have, on average, a better coverage and are closer to the reference set.

We have also explicitly examined the Pareto approximations from these techniques for 39 VRPTW problem instances, and found that solutions obtained by techniques J-A and E-A together dominated the others in more than a half of the instances, and in more if they are combined with those solutions obtained by techniques J-P and E-P.

The execution time of our algorithm was also reviewed. We found that the technique F, with no similarity measure, was the quickest, followed by J+A and J-A, and that E-A and E-P were the slowest.

When the solutions from our algorithm using J-A and E-A are compared with those from recent publications, although our results are not the overall best, they are better than some, and, on average, competitive. Additionally, our algorithms managed to find solutions for which the accumulated travel distance was lower than others, despite their having a higher number of routes, illustrating the need to treat the VRPTW as a multi-objective problem.

Given the promising performance of our algorithm, we are now looking at the possibility of minimising one objective more, which could be the waiting time, to see if our algorithm preserves its good performance. We are also pursuing the comparison of our results with other evolutionary multi-criterion optimisation methods.

8. ACKNOWLEDGMENTS

The first author acknowledges support from the Mexican National Council of Science and Technology (CONACYT), through scholarship 179372, to pursue his graduate studies.

9. REFERENCES

[1] O. Bräysy and M. Gendreau. Vehicle routing problem with time windows, part I: Route construction and local search algorithms. *Transport. Sci.*, 39(1):104–118, 2005.

[2] O. Bräysy and M. Gendreau. Vehicle routing problem with time windows, part II: Metaheuristics. *Transport. Sci.*, 39(1):119–139, 2005.

[3] K. Deb and S. Jain. Running performance metrics for evolutionary multi-objective optimization. In L. Wang, K. C. Tan, T. Furuhashi, J.-H. Kim, and X. Yao, editors, *4th Asia-Pacific Conference on Simulated Evolution and Learning*, pages 13–20, Singapore, November 2002. Nanyang Technical University.

[4] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE T. on Evolut. Comput.*, 6(2):182–197, 2002.

[5] M. Desrochers, J. Desrosiers, and M. Solomon. A new optimization algorithm for the vehicle routing problem with time windows. *Oper. Res.*, 40(2):342–354, 1992.

[6] A. Garcia-Najera and J. A. Bullinaria. Bi-objective optimization for the vehicle routing problem with time windows: Using route similarity to enhance performance. In M. Ehrgott, C. Fonseca, X. Gandibleux, J. K. Hao, and M. Sevaux, editors, *5th International Conference on Evolutionary Multi-Criterion Optimization*, 2009. To appear.

[7] J. Homberger and H. Gehring. A two-phase hybrid metaheuristic for the vehicle routing problem with time windows. *Eur. J. Oper. Res.*, 162:220–238, 2005.

[8] N. Jozefowicz, F. Semet, and E.-G. Talbi. Multi-objective vehicle routing problems. *Eur. J. Oper. Res.*, 189:293–309, 2008.

[9] J. Knowles, L. Thiele, and E. Zitzler. A tutorial on the performance assessment of stochastic multiobjective optimizers. TIK Report 214, Computer Engineering and Networks Laboratory (TIK), ETH Zurich, 2006.

[10] A. Le Bouthillier and T. G. Crainic. A cooperative parallel metaheuristic for vehicle routing with time windows. *Comput. Oper. Res.*, 32:1685–1708, 2005.

[11] V. I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Sov. Phys. Doklady*, 10(8):707–710, 1966.

[12] B. Ombuki, B. J. Ross, and F. Hanshar. Multi-objective genetic algorithms for vehicle routing problem with time windows. *Appd. Intel.*, 24(1):17–30, 2006.

[13] M. M. Solomon. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Oper. Res.*, 35(2):254–265, 1987.

[14] K. Sörensen. Distance measures based on the edit distance for permutation-type representations. *J. Heuristics*, 13(1):35–47, 2007.

[15] K. C. Tan, Y. H. Chew, and L. H. Lee. A hybrid multiobjective evolutionary algorithm for solving vehicle routing problem with time windows. *Comput. Optim. and Appl.*, 34(1):115–151, 2006.

[16] P. Toth and D. Vigo. *The vehicle routing problem.* SIAM, Philadelphia, PA, USA, 2001.

[17] R. A. Wagner and M. J. Fischer. The string-to-string correction problem. *J. Assoc. Comput. Mach.*, 21(1):168–173, 1974.

[18] E. Zitzler, K. Deb, and L. Thiele. Comparison of multiobjective evolutionary algorithms: Empirical results. *Evol. Comput.*, 8(2):173–195, 2000.

[19] E. Zitzler, M. Laumanns, and S. Bleuler. A tutorial on evolutionary multiobjective optimization. In X. Gandibleux, M. Sevaux, K. Sörensen, and V. T'kindt, editors, *Metaheuristics for Multiobjective Optimisation*, pages 3–38. Springer-Verlag, 2004.

[20] E. Zitzler, M. Laumanns, and L. Thiele. SPEA 2: Improving the Strength Pareto Evolutionary Algorithm for multiobjective optimization. In K. Giannakoglou, D. Tsahalis, J. Periaux, K. Papailiou, and T. Fogarty, editors, *Evolutionary Methods for Design, Optimisation and Control*, pages 19–26. CIMNE, Barcelona, Spain, 2002.

[21] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. G. da Fonseca. Performance assesment of multiobjective optimizers: An analysis and review. *IEEE T. Evolut. Comput.*, 7(2):117–132, 2003.