Algorithms for the Multi-Objective Vehicle Routing Problem with Hard Time Windows and Stochastic Travel Time and Service Time

Douglas M. Miranda [a], Juergen Branke [b], Samuel V. Conceição [a]

[a] Department of Production Engineering, Federal University of Minas Gerais, 6627, Antônio Carlos Avenue, Belo Horizonte, Brazil

[b] Warwick Business School, The University of Warwick, Coventry CV4 7AL, UK

E-mails:
Douglas M. Miranda
douglasmiranda@ufmg.br

Juergen Branke
Juergen.Branke@wbs.ac.uk

Samuel V. Conceição
svieira@dep.ufmg.br

Corresponding Author:

Douglas Moura Miranda, PhD.
Federal University of Minas Gerais
6627, Antônio Carlos Avenue, Belo Horizonte, Brazil
Tel: (+55) 31 - 992059811
Email: douglasmiranda@ufmg.br

Algorithms for the Multi-Objective Vehicle Routing Problem with Hard Time Windows and Stochastic Travel Time and Service Time

ABSTRACT

This paper introduces a multi-objective vehicle routing problem with hard time windows and stochastic travel and service times. This problem has two practical objectives: minimizing the operational costs, and maximizing the service level. These objectives are usually conflicting. Thus, we follow a multi-objective approach, aiming to compute a set of Pareto-optimal alternatives with different trade-offs for a decision maker to choose from. We propose two algorithms (a Multi-Objective Memetic Algorithm and a Multi-Objective Iterated Local Search) and compare them to an evolutionary multi-objective optimizer from the literature. We also propose a modified statistical method for the service level calculation. Experiments based on an adapted version of the 56 Solomon instances demonstrate the effectiveness of the proposed algorithms.

Keywords: vehicle routing with time windows; stochastic travel times; evolutionary and memetic algorithms; iterated local search; multiobjective optimization.

1. Introduction

The classic vehicle routing problem with time windows (VRPTW) assumes the travel times between customers and the service times at the customers are known and deterministic. However, in real life applications, these two parameters are often stochastic, and models that takes into consideration this uncertainty can provide more accurate answers for the decision maker. Over the years, the customer expectations in the logistics sector have continuously increased and more customer-oriented business models have been demanded, for instance, ensuring a particular level of service at individual customers.

In business terms, the minimization of the operational costs of delivering the goods to the customers is a common objective. Defining service level as the probability of a vehicle visiting a customer before the end of the time window, it is plausible to say that the higher the customer service level, the higher is the customer satisfaction, motivating the inclusion of the service level as one objective function of the model.

Because operational costs and service level have very different dimensions, combining them into one single objective is not straightforward. Additionally, there seem to be a conflict between these objectives, motivating the modeling of the problem as multiobjective (MO-VRPTW). A MO-VRPTW can provide the decision maker with more comprehensive information about the problem, and once they acquire a knowledge from a set of non-dominated solutions, it is easier to define criteria to pick a single solution, by intuition or through the use of multi-criteria decision-making techniques (MCDM), e.g. Shi et al. (2009).

In this context, this paper tackles a problem considering realistic features presented in real applications such as stochastic travel time and service time, a constraint with a lower bound for the customer service level, and the consideration of two objectives (maximizing service level and minimizing operational costs).

The main contributions of this paper are:

1. The introduction of the multiobjective vehicle routing problem with hard time windows and stochastic travel and service times.
2. The design and implementation of two new algorithms (a Memetic Algorithm and Iterated Local Search) discussing specific components such as the design and utilization of the local searches, specific strategies for selection, cross-over, exit criteria and speed-up techniques.
3. An improved version of the method to calculate service levels from Miranda and Conceição (2016) that better reflects the lower bound constraint on the distribution of travel and service times.

The problem approached in this study has hard time windows which means the vehicle has to wait in case it arrives before the start of the time window. There is a single depot, the vehicles have the same capacity and they always visit and deliver the service at the customers. The service time and the travel time are both random variables with a nonnegative normal probability distribution. Common distributions used for the travel time are normal, lognormal and Gamma distributions (Tas et al. 2013). Many studies consider the travel time normally distributed including Kenion & Morton (2003), Jie et al. (2010), Hofleitner et al. (2012), and Chen et al. (2014).

The paper is structured as follows. We give a literature review in Section 2, followed by a mathematical formulation of the problem in Section 3. Next, in Section 4, we present the developed method to calculate the service level of the customers. Section 5 explains the metaheuristics developed to solve the Multiobjective VRP with hard time windows and stochastic travel time and service time. Section 6 has experiments testing the performance of the proposed statistical method and heuristics. Finally, the main findings and conclusions are highlighted in Section 7.

2. Literature Review

Literature reviews of the Stochastic Vehicle Routing Problem (SVRP) have been provided for example by Gendreau et al. (1996, 2014) and Zeimpekis et al. (2007). Berhan et al. (2014) present a comprehensive survey on the SVRP and a classification of the papers. Most SVRP literature focuses on stochastic demand (Laporte et al., 2002 and Dror, 2016), customers' presence (see, e.g. Gendreau et al., 1995) and both demand and presence (Balaprakash, 2015).

In the context of the VRPTW and stochastic travel times and/or service times it is important to separate the formulations without waiting time from the formulations with waiting time for the vehicles. Models without waiting time benefit from the use of convolution properties while summing the random variables. The probability distribution of the sum of two or more independent random variables is the convolution of their individual distributions and many well-known distributions have simple convolutions when there is no waiting time.

For the VRPTW with stochastic travel/service time and no propagation of the waiting time, some relevant studies were conducted by Russell & Urban (2008), Tas et al. (2013), Tas et al. (2014) and Vareias et al. (2017). In the context of VRPTW with stochastic travel/service time and with waiting time (in which the present paper is situated), studies were conducted by Jula et al. (2006), Chang et al. (2009), Li et al. (2010), Miranda (2011), Errico et al. (2013 and 2016), Zhang et al. (2013), Ehmke et al. (2015), Binart et al. (2016), Miranda and Conceição (2016) and Gutierrez et al. (2016). These studies are discussed below in more detail.

Jula et al. (2006) suggest methods of estimating the mean and variance of a vehicle's arrival time at customers using first-order approximation of Taylor series. Chang et al. (2009) studied the relationship between arrival time and time window, and developed an approach to calculate the mean and variance of the arrival time with the assumption of normality. Li et al. (2010) use Monte Carlo simulation to calculate the service level. Errico (2013) and Errico et al. (2016) propose a formulation considering a symmetric triangular distribution with an exact solution. Zhang et al. (2013) adapt the method α-discrete from Miller-Hooks & Mahmassani (1998) to estimate the arrival time distribution at a customer. The method is applied to a normal distribution and log normal distribution for the travel time, and normal distribution for the service time.

Ehmke et al. (2015) propose a method based on the application of extreme value theory that allows the computation of the distribution of the maximum of two normal variables. Even though the arrival time distribution is not normal, they assume so, and evaluate the error of this assumption in an experiment. Binart et al. (2016) solve a variant of the VRP where it's assumed that service as well as travel times are stochastic, both with discrete triangular distributions. Miranda and Conceição (2016) introduce a method to compute the probability of the vehicle arriving before the end of a time window for the case where service time and travel time follow a Gaussian distribution. A single objective VRPTW is solved through Iterated Local Search. Finally, Gutierrez et al. (2016) develop a memetic algorithm to solve a single objective version of the VRPTW with stochastic travel and service times. The method used to compute the service level comes from Ehmke et al. (2015) and the only modification is that both travel time and service time are stochastic.

Still regarding the variant with propagation of waiting times, papers mentioned previously use triangular distributions such as Binart et al. (2016), Errico (2013) and Errico et al. (2016), which is a poor representation for real applications. Li et al. (2010) use Monte Carlo simulation (too expensive to be used in an NP-hard problem). Jula et al. (2006) use first-order approximation of Taylor series which deliver poor results. Ehmke et al. (2015) use extreme value theory but also assuming the arrival time distribution is normal. Zhang et al. (2013) does not assume the arrival time is normal even with normal travel time and service time, and its method is used as a benchmark for the statistical method of our paper.

All papers mentioned previously consider the service level as a constraint in the model and they use operational cost as single objective. Several papers propose additional objectives for the VRPTW, such as: the number of used vehicles, total traveled distance, traveling time of the longest route (makespan), total waiting time, and total delay time (Castro et al., 2011). To the best of our knowledge, no publication so far used service level.

Our paper differs from and extends the existing work in the following ways: It is the first attempt to approach a bi-objective variant of the VRPTW with stochastic travel times and service times, where the service level is one of the objectives. It proposes a new approach to embed a local search into a multi-objective problem. In addition to that, our statistical method to calculate the service level (already validated in Miranda and Conceição, 2016) has been changed to consider a more realistic representation of real problems (discussed in Section 4).

3. Background

3.1 – Multi-objective Optimization (MOO)

MOO is the process of simultaneously optimizing two or more conflicting objectives. In mathematical terms, a multiobjective optimization problem (MOP) cab be written, without loss of generality, as $min\, f(x) = (f_1(x), f_2(x), \dots, f_p(x))$ subject to $x \in X \subseteq \Re^n$ where X is a constraint set in the multi-dimensional space of the problem specified by $X = \left\{x \in \Re^n \colon g_i^{(x)} \leq 0, i = 1, \dots, m;\ h_j^{(x)} = 0, j = 1, \dots, l\right\}$. Given two feasible solutions x and y, we say that x dominates y, if $\forall i \colon f_i(x) \leq f_i(y)$ and $\exists j \colon f_j(x) < f_j(y)$. Moreover, x is said to be Pareto optimal if and only if it is not dominated by any other feasible solution. The aim is to find the set of Pareto optimal solutions usually called Pareto set. This set maps to a number of non-dominated points in the objective space, the so-called Pareto Front.

Pareto ranking is often used to rank solutions: All non-dominated solutions are assigned rank 1, and inductively, all solutions non-dominated once ranks 1…i have been removed are assigned rank $i\ + 1$.

3.2 – Problem statement

Let $G = (V_0, A)$ be a complete digraph, where $V_0 = \{0, \dots, n\}$ is a set of vertices and $A = \{(i,j) \colon i, j\ \in\ V_0, i \neq j\}$ is a set of arcs. Vertex 0 represents the depot where m_0 vehicles with capacity Q are available. The set of customers is $V = V_0 \backslash \{0\} =\ \{1, \dots, n\}$. Each customer $i \in V$ has a non-negative demand q_i, service time ST_i, and a time window $[e_i, l_i]$, where e_i is the start of the time window (earliest time) and l_i is the end of the time window (latest time). If the vehicle arrives at customer i before e_i, it is necessary to wait until e_i. A travel time TT_{ij} is assigned to each arc $(i,j) \in A$. Both TT_{ij} and ST_i are random variables with known and independent probability density. $SL_i = P(AT_i \leq l_i)$ is the service level at customer i. The vector $SL = (SL_1, \dots, SL_n)$ summarizes the service level of each customer. Other delimitations are: $Q \geq q_i, i \in V$ (i.e., each vehicle has enough capacity to serve at least one customer) and $m_0 * Q \geq \sum_{i=1}^{n} q_i$ (i.e. the fleet is big enough to serve all the customers). There is no time window for the depot, i.e. $[e_0, l_0] = [0, \infty]$. Further notation includes:

f	Fixed cost for one vehicle
m	Number of vehicles in a feasible solution, $m \leq m_0$
c	Fixed cost for each unit of the travel time TT_{ij}

K	Set of required vehicles in a feasible solution $K = \{1, \dots, m\}$.
x_{ijk}	Boolean variable with value 1 if vehicle k serves arc (i, j)
AT_i	Arrival time at customer i
SS_i	Service start time at customer i
α_i	Required service level by customer i where $\alpha_i \in [0,1]$

The decision variables of the problem are x_{ijk} and m. The model for the problem can be described as follows:

$$Min\, f * m + \sum_{(i,j)\in A} \sum_{k\in K} E(TT_{ij}) * c * x_{ijk} \tag{1}$$

$$Min\, -E[SL] \tag{2}$$

Subject to:

$$\sum_{j\in V_0} \sum_{k\in K} x_{ijk} = 1, \forall i \in V \tag{3}$$

$$\sum_{j\in V} x_{0jk} = 1, \forall k \in K \tag{4}$$

$$\sum_{i\in V} x_{i0k} = 1, \forall k \in K \tag{5}$$

$$\sum_{i\in V_0} x_{ijk} - \sum_{i\in V_0} x_{jik} = 0,\ \forall j \in V, k \in K \tag{6}$$

$$\sum_{i\in V} q_i \sum_{j\in V_0} x_{ijk} \le Q, \forall k \in K \tag{7}$$

$$P(AT_i \le l_i) \ge \alpha_i,\ \forall i \in V \tag{8}$$

$$e_i \le SS_i, \forall i \in V \tag{9}$$

The two objectives are given by Equations (1) and (2). Equation (1) is formed by two parts: the vehicle fixed cost (number of vehicles m multiplied by the fixed cost f) and the variable cost (the sum of the mean of each activated arc multiplied by the cost c per unit of travel time). Therefore, both terms are expressed by a financial unit. The second objective is given by Equation (2) and maximizes the service level of the solution, defined here as the mean of the service levels of all customers.

Equation (3) ensures each customer is visited only once. Equations (4) and (5) ensure each vehicle starts the route in the depot and also returns to it. Equation (6) ensures that each vehicle departs from a customer location after it visits the customer. Equation (7) is the capacity constraint. Equation (8) is the service level required by each customer. It reflects that the probability of a vehicle arriving at the customer before the end of the time window should be greater than a given threshold, for any customer. Equation (9) ensures the hard time windows, where the service will only start after the start of the time window, therefore, if the vehicle arrives before e_i, it must wait until e_i. The service start time is given by Equation 10:

$$SS_i = max\{AT_i, e_i\}, \tag{10}$$

where $AT_i = SS_{i-1} + ST_{i-1} + TT_{i-1,i}$.

Note that the departure time from the depot is defined as zero and a departure time greater than zero does not improve any of the objectives.

Regarding the service level objective, we use the mean service level rather than median or percentage of customers within service level because we assume there is also a value in over-achieving the desired service level α_i.

4 – Computation of the service level

This section presents the statistical method to calculate the probability of a vehicle arriving at a customer before the end of its time window (service level). It is based on the method introduced by Miranda and Conceição (2016) but uses a lower bound for the service and travel time that is more representative of real applications.

4.1 – Statistical method to compute the service level

For the sake of simplicity, we adopt a specific notation for the statistical problem where a random variable X_i is the arrival time at the i-th customer, and after its truncation at the start of the time window, it becomes the truncated variable $X_i^t$with function f_x left-truncated at point t_i, where $X_i^t = \max(x_i, t_i)$, not removing the early arrival. The variable Y_i is the sum of the service time (*ST*) and travel time (*TT*), assuming they are both normally distributed: $ST_i + TT_{i,i+1} = N(\mu_{STi}, \sigma_{STi}{}^2) + N(\mu_{TTi,i+1}, \sigma_{TTi,i+1}{}^2) = N(\mu_{STi} + \mu_{TTi,i+1}, \sigma_{STi}{}^2 + \sigma_{TTi,i+1}{}^2) = Y_i$. By doing this, the problem becomes solving $X_{i+1} = X_i^t + Y_i$ recursively from $i = 1$ to $i = n$, where n is the number of customers of a given route. The service level for customer $i + 1$ is $P\{X_{i+1} \leq c\}$ where c is equivalent to the end of the time window l_{i+1}. Figure 1 illustrates the case and also highlights the fact that $X_i^t = t \; if \; x \leq t$ with a peak at the beginning of the service window.

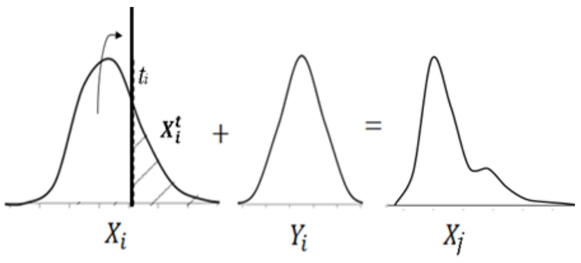

Figure. 1: Sum of random variables

The cumulative distribution function of $X_{i+1} = X_i^t + Y_i$ is obtained as follows:

$$F_{X_i^t+Y_i}(c) = \int_{-\infty}^{\infty} F_{X_i^t}(c-y)\ f_{Y_i}(y)\,dy \tag{11}$$

Equation (11) is known as the convolution of the marginal distributions. This equation is approximated by Equation (12), a discrete function with I intervals, where y_f and y_0 are the upper and lower bounds of the integration, respectively:

$$\int_{-\infty}^{\infty} F_{X_i^t}(c-y)\ f_{Y_i}(y)\,dy \cong \frac{y_f - y_0}{I} * \sum_{k=1}^{I}\left(\frac{g(y_{k-1}) + g(y_k)}{2}\right) \tag{12}$$

where $g(y_k) = F_{X_i^t}(c-y_k)\ f_{Y_i}(y_k), \forall k \in \{1, \dots I\}$.

The lower bound is computed as: y_0=max$(\mu_{Y_i} - \mathrm{S} * \sigma_{Y_i}, 0)$ where S is a parameter specifying the number of standard deviations used. Note that negative values are not allowed because both service time and travel time cannot have negative values. The upper bound is y_f=min$(\mu_{Y_i} + \mathrm{S} * \sigma_{Y_i},\ c - t_i)$. It is desired $P(Y_i \leq \mu_{Y_i} - \mathrm{S} * \sigma_{Y_i}) \cong 0$ and $P(Y_i \leq \mu_{Y_i} + \mathrm{S} * \sigma_{Y_i}) \cong 1$. Due to the truncation point t_i, we have $c - y_k \geq t_i$, therefore $y \leq c - t_i$.

To compute $g(y_k)$ we need to calculate $F_{X_i^t}(c-y_k)$ and $f_{Y_i}(y_k)$. Because Y_i is assumed to be a Gaussian distribution, we have $f_{Y_i}(y) = \frac{1}{\sigma_{Y_i}\sqrt{2\pi}}\, exp\left(-0.5\frac{(y-\mu_{Y_i})^2}{\sigma_{Y_i}{}^2}\right)$. The cumulative function $F_{X_i^t}(c - y_k)$ is assumed to have been calculated in the previous iteration of the recursion and its discrete function is stored in a matrix $mprob_i$ comprised of $nint$ points, as follows:

$$mprob_i = \begin{bmatrix} c_1^i & \dots & c_q^i & \dots & c_{nint}^i \\ F_{X_i^t}(c_1^i) & \dots & F_{X_i^t}(c_q^i) & \dots & F_{X_i^t}(c_{nint}^i) \end{bmatrix}$$

where $q \in \{1, .., nint\}, F_{X_i^t}(c_1^i) \cong 0$ and $F_{X_i^t}(c_{nint}^i) \cong 1$.

Because $mprob_i$ stores discretized values of the cumulative function, any value of the continuous function $F_{X_i^t}$ is calculated by a linear interpolation of the adjacent points. Therefore, $F_{X_i^t}(c-y_k)$ is computed through a linear interpolation of $F_{X_i^t}(c_q^i)$ and $F_{X_i^t}(c_{q+1}^i)$ where $c_q^i \leq c - y_k \leq c_{q+1}^i$. As mentioned before, if c is equivalent to the end of the time window l_i, then calculating $P\{X_{i+1} \leq c\}$ is the same of calculating the service level at customer $i+1$.

In order to allow the computation of the next iteration, it is necessary to generate the matrix $mprob_{i+1}$ by calculating the discrete cumulative function $F_{X_i^t}(c_q^{i+1})$ for each $q = \{1, \dots nint\}$. The bounds are computed as: $c_1^{i+1} = \max\left(F_{X_i^t}^{-1}(\cong 0), t_i\right) + \max(\mu_{Y_i} - \mathrm{S} * \sigma_{Y_i}, 0)$ and $c_{nint}^{i+1} = \max\left(F_{X_i^t}^{-1}(\cong 1), t_i\right) + (\mu_{Y_i} + \mathrm{S} * \sigma_{Y_i})$. Note that $F_{X_i^t}^{-1}(\cong 0)$ and $F_{X_i^t}^{-1}(\cong 1)$ were computed in the previous iteration.

So far, this allows to calculate $P\{X^t + Y \leq c\}$ where Y is assumed to be a Gaussian distribution. Next, we describe some considerations for the service time and travel time that differentiates this approach from Miranda and Conceição (2016).

Let X be a normal random variable that can represent ST or TT with probability density function $f(x)$. Because the travel time and service time are normal distributions where negative values cannot be censored, the resulting distributions are left-truncated normal distributions (random variables represented by ST^t and TT^t) with probability density function given by Equation 13:

$$f^t(x) = \begin{cases} 0, & x < 0 \\ \dfrac{f_x(x)}{\int_0^\infty f_x(x)dx}, & x \geq 0 \end{cases} \tag{13}$$

Function $g(y_k)$ has $f_{Y_i}(y) = \frac{1}{\sigma_{Y_i}\sqrt{2\pi}}\ exp\left(-0.5\frac{(y-\mu_{Y_i})^2}{\sigma_{Y_i}{}^2}\right)$ where $\mu_{Y_i} = E[ST^t] + E[TT^t]$ and $\sigma_{Y_i}{}^2 = Var[ST^t] + Var[TT^t]$. The components $E[ST^t], E[TT^t], Var[ST^t]$ and $Var[ST^t]$ are calculated by Equations 14 and 15, where X represents a truncated normal variable with mean μ, standard deviation σ, left truncation point t, $\alpha = (t - \mu)/\sigma$, probability density function $\phi(*)$ and cumulative distribution function. $\Phi(*)$.

$$E[X|X > t] = \mu + \sigma\ \frac{\phi(\alpha)}{1 - \Phi(\alpha)} \tag{14}$$

$$Var[X|X > t] = \sigma^2 \left[\left(1 + \frac{\alpha\phi(\alpha)}{1 - \Phi(\alpha)}\right) - \left(\frac{\phi(\alpha)}{1 - \Phi(\alpha)}\right)^2\right] \tag{15}$$

To calculate $P\{X_i^t + Y_i \leq c\}$, we have $Y_i = ST_i^t + TT_i^t$. The utilization of Equations 14 and 15gives the correct values for the mean and variance of Y_i but using a Gaussian equation for $f_{Y_i}(y)$ is an approximation. Because probabilities $P(ST_i^t < 0)$ and $P(TT_i^t < 0)$ are very small, the impact of truncation in the distribution of Y_i is small. The accuracy of the method is discussed in Section 6.2.

4.2 – Handling non-negative values of the service time

This paper handles the non-negative values of service time and travel time differently from Miranda and Conceição (2016), where the distribution function for travel time and service time with non-negative values are given by Equation 16 and not 13. This change is made because we believe Equation 13 is more representative of real world scenarios than Equation 16, as illustrated by Figures 2 and 3. Observe these two representations solve different problems therefore a direct comparison is not sensible.

$$f^t(x) = \begin{cases} 0, & x < 0 \\ f_x(x), & x \geq 0 \end{cases} \tag{16}$$

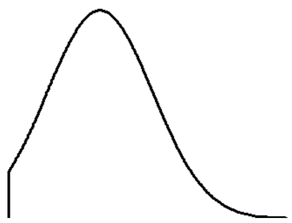
Figure 2: function for equation 13

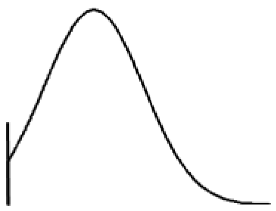
Figure 3: function for equation 16

Figures 2 and 3 show exemplary probability distributions based on Equations 13 and 16, respectively. As Figure 3 shows, because the negative values are simply accumulated at zero, there is a concentration of probability at zero which does not seem to be very realistic. This is different from Figure 2 where the subject is the truncation at the start of the time window.

This change in the formulation of the problem induced a change while computing $f_{Y_i}(y)$. For Miranda and Conceição (2016), $\mu_y = E[ST] + E[TT]$ and $\sigma_y{}^2 = Var[ST] + Var[TT]$, not using Equations 14 and 15, while for the present paper, we have $\mu_y = E[ST^t] + E[TT^t]$ and $\sigma_y{}^2 = Var[ST^t] + Var[TT^t]$.

5 –Algorithms to solve the Multiobjective Stochastic VRPTW

In this paper, we devise two metaheuristics successfully applied in variants of VRP in the literature: a Multiobjective Memetic Algorithm (MMA) and a Multiobjective Iterated Local Search (MILS). In order to better evaluate the results, we also implement two versions of the Multiobjective Evolutionary Algorithm (MOEA) proposed by Garcia-Najera and Bullinaria (2011), totaling four algorithms. This section presents the algorithms in which the statistical method of the previous section is embedded. Any heuristic can be adapted to use the proposed method for service level calculation.

5.1 – Multiobjective Memetic Algorithm (MMA)

A Memetic Algorithm (see concepts in Moscato, 1999) is an approach combining an evolutionary algorithm with a local improvement procedure. The pseudo-code of the main loop can be found in Algorithm 1, where HV stands for hypervolume, classic metric for multi-objective problems detailed in Emmerich et al. (2005).

Algorithm 1 : Main loop of the Multiobjective Memetic Algorithm

```
1   Initialize parameters for the VRPTW instance
2   **While** HV improvement in the last 3 generations > *minHV*
3       **if** $(gen = 1)$ then
4           Generate *nIniSol* initial solutions and return *Offspring*
5       **else**
6           Apply selection by SUS in *Pop*
7           **for** $i = 1: nChi$
8               Apply crossover and add child to *Offspring*
9           **end**
10      **end**
11      *Offspring* ←Pre_Intensification_VND(*Offspring*)
12      Rank *Offspring* and return *Offspring Rank1*
13      *Offspring'* ← Intensification_VND(*Offspring Rank1*)
14      Compute rank and HV contribution in *Pop*
15      Cut-off of the population *Pop*
16      Add *Offspring'* to *Pop*
17      Compute HV contribution in *Pop*, update *BestPop* and HV improvement
18  **end**
19  *BestPop* ← Post_Intensification_LocalSearch(*BestPop*)
```

Line 1 initializes the parameters of the VRPTW instance to be solved. Loop 2-18 are executed while there is a significant improvement (> *minHV*) of the hypervolume of the Pareto front *BestPop*. Line 4 is executed only in the first iteration, and creates $nIniSol$ initial solutions to form a population called *Offspring* that is used to initialize the main population *Pop*. Otherwise there is already a population *Pop* and then $nChi$ solutions are selected in line 6 using Stochastic Universal Sampling (SUS) and $nChi$ new solutions are generated by the crossover operator in line 8. These new solutions form the population *Offspring* . Line 11 improves the *Offspring* through the application of a specific set of local searches identified as "Pre-Intensification". Line 12 computes the rank of the solutions in *Offspring* and only those solutions with rank 1 (non-dominated solutions) form the population *Offspring Rank 1* that is processed by another set of local searches called "Intensification" and returns *Offspring'*. In line 14, the rank is updated and the individual contribution of each solution in *Pop* is calculated. Line 15 checks whether the current population *Pop* exceeds a specified maximum number of solutions. If yes, the solutions with smaller rank and hypervolume contribution are eliminated. In line 16, all solutions of the population *Offspring'* are added to the main population *Pop*. Finally, line 17 updates the rank and hypervolume contribution for each solution from *Pop* (to be further used in line 6 during selection), updates *BestPop* (Pareto front with the best set of non-dominated solutions obtained by the algorithm up until the current iteration) and calculates the hypervolume of *BestPop* to update the accumulated hypervolume improvement in the last 3 generations (exit criterion in line 2). After the main loop, in line 19, the current *BestPop* is processed by a set of local searches called "Post-Intensification" aiming at an additional increase of the hypervolume.

The individual hypervolume contribution used in line 14 and the hypervolume calculation used in line 17 were implemented according to Fleischer (2003) and Emmerich et al. (2005), respectively. The initial solution (line 4) is based on the well-known $I1$ heuristic from Solomon (1987), a sequential insertion heuristic that considers the insertion cost of an unrouted customer u between two adjacent customers i_{p-1} and i_p in a partially finished route $(i_0, i_1, \dots, i_0)$ in which i_0 represents the depot. The insertion cost is a weighting of the travel time to the adjacent customers and the new service start time for i_p. When it is not possible to insert a new customer in the route, a new route is started with a customer that comes from a list of unrouted customers (seed list) sorted by a score given by a weighting between the distance to the depot and the end of the time window. Different weighting parameters for the insertion cost and seed list are used to generate a number of solutions. In order to avoid additional computation of the service level in this initial solution phase, the time windows feasibility is tested firstly according to the classic deterministic approach, assuming mean values all travel and service times. If feasible by this test, the stochastic test is performed, computing the service levels of all customers of the route and checking if they are higher than a specified threshold α. It is possible to generate initial solutions that violate the required service level, and in this case, a penalty proportional to the violation is added to both objectives.

5.2– Multiobjective Iterated Local Search (MILS)

The metaheuristic "Iterated Local Search" (ILS) has been used successfully for a variety of single-objective combinatorial problems and it has been also applied for multiobjective routing problems such as Assis et al. (2012), Aquino et al., (2014), and de Souza Lima (2017). The main loop of the MILS proposed here is described in Algorithm 2. We have a working set of solutions (S^W), a local set of non-dominated solutions (S^L) and the non-dominated set of solutions (S^*) that is the best Pareto-front obtained by the algorithm.

In the first iteration (it), a set of initial solutions are created in line 4 initializing S^W. In line 5, S^* is initialized with S^W. For other iterations, 2 solutions (s_1 and s_2) are selected from S^W using SUS (line 7). A new solution s is generated via crossover (line 8). Note that from the context of Iterated Local Search, we are using the crossover operator as a perturbation operator. After that, in line 10, a set of local searches based on Variable Neighborhood Descent (VND) is applied to s returning the local set of non-dominated solutions S^L. The VND works as described in Algorithm 4. Lines 11 and 12 update the hyper volume contribution for each solution in S^W and perform the cut-off to adjust the size of the working population S^W. This is done before adding S^L to S^W (line 13) so the solutions from S^L still have some chance to be selected by SUS at the beginning of the next iteration. Note that even if a solution from S^L is dominated, because they come from a local set of non-dominated solutions, they still have a chance to contribute. In line 11, S^W and S^* are updated. If there is improvement in the S^* hypervolume then the counter for the number of perturbations ($nPert$) resets to zero, otherwise it is incremented. The process continues while the maximum number of perturbations ($maxPert$) or iterations ($maxIt$) is not exceeded.

Algorithm 2: MILS

```
1  Initialize parameters for the VRPTW instance
2  While ($\mathrm{nPert} \leq \mathrm{maxPert}$) And ($\mathrm{it} < \mathrm{maxIt}$)
3      if ($\mathrm{it} = 1$) then
4          Generate nIniSol initial solutions and return $S^W$
5          Add $S^W$ to $S^*$, update HV.
6      else
7          Select 2 solutions ($s_1$ and $s_2$) from $S^W$ using SUS
8          s ← Crossover ($s_1$,$s_2$).
9      end
10     $S^L \leftarrow$ VND(s).
11     Compute rank and HV contribution in $S^W$
12     Cut-off $S^W$
13     Add $S^L$ to $S^W$
14     Compute HV contribution in $S^W$, update $S^*$ and HV improvement
15     if ($\mathrm{HV\ improvement} = \mathrm{true}$) then
16         $\mathrm{nPert} = 0$;
17     else
18         $\mathrm{nPert} = \mathrm{nPert} + 1$;
19     end
20     $\mathrm{it} = \mathrm{it} + 1$;
21 end
```

5.3– Operators

In this section, we discuss operators such a local searches and crossover used in the metaheuristics. Usually, for single objective and even multi-objective problems, local searches receive one solution as input and return one solution as output. Differently, all the local searches implemented in this work receive one solution and return a set of non-dominated solutions.

Algorithm 3 shows the general structure of the local searches. A local population $popLocal$ is initialized with the solution s. The loop 3-17 keeps exploring new moves while a new non-dominated solution is found. Loop 5-16 performs the moves. In line 5, because the service level computation impacts the running time, not all moves have the objective function evaluated, avoiding service level calculations. The considered moves are said “eligible” and explained in more detail in Algorithm 5 (lines 6, 8 and 9). Line 9 replaces s by s' if function $Better$ returns true, i.e., s' dominates s or if the percentage improvement in one objective is higher than the percentage deterioration in the other objective compared with the objective evaluations of s. Lines 11-15 add the new solution s’ to the local population in case it is not dominated by any solution, delete the dominated solutions, and update the boolean variable. In line 18, the local population *popLocal* and the index of the current solution s in the local population are returned by the function. Note that line 9 is only a strategy to guide the local search by exploring a good solution. Even if the new solution s' does not replace s, if s' is not dominated it will still be added to the population.

Algorithm 3: Basic structure of the local search

```
1  $popLocal(1) \leftarrow s$
2  improve ← true
3  **while** improve = true do
4      improve ←false
5      **for** each eligible and feasible move of the solution s
6          Evaluate objectives of the new solution $s'$
7          Check if s' is better than $s$
8          **if** $Better(s', s)$
9              $s \leftarrow s'$
10         **end**
11         **if** $s'$ is not dominated by any solution of $popLocal$
12             Add $s'$ to $popLocal$
13             Delete any solution in *popLocal* dominated by $s'$
14             improve ← true
15         **end**
16     **end**
17 **end**
18 Return $popLocal$ and index of $s$
```

For the MMA algorithm, there are three different sets of local searches: “Pre-Intensification”, “Intensification” and “Post-Intensification”, and for the MILS algorithm there is only one named “VND-MILS”. These searches are based on the Variable Neighborhood Descent (VND), see Hansen & Mladenovic (2003). Many frequently applied neighborhood operators in VRP were implemented and those successfully tested were kept in the algorithms. Six local searches are used by these three VND functions: 2-opt (Lin, 1965), Reallocation (Osman, 1993), Interchange (Osman, 1993), 2opt* (Potvin & Rousseau, 1995) , “Intra Exchange” and “Reallocation Chain”. The local search “Intra Exchange” is analog to “Reallocation” but all movements are performed in the same route. The operator Cross-exchange (Taillard et al. 1997) was also tested but identified as not helpful and it is not included.

The local search “Reallocation Chain” was specifically designed to focus on route elimination. Given a solution, starting with the route $r1$ with the least customers, the local search “Reallocation” is applied to move the customers from route $r1$ to the other routes. When this is not possible anymore and there are still remaining customers in $r1$, the local search “Interchange” replaces one customer from $r1$ by another from the other routes and after that, the “Reallocation” is applied again.

The ‘Pre-Intensification VND’ used in line 11 of Algorithm 1 is described in Algorithm 4, and for the sake of brevity, also used to explain the other types of VND. The function receives a population “pop” that in the case of the ‘Pre-Intensification VND’ is the *Offspring* (from lines 4 and 8, Algorthm 1). Lines 1 and 2 initialize an auxiliary population pop' and a local population $localPop$. Loop 3-17 is applied to each solution s in pop in which the local population is initialized with solution s. Loop 7-14 applies K local searches while a non-dominated solution is found. In line 8, the first local search ($k = 1$) initially performs the moves in the original solution $s = pop(\text{i})$. The local search accepts the move according to the function $Better$ (line 8, Algorithm 3), where s receives the new solution s'. The local population $localPop$ is updated according to the loop lines 11-15, Algorithm 3. It means that the local search k updates s and $localPop$ passed as input to the next local search $k + 1$. In line 15, the solutions from $localPop$ are included in pop' (only non-repeated solutions), and in the next line the local population is reset. By doing so, each solution $pop(\mathrm{i})$ generates its own set of non-dominated solutions that are all included in pop'. This is done to have more diversity on the search, still working with high quality solutions (a non-dominated set of solutions).

Algorithm 4: Pre-IntensificationVND(pop)

```
 1 $pop' \leftarrow \emptyset$;
 2 $localPop \leftarrow \emptyset$;
 3 for i = 1 to $npop$
 4     s ← $pop$(i);
 5     $localPop(1) \leftarrow s$;
 6     k ← 1;
 7     while k ≤ K do
 8         Local_Search (s, $localPop$, k);
 9         if non-dominated solution found then
10             k ← 1;
11         else
12             k ← k + 1;
13         end
14     end
15     Add $localPop$ to $pop'$
16     $localPop \leftarrow \emptyset$;
17 end
18 Return $pop'$;
```

The ‘Intensification VND’ used in line 13 of Algorithm 1 works similarly to Algorithm 4, with the following differences: pop passed as input is *Offspring Rank1* (from line 12, Algorthm 1); there are no lines 1 and 5; in line 2 we have $localPop \leftarrow pop$; we do not have lines 15 and 16, meaning there is no reset of the local population, aiming to increase the intensification; and the function returns $localPop$. The ‘Post-Intensification VND’ used in line 19 of Algorithm 1 is the same of ‘Intensification VND’ with differences described in Table 1. The VND-MILS used in line 10, Algorithm 2, works according to Algorithm 4 but with $npop = 1$, because it receives a single solution and still returns a local set of non-dominated solutions. Other differences among the four VND strategies are summarized in Table 1, showing the input of the VND and its sequence of local searches.

Table 1: VND functions

VND Type	**Input**	**Local Searches**
Pre-Intensification	Offspring population (*Offspring*)	2-opt, Reallocation Chain, Interchange, Reallocation
Intensification	Non-dominated solutions of the Offspring (*Offspring Rank 1*)	2-opt, Intra Exchange, Interchange, Reallocation, 2-opt*
Post-Intensification	Pareto-front (*Best Pop*)	2-opt, Reallocation, 2-opt*
VND-MILS	Single solution	2-opt, Reallocation Chain, Intra Exchange, Interchange, Reallocation, 2-opt*

Algorithm 5 presents the pseudo-code for the main structure of the local search “Interchange” in the context of the multiobjective and stochastic VRPTW. The general idea is to interchange customers from two different routes in order to find new non-dominated solutions. In this algorithm, “SL” stands for the service level of the customers.

Algorithm 5: Main structure of the local search "Interchange"

```
 1  for each route r1
 2      for each route r2 (r2 different from r1)
 3          if r1 and r2 are not tabu
 4              for each customer in r1
 5                  for each customer in r2
 6                      if within circular zone
 7                          if capacity constraint holds
 8                              if move is not tabu
 9                                  if deterministic time windows holds
10                                      Compute SL for the customers in the modified r1
11                                      if penalty for the SL has not increased
12                                          Compute SL for the customers in the modified r2
13                                          if penalty for the SL has not increased
14                                              Evaluate Objectives
15                                              if Better = true
16                                                  bestSol←newSol
17                                              end
18                                              Update local population
19                                          end
20                                  … end(s)
21  end
```

The loops starting in lines 1- 2 and 4 - 5 explore different customers and different routes to perform the moves. In line 3, only pairs of routes classified as “not tabu” are explored. In this context, the word “tabu” means “forbidden”, and that pair of routes is not considered eligible. The first time in the whole algorithm a pair of routes is explored by the local search, if it does not lead to non-dominated solutions, the pair of routes is classified as “tabu”. In line 6, only moves involving customers within a given circular zone are eligible. Line 8, the idea of “tabu moves” is analog to the idea of “tabu routes”, in which if an interchange move is tested and it does not lead to a non-dominated solution, this move is declared “tabu”. Line 9 performs the time windows feasibility test using a deterministic approach. Lines 10 and 12 compute the service level of the customers in the new routes. Lines 14-18 work according to Algorithm 3. The conditional clauses in lines 3, 6,7,8,9 and 11 aim to prevent additional computation of the service level of the customers for the new routes. Note that while in principle it may be possible that the deterministic time windows feasibility test performed in line 9 could reject a feasible solution, preliminary experiments showed this happens rarely and only for small values of the required service level ($\leq 50\%$) which is not the case for the experiments described in the next sections. This is supported by the fact that the mean of the service start time is greater than the deterministic start time because in the stochastic formulation, the truncation point at the start of the time windows shifts the mean to right, which doesn't happen in the deterministic formulation

In order to quickly check whether a route is tabu or not, we created an indexation mechanism to give a unique number to pairs of routes (in the case of local searches in which the moves involves two routes) according to equation (17). Let b_i be an array with the deterministic service start time of the customer i for $i = 1, \dots, N$ customers. Let $r1$ be an array with a sequence of $nc1$ customers visited by the vehicle where $k = 1, \dots, nc1$; and another array for the route $r2$ with $nc2$ customers with $k = 1, \dots, nc2$. SL is an array of length N with the service level of the customers. The indexation is used in a boolean array $TR_{\,idxR}$. If $TR_{\,idxR} = true$, the pair of routes is considered tabu.

$$idxR = \left\| \left(\sum_{k=1}^{nc1} b_{r1_k} + \sum_{k=1}^{nc2} b_{r2_k} \right) * 1000 \right\| + \left\| \sum_{k=1}^{nc2} SL_{r2_k} \right\| + r1_1 \qquad (17)$$

In a similar way, in order to index also the moves, we used Equation (18) where Δ_1 is the variation of the total travel time for the route $r1$ when the customer $u2$ from route $r1$ is inserted in $r2$ replacing customer $u1$. Δ_2 is analog for route $r2$.

$$idxMov = idxR + \|(\Delta_1 + \Delta_2) * 100\| + \|(\mathrm{b}_{u1} + \mathrm{b}_{u2}) * 100\| \qquad (18)$$

This index is used in a boolean array $TM_{\,idxMov}$. If $TM_{\,idxMov} = true$, the move is tabu. This indexation equations were created through preliminary experiments in order to minimize the probability of declaring tabu a pair of routes/moves that was never explored previously. The effectiveness of such strategy is shown in the experiments of the next section.

The circular zone used in line 6 of Algorithm 5 defines a maximum mean travel time (TT_{max}) to evaluate moves. The insertion of customer i between customers $i - 1$ and $i + 1$ is only considered eligible if $TT_{i-1,i} \leq TT_{max}$ and $TT_{i,i+1} \leq TT_{max}$. The utilization of the circular zone, tabu route/moves and the deterministic time windows test aims to reduce computational time and the effectiveness of these strategies is discussed in the next subsection.

The selection strategy used in the algorithms 1 and 2 is the Stochastic Universal Sampling (SUS), introduced by Baker (1987). It exhibits no bias and minimal spread, and it as an alternative for the well-known roulette wheel selection. The crossover operator developed in this work (used in both MMA and MILS) receives an array with the index of the solutions in the population selected by SUS. The function picks two solutions (s_1 and s_2) and combines them to generate a new solution (s'). It is common in the literature having crossover operators that copy routes from both parents, but our operator has a step to reduce the number of conflicts (common customers) while selecting the routes from both parents.

	Parent 1 (s_1)				Child (s')				Parent 2 (s_2)			
Route 1	1	14	9	4	10	11	12	13	10	11	12	13
Route 2	5	6	7	8	5	6	7	8	14	15	7	8
Route 3	10	11	12	13	1	2	9		3	4	5	6
Route 4	3	2	15		3				1	2	9	

Figure 4: Crossover Operator

The following steps describe the crossover and are applied to the example in Figure 4.

1) For each route in s_1, calculate the number of routes in s_2 with any customer in common and generate a matrix ($L1$) sorted in ascending order, storing in one row the number of common customers and in another row the index of the route. Repeat the same for s_2 in comparison with s_1 generating $L2$. In the example of Figure 4:

L1

#	1	2	3	3
route	$r3$	$r2$	$r1$	$r4$

L2

#	1	2	3	3
route	$r1$	$r4$	$r2$	$r3$

2) Initialize s' with the first route in $L2$, In the example, it is $r1$.
3) In $L1$, select the first route from s_1 in which its first customer is not in s'. Copy the route to s', from index 1 to index p, where $p + 1$ is the index with the first customer already routed in s'. In the example, the preference is for $r3$ but its first customer is already in s', then it is $r2$, the whole route in this case.
4) In $L2$, select the first route from s_2 in which its first customer is not in s'. Copy the route to s', from index 1 to index p, where $p + 1$ is the index with the first customer already routed in s'. In the example, it is $r4$, the whole route.
5) Repeat steps 3 to 4 until one of them is depleted. In the example, repeating step 3 returns $r4$ until index $p = 1$, because $p + 1$ is a customer already inserted in s'. Then, $L1$ is depleted and step 5 is finished.
6) If there are customers still not inserted in s', apply a constructive heuristic to complete s'. We choose a modified version of the algorithm used to generate the initial solutions. In the example, the new solution s' inherited 12 out of 15 customers, with information of sequence and direction.

5.4– Notes on the features of the local search schemes

In this section, we emphasize the main features that make the proposed algorithms unique when compared with others from the literature. One point is related to the way the local searches are guided. Evolutionary algorithms such as Castro et al. (2011) and Chiang and Hsu (2014) have neighborhood structures that are used as mutation operators not considering the objective function, usually moving to the first feasible solution found, receiving one solution and returning one modified solution. Assis et al. (2013) suggest the adoption of many distinct specialized neighborhoods for each addressed single objective, and the local searches accept the first solution that improves the single objective of the respective local search. In Baños et al. (2013), if the new solution is not dominated by the parents, it is accepted, otherwise the solution is accepted or not according to the criterion of Metropolis in a simulated annealing context.

Qi et al. (2015) force single-objective sub-problems, so the local search is guided as in any typical single-objective approach. Sivaramkumar et al. (2015) normalize the objectives and work with the averages to have single-objective problems, therefore there are no additional challenges while guiding the local search. In Lima et al. (2017), the incumbent frontier set is passed as an input to the local search phase so that the dominance checking can be done along the search inside the neighborhood structure and the local search is performed separately for each objective one at a time.

Differently from those studies, in the current paper, the developed local searches receive one solution and return a local set of non-dominated solutions. In addition to that, the local search accepts the move if the new solution dominates the original solution or if the percentage improvement in one objective is higher than the percentage deterioration in the other objective. Note that this criterion works with the objectives simultaneously, prevents an undesirable zigzag phenomenon and ensures a certain level of quality for the current solution guiding the local search.

Another point is related to the way the local searches are embedded in the framework of the meta-heuristic. Assis et al. (2013) and Lima et al. (2017) count on one VND in their algorithms, but differently, in our proposed MMA we have two sets, working in two phases: pre-intensification and intensification (Algorithm 4). In the first, each solution in 'Offspring' is explored by local searches in a way that each solution generates its own local set of non-dominated solutions, that are then merged to form a new 'Offspring' comprised of a diverse set of high quality solutions. After that, in preparation for the second VND, these solutions are ranked and the non-dominated solutions form the 'Offspring rank 1' that is passed as input for the second VND. Each solution in 'Offspring rank 1' is explored by a set of local searches, but differently from the first VND, the local population is initialized with 'Offspring rank 1' and it is not reinitialized while interacting over each solution of 'Offspring rank 1'. This strategy favors diversification in the first VND and intensification in the second VND.

5.5– Multiobjective Evolutionary Algorithm (MOEA)

For comparison purposes, we also implement the Multiobjective Evolutionary Algorithm (MOEA) proposed by Garcia-Najera and Bullinaria (2011) to solve a multi-objective VRPTW. The algorithm comprises a typical evolutionary framework enhanced by a diversification mechanism based on a similarity measure that has outperformed the popular NSGAII. The main loop is presented in Algorithm 6.

Algorithm 6: MOEA

```
 1  Initialize parameters for the VRPTW instance
 2  While $gen \leq numGen$
 3      if ($gen = 1$) then
 4          Generate popSize initial solutions and return Offspring
 5      else
 6          Apply selection using modified tournament
 7          for $i = 1{:}nChi$
 8              Apply crossover and add child to Offspring
 9          end
10      end
11      Offspring' ← Mutation(Offspring)
12      Combine Offspring and Pop
13      Cut-off of the population Pop
14      Compute rank and similarity
15      $gen \leftarrow gen + 1$
16  end
```

In Algorithm 6, line 4, the MOEA starts with a set of *popSize* solutions, each being a randomly generated feasible route. The parent selection (line 6) consists in a modified 2-tournament method, where the first of two parents is chosen on the basis of rank, and the second on the basis of a similarity measure based on Jaccard's similarity coefficient, aiming to maintain population diversity. Regarding the crossover operator (line 8), a random number of routes are chosen from the first parent and copied into the offspring, then all those routes from the second parent which are not in conflict with customers already copied from the first, are copied into the offspring. There are three mutation operators: 'Reallocation' which takes a number of customers from a given route and allocates them to another, 'Exchange' which swaps sequences of customers between two routes, and 'Reposition' which selects one customer from a specific route to reinsert it into the same route. In line 12 both populations are combined, and in line 13, when the population size is exceeded in the last selected front, similarity is computed for the solutions in that front, and the least similar are chosen. Finally, line 13 updates the rank and similarity, and line 14 increments the generation counter.

We implement two versions, both using the same operators from Garcia-Najera and Bullinaria (2011) adapted to our problem formulation. The first version (MOEA) works exactly as described in Algorithm 6 (original form), and in the second version (MOEA-VND) we add a VND loop between lines 11 and 12, using the configuration "Pre-Intensification" in Table 1, Section 5.3. The purpose of the second version is to evaluate whether the inclusion of local searches is able to boost the algorithm's performance, which we believe is a relevant discussion in terms of algorithm design.

6 – Experiments and Results

This section is divided into four sub-sections with different experiments. Section 6.1 shows the influence of different parameters of the statistical method used to compute the service level. Section 6.2 aims to validate the statistical method by comparing it with a benchmark (method proposed by Zhang *et al.*, 2013). Section 6.3 presents results for some specific features of the optimization algorithms (MMA and MILS), and finally, in section 6.4 we have the application of the algorithms to all Solomon instances.

Regarding the instances for Sections 6.1 and 6.2, all 56 well known Solomon's benchmark problems with 100 customers were adapted to generate instances with different probabilities of waiting time (from 0 to 100%). Approximately 160 routes were generated for each instance, using the construction heuristic I1 from Solomon (1987), resulting in a database formed by 9037 routes with 3 to 55 customers. For each of the total number of 101701 customers, we calculated $P(AT_i \leq l_i)$ and $P(AT_i \leq e_i)$, i.e., the service level and the waiting time probability, respectively. We then compute the error of the 203402 calculations as the absolute difference between the computed value and the true value approximated by simulation (Li *et al.*, 2010), using 100000 iterations.

All instances used in the experiments (from Section 6.1 to 6.4) have the same features of the original instances of Solomon, such as vehicle capacity, customer location, time windows and service time. Only a standard deviation for the travel time and service time is added, as explained next.

Let $dis(i,j)$ be the distance between two customers in the original instance, the travel time is a normal distribution left truncated at zero with average $E[TT_{i,j}] = dis(i,j)$ and standard deviation $dev[TT_{i,j}]$ generated by $U[0.1; 0.6] * dis(i,j)$ where U is a uniform distribution. The service time is also a normal distribution left truncated at zero with average $E[ST_i] = s_i$ where s_i is the deterministic service time of the original instance, and standard deviation $dev[ST_i] = U[0.1; 0.6] * E[ST_i]$. As a reference, the range $U[0.1; 0.6]$ is larger than used by Zhang *et al.* (2013) which was $U[0.2; 0.6]$. A second reference is Li *et al.* (2010) with a proportion that varies from 0.07 to 0.2 the value of the mean. In Ehmke et al. (2015), the variation of the travel time is randomly generated uniformly between 0.1 and 0.3. For practical applications, in case service and travel times are Gaussian, we believe it is very unlikely to find a variation outside the range used in this paper.

All algorithms were coded in Matlab R2015a, and tested in a computer dual core Intel i7 2.6 GHz with 16GB RAM, running Windows 10 Pro.

Sections 6.3 and 6.4 report the hypervolume indicator used in Emmerich et al. (2005). Preliminary tests executed Algorithm 1 a number of times in order to have reference points for the max and min values for each objective and for each instance. A normalization is performed on non-dominated solutions for each instance by using $val_m^{new} = \left(val_m - min_m^i\right) * 100/\left(max_m^i - min_m^i\right)$ where m is the index for the objective and i is the index for the instance. The reference point is set as (2000, 2000) and the hypervolume is divided by 400 just for better readability.

6.1 – Parameters discussion for the method to compute the service level

The method has three main parameters: $nint$ (number of intervals in the discretized cumulative function), I (number of intervals to solve the convolution) and S (number of standard deviations), all mentioned in Section 4, where I is used in equation 12, $nint$ in the matrix $mprob_i$ and S to calculate bounds over which we integrate in Equation 12. For this experiment, values $(10,15,20,25,30,40,50)$ were used for $nint$ and I, and values $(2.0,2.5,3.0,3.5,4.0,4.5,5.0)$ for S, totaling $7 * 7 * 7 = 343$ scenarios.

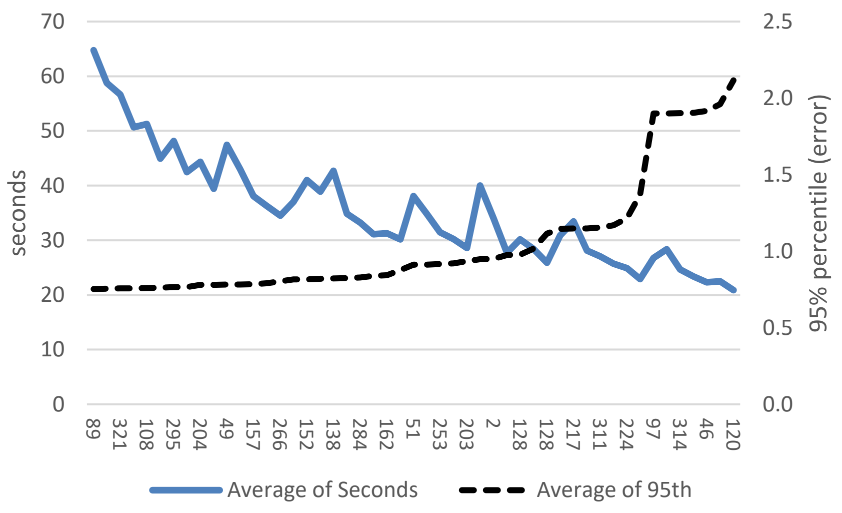

Figure 5: Running time and error for the 49 scenarios

Figure 5 shows the relation between the error (95% percentile of the mean absolute error in percentage points) and the running time, with the ID of the tested scenarios on the x-axis. It helps to choose a convenient value for the parameters. Naturally, the choice depends on the context of the application and in this case, it was selected the scenario 203 with error of 0.93% and running time of 28 seconds, parameters $nint = 20$, $I = 20$ and $S = 3.5$ that presented a good trade-off. Note that the service level computation in a given customer is used when evaluating a move in the local searches. Therefore, the faster the service level calculation, the faster the local search and consequently the metaheuristic.

6.2– Validation of the method to compute the service level

For this experiment, we set $nint = 20$, $I = 20$, and $S = 3.5$ based on the previous section. Table 2 displays the results. The first three metrics give descriptive information for the error (absolute error in percentage points) in 203402 calculations performed: the mean of the error, the standard deviation of the error and the 95th percentile for the error. Metric 4 gives the computational time in seconds (average of 5 runs). The running time is the elapsed time to compute the probabilities for all customers in all routes. The third column of the table has the results for the proposed method, the last two columns refer to the benchmark using two configurations for the discrete parameter L (the main parameter of the benchmark method).

Table 2: Results for the experiment with benchmark

N	Metric	Proposed	L=10	L=20
1	Mean Error (p.p.)	0.198	0.571	0.245
2	Std. Dev. Error (p.p.)	0.337	0.998	0.427
3	95th Percentile (p.p.)	0.930	2.804	1.168
4	Time (seconds)	27.14	109.43	231.75

Table 2 shows that the proposed method obtained better results in all four metrics when compared with the benchmark using $L = 10$. The computational time of the proposed method was 4 times faster. When compared with $L = 20$, the proposed method had slightly better results for the error (metrics 1 to 3) and it was almost 9 times faster. In order to better illustrate the behavior of the error, the error distribution is presented in figures 6 and 7.

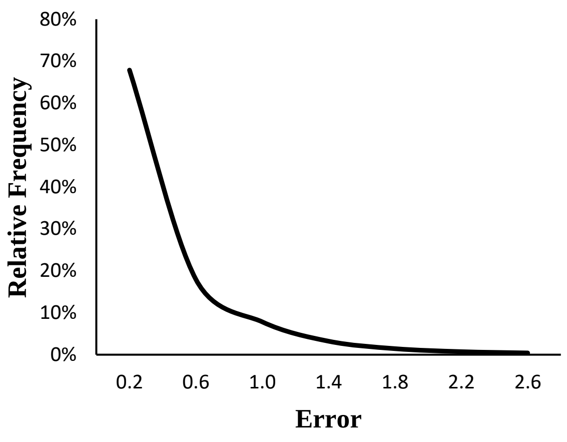

Figure 6: Relative frequencyof the error

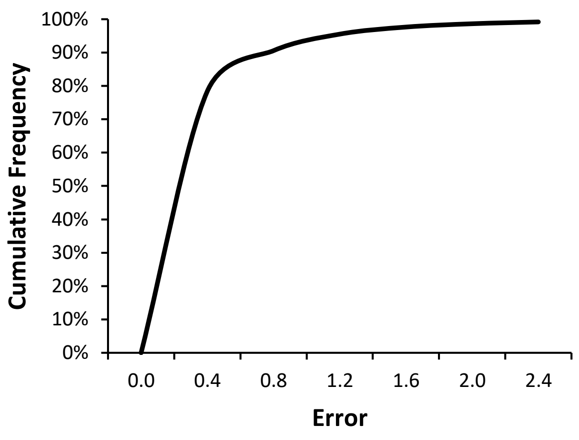

Figure 7: Cumulative distribution of the error

:

Figure 6 shows that the frequency distribution has a positive skew with a long tail on the right. Errors close to zero have high relative frequency evidencing the quality of the results. Figure 7 gives the percentiles.

6.3 –Strategies effectiveness of the algorithms

In order to save computational time, strategies using circular zones and routes/moves tabu are adopted (used in local searches, for instance Algorithm 5, lines 3, 6 and 8). The effectiveness of these strategies is tested in this section.

The circular zones used in the local searches (for example line 6 of Algorithm 5) define a maximum value TT_{max} for the travel time of the customers participating of a move in a local search. Considering the travel times among all customers in the travel time matrix NxN (where N is the number of customers and each element is $E[TT_{i,j}]$), we study the influence of different limit values using percentiles of 20, 30, 40, 50 and 60%, e.g. TT_{max} for 60th percentile means 60% of the arcs of the travel time matrix are smaller than TT_{max}.

Figure 8 shows the influence of different percentiles used to obtain TT_{max} in the computational time and hypervolume of the Pareto front returned by the algorithm. The result from 60th percentile is used as a reference for the other values, dividing the hypervolume obtained with a certain percentile by the hypervolume obtained with 60th percentile. Analog for the running time. The results are expressed as the mean of the running time and hypervolume for five runs of six Solomon instances, the first of each one of the six classes: C101, C201, R101, R201, RC101 and RC201.

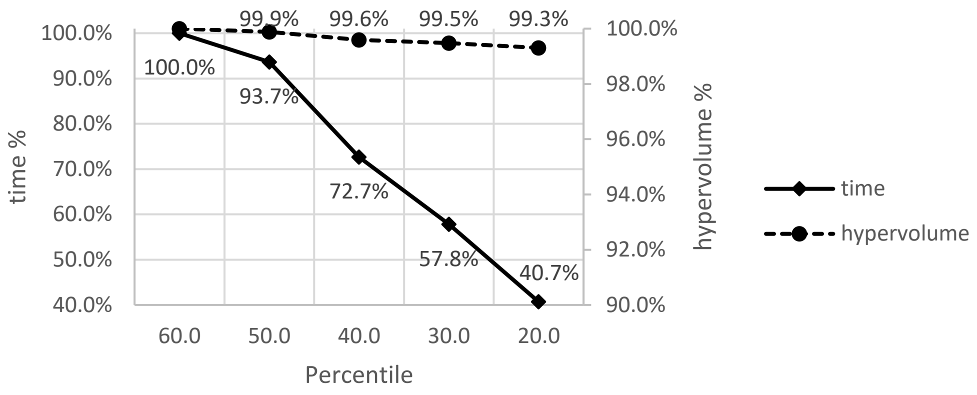

Figure 8: Influence of the circular zone

Figure 8 demonstrates that the lower the percentile, the lower the computational time and the lower the hypervolume. For instance, the computational time for the 20th percentile is 0.407 times (40.7%) the running time of 60th percentile, while the hypervolume for the 20th percentile is 0.993 (99.3%) the hypervolume of 60th percentile. It is relevant to observe that the reduction of the running time is much bigger than the reduction of the hypervolume. Therefore, depending on the context of the application, if a small deterioration of the hypervolume is admissible, then it is possible to reduce substantially the computational time with the utilization of the circular zone.

The strategy involving the routes/moves tabu (for example lines 3 and 8 of Algorithm 5) is also tested. Regarding its influence in the hypervolume, hypothesis tests showed no statistical differences for the variance (Levene's test) and for the mean (ANOVA), with 95% confidence level. Analog tests also showed no statistical difference for the average number of vehicles in the solutions. Figure 9 shows the influence of the routes/moves tabu in the computational time for the same six instances. There are two cases: one using the tabu strategy only for the routes ($1T$) and another using the strategy for routes and also for moves ($2T$). The result with no utilization of any tabu strategy is used as reference: $time\% = time(tabu\ case)/time(without\ tabu)$. We also present the average number of vehicles in the solutions.

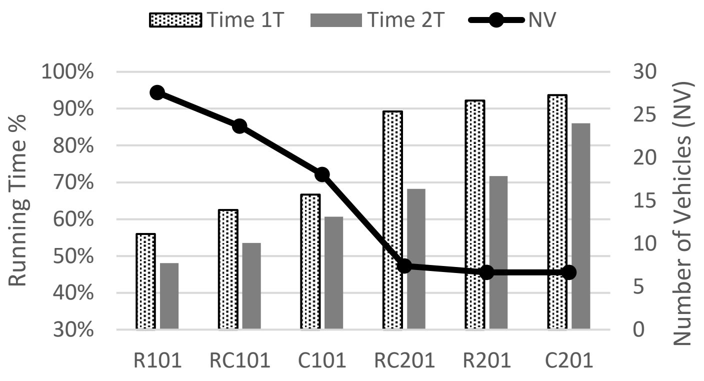

Figure 9 : Influence of the tabu strategy

Figure 9 shows a significant reduction of the computational time for all instances. It can also be seen that the utilization of $2T$ outperforms $1T$. For example, in the instance C101, there is a reduction of approximately 30% of the time by using $1T$, and with the utilization of $2T$, this reduction improved to 40%. The savings strongly depend on the problem instance. The lower the number of vehicles, the lower the savings with the tabu strategy. For instance, with few vehicles, the routes have more customers and consequently much more possible moves in the local search, therefore, the higher the odds of finding a move that will lead to a non-dominated solution and consequently it won't be declared tabu, reducing the savings.

6.4 – Results for the multi-objective algorithms

Because there is no benchmark in the literature approaching the same problem studied here, we decided to implement the MOEA from Garcia-Najera and Bullinaria (2011) using their operators but adapted to our problem formulation described in section 3.2. In addition to that, we decided to run an extended experiment with very long runtime (that could be considered prohibitive in real applications) to have a better idea of results that could potentially be found. By doing this, we run experiments to evaluate the performance of the three developed algorithms: MMA, MILS and MOEA-VND.

For this test, we used the 56 instances of Solomon (1987). As mentioned in the introduction of Section 6, for the instances used in this experiment, the standard deviation ($dev[TT_{i,j}]$) to travel from customer i to j was generated by $U[0.1; 0.6] * dis(i,j)$ where U is a uniform distribution and $dis(i,j)$ is the original distance, here used as the mean of the travel time. Analog for the service time. The instances can be downloaded from the journal website or requested from the corresponding author.

A number of preliminary experiments were performed to find a suitable configuration for all algorithms, involving different configurations for the VND (for example using a random or deterministic sequence of neighborhoods), exit criteria, dynamic adjustments of parameters and others. In these experiments, the best configuration for the parameters was selected such that execution time is approximately the same.

Parameters used for the MMA algorithm: As exit criterion, the main loop of Algorithm 1 is executed while the sum of the hypervolume improvement percentage of the last three generations is greater than 0.5%;

the percentile used for the circular zone is 30%; the number of initial solutions is 15; the number of solutions generated in the crossover is 10; both tabu strategies are used (for routes and moves); the cut-off of the population is made for each generation and solutions with rank 3 or above are eliminated;

Parameters used for MILS (algorithm 2): $maxPert = 10$, $maxIt = 200$. The percentile for the circular zone is dynamically adjusted. After two consecutive perturbations with no improvement, the percentile used is increased in 10%, with maximum value of 60%. When there is an improvement, it is reinitialized with 30%. This dynamic adjustment allows the algorithm to test more possibilities when it is stuck in the current best solution. MILS use one type of VND (Table 1) and not three like MMA. The other parameters are the same as used for MMA. For each one of the 56 instances, we perform 10 runs.

Parameters used for MOEA (Algorithm 6): $popSize = 50$ and $numGen = 500$. For the MOEA-VND: $popSize = 40$ and $numGen = 400$. For all algorithms, the fixed the cost of the vehicle in the objective function of equation (1) to $f = 1000$; the variable cost for the travel time in the objective function of equation (1) is $c = 1$ and the minimum required service level for each customer is 70%.

In the extended experiments, we used a more aggressive configuration aiming for quality (higher hypervolumes) and not running time. For the MMA: Circular zone with 60th percentile and the main loop of algorithm 1 executed while the sum of the hypervolume improvement percentage of the last five generations is greater than 0% (exit criterion). For MILS: $maxPert = 30, maxIt = 1000$. For MOEA: $popSize = 50$ and $numGen = 2000$. For the MOEA-VND: $popSize = 40$ and $numGen = 1600$. As a benchmark we considered the best result among 10 runs of the extended experiment of all algorithms (column “benchmark”, Table 3).

Table 3 summarizes the results presenting the overall hypervolume gap among the 10 runs of the 56 instances, where $gap(\%) = (Benchmark - Mean)/Benchmark) * 100$ and the mean of the computational time in seconds. Figure 10 presents the gaps for each one of the six classes of Solomon instances. We see in Table 3 that MMA presents the best gap among the four algorithms, followed MOEA-VND, while MILS and MOEA have very similar gaps. When we split the results per class, in Figure 10, we see the same behavior from Table 3, with MOEA and MILS presenting very similar gaps.

Table 3: summary of results

Alg.	**MMA**	**MILS**	**MOEA**	**MOEA-VND**
gap	0.396%	1.253%	1.249%	0.915%
seconds	319.54	316.71	321.69	335.41

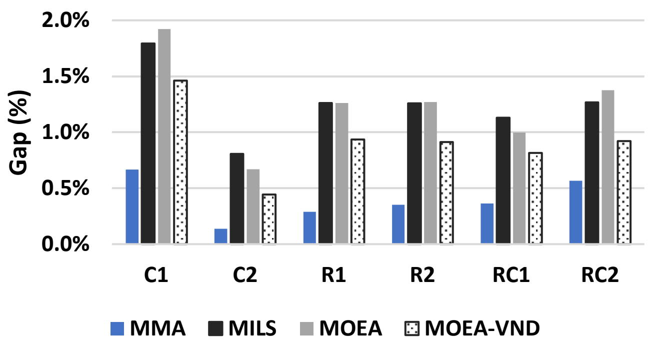

Figure 10: Gaps for Solomon classes

We complement this analysis checking whether there is statistical difference for the mean of the hypervolume among the four heuristics. Because normality tests rejected the hypothesis of normal distribution for the difference of some pairs of heuristics, we apply Wilcoxon signed rank test for a pairwise comparison, testing for no difference in the null hypothesis, with 95% confidence level ($\alpha = 0.05$). The results for this analysis are shown in Table 4 and Figure 11.

Table 4: Statistical Analysis- Wilcoxon pairwise comparison

	Mean difference (%)	**Median difference (%)**	**p-value**
MMA vs MILS	0.83%	0.86%	0.000
MMA vs MOEA	0.84%	0.75%	0.000
MMA vs MOEA-VND	0.50%	0.51%	0.000
MILS vs MOEA	0.01%	-0.11%	0.688
MILS vs MOEA-VND	-0.33%	-0.35%	0.000
MOEA vs MOEA-VND	-0.34%	-0.24%	0.000

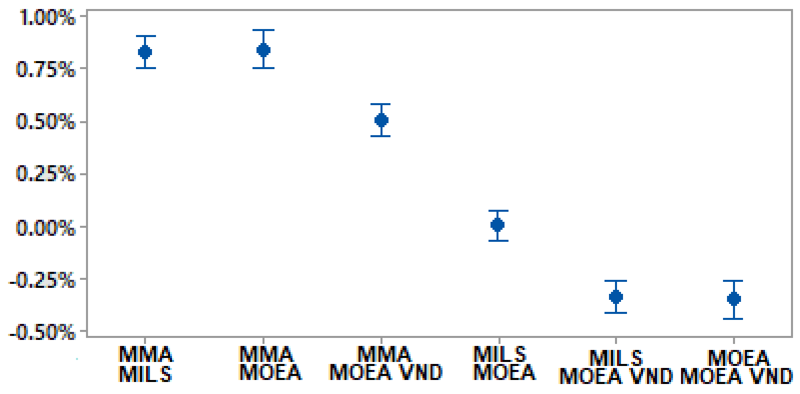

Figure 11: 95% Confidence Interval for the mean difference

We see in Table 4 that there is enough evidence to reject the null hypothesis in all scenarios but MILS vs MOEA, a fact reinforced by Figure 11 with zero within the confidence interval. We also see that there is evidence supporting that MMA outperforms the other three algorithms, which indicates that the proposed strategy to guide the local searches worked properly. While there is no difference between MILS and MOEA, they both are outperformed by MOEA-VND, therefore we can argue that the inclusion of the VND in the main loop of MOEA was successful, working as an intensification mechanism. We also see that MILS is outperformed by MMA. Considering the main difference between these two algorithms is the VND configuration, where MILS has one type and MMA has two types working together (pre-intensification and intensification), we believe it helps to find more diverse solutions and improve the hypervolume.

For a multi comparison study (Derrac et al.,2011) we apply Friedman test. Table 5 shows the Friedman ranks and Table 6 the unadjusted and adjusted p-values. This study reinforces the results of the pairwise study. We see that MMA presented the best median for the hypervolume (the higher the hypervolume the better) and also the best rank (the lower the rank the better). In Table 6, MMA is the control algorithm, confirming that the proposed algorithm indeed outperforms the other 3 algorithms. We used Bonferroni-Dunn as a post-hoc method to adjust the p-value and although this method has less power than others, it was enough to detect difference among the algorithms.

Table 5: Friedman test - Ranks

Algorithm	Median (Hypervolume)	Average Rank
MILS	10079	3.05
MMA	10166	1.66
MOEA	10075	2.91
MOEA2	10102	2.38

Table 6: P-values for the Friedman test (MMA is the control algorithm)

Algorithm	Unadjusted p-value	Adjusted p-value (Bonferroni-Dunn)
MILS	0.0000	0.0000
MOEA	0.0000	0.0000
MOEA2	0.0034	0.0102

Figure 12 shows the mean of the computational time for the six classes of instances for MMA. It is seen that the classes R2, C2 and RC2 take much longer than the others. These classes have bigger capacity of the vehicles and a larger range for the time windows, consequently a higher number of eligible and feasible moves and a higher number of service level computations.

One reason for generating a Pareto front is to give to the decision maker different solutions with a significant range of values for the objectives. Figure 13 shows the average range for each class, for each of the two objectives (service level and operational cost), using MMA. The class C1 obtained the largest range for both objectives, offering to the decision maker solutions with a range up to 5 percentage points for the service level and approximately 4000 units for the cost. The classes RC2 and R2 obtained tighter ranges for the service level and costs.

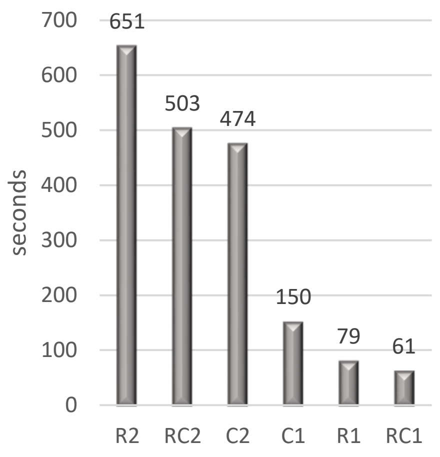

Figure 12: Computational time

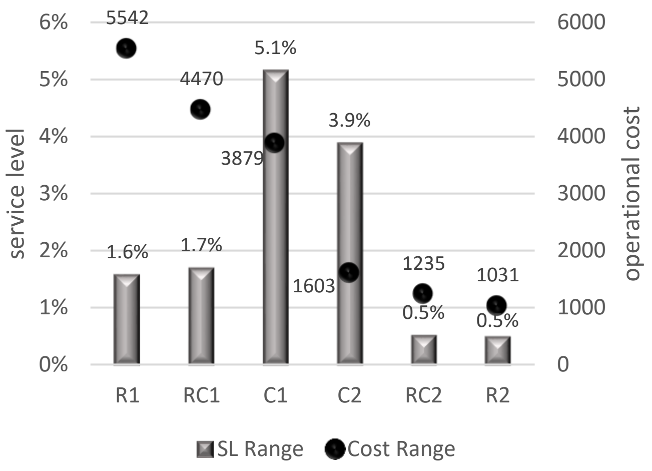

Figure 13: Objectives range

:

Figure 14 presents the Pareto front for 9 instances: C101 (a), C201 (b), C205 (c), R103 (d), R202 (e), R206 (f), RC105 (g), RC205 (h) and RC208 (i). We selected Pareto fronts from instances with different features to give the reader an idea of the types of fronts obtained, in this case from the run with the closest hypervolume to the mean of the respective instance, using MMA. The horizontal axes is the operational cost (divided by 1000) and the vertical axes is the service level (multiplied by -1 so both objectives are minimized).

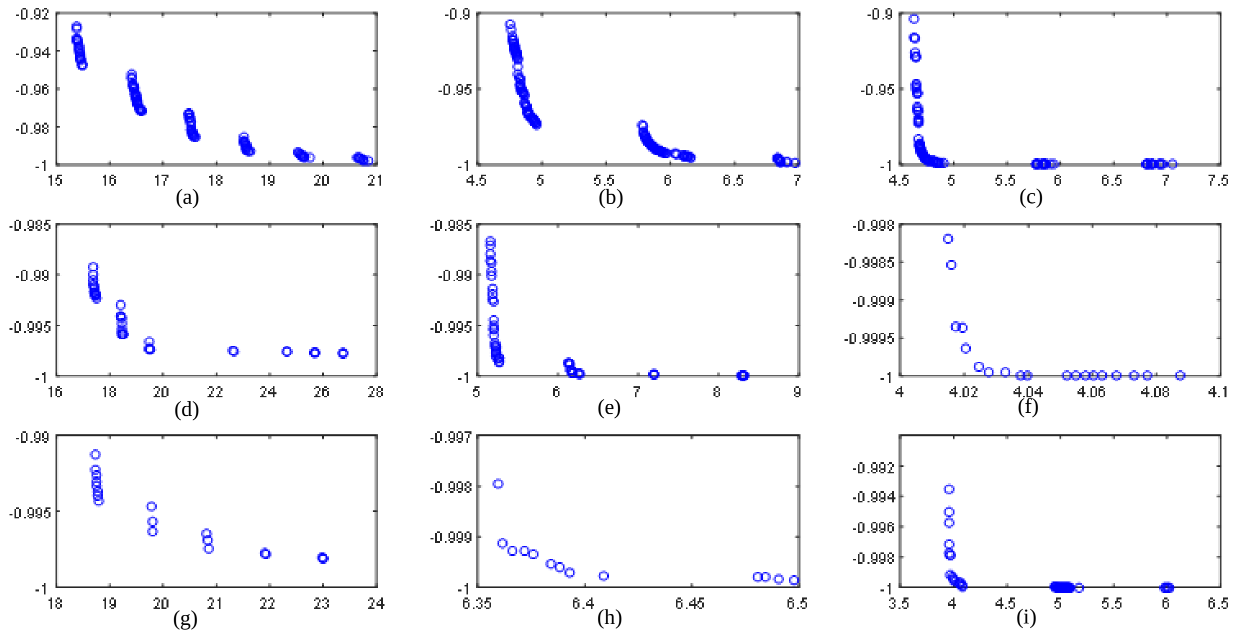

Figure 14: Pareto front for 9 instances

In Figure 14, the Pareto fronts have a discontinuity in the operational cost caused mainly by the number of vehicles. Solutions with the same number of vehicles have a smaller difference in the costs due to the travel time. Generally speaking, it can be observed that the algorithm returns numerous and diverse non-dominated solutions for the decision maker. Some Pareto fronts such as C101 (Figure 14a) have a bigger range for both objectives, offering solutions with significant differences for the operational cost and service level. Other Pareto fronts such as R206 (Figure 14f) present a small range for both objectives. It happened mostly in the class of instances type 2 (C2, R2 and RC2) characterized by a larger range of the time windows when compared with other instances. For instances with large time windows, the service level does not represent a big challenge.

7 – Concluding Remarks and Further Research

We introduced a new practical variant of the VRP, the multiobjective vehicle routing problem with hard time windows and stochastic travel and service times, in which the objectives are the minimization of the transportation cost and the maximization of the service level at customers.

We have interesting findings from the experiments. First, modelling the problem with multi-objective optimization provided indeed a significant range of solutions helping the decision maker to analyze trade-offs between costs and service levels. We also have found that the Pareto-fronts present a broader range of values for both objectives in classes of problems in which the presence of time windows is more significant, especially cases with a tight range for the time windows. Problems with a large range for the time windows tend to present higher service levels, therefore the range of options in the Pareto-front is reduced.

In this paper, the travel time and the service time are modelled as truncated non-negative normal distributions. The results show that the method developed to calculate the service levels obtained small errors, dealing properly with the challenge of having waiting times propagating along the route, affecting the arrival time distribution. Because service level computation is performed in the objective function evaluation, it it has a big impact on computational time. We studied suitable parameters for the statistical method in order to balance time and error. We also incorporated strategies such as circular zones and route/moves tabu to speed up optimization. We found these approaches useful, improving significantly the computation time with a tiny deterioration of the hypervolume.

Finally, through a statistical analysis, we found that the proposed memetic algorithm (MMA) presented better results than the other algorithms. That is an evidence that the innovative features of our memetic algorithm design, such as different VND configurations and local searches returning a set of non-dominated solutions are useful. Future research might focus on the utilization of other probability distributions for the travel time, resolution of time-dependent variants, and larger instances.

8 – References

Aquino, F., Arroyo,R., Claudio J. (2014). A hybrid multi-objective iterated local search heuristic for vehicle routing problem with time windows.14th International Conference on Hybrid Intelligent Systems, pp.117-122

Assis, L. P., Maravilha, A. L., Vivas, A., Campelo, F., & Ramírez, J. A. (2013). Multiobjective vehicle routing problem with fixed delivery and optional collections. *Optimization letters*, 7(7), 1419-1431

Baker, J. E. (1987). Reducing bias and inefficiency in the selection algorithm. In *Proceedings of International Conference on Genetic Algorithms*, pp. 14-21

Balaprakash, P., Birattari,M., Stützle,T. and Dorigo, M. (2015) Estimation-based metaheuristics for the single vehicle routing problem with stochastic demands and customers. *Computational Optimization and Applications*, 61(2), pp 463–487

Baños, R., Ortega, J., Gil, C., FernáNdez, A., & De Toro, F. (2013). A simulated annealing-based parallel multi-objective approach to vehicle routing problems with time windows. *Expert Systems with Applications*, 40(5), 1696-1707.

Berhan, E., Beshah, B. and Kitaw, D. (2014). Stochastic Vehicle Routing Problem: A Literature Survey. *Journal of Information & Knowledge Management*, *13*(03), 1450022.

Binart, S.;Dejax,P.; Gendreau,M. and Semeta F. (2016). A 2-stage method for a field service routing problem with stochastic travel and service times. *Computers & Operations Research*, 65, pp. 64–75

Blickle, T. & Thiele, L. (1996). A comparison of selection schemes used in evolutionary algorithms. *Journal Evolutionary Computation*, 4(4), 361-394.

Castro-Gutierrez, J., Landa-Silva, D., & Pérez, J. M. (2011). Nature of real-world multi-objective vehicle routing with evolutionary algorithms. In IEEE International Conference on Systems, Man, and Cybernetics (SMC), IEEE, pp. 257-264

Chang. T., Wan Y. & Tsang W. (2009). A stochastic dynamic traveling salesman problem with hard time Windows. *European Journal of Operational Research*, 198(3), 748-759.

Chen, Lu, et al. (2014). Optimizing road network daily maintenance operations with stochastic service and travel times." *Transportation Research Part E: Logistics and Transportation Review* 64, pp. 88-102.

Chiang, T. C., & Hsu, W. H. (2014). A knowledge-based evolutionary algorithm for the multiobjective vehicle routing problem with time windows. Computers & Operations Research, 45, pp.25-37.

Derrac, J., García, S., Molina, D., & Herrera, F. (2011). A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. Swarm and Evolutionary Computation, 1(1), 3-18.

Dror, M. (2016). Vehicle routing with stochastic demands: Models & computational methods. In International Series in Operations Research and Management Science, Springer, Vol. 46, pp. 425-449

Ehmke, Jan Fabian, Ann Melissa Campbell, and Timothy L. Urban. (2015). Ensuring service levels in routing problems with time windows and stochastic travel times. *European Journal of Operational Research* 240(2), pp. 539-550.

Emmerich, M.; Beume, N.; Naujoks, B. (2005). An EMO Algorithm Using the Hypervolume Measure as Selection Criterion. Chapter in: Evolutionary Multi-Criterion Optimization. *Lecture Notes in Computer Science*. Volume 3410. pp 62-76.

Errico, F., Desaulniers, G., Gendreau, M., Rei, W., & Rousseau, L. M. (2013). The vehicle routing problem with hard time windows and stochastic service times. *EURO Journal on Transportation and Logistics*, 1-29.

Fleischer, M. (2003). The Measure of Pareto Optima Applications to Multi-objective Metaheuristics. Chapter in: Evolutionary Multi-Criterion Optimization. *Lecture Notes in Computer Science*. Volume 2632, pp. 519-533.

Garcia-Najera, A., & Bullinaria, J. A. (2011). An improved multi-objective evolutionary algorithm for the vehicle routing problem with time windows. *Computers & Operations Research*, *38*(1), pp. 287-300.

Gendreau, M., Laporte, G., & Séguin, R. (1995). An exact algorithm for the vehicle routing problem with stochastic demands and customers. *Transportation Science*, 29(2), pp. 143–155.

Gendreau M.; Laporte G.& Seguin R. (1996). Stochastic vehicle routing. *European Journal of Operational Research*, 88(1), 3-12.

Gendreau M., Jabali O., Rei W. , (2014). Stochastic Vehicle Routing Problems. In: Toth, P., Vigo, D. (Eds.), Vehicle Routing: Problems, Methods, and Applications, Second Edition. SIAM.

Hansen, P., & Mladenovic, N. (2003). *A tutorial on variable neighborhood search*. Groupe d'études et de recherche en analyse des décisions, HEC Montréal.

Hofleitner, A., Herring, R., Abbeel, P. and Bayen, A. (2012). Learning the dynamics of arterial traffic from probe data using a dynamic Bayesian network. *IEEE Transactions on Intelligent Transportation Systems*, *13*(4), 1679-1693.

Laporte, G., Louveaux, F. V., & Hamme, L. (2002). An integer L-shaped algorithm for the capacitated vehicle routing problem with stochastic demands. *Operations Research*, 50(3), 415–423.

Jie G. (2010). Model and Algorithm of Vehicle Routing Problem with Time Windows in Stochastic Traffic Network. *Logistics Systems and Intelligent Management*, vol. 3, pp. 848-851.

Jula H. Dessouky M. & Iannou P. (2006). Truck route planning in nonstationary stochastic networks with time windows at customer locations. *IEEE Transactions on Intelligent Transportation Systems*, 7(1), 51-62.

Kenyon A. S.; Morton D. P. (2003). Stochastic Vehicle Routing with Random Travel Times. *Transportation Science*, *37*(1), 69-82.

Li X.; Tian P. & Leung S. C. H. (2010). Vehicle routing problems with time windows and stochastic travel and service times: models and algorithms. *International Journal on Production Economics*, vol. 125, 137-145.

Lima, F. M., Pereira, D. S., da Conceição, S. V., & de Camargo, R. S. (2017). A multi-objective capacitated rural school bus routing problem with heterogeneous fleet and mixed loads. 4OR, 1-28.

Lin, S. (1965). Computer solutions of the traveling salesman problem. *The Bell system technical journal*, *44*(10), 2245-2269.

Miller-Hooks E, Mahmassani HS (1998). Optimal routing of hazardous materials in stochastic, time-varying transportation networks. *Transportation Research Record*., 1645, pp.143–151

Miranda, D. M. (2011). Metaheuristicas para as variantes do problema de roteameto de veiculos: capacitado, com janela de tempo e com tempo de viagem estocástico. Dissertation: Master’s Degree. Federal University of Minas Gerais (UFMG)

Miranda, D. M. and Conceição, S. V. (2016). The vehicle routing problem with hard time windows and stochastic travel and service time. *Expert Systems With Applications*, 64, pp. 104–116.

Moscato, P., 1999. Memetic algorithms: a short introduction. In: Corne, D., Dorigo, M., Glover, F. (Eds.), New Ideas in Optimization. McGraw Hill, pp. 219–234

Osman I.H. (1993). Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem. *Annals of Operations Research*, *41*(4), 421-451.

Potvin, J.-Y., J.-M. Rousseau. (1995). An exchange heuristic for routing problems with time windows. *Journal of the Operational Research Society, 46*(12), 1433-1446.

Qi, Y., Hou, Z., Li, H., Huang, J., & Li, X. (2015). A decomposition based memetic algorithm for multi-objective vehicle routing problem with time windows. *Computers & Operations Research*, 62, pp. 61-77.

Russell. R. A. and T. L. Urban (2008). Vehicle routing with soft time windows and Erlang travel times. *Journal of Operational Research Society,* 59, pp. 1220-1228.

Shi, Y., Wang, S., Kou, G., Wallenius, J. (2009). New State of MCDM in the 21st Century. *Lecture Notes in Economics and Mathematical Systems*. Springer.

Sivaramkumar, V., Thansekhar, M. R., Saravanan, R., & Miruna Joe Amali, S. (2015). Multi-objective vehicle routing problem with time windows: Improving customer satisfaction by considering gap time. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, 0954405415586608.

Solomon. M. M. (1987). Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations* Research, 35(2), 254-265.

Tas Duygun. Nico Dellaert. Tomvan Woensel. Tonde Kok (2013). Vehicle routing problem with stochastic travel times including soft time windows and service costs. *Computers & Operations Research*, vol. 40, pp. 214–224

Taş, D., Dellaert, N., van Woensel, T., & de Kok, T. (2014). The time-dependent vehicle routing problem with soft time windows and stochastic travel times. *Transportation Research Part C: Emerging Technologies*, *48*, 66-83.

Taillard, É., Badeau, P., Gendreau, M., Guertin, F., & Potvin, J. Y. (1997). A tabu search heuristic for the vehicle routing problem with soft time windows. *Transportation science*, 31(2), pp. 170-186.

Zeimpekis, V. S., Tarantilis, C. D., Giaglis, G. M., & Minis, I. E. (Eds.). (2007). *Dynamic fleet management: Concepts, Systems, Algorithms & Case Studies* (Vol. 38). Springer Science & Business Media.

Zhang J. & William H. K. Lam & Bi Yu Chen (2013). A Stochastic Vehicle Routing Problem with Travel Time Uncertainty: Trade-Off between Cost and Customer Service. *Networks and Spatial Economics*, 13, pp. 471- 496