A statistical learning based approach for parameter fine-tuning of metaheuristics

Laura Calvet[1,*], Angel A. Juan[1], Carles Serrat[2] and Jana Ries[3]

Abstract

Metaheuristics are approximation methods used to solve combinatorial optimization problems. Their performance usually depends on a set of parameters that need to be adjusted. The selection of appropriate parameter values causes a loss of efficiency, as it requires time, and advanced analytical and problem-specific skills. This paper provides an overview of the principal approaches to tackle the Parameter Setting Problem, focusing on the statistical procedures employed so far by the scientific community. In addition, a novel methodology is proposed, which is tested using an already existing algorithm for solving the Multi-Depot Vehicle Routing Problem.

Keywords: Parameter fine-tuning, metaheuristics, statistical learning, biased randomization.

1. Introduction

Mathematical optimization plays an important role both in research and in our everyday lives. Management of portfolios, vehicle routing or DNA sequence assembly, are only some of the fields in which optimization techniques are employed.

Most of the existing proposals to solve optimization problems can be classified into exact methods or heuristic/metaheuristic approaches (Talbi, 2009). The former guarantee the optimality of the solution found. Unfortunately, a number of relevant problems are particularly complex, and tackling them with state-of-the-art exact methods would

[*] *Corresponding author:* Laura Calvet, lcalvetl@uoc.edu

[1] Department of Computer Science, Open University of Catalonia, IN3, 08018 Barcelona, Spain, lcalvetl@uoc.edu, ajuanp@uoc.edu

[3] Department of Mathematics, Technical University of Catalonia, 08028 Barcelona, Spain, carles.serrat@upc.edu

[4] Portsmouth Business School, University of Portsmouth, PO1 3DE, UK, jana.ries@port.ac.uk

Received: March 2015
Accepted: May 2016

require substantial computer memory and time. Problems of this kind are known to be NP-hard (Bovet and Crescenzi, 1994). The Facility Location Problem, the Knapsack Problem and the Multi-Depot Vehicle Routing Problem (MDVRP) are some examples of NP-hard problems. In these cases, heuristics present some experience-based techniques that implement strategies to obtain a sufficiently good solution in a reasonable amount of time. Although they do not provide any theoretical guarantee, they are a popular choice when solving NP-hard problems. Owing to its nature, any heuristic is problem-dependent, which restricts its application to one particular class of problems. Also, heuristics usually provide sub-optimal solutions. These factors have led to the introduction of metaheuristics.

Birattari and Kacprzyk (2009) defines metaheuristics as "general algorithmic templates that can be easily adapted to solve the most different optimization problems". A number of them are nature-inspired, include stochastic components and have several parameters (Boussaïd et al., 2013). They are present in a large number of research areas such as telecommunications (Martins and Ribeiro, 2006), machine learning (Carvalho et al., 2011), and vehicle routing (Gendreau et al., 2008), among others.

Although the performance of metaheuristics is known to depend on its parameter values, the scientific community has not formally addressed the so-called Parameter Setting Problem (PSP) until the end of the last century. According to Eiben et al. (1999), during the first decades of metaheuristics research, many scientists based their choices on tuning the parameters "by hand", i.e. experimenting with different values and selecting the ones that provide the best outputs, or "by analogy", applying settings that have been proven successful for similar problems. More recently, the need for a systematic approach towards setting of metaheuristic parameters has been increasingly outlined in the literature (Hooker, 1995; Johnson, 2002). Subsequently, researchers employ a scientific approach to tackle the PSP more frequently. It is important to highlight that the selection of a systematic methodology leads to a gain of efficiency, as in general, less time is required to fine-tune the parameters while the performance of the metaheuristic is the same if not improved. However, there is no methodology commonly accepted by the scientific community and there is also a lack of publications that compare, in an exhaustive and objective manner, the main approaches and the techniques used so far. Moreover, some of the proposed methodologies are not easily reproducible or are highly metaheuristic and problem dependent. These are some of the reasons why, in spite of the amount of parameter fine-tuning works, many practitioners go on tuning by hand or designing algorithms without parameters (or with a very low number of them), even in the case when more parameterized algorithms could lead to better performances.

This article aims to contribute to the literature by proposing a general and automated statistical learning based procedure to tackle the PSP. It is accompanied by some methodological guidelines to validate the results. In order to test the methodology and illustrate its application, the approach is employed to fine-tune a hybrid algorithm implemented to solve the MDVRP.

The remainder of this article is organized as follows. Section 2 presents a formal definition of the PSP, the existing approaches, and their main contributions. Our methodology is outlined in Section 3, followed by Section 4, which shows its application on a hybrid algorithm. A discussion of the results is reported in Section 5. Finally, Section 6 presents concluding remarks.

2. Related work on the Parameter Setting Problem

Ries et al. (2012) define the PSP as the search for a set of parameter values θ^* in the parameter space Θ such that $\forall \theta \in \Theta : \theta^* \succeq \theta$ (where $\succeq$ denotes a relation of preference), for a given metaheuristic m in the metaheuristic space M, and a given instance x or group of them X in the instance space I. In practice, the amount of time available for experimenting T may be a restriction. In this case, the solution is approximate $(\hat{\theta})$. With regards to the difficulty of this problem, Montero et al. (2014) states that: (a) it is time consuming; (b) the best set of parameter values depends on the problem at hand; and (c) the parameters can be interrelated.

During the last decades, a large number of methodologies have been put forward to solve the PSP. These proposals can be classified in two groups (Birattari and Kacprzyk, 2009): Parameter Control Strategies (PCS), and Parameter Tuning Strategies (PTS). This classification is extended by Instance-specific Parameter Tuning Strategies (IPTS), which include features of the aforementioned groups.

This section provides a brief description of each approach and some of the most cited works. We refer the interested reader to more specific publications such as Eiben et al. (1999), De Jong (2007) and Battiti and Brunato (2010) for an expanded review of PCS, Birattari and Kacprzyk (2009) in the case of PTS, and Ries (2009) for IPTS.

2.1. Parameter Control Strategies (PCS)

These methodologies aim for a dynamic fine-tuning of the parameters by controlling and adapting their values while solving a problem instance. They follow two basic steps: firstly, an initial set of parameter values is chosen; secondly, an adaptation mechanism is integrated which changes relevant parameter values. Most of these strategies apply Adaptive Parameter Control, which means that their adaptation mechanism is based on the assessment of particular information that is stored during the iterative process of a metaheuristic. This information is usually related to the goodness of intermediate solutions. Figure 1 outlines the main instructions of a PCS based on Adaptive Parameter Control. The main drawbacks of this approach are the potentially high computational effort required and the lack of acquired understanding about good parameter values each time an instance is solved.

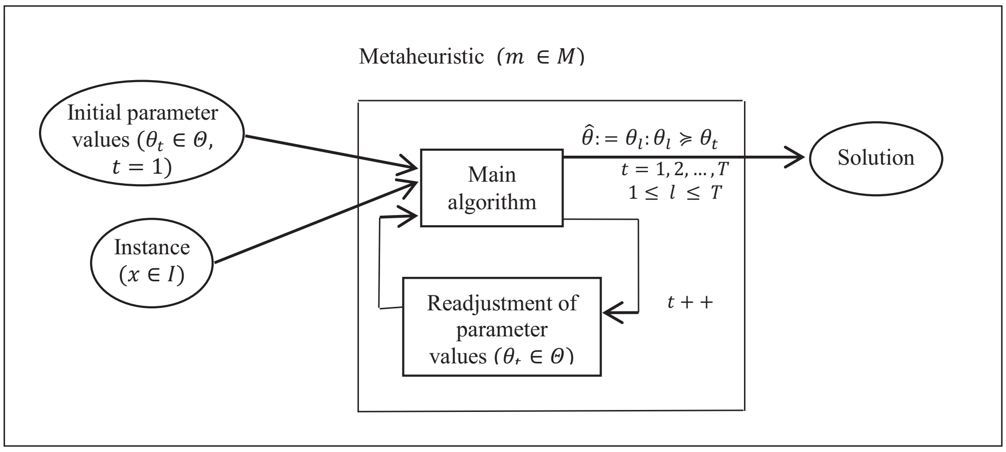

Metaheuristic $(m \in M)$
Initial parameter values $(\theta_t \in \Theta, t = 1)$
Instance $(x \in I)$
Main algorithm
Readjustment of parameter values $(\theta_t \in \Theta)$
$\hat{\theta} := \theta_l : \theta_l \succcurlyeq \theta_t$
$t = 1, 2, \dots, T$
$1 \leq l \leq T$
$t++$
Solution

Figure 1: *Scheme of PCS applying an Adaptive Parameter Control.*

Eiben et al. (1999) addressed the PSP in Evolutionary Algorithms (EAs). Three categories were defined to classify the PCS. The first one, Deterministic Parameter Control, alters the value of a parameter by some deterministic rule, which is usually time based. The second category, Adaptive Parameter Control, does employ feedback to determine the direction and/or magnitude of a parameter change. This is the most used kind of control. Consequently, we will focus on it. The third, Self-Adaptive Parameter Control (Smith, 2008), encodes the parameters to be adapted into the chromosomes of an EA. De Jong (2007) described the main motivations to use dynamic parameter setting strategies in EAs: first, as the running proceeds, information about the fitness landscape is generated, which may be used to improve the performance; also, changing the parameters is needed as an EA "evolves from a more diffuse global search process to a more focused converging local search process".

Table 1: *Representative works employing PCS.*

Work	Main techniques	Metaheuristic	Optimization problem
Battiti and Tecchiolli (1994) and Battiti and Brunato (2005)	Reactive Scheme	Tabu Search (TS)	Quadratic Assignment Problem (QAP), and Maximum Clique Problem
Zennaki and Ech-Cherif (2010)	Support Vector Machines	TS	TSP
Lessmann et al. (2011)	Regression Models	Particle Swarm Optimization (PSO)	Water Supply Network Planning Problem

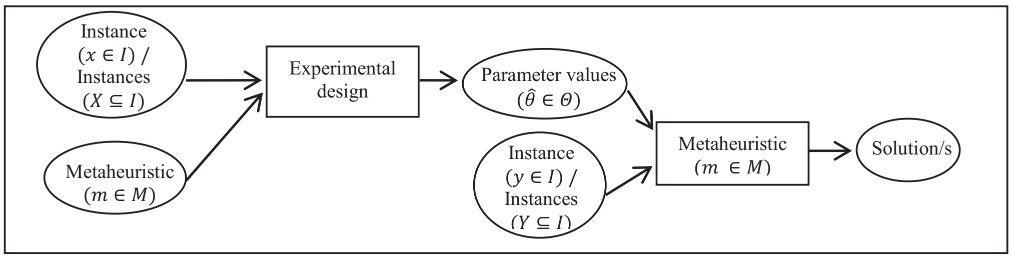


Figure 2: *Scheme of PTS.*

Table 1 gathers a few representative works following this approach. Nowadays, it constitutes a popular choice, mostly in EAs. From the literature, it can be concluded that the parameter fine-tuning is a difficult task, partly due to the potential interactions between parameters (Eiben et al., 1999; De Jong, 2007 and Smith, 2008). The worth of applying PCS is sometimes doubted (Beasley et al., 1993) or not recommended for static optimization problems (De Jong, 2007). However, most authors agree that this approach has a long way to go.

2.2. Parameter Tuning Strategies (PTS)

This approach relies on the concept of robustness (Viana et al., 2005). A robust algorithm provides good results for a given set of instances of a problem using a fixed set of parameter values. The basic procedure (Figure 2) involves finding a set of parameter values providing satisfactory results for a set of instances, usually using statistical and/or optimization techniques. Some authors analyse only a representative subset of instances and apply the set of parameter values found to solve all the instances. This approach also includes the case of solving one instance.

The work of Czarn et al. (2004) is an outstanding contribution from a statistical point of view. It addresses the issues of blocking when using design of experiments (DOE) for variation or noise due to seed, testing individual parameters and interactions, and performing power analyses, among others.

Table 2 shows some works relying on this approach. Many authors focus on minimizing the number of runs, presenting simple models without interactions (e.g., Coy et al., 2001; Pongcharoen et al. 2007 and Xu et al., 1998). DOE and regression analysis are the most employed techniques. The main criticism these works may receive is that most need an initialization of methodology-specific parameters that in some cases is not fully reported. Fortunately, the number of papers that report applications of their methodology in more than one problem or in real-world problems is increasing.

Table 2: *Representative works implementing PTS.*

Work	Main techniques	Metaheuristic	Optimization problem
Park and Kim (1998)	Simplex method	SA	Graph Partitioning Problem, Permutation Flow Shop Scheduling Problem, and Short-term Production Scheduling Problem
Xu et al. (1998)	Tree growing and pruning method based on statistical tests	TS	Steiner Tree-Star Problem
Coy et al. (2001)	DOE and Linear Regression	Routing heuristics	Vehicle Routing Problem
Bartz-Beielstein et al. (2004)	DOE, Classification and Regression Trees, and Design and Analysis of Computer Experiments	PSO and Nelder-Mead Simplex Algorithm	Elevator Group Controller Problem
Ramos et al. (2005)	Logistic Regression	EA	TSP
Birattari and Kacprzyk (2009), Birattari et al. (2010)	Racing Algorithm (Maron and Moore, 1993) and the Friedman's two-way analysis of variance by ranks (Conover, 1999)	Iterated Local Search (ILS) and Ant Colony Optimization (ACO)	QAP and TSP
Adenso-Díaz and Laguna (2006)	DOE and Local Search	Neighbourhood structure, TS, SA, TS, Heuristic based on the SA and the TS, and TS	Steiner Problem, Part-Machine Grouping Problem, Part-Machine Grouping Problem, Single-Machine Scheduling, Proportionate Flowshops, and Bandwidth Packing
Pongcharoen et al. (2007)	DOE	GA	TSP
Ridge and Kudenko (2007)	DOE and Desirability Functions	ACO	TSP
Gunawan et al. (2013)	DOE, Response Surface Methodology and ParamILS (Hutter et al., 2009)	SA	Industry Spares Inventory Optimization Problem

2.3. Instance-specific Parameter Tuning Strategies (IPTS)

As in the case of PCS, IPTS aim for an instance-specific tailoring of the parameters. At the same time, these strategies use a fixed set of parameter values, as the PTS, avoiding the need of modifying the metaheuristic algorithm and reducing the potential computational effort required to adapt parameter values during the algorithmic run. In order to implement these strategies the relation between the parameter values and the performance of the metaheuristic has to be analysed, taking into account instance features. The next step consists in developing a mechanism able to use the features of a new instance to recommend a set of parameter values. The key element is the selection of instance features that are easy and fast to compute, and good at discriminating instances on the shape of their fitness landscapes, which analyse the relationship between the objective function values and the parameters. This learning may take a non-negligible amount of time, but it is assumed that this approach requires less computational time than the PCS approach does. The procedure is shown in Figure 3.

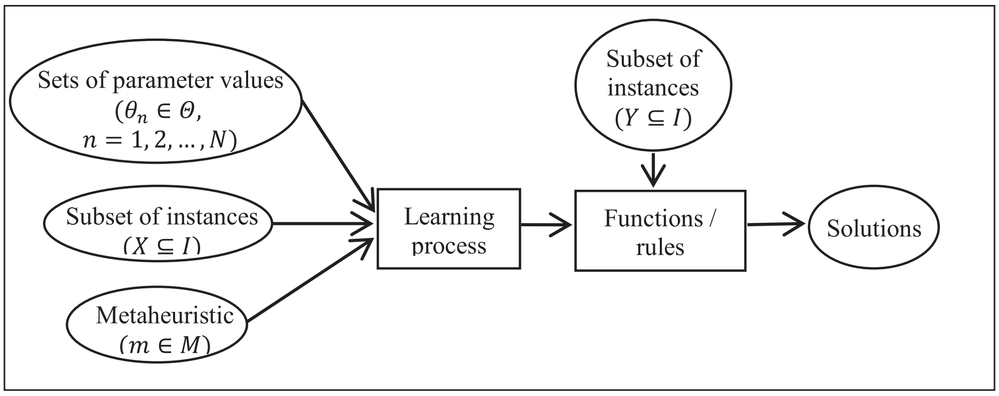

Sets of parameter values ($\theta_n \in \Theta$, $n = 1, 2, \ldots, N$)
Subset of instances ($X \subseteq I$)
Metaheuristic ($m \in M$)
Learning process
Subset of instances ($Y \subseteq I$)
Functions / rules
Solutions

Figure 3: *Scheme of IPTS.*

Some contributions are included in Table 3. The number of works is low since it is relatively new. As in the previous cases, they employ a variety of techniques and analyse several problems.

Table 3: *Representative works implementing PTS.*

Work	Main techniques	Metaheuristic	Optimization problem
Ries (2009)	DOE and Fuzzy Logic	Guided Local Search and GA	TSP
Pavón et al. (2009)	Case-Based Reasoning and Bayesian Networks	GA	Root Identification Problem
Dobslaw (2010)	DOE and Artificial Neural Networks	PSO	TSP

It has been seen that the literature on the PSP is relatively diverse. However, more research is needed to fully explore and compare the performance of different techniques from statistics and operations research (OR), and to achieve that researchers and practitioners become aware of the relevant effect that an adequate parameter-fine tuning may have. In this paper we mainly focus on the parameter fine-tuning of metaheuristic algorithms from an OR perspective. Notice, however, that the literature on parameter fine-tuning of general algorithms is much more extensive, and it has been mainly developed by the computer science community. This community addresses a larger variety of problems (not only of optimization nature), tends to employ algorithms with a larger number of parameters, and uses to consider more complex and/or time-consuming approaches for setting the parameters of different types of algorithms, including searching and classification algorithms, etc. Thus, for example, Ansótegui et al. (2015) or Hutter et al. (2011) describe general but complex methods that can be used in the fine-tuning process of several types of algorithms. These general approaches are rarely considered by the OR community. Accordingly, one of the main contributions of this paper is to provide the OR community with an alternative methodology, which is easier to use and faster, and that can be employed to simplify and make more agile the fine-tuning process of metaheuristic algorithms.

2.4. Approaches comparison

All approaches have different advantages. The dynamic adaptation of the parameter values that characterizes PCS usually provides better results. However, the computational effort tends to be higher. On the other hand, the PTS approach is the easiest and fastest to use, once a set of parameter values is selected. Although the code of the algorithm is not changed, finding an adequate set may be also time-consuming. The last group of strategies represents a compromise solution: it takes less computational time than the PCS approach, but requires implementing a learning mechanism, for which statistical learning skills are needed.

Therefore, there is no approach that stands out from the others. Probably, the most adequate depends on the specific problem to tackle, the instances to solve, the available time and the skills of the researcher. Despite this fact, some general guidelines can be formulated. PTS can be considered as the best option when working with robust algorithms. Regarding IPTS, they are more complex than PTS but provide better results when the algorithm is not robust. In case of prioritizing the algorithm performance, PCS usually constitute the most recommendable approach.

3. Our approach

We propose a methodology that follows the PTS approach. There are several reasons for choosing it. Firstly, it is not computationally intensive, since it may focus on a subset of instances. The inference from a representative sample of benchmark instances to the whole set usually provides good results, specifically if the analysed algorithm is robust. There are two conditions that imply robustness. First, the algorithm has to be little sensitive to small changes in the parameter values, and second, the fitness landscapes for different instances have to be similar. These conditions guarantee that the best set of parameter values for one instance will probably provide good results for the others. The high number of works following this approach, which cover several metaheuristics and optimization problems, shows that many metaheuristic algorithms can be considered robust. Another reason for focusing on PTS is that there is no methodology based on this approach and widely employed, but at the same time, there are plenty of techniques that can be used. Some of them have been intensively tested as DOE and regression analysis. However, others remain to be investigated.

Our methodology is based on clustering (Hastie et al., 2009) and DOE (Montgomery, 2012). These are two well-established techniques that can be easily implemented using free statistical software. Clustering groups instances that have a similar fitness landscape. It facilitates the selection of representative instances and also provides information that can be used to perform a more flexible fine-tuning if each group is treated independently, i.e. exploring the fitness landscape of an instance to find a good set of parameter values and applying it to solve the instances assigned to the same group. Regarding DOE, it enables experimenters to identify and quantify the effects of several parameters and their interactions on the objective function value.

The remainder of this section presents a statistical learning based methodology to obtain a list of sets of parameter values, and a more global procedure to validate and assess its goodness.

3.1. General methodology

A four-step procedure is exposed herein. It is assumed that the experimenter has described and modelled a problem, and has chosen the metaheuristic to tackle it and a set of benchmark instances.

- The first step involves choosing a subset of the instances. Their fitness landscapes will be analysed in order to obtain sets of parameter values that provide good results for them. The subset has to be representative as these sets of parameter values will be used to solve the whole set of instances. An approach to select a representative subset is, firstly, to determine the instance features that have a major influence on which set of parameter values is the most adequate, and then, choose the instances in such a way that the feature values of the subset are representative of those of the entire set of instances. For example, if we have a parameter for which its optimum value is known to depend on the instance size, a representative subset of the instances will present the same proportion of instances of a given size that the whole set does. This approach can be particularly difficult when there are several non-independent parameters. A possible simplification for feature selection consists of choosing those that are commonly used to discriminate instances of a specific problem. Several examples can be found in the literature. Coy et al. (2001) considered, when addressing the Capacitated Vehicle Routing Problem (CVRP), the distribution of customers, the distribution of demand and the location of the depot. Ries et al. (2012) studied the size, the distance metric, a ratio to describe the shape of the area within which a set of cities is distributed and a measure of clustering for the TSP.

In contrast, a problem-independent approach is proposed here. Initially, for a given number of randomly generated sets of parameter values, each instance is solved several times using different seeds for the random number generator of the algorithm (or only once if the algorithm is deterministic). Alternatively, the sets could also be generated using more advanced statistical techniques such as DOE. We consider the median of the objective function values found with the same parameter values but different seeds. The median is a robust measure to aggregate data, but many others could be employed. It is essential to remark the importance that a seed may have in the performance of an algorithm (Juan et al., 2015 and Czarn et al., 2004). Afterwards, feature scaling is applied to the values obtained for each instance. Then, this data is used to cluster instances and select a representative one from each cluster. These instances form the subset to analyse.

Although it is a computationally intensive approach, we think it is effective to assess which instances show a similar relation between parameter values and the performance of an algorithm.

For each instance of the subset, the steps ranging from the second to the fourth are implemented as follows.

- The second step requires selecting the range over which each parameter can be set. Some experience or knowledge about the problem and the metaheuristic may be highly valuable. The ranges should be large enough to cover at least one set of parameter values that can provide a sufficiently good solution with a high probability. On the other hand, a smaller range would allow the experimenter to describe more accurately, with the same resources, the relationship between the parameter values and the objective function value. If there is no a priori information about which are the best regions of the parameter space, a suitable procedure is to perform a rough and fast landscape analysis. Specifically, some possible combinations of parameter values can be selected and utilised to run the algorithm. The best results will identify promising regions. There are several ways of choosing the combinations, as equally-spaced or randomly generated sets. This analysis holds a trade-off between the computational time required and the reliability of the conclusions.

- The third step consists of designing an experiment. A Central Composite Design is studied. Each metaheuristic parameter is considered a factor and the extreme values of its range define the levels. According to this design, the algorithm is executed also several times for each combination of factor values, each one with a different seed.
- In the fourth step, a procedure is developed to search the neighbourhood of the best set of parameter values found. Specifically, another Central Composite Design centred on this set is applied.

Finally, the upshot is a list of recommended sets of parameter values, one per cluster; in particular, those that reported the best results on the last step. The procedure is shown in Figure 4.

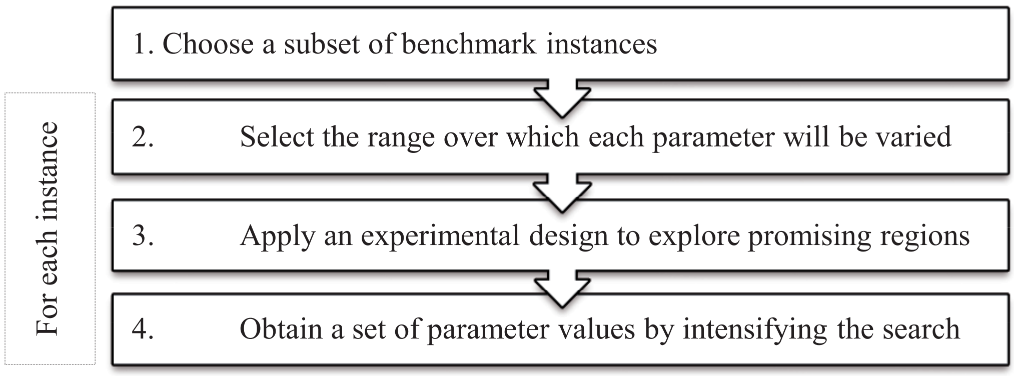
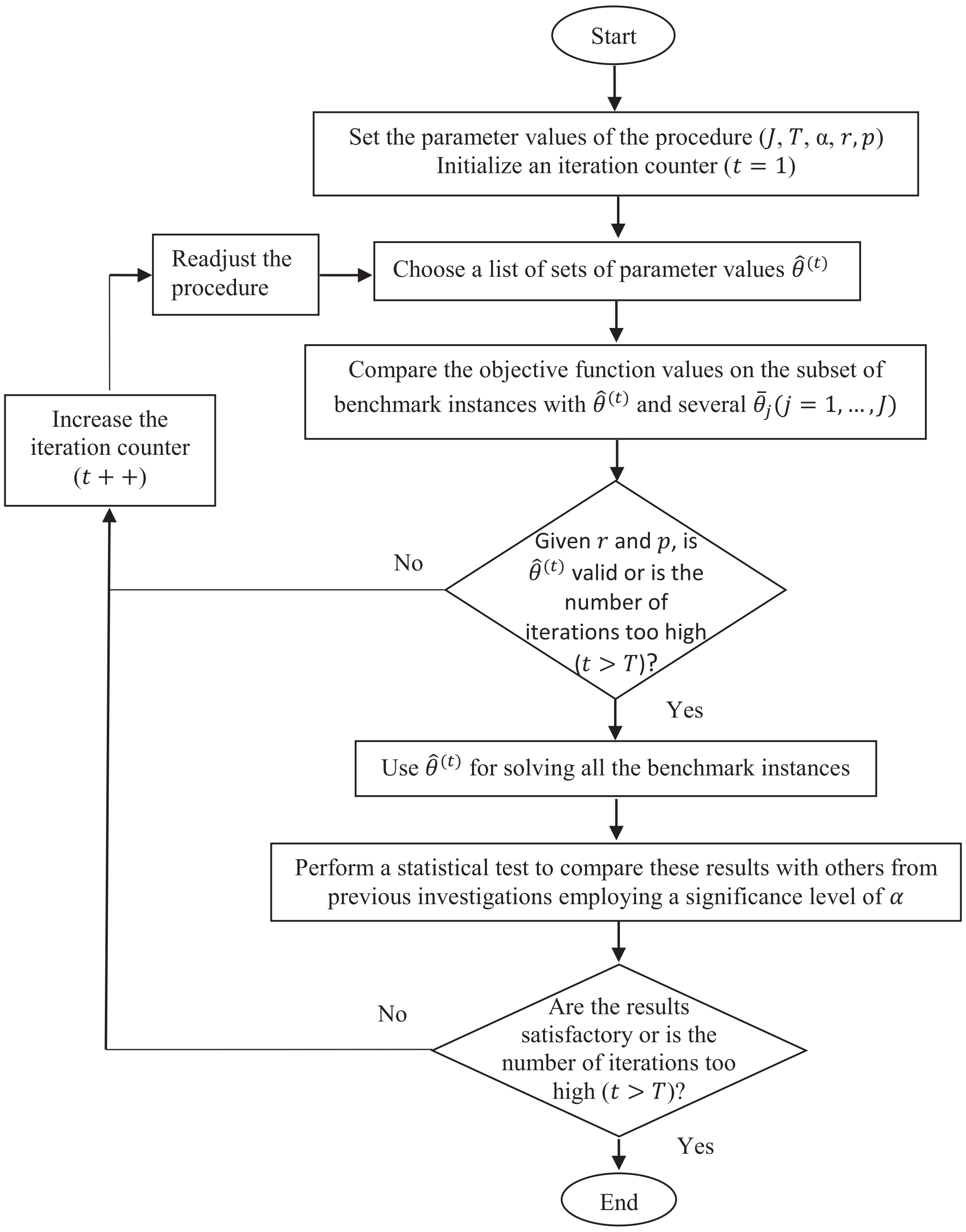

1. Choose a subset of benchmark instances
2. Select the range over which each parameter will be varied
3. Apply an experimental design to explore promising regions
4. Obtain a set of parameter values by intensifying the search
For each instance

Figure 4: *Outline of the procedure for parameter fine-tuning.*

An extended proceeding (Figure 5) is described below in order to validate the list of sets of parameter values obtained and analyse the results provided by it.

Before all else, a list of sets of parameter values, $\hat{\theta} = (\hat{\theta}_1, \hat{\theta}_2, \ldots, \hat{\theta}_K)$ where K is the number of clusters, is chosen as has been explained in the precedent section. Later on, each instance of the subset used to select $\hat{\theta}$ is solved with the corresponding set of $\hat{\theta}$, and with different sets, $\bar{\theta}_j$ $(j = 1, 2, \ldots, J)$ (equally spaced, randomly selected or relatively close to the set of $\hat{\theta}$ according to some distance measure). To assess the performance of a set of $\hat{\theta}$ in a specific instance regarding the other sets, the associated solutions are compared. Given a decision level parameter r $(1 \leq r \leq J+1)$, if the rank of the objective function value provided by the proposed set is equal or lower than r, then it is considered a good set for that instance. Once all the instances of the subset are examined, the proportion of them in which the corresponding set has been classified as good can be calculated. $\hat{\theta}$ is validated by comparing this proportion with a predefined parameter

p $(0 < p < 1)$; if the proportion is higher, then the experimenter has enough evidence of the quality of $\hat{\theta}$ to go on to test it with other instances in the next step.



Start
Set the parameter values of the procedure (J, T, α, r, p)
Initialize an iteration counter $(t = 1)$
Readjust the procedure
Choose a list of sets of parameter values $\hat{\theta}^{(t)}$
Compare the objective function values on the subset of benchmark instances with $\hat{\theta}^{(t)}$ and several $\bar{\theta}_j (j = 1, \dots, J)$
Increase the iteration counter $(t++)$
No
Given r and p, is $\hat{\theta}^{(t)}$ valid or is the number of iterations too high $(t > T)$?
Yes
Use $\hat{\theta}^{(t)}$ for solving all the benchmark instances
Perform a statistical test to compare these results with others from previous investigations employing a significance level of α
No
Are the results satisfactory or is the number of iterations too high $(t > T)$?
Yes
End

Figure 5: *Flowchart representing the proposed methodology.*

If $\hat{\theta}$ is not validated, the process has to be readjusted and restarted. This readjustment may be done in several ways, some options are: checking the robustness and the adequacy of the clustering, adapting the ranges, dedicating more resources to the search, etc. The best strategy is problem-dependent. As a consequence, the choice should rely on the opinion of the experimenter, who will have acquired valuable information from the outputs observed.

Once the list of sets of parameter values has been labelled as valid, it is applied for solving the other instances (each one with the set proposed for the representative instance of the cluster where it has been assigned). To examine the effectiveness of the procedure, it is desirable to compare the solutions with others reported in the literature for the same instances, by performing the t-test for paired samples if data are normal, or the Wilcoxon signed rank test otherwise. If the means (or the mean ranks if data are not normal) do not differ significantly, it may be classified as a satisfactory outcome as it will mean that the proposed methodology, automated and general, has been proven to be competitive. If the results are unsatisfactory, the procedure should be modified and reinitiated.

It is useful to consider that, since the available resources are usually limited, the possible readjustments should be also limited (T represents this limit). Consequently, the process may end without a satisfactory list of sets of parameter values. In this case, the list which provides on average the best solutions will be accepted.

4. Experimental results

4.1. Case study: Biased randomization and ILS for solving the Multi-Depot Vehicle Routing Problem (MDVRP)

In order to test our methodology, it was implemented to fine-tune the parameters of the hybrid algorithm described in Juan et al. (2015), which combines biased randomization and the ILS metaheuristic to address the MDVRP. A brief introduction to both the problem and the algorithm are presented in this subsection.

The MDVRP is a variant of the well-known CVRP that consists in planning routes to service a number of customers with a homogeneous fleet of vehicles that have a maximum capacity. All routes begin and end at one depot, where all resources are initially located. The objective is to find a solution (Figure 6) that minimizes the total cost while satisfying the associated constraints. Typically, these constraints imply that a single vehicle supplies each customer and it cannot stop twice at the same customer. The MDVRP integrates an allocation problem, in which the customers are assigned to one depot, with several CVRPs, one per depot. In the test case, there is also a maximum number of vehicles per depot and a maximum route length. It is considered a challenging problem as allocation and routing issues are interrelated.

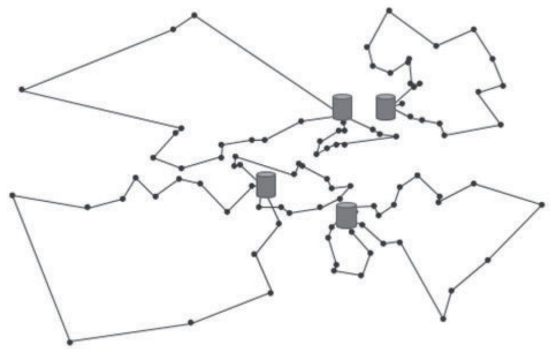


Figure 6: *Solution for a medium-size MDVRP with 4 depots (cylinders).*

The algorithm follows several steps. Initially, a priority list of potentially eligible customers is computed for each depot. The lists are sorted according to a distance-based criterion. Then, they are randomized based on a geometric distribution and used to allocate customers to depots. Afterwards, an initial solution is built by solving each routing problem independently with a version of the Clarke & Wright's Savings (CWS) heuristic (Clarke and Wright, 1964). In short, CWS starts building an initial solution in which each route includes just one customer. Following that, the heuristic considers the possibility of merging two routes if the total cost is reduced. This operation is repeated until no more merges are possible. For this project, the authors developed a biased-randomized version (Juan et al., 2011); while the original seeks always the best possible merging, this version applies biased randomization to select one merging (i.e., multiple solutions can be obtained). In the next phase, an ILS procedure is implemented. A new solution is computed by perturbing the current solution, which implies the reallocation of a given percentage of customers. The new solution replaces the current solution if the former is better. If it is also better than the best solution found so far, the latter is updated. On the other hand, if the new solution is worse than the current one, an acceptance criterion is applied and, consequently, the current base solution can still be modified. This phase ends after a fixed number of iterations. Finally, a post-optimization process is applied to the five best solutions.

This algorithm has three main parameters:

- bM: the parameter of the distribution assigning nodes to depots.
- bR: the parameter of the distribution selecting edges in the CWS heuristic.
- $p*$: the percentage of nodes that are reallocated in the ILS phase.

Note that these parameters take values between 0 and 1.

5. Implementation details

The first step is the selection of a representative subset of instances. Initially, 10 randomly generated sets of parameter values, 7 seeds and the 33 benchmark instances solved in Juan et al. (2015) were selected. Therefore, information from 2310 runs was stored. Data from different seeds was aggregated by computing the median; then feature scaling was applied. The instances that were considered easy-to-solve, those that presented no variation in the results, were separated. This was done to focus the analysis on the instances for which results could be improved by fine-tuning the parameters. Afterwards, a clustering using the *k*-medoids algorithm (Theodoridis and Koutroumbas, 2009) was performed. The range of values considered for setting the value of *k* was 2-12. The final value was selected employing the average silhouette criteria (Rousseeuw, 1987). The composition of the clusters and the representative instances (or medoids) can be observed in Table 4.

Table 4: *Clustering of the benchmark instances.*

Medoids	Clusters
p01	p01
p07	p04, p07, p11, p18, pr02, pr05, pr09
p09	p03, p09, pr04, pr10
p17	p17
p19	p19
p22	p22
p23	p20, p23
pr06	p05, p06, p08, p10, p15, pr01, pr03, pr06, pr07, pr08

Once the subset of instances was formed, the second step, setting the ranges of the parameters, was carried out. After a statistical analysis, it was concluded that just two parameters, *bM* and *bR*, did significantly affect the performance of the algorithm. Therefore, only those two parameters were studied. Five equally spaced values ranging from 0 to 1 were analysed for each parameter. Each instance was solved seven times (considering different seeds) for each possible combination of parameter values. The objective function values were aggregated as before. Then, the values for other possible combinations were estimated by linear interpolation.

The ranges were set to cover the smallest rectangular area of the parameter space that included the lowest objective function values. In particular, the values labelled as the lowest were those meeting the following condition:

$$\text{Objective solution} \leq \text{minimum value} + \beta \cdot (\text{maximum value} - \text{minimum value})$$

The value of β was set at a different value for each instance. More precisely, it was the minimum value that encompassed, at least, 5% of the search space. Figure 7 shows the contour plot and the area in which the search was intensified for each instance.

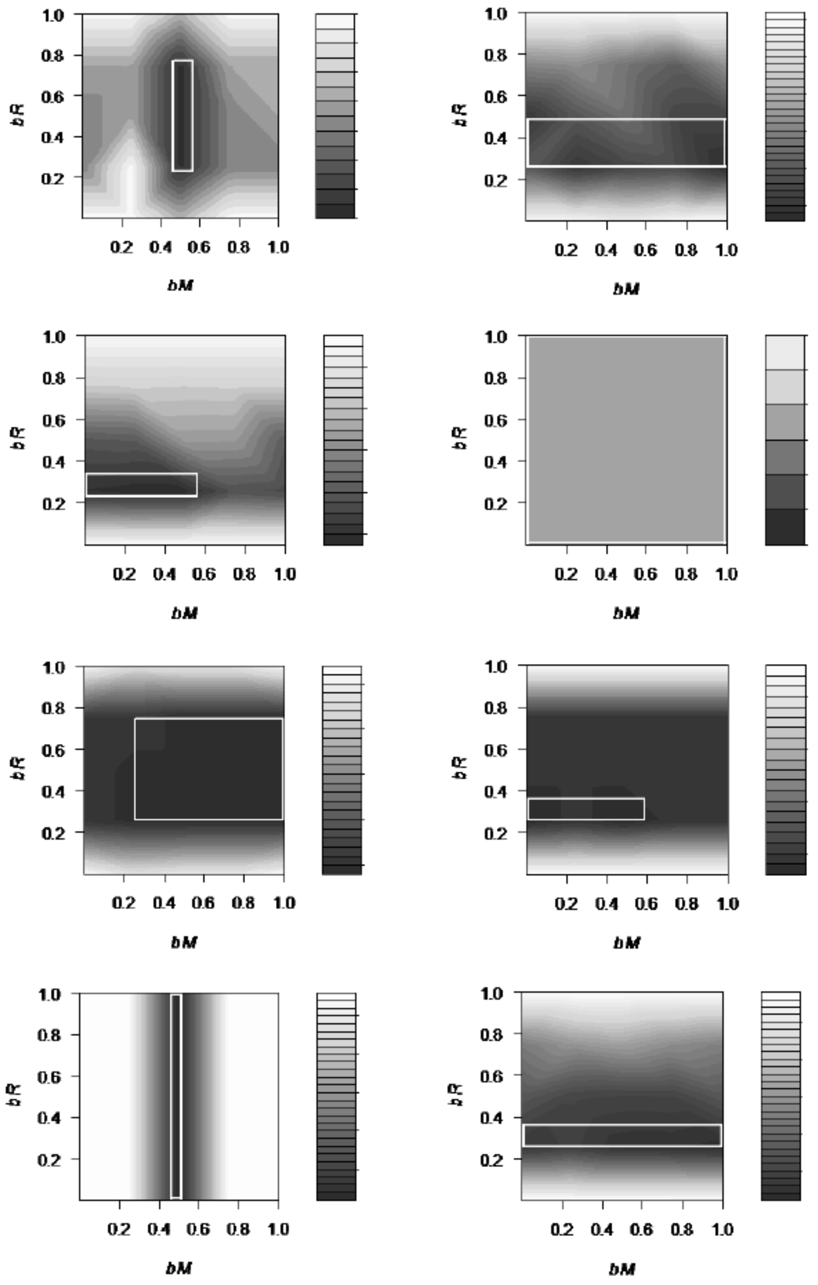


Figure 7: *Contour plots of the medoids sorted from left to right, and top to bottom.*

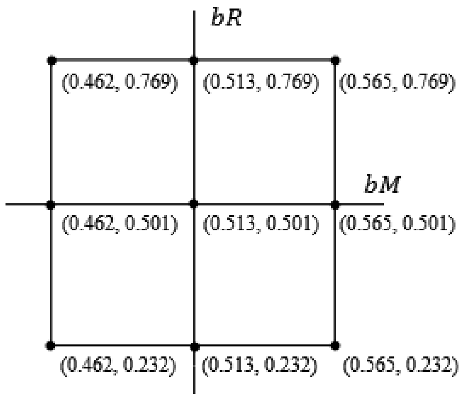


Figure 8: *Scheme of the FCC Design applied to the instance p01.*

The next step was applying a design for each instance of the subset. It was performed to better analyse the relation between the metaheuristic performance and the parameter values. A Face-Centred Central Composite (FCC) Design was selected, as in most of the cases the space parameter could not be expanded (since all parameters could only take values between 0 and 1). Figure 8 displays the scheme for instance p01. The objective function values for the same instance are represented in Figure 9.

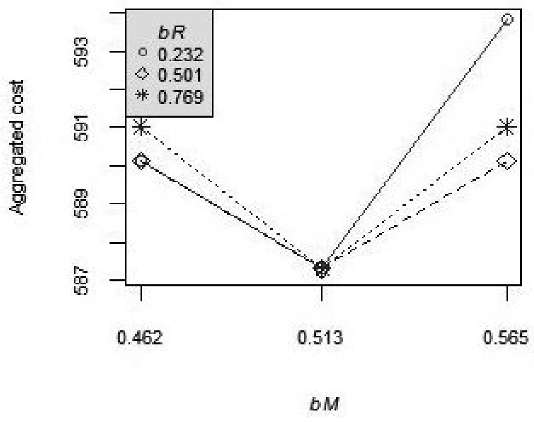


Figure 9: *Solutions of the instance p01.*

Then, the neighbourhood of each set that provided the best solution for an instance was explored applying another FCC Design, centred on that set and covering half of the area analysed with the previous design. The sets that finally presented the best performance were stored. They are outlined in Table 5. Random values were assigned to the instances that did not present variations in the results when changing the parameter values.

Table 5: *Proposed list of sets of parameter values.*

Medoids	Clusters	*bM*	*bR*
p01	p01	0.513	0.501
p07	p04, p07, p11, p18, pr02, pr05, pr09	0.001	0.372
p09	p03, p09, pr04, pr10	0.283	0.283
	p17	random	random
p19	p19	0.443	0.378
p22	p22	0.001	0.231
p23	p20, p23	0.449	0.250
pr06	p05, p06, p08, p10, p15, pr01, pr03, pr06, pr07, pr08	0.500	0.231
	p02, p12, p13, p14, p16, p21	random	random

5.1. Results

The following parameters were chosen to validate the list of sets: $J = 10$, $T = 3$, $\alpha = 0.05$, $r = 6$, $p = 0.7$. The number of sets randomly generated was fixed considering the trade-off between the reliability of our comparisons and the computational time required. The number of iterations was set considering only the time available. The significance level is the one most commonly used in the literature. The value of the fourth parameter is the mean rank that could be expected due to randomness with 11 solutions (1 set proposed and 10 randomly generated). The last parameter was calibrated to force the algorithm to provide good results at most of the instances.

The algorithm was run 7 times with different seeds for each combination of parameter values, the medians and the minimum values were stored. The ranks of the results obtained are detailed in Table 6. Ties receive a rank equal to the average of the ranks they span, shown inside the parentheses.

Table 6: *Ranks of the results provided by our list and by* 10 *random sets.*

Medoids	Rank (medians)	Rank (minimum values)
p01	1	3.5 (1-6)
p07	5	3.5 (1-6)
p09	2	2
p17	2 (1-3)	1
p19	6.5 (2-11)	10.5 (10-11)
p22	11	11
p23	1.5 (1-2)	1
pr06	5	1.5 (1-2)
Valid instances	0.75	0.75

Table 7: *Sets of parameter values for comparison.*

bM	bR	p*
Uniform (0.5, 0.8)	Uniform (0.1, 0.2)	Uniform (0.1, 0.5)

Table 8: *Instances experimental results.*

Inst.	OR medians (1)	OR, minimum values (2)	JR, medians (3)	JR, minimum values (4)	% Gap (1)-(3)	% Gap (2)-(4)
p01	585.000	576.866	593.829	576.866	−1.509	0.000
p02	480.261	476.660	480.261	476.660	0.000	0.000
p03	644.464	641.186	649.229	641.186	−0.739	0.000
p04	1022.085	1019.570	1024.473	1024.062	−0.234	−0.441
p05	760.341	756.281	764.325	754.882	−0.524	0.185
p06	882.827	879.072	880.418	879.763	0.273	−0.079
p07	899.709	897.974	906.395	897.974	−0.743	0.000
p08	4440.534	4434.552	4438.407	4426.747	0.048	0.176
p09	3920.743	3906.561	3923.248	3900.274	−0.064	0.161
p10	3706.763	3667.344	3705.012	3687.054	0.047	−0.537
p11	3598.972	3584.691	3592.891	3585.690	0.169	−0.028
p12	1318.955	1318.955	1318.955	1318.955	0.000	0.000
p13	1318.955	1318.955	1318.955	1318.955	0.000	0.000
p14	1360.115	1360.115	1360.115	1360.115	0.000	0.000
p15	2573.393	2556.846	2573.393	2557.528	0.000	−0.027
p16	2605.565	2585.373	2605.565	2600.099	0.000	−0.570
p17	2720.231	2714.663	2725.799	2725.799	−0.205	−0.410
p18	3831.996	3806.783	3835.388	3806.783	−0.089	0.000
p19	3883.686	3883.686	3883.686	3881.427	0.000	0.058
p20	4080.348	4074.779	4091.482	4091.482	−0.273	−0.410
p21	5706.530	5692.789	5701.902	5692.789	0.081	0.000
p22	5808.738	5806.370	5806.480	5786.288	0.039	0.346
p23	6134.441	6128.873	6145.576	6123.306	−0.182	0.091
pr01	861.319	861.318	861.319	861.318	0.000	0.000
pr02	1330.495	1310.679	1331.543	1314.364	−0.079	−0.281
pr03	1813.634	1813.634	1814.452	1813.634	−0.045	0.000
pr04	2084.843	2077.582	2089.785	2079.832	−0.237	−0.108
pr05	2379.075	2359.947	2379.797	2368.525	−0.030	−0.363
pr06	2709.792	2693.680	2713.593	2696.504	−0.140	−0.105
pr07	1109.235	1109.235	1109.235	1109.235	0.000	0.000
pr08	1680.896	1674.930	1678.872	1674.594	0.120	0.020
pr09	2148.216	2147.192	2153.317	2142.650	−0.237	0.212
pr10	3016.255	3008.129	3028.606	3014.874	−0.409	−0.224

According to our methodology, the list of sets can be considered valid as it presents a rank equal to or below 6 in 75% of the analysed instances, both considering medians and minimum values. In order to test our results, the algorithm was executed with the parameter values suggested in Juan et al. (2015). Both series of results are comparable as were obtained using the same computer and stopping criteria based on the number of iterations. Table 7 presents the parameter values used in the aforementioned paper. Instead of setting fixed values, the authors introduced randomness by employing uniform distributions. The lower and upper bounds were selected after some tests.

Table 8 shows the results obtained solving all instances with the proposed list of sets (our results, OR), and with the set proposed in Juan et al. (2015) (indicated as JR in the table).

6. Discussion of the results

The comparison of the solutions shows that our procedure achieves better results in most of the instances. Table 9 presents the average and the standard deviation of the differences, and the p-values of the test to compare the mean ranks of the results. It is a non-parametric test as the null hypothesis of the Shapiro-Wilk test, a test of normality, was rejected in all cases. The means are negatives, indicating that our methodology provides better solutions. The p-values reveal that the differences of the mean ranks are not statistically significant. Even though, the magnitude of the mean difference can be considered relevant in the context of the MDVRP.

Table 9: *Means and standard deviations of the differences and statistical tests.*

		Mean of the differences	Standard deviation of the differences	P-value of the comparison of mean ranks
All instances	Medians	−0.149	0.330	0.954
	Minimum values	−0.070	0.219	0.980
All instances except the studied subset and those not analysed	Medians	−0.117	0.247	0.942
	Minimum values	−0.100	0.217	0.942

Results on all instances except the subset of representative instances selected initially and those not analysed because of the null variation of their results allow us to demonstrate the good performance of our methodology, which is not directly attributed to the instances deeply studied but to their representativeness, without considering the changes in the instances that where discarded, which are due to randomness.

7. Conclusions

This paper has addressed the Parameter Setting Problem which, due to the relevance of metaheuristics in a number of fields, is increasingly getting more attention.

We have presented an overview of the main approaches: Parameter Control Strategies (PCS), Parameter Tuning Strategies (PTS), and Instance-specific Parameter Tuning Strategies (IPTS). While PCS dynamically adapt the parameter values during the resolution of an instance, PTS leave the parameter values fixed and employ them to solve several instances. IPTS represent a compromise solution, the parameter values are not modified during the search but they can be different for each instance, depending on its features. The benefits and pitfalls of each approach have been discussed. In addition, a new methodology which stands out for being automated and, problem- and metaheuristic-independent, has been presented. It incorporates techniques of clustering, which allows splitting the set of instances and, as a consequence, gives more flexibility to the fine-tuning by analysing each subset independently, and design of experiments. As a result, we have developed a methodology that avoids the strictness of common PTS, which present only a set of parameter values, and the need of modifying the main algorithm and spending more time on the resolution of instances that characterizes PCS. At the same time, our methodology is simpler than IPTS as it does not require a learning procedure able to recommend an instance-specific set of parameter values. In order to illustrate and test our methodology, it has been applied to a hybrid algorithm. The case study provides promising results.

Acknowledgments

This work has been partially supported by the Spanish Ministry of Economy and Competitiveness (TRA2013-48180-C3-P, MTM2012-38067-C02-01, TRA2015-71883-REDT), and FEDER. Likewise, we want to acknowledge the support received by the Department of Universities, Research and Information Society of the Catalan Government (2014-CTP-00001).

References

Adenso-Diaz, B. and Laguna, M. (2006). Fine-tuning of algorithms using fractional experimental designs and local search. *Operations Research*, 54 , 99–114.

Ansótegui, C., Malitsky, Y., Samulowitz, H., Sellmann, M. and Tierney, K. (2015). Model-based genetic algorithms for algorithm configuration. In *Proceedings of the 24th International Conference on Artificial Intelligence* IJ-CAI'15 (pp. 733-739). AAAI Press.
URL: http://dl.acm.org/citation. cfm?id=2832249.2832351.

Bartz-Beielstein, T., Parsopoulos, K. E. and Vrahatis, M. N. (2004). Design and analysis of optimization algorithms using computational statistics. *Applied Numerical Analysis & Computational Mathematics*, 1, 413–433.

Battiti, R. and Brunato, M. (2005). *Reactive search: machine learning for memory-based heuristics*. Technical Report Teofilo F. Gonzalez (Ed.), Approximation Algorithms and Metaheuristics, Taylor Francis Books (CRC Press.

Battiti, R. and Brunato, M. (2010). Reactive search optimization: learning while optimizing. In *Handbook of Metaheuristics* (pp. 543–571). Springer.

Battiti, R. and Tecchiolli, G. (1994). The reactive tabu search. *ORSA journal on computing*, 6, 126–140.

Beasley, D., Bull, D.R., Martin, R.R. et al. (1993). An overview of genetic algorithms: Part 2, research topics. *University computing*, 15, 170–181.

Birattari, M. and Kacprzyk, J. (2009). *Tuning metaheuristics: a machine learning perspective*, volume 197. Springer.

Birattari, M., Yuan, Z., Balaprakash, P. and Stützle, T. (2010). F-race and iterated f-race: An overview. In *Experimental methods for the analysis of optimization algorithms* (pp. 311–336). Springer.

Boussaïd, I., Lepagnot, J. and Siarry, P. (2013). A survey on optimization metaheuristics. *Information Sciences*, 237 , 82–117.

Bovet, D.P. and Crescenzi, P. (1994). *Introduction to the Theory of Complexity*. Hertfordshire, UK, UK: Prentice Hall International (UK) Ltd.

Carvalho, A.R., Ramos, F.M. and Chaves, A.A. (2011). Metaheuristics for the feedforward artificial neural network architecture optimization problem. *Neural Computing and Applications*, 20 , 1273–1284.

Clarke, G. and Wright, J.W. (1964). Scheduling of vehicles from a central depot to a number of delivery points. *Operations research*, 12, 568–581.

Conover, W.J. (1999). *Practical Nonparametric Statistics*. (3rd ed.). John Wiley & Sons.

Coy, S. P., Golden, B.L., Runger, G.C. and Wasil, E.A. (2001). Using experimental design to find effective parameter settings for heuristics. *Journal of Heuristics*, 7, 77–97.

Czarn, A., MacNish, C., Vijayan, K., Turlach, B. and Gupta, R. (2004). Statistical exploratory analysis of genetic algorithms. *Evolutionary Computation, IEEE Transactions on*, 8, 405–421.

De Jong, K. (2007). Parameter setting in eas: a 30 year perspective. In *Parameter setting in evolutionary algorithms* (pp. 1–18). Springer.

Dobslaw, F. (2010). A parameter tuning framework for metaheuristics based on design of experiments and artificial neural networks. In *International Conference on Computer Mathematics and Natural Computing*. WASET.

Eiben, A.E., Hinterding, R. and Michalewicz, Z. (1999). Parameter control in evolutionary algorithms. *Evolutionary Computation, IEEE Transactions on*, 3, 124–141.

Gendreau, M., Potvin, J.-Y., Bräumlaysy, O., Hasle, G. and Løkketangen, A. (2008). *Metaheuristics for the vehicle routing problem and its extensions: A categorized bibliography*. Springer.

Gunawan, A., Lau, H.C. and Wong, E. (2013). Real-world parameter tuning using factorial design with parameter decomposition. In *Advances in Metaheuristics* (pp. 37–59). Springer.

Hastie, T., Tibshirani, R. and Friedman, J. (2009). *The elements of statistical learning*. (2nd ed.). Springer.

Hooker, J.N. (1995). Testing heuristics: We have it all wrong. *Journal of Heuristics*, 1 , 33–42.

Hutter, F., Hoos, H.H. and Leyton-Brown, K. (2011). Sequential model-based optimization for general algorithm configuration. In *Learning and Intelligent Optimization* (pp. 507–523). Springer.

Hutter, F., Hoos, H.H., Leyton-Brown, K. and Stützle, T. (2009). Paramils: an automatic algorithm configuration framework. *Journal of Artificial Intelligence Research*, 36, 267–306.

Johnson, D.S. (2002). A theoreticians guide to the experimental analysis of algorithms. *Data structures, near neighbor searches, and methodology: fifth and sixth DIMACS implementation challenges*, 59, 215–250.

Juan, A.A., Faulin, J., Jorba, J., Riera, D., Masip, D. and Barrios, B. (2011). On the use of monte carlo simulation, cache and splitting techniques to improve the clarke and wright savings heuristics. *Journal of the Operational Research Society*, 62 , 1085–1097.

Juan, A.A., Pascual, I., Guimarans, D. and Barrios, B. (2015). Combining biased randomization with iterated local search for solving the multidepot vehicle routing problem. *International Transactions in Operational Research*, 22, 647–667.

Lessmann, S., Caserta, M. and Arango, I.M. (2011). Tuning metaheuristics: A data mining based approach for particle swarm optimization. *Expert Systems with Applications*, 38, 12826–12838.

Maron, O. and Moore, A.W. (1993). Hoeffding races: Accelerating model selection search for classification and function approximation. *Robotics Institute*, (p. 263).

Martins, S.L. and Ribeiro, C.C. (2006). Metaheuristics and applications to optimization problems in telecommunications. In *Handbook of optimization in telecommunications* (pp. 103–128). Springer.

Montero, E., Riff, M.-C. and Neveu, B. (2014). A beginner's guide to tuning methods. *Applied Soft Computing*, 17, 39–51.

Montgomery, D.C. (2008). *Design and analysis of experiments*. (8th ed.). John Wiley & Sons.

Park, M.-W. and Kim, Y.-D. (1998). A systematic procedure for setting parameters in simulated annealing algorithms. *Computers & Operations Research*, 25, 207–217.

Pavón, R., Díaz, F., Laza, R. and Luzón, V. (2009). Automatic parameter tuning with a bayesian case-based reasoning system. a case of study. *Expert Systems With Applications*, 36, 3407–3420.

Pongcharoen, P., Chainate, W. and Thapatsuwan, P. (2007). Exploration of genetic parameters and operators through travelling salesman problem. *Science Asia*, 33, 215–222.

Ramos, I.C., Goldbarg, M.C., Goldbarg, E.G. and Neto, A.D.D. (2005). Logistic regression for parameter tuning on an evolutionary algorithm. In *Evolutionary Computation, 2005. The 2005 IEEE Congress on* (pp. 1061-1068). IEEE volume 2.

Ridge, E. and Kudenko, D. (2007). Analyzing heuristic performance with response surface models: prediction, optimization and robustness. In *Proceedings of the 9th annual conference on Genetic and evolutionary computation* (pp. 150–157). ACM.

Ries, J. (2009). *Instance-based flexible parameter tuning for meta-heuristics using fuzzy-logic*. Ph.D. thesis University of Portsmouth.

Ries, J., Beullens, P. and Salt, D. (2012). Instance-specific multi-objective parameter tuning based on fuzzy logic. *European Journal of Operational Research*, 218, 305–315.

Rousseeuw, P.J. (1987). Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20 , 53–65.

Smith, J.E. (2008). Self-adaptation in evolutionary algorithms for combinatorial optimisation. In *Adaptive and Multilevel Metaheuristics* (pp. 31–57). Springer.

Talbi, E.-G. (2009). *Metaheuristics: from design to implementation*, volume 74. John Wiley & Sons.

Theodoridis, S. and Koutroumbas, K. (2009). *Pattern Recognition*, volume 74. John Wiley & Sons.

Viana, A., Sousa, J.P. and Matos, M.A. (2005). Constraint oriented neighbourhoodsa new search strategy in metaheuristics. In *Metaheuristics: progress as real problem solvers* (pp. 389–414). Springer.

Xu, J., Chiu, S.Y. and Glover, F. (1998). Fine-tuning a tabu search algorithm with statistical tests. *International Transactions in Operational Research*, 5, 233–244.

Zennaki, M. and Ech-Cherif, A. (2010). A new machine learning based approach for tuning metaheuristics for the solution of hard combinatorial optimization problems. *Journal of Applied Sciences*, 10, 1991–2000.