
Fine-tuning of Algorithms Using Fractional Experimental Designs and Local Search

BELARMINO ADENSO-DÍAZ

*Escuela Superior de Ingenieros Industriales, Campus de Viesques,
Universidad de Oviedo, 33204-Gijón (Spain)*
adenso@etsiig.uniovi.es

MANUEL LAGUNA ^(A)

*Leeds School of Business
University of Colorado, Boulder, CO 80309-0419*
laguna@colorado.edu
Tel. (303) 492-6368 Fax. (303) 492-5962

Latest revision: May 27, 2004

^(A) Corresponding author

Abstract — Researchers and practitioners frequently spend more time fine-tuning algorithms than designing and implementing them. This is particularly true when developing heuristics and metaheuristics, where the “right” choice of values for search parameters has a considerable effect on the performance of the procedure. When testing metaheuristics, performance typically is measured considering both the quality of the solutions obtained and the time needed to find them. In this paper, we describe the development of CALIBRA, a procedure that attempts to find the best values for up to five search parameters associated with a procedure under study. Since CALIBRA uses Taguchi’s fractional factorial experimental designs coupled with a local search procedure, the best values found are not guaranteed to be optimal. We test CALIBRA on six existing heuristic-based procedures. These experiments show that CALIBRA is able to find parameter values that either match or improve the performance of the procedures resulting from using the parameter values suggested by their developers. The latest version of CALIBRA can be downloaded for free from http://coruxa.epsig.uniovi.es/~adenso/file_d.html.

Keywords: Parameter setting, Taguchi design of experiments, heuristic search.

1. INTRODUCTION

The goal of our study is to address the issue of fine-tuning algorithms from the point of view of creating a system that can assist researchers and practitioners in this important and time-consuming task. The time required to fine-tune an algorithm sometimes far exceeds the development time. For the purpose of this study, we consider three types of algorithms:

Exact — Procedures that guarantee finding an optimal solution if allowed to complete their execution. Branch and bound, cutting planes and dynamic programming procedures fall in this category.

Heuristic — Procedures that do not guarantee finding an optimal solution and that do not attempt to escape local optimality by mechanisms such as solution perturbation or multi-starting. Local search procedures such as those used in nonlinear optimization, and constructive procedures such as those used in combinatorial optimization, fall in this category.

Metaheuristic — Procedures that guide and modify heuristics to produce solutions beyond those that are normally generated in a quest for local optimality. Procedures based on evolutionary approaches, tabu search, simulated annealing, and multi-start strategies fall in this category. Hybrid procedures based on metaheuristic frameworks are also considered metaheuristics.

There is anecdotal evidence that about 10% of the total time dedicated to designing and testing of a new heuristic or metaheuristic is spent on development, and the remaining 90% is consumed fine-tuning parameters. The method introduced in this study has the goal of assisting in the process of designing, implementing and testing algorithmic approaches for hard optimization problems. The fine-tuning procedure proposed here is not customized to either heuristic, metaheuristic or exact procedures; however, the quality of the parameter values might have to be measured differently in each case. For instance, the effectiveness of a given set of parameter values in the context of a constructive heuristic depends on the quality of the solutions found, since speed is typically not a factor. The effectiveness of a given set of parameter values in the context of a metaheuristic procedure typically depends on the combination of quality and speed. This is also true for global optimization procedures for nonlinear optimization. Finally, the effectiveness of a given set of parameter values in the context of an exact procedure (e.g., a branch and bound method for integer programming) depends solely on speed since the solutions are confirmed to be optimal if the procedure is allowed to run to completion. Some algorithms, for example, may be conceived and implemented in a relatively short period of time (one or two weeks) and then require much longer to be fine-tuned (one or two months). Barr, et al. (1995) state the importance of fine-tuning heuristic and metaheuristic procedures:

“The selection of parameter values that drive heuristics is itself a scientific endeavor and deserves more attention than it has received in the operations research literature. This is an area where the scientific method and statistical analysis could and should be employed.”

Fink and Voss (2002) also address the importance of fine-tuning when creating a framework of reusable software components in the metaheuristic domain:

“The adaptation of metaheuristics to a specific type of problem may concern both the static definition of problem-specific concepts such as the solution space or the neighborhood structure as well as the tuning of run-time parameters (calibration). The latter aspect of designing robust (auto-adaptive) algorithms, though being an important and only partly solved research topic, may be mostly hidden from the user.”

Our current development employs statistical analysis techniques and a local search procedure to create a systematic way of fine-tuning algorithms within a specified range of values. We created this procedure (referred to as CALIBRA) with the goal of providing a tool for fine-tuning algorithms that helps an analyst to search for parameter values in a range of interest using a training set of instances and guided by a measure of performance. CALIBRA performs a systematic search for parameter values within the specified ranges employing the measure of performance specified by the analyst (e.g., solution quality) as a guiding mechanism.

A casual review of recent publications in operations research or management science journals would reveal procedures whose performance depends on the selection of values for a set of parameters. This is more evident in the case of metaheuristic-based procedures, which have become the solution methodology of choice in a variety of application areas. In the literature, the selection of particular parameter values is commonly justified in one of the following ways:

1. Most of the studies simply provide a list of the parameters and their preferred value. The specific choices are justified with statements such as: “... parameter values have been established experimentally.” In most cases, however, there is no specific reference to the type of experiments employed to arrive to the selected parameter values. Also, there is no discussion about the sensitivity of the parameter values with respect to the performance of the procedure.
2. Many other studies provide the parameter values without any further explanation. Frequently, the parameter values are different for each problem class (or even each instance within a class) and no explanation is given.
3. In a number of cases, the authors choose parameter values that have been reported in other studies to be effective. Frequently, this is done without confirming that the values are also effective in the current context. This practice is more common in implementations of simulated annealing and genetic algorithms. In the case of simulated annealing, for example, some general cooling schedules have been established in the literature (e.g. see van Laarhoven and Aarts, 1987) and other authors use them without further experimental confirmation of their effectiveness. In the case of genetic algorithms, several authors (DeJong and Spears, 1990; Grefenstette, 1986) have defined what they consider “good practices” for the implementation of procedures based on genetic algorithms. These common practices then are utilized in many other studies.
4. In a small number of cases, the papers report the experimental designs that were employed to establish the parameter values used for computational testing.

We believe that the application of a systematic method, such as the one that we propose here, is a superior approach to parameter fine-tuning than the approaches outlined above, with the possible exception of the fourth one. Best practices associated with the application of experimental design techniques for computational testing can be found in Barr, et al. (1995)

and Coy, et al. (2001). Our paper is organized as follows. In section 2, we describe two systematic techniques that have been used to fine-tune algorithms. We describe our proposed procedure in section 3. In section 4, we apply the proposed methodology to six existing solution algorithms. We finish the paper with some concluding remarks in section 5.

2. FINE-TUNING TECHNIQUES

Fine-tuning of parameters generally is regarded as a painful but necessary task. The process by which the values of parameters are set varies considerably from one research project to another. In some cases, parameter values are found using ad-hoc experimentation on a reduced set of problem instances. This may lead to values that produce inferior results, limiting the performance of the procedure. A more rigorous way of fine-tuning a procedure involves the use of either experimental design or a global optimizer. Regardless of the fine-tuning method, a single set of values may not generalize and produce good results on the entire set of problems. In other words, a single set of parameter values applied to an entire set of problems may produce equal results but not better than individual parameter values customized to each problem instance in the set. We address this issue in section 4.10.

2.1 Design of Experiments

Statistical design of experiments is the process of first planning an experiment so appropriate data are collected and then analyzing these data by statistical methods to be able to draw valid and objective conclusions. Thus, there are two aspects to any experimental problem: the design of the experiment and the statistical analysis of the data. These two subjects are closely related, since the method of analysis depends directly on the design employed (Montgomery, 1997).

The design of experiment refers to the structure of the experiment, with particular reference to:

1. The set of treatments included in the study
2. The set of experimental units included in the study
3. The rules and procedures by which the treatments are assigned to the experimental units
4. The measurements that are made on the experimental units after the treatments have been applied

Statistical designs of experiments are concerned with the third element—the rules and the procedures whereby treatments are assigned to the experimental units. Neter, et al. (1996) provide a comprehensive treatment of statistical experimental design and analysis.

One of the most popular designs for exploratory experimental studies is the factorial design. In a factorial design, the analyst selects k factors that are believed to affect the response of the system. Then, two or three critical values for these factors are chosen. If two values are selected, the design is known as a 2^k factorial design. Similarly, a 3^k design corresponds to choosing three values (or levels) for each of the k factors. In the context of fine-tuning algorithms, the factors correspond to the parameters and the levels to the critical values to be examined. All combinations of the parameters are tested in a full factorial design. Therefore, a full factorial design of a procedure with 5 parameters and 3 critical values would require $3^5 = 243$ experiments. When the procedure is a metaheuristic that includes random elements in its search strategies (for example a genetic algorithm), replications of each experiment may be necessary to collect meaningful data for analysis.

Full factorial designs quickly can become impractical, because the number of experiments exponentially increases with the number of parameters. A more practical alternative is to employ fractional factorial designs. These designs draw conclusions based on a fraction of experiments that are strategically selected from the set of all possible experiments in the full factorial design. Fractional factorial designs allow the study of a large number of factors with relatively few experimental trials. One of the most notable proponents of the use of fractional factorial designs is Genichi Taguchi (Roy, 1990). Taguchi proposed a special set of orthogonal arrays to lay out experiments associated with quality improvement in manufacturing. A balanced array is an $n \times k$ array A with entries from a finite set S ($|S|=s$) satisfying the following: Let y be a t -vector of S^t , then in any t rows of A , y appears $\lambda(y)$ times as columns; for any permutation P on the coordinates of y , $\lambda(y) = \lambda(P(y))$.

If $\lambda(y)$ is a constant number for any y of S^t then the array A is called an *orthogonal array*. An orthogonal array with n runs, k factors, s levels, and strength t is an array of size $n \times k$, with entries from 0 to $s-1$, with the property that in any t columns the s^t possible combinations appear equally often. For example, the following is an orthogonal array with $n = 8$, $k = 5$, $s = 2$ and $t = 2$:

```

0 0 0 0 0
1 0 0 1 1
0 1 0 1 0
0 0 1 0 1
1 1 0 0 1
1 0 1 1 0
0 1 1 1 1
1 1 1 0 0

```

The aim of an experiment is to investigate not only the effects of the individual variables (or factors) on the outcome, but also how the variables interact. As mentioned before, even with a moderate number of factors and a small number of levels for each factor, the number of possible level combinations of the factors increases rapidly. Therefore, it may not be feasible to make even one observation at each of the level combinations. In such cases, observations are made at only some of the level combinations and the purpose of the orthogonal array is to specify which level combinations are to be used. Such experiments are called fractional factorial experiments. A comprehensive study of orthogonal arrays can be found in Hedayat, Sloane and Stufken (1999).

Taguchi's orthogonal arrays are the result of combining orthogonal Latin squares¹ in order to create "versatile recipes" that apply to several experimental conditions. For example, a design of experiments involving 4, 5, 6, or 7 factors can all be performed using the same orthogonal array (L_8). We use Taguchi's $L_9(3^4)$ orthogonal array in CALIBRA. This array can handle up to 4 parameters with 3 critical values running only 9 experiments.

Although the use of the Taguchi method to algorithmic parameter fine-tuning is new, the application of statistical experimental design to algorithmic performance analysis has a rather long history. Discussions on the experimental design and statistical analysis of computational

¹ A Latin square design is a design with r treatments and two blocking variables, each containing r classes. Each row and each column in the design square contains all treatments; that is, each class of each blocking variable constitutes a replication (Neter, et al. 1996).

studies can be found in Barr, et al. (1995), Crowder, et al. (1978), Greenberg (1990), and Hoaglin and Andrews (1975). An application of design of experiments to network reoptimization algorithms is presented in Amini and Barr (1993). Amini and Racer (1994) also use experimental design techniques to compare alternative solution methods for the generalized assignment problem. Coy, et al. (2001) apply design of experiments and gradient descent to find effective parameter values for two vehicle routing heuristics. Additional fine-tuning references are van Breedam (1995) in a vehicle routing application and Xu, Chiu and Glover (1998) in a tabu search procedure for a telecommunications network design problem.

2.2 Global Optimization

Another systematic way of searching for optimal parameter values is to employ global optimization techniques. This form of fine-tuning can be achieved only with global optimizers that can treat the objective function evaluation as a black box. Consider, for example, the GENOCOP system, developed by Michalewicz (1994) and his group at the University of North Carolina at Charlotte (<http://www.coe.uncc.edu/~zbyszek/gchome.html>). GENOCOP is a global optimizer based on genetic algorithms that treats the objective function evaluation as a black box. Therefore, it is conceivable to apply GENOCOP to the problem of fine-tuning parameters.

This possibility may be at first glance very appealing. However, one should keep in mind that procedures such as GENOCOP were designed under the assumption that the evaluation of the objective function is not computationally time consuming. Therefore, global optimizers such as GENOCOP tend to give good results after a large number of solution trials have been generated. Since each solution trial requires a call to the evaluation function, the number of experiments (i.e., runs of the algorithm being fine-tuned) is also large. The advantage of this approach, on the other hand, is that the search does not have to be limited to two or three critical parameter values. Global optimizers such as GENOCOP search for optimal values of integer or continuous variables that are bounded properly. Therefore, the analyst needs only to specify a range for each parameter of the algorithm as well as the appropriate domain. For example, one of the most important parameters in a tabu search implementation is the so-called tabu tenure. This parameter refers to the number of iterations a particular attribute of a solution (e.g., the position of a job in a sequencing problem) is classified as “forbidden” (or “tabu” in the methodology’s jargon). Typically, this parameter would be defined as an integer variable in a given range that is believed to contain an effective tabu tenure value for a class of problems.

Some commercial global optimizers also can be adapted for the problem of fine-tuning algorithms. For example, the global optimizer known as OptQuest (www.opttek.com) is embedded in simulation software such as Crystal Ball and Arena and is used to search for optimal values of input parameters of simulation models. By replacing a simulation model with an algorithm and a set of training problems, the problem of optimizing a simulation model becomes equivalent to the problem of fine-tuning parameters of an algorithm. Evolver (www.palisade.com) is another commercial global optimizer that might be suitable for the fine-tuning problem.

3. DESCRIPTION OF CALIBRA

The fine-tuning support tool that we propose is based on combining two technologies: experimental designs and local search. The experimental designs provide a way of focusing the local search on promising regions of the search space. That is, instead of initializing the

search from an arbitrary point (a set of values for the parameters), a design of experiments is used to find a starting point for the search and to assess the effect of changing the parameter values. Design of experiments is also used beyond the initial stage to keep the search focused around the parameter values suggested by the analysis of the experimental results. In the Taguchi methodology, the analysis of the results provides the “optimal condition”, that is, the optimal values for the factors. However, this optimal condition is found by the combination of calculating the average performance of each factor at each specified level and then finding the “main effect” or “factorial effect”. The key assumption is that the relationship is linear and therefore the main effect of a bi-level design is calculated as the difference between the two average effects of the factor at the two levels. The so-called optimal condition need not be one of the experiments in the fractional design. When this happens, a confirmation run is performed to measure the actual response and compare it with the estimated response from the analysis. Because of the linear assumption, this predicted “optimal condition” could be very different than the true optimum. Since in most fine-tuning applications the linearity assumption does not hold, CALIBRA uses the results of the analysis only as a guideline to narrow the search and initiate the next round of experiments. As the search focuses on narrower ranges for each parameter, the linear assumption becomes less restrictive and the predicted “optimal condition” approaches the true optimal condition for the current experiment. The proposed approach contrasts with that taken by the global optimizers mentioned above, which can intensify the search only after sampling the feasible ranges for the parameter values and therefore employing considerably more computational effort. Also, the intensification does not include narrowing the ranges for the parameter values because global optimizers are typically designed to create a balance between diversifying the search (i.e., exploration of the solution space) and intensifying the search (i.e., exploitation of the solution space).

Given any algorithm (ALGO) with several parameters to be fine-tuned, CALIBRA sets up a series of experiments based on the Taguchi methodology. These experiments have the goal of finding the “best” value for each search parameter in ALGO. The notion of best depends on how the performance of ALGO is measured. For instance, if performance is measured by the quality of the solutions to a set of problems, then the best parameter values are those that yield the solutions with the highest quality. CALIBRA controls the calls to ALGO and passes the appropriate values of the parameters corresponding to each experiment. CALIBRA then reads a text file generated by ALGO, where the response value for the experiment has been recorded. CALIBRA uses the response to guide the search for the best values for the parameters. The current version of CALIBRA is limited to a maximum of 5 parameters. If the algorithm being fine-tuned has more than 5 parameters, we recommend the use of Taguchi’s $L_{16}(2^{15})$ array to determine the 5 most significant parameters and fix the others to appropriate values. The $L_{16}(2^{15})$ can handle up to 15 parameters, which we anticipate would be an upper bound on the number of parameters that would be reasonable to use in an algorithm. In order to perform such preliminary experimentation, all the parameters of the algorithm must be assigned to appropriate columns in the $L_{16}(2^{15})$ array. If the algorithm has less than 15 parameters, the additional columns can be used to study some interaction effects between parameters. The specific design depends on each situation and should follow the guidelines of the Taguchi methodology, outlined and applied in Taguchi (1987), Roy (1990), Peace (1993), Taguchi and Yokoyama (1994) and Song, Mathur and Pattipati (1995).

Since CALIBRA uses a full factorial 2^k design, Taguchi’s $L_9(3^4)$ design and a local search procedure, a suitable termination criterion must be defined to stop its execution. It is customary in design of experiments to choose a design based on the available budget to perform experiments. That is, an analyst first determines the number of experiments that can

be afforded with the current budget and then decides which design will be most appropriate. CALIBRA emulates this process by asking the analyst to enter the maximum number of experiments (i.e., calls to ALGO) that can be performed to fine-tune the algorithm. Before describing the procedure using a pseudo-code, we introduce the following notation:

MAXEX	:	User's specified limit on the number of experiments.
k	:	Number of parameters.
u_i	:	Upper limit on the value of parameter i (for $i = 1, \dots, k$).
l_i	:	Lower limit on the value of parameter i (for $i = 1, \dots, k$).
ROUND(x)	:	A function that rounds the value of x . If the fractional part of x is less than 0.5 then the function returns the integer part of x , otherwise it returns ROUND($x+0.5$).
$p_i^*(iter)$:	Value of the best level for parameter i at iteration $iter$.
$p_i^-(iter)$:	Value of the worst level for parameter i at iteration $iter$.
$p_i^n(iter)$:	Value of level n for parameter i at iteration $iter$. Since 3-level experiments are performed during the local search, $n = 1, 2$ or 3 .
P_i^s	:	Value for parameter i in the seed solution s .
$iter$:	Iteration counter. This counts the number of $L_9(3^4)$ experimental designs performed during the local search.
<i>NumSol</i>	:	Number of solutions (sets of parameter values) found.
<i>NumCall</i>	:	Number of experiments (i.e., number of calls to ALGO).
$s1$ and $s2$:	Seed solutions to initiate a new local search when <i>NumSol</i> > 3 in Phase 2

The procedure consists of four phases (Figure 1). The first phase starts with a 2^k full factorial design using the first and third quartile within the range of each parameter. CALIBRA works with integer arithmetic; therefore parameters with continuous values must be discretized by specifying a level of accuracy (i.e., two or three decimal places). For instance, a continuous parameter in the range 2.35 to 4.21 with an accuracy of two decimal places is internally handled as an integer variable with a lower limit of 235 and an upper limit of 421. This approach does not represent a limitation for CALIBRA, since its dichotomous search is capable of quickly converging towards the best values in the range, even when the desired accuracy for continuous parameters is high.

— FIGURE 1 —

The second phase determines an initial solution for the local search. The initial solution is constructed to exploit the most promising regions first (as given by the response of previous experiments) and then diversifying the search to other regions. If the number of parameters is 5, one of the parameters is fixed to a chosen value in this phase. This is necessary because the local search uses the $L_9(3^4)$ design, which can handle a maximum of 4 parameters.

The third phase is the local search. The goal of the search is to find a local optimal solution (i.e., a set of parameter values) by incrementally reducing the range of each parameter value. As the range is reduced, experiments are performed using the bounds and the midpoint of the updated range. This is done until no significant change is detected or the range converges to a single point.

The fourth phase starts by checking if more experiments can be performed within the allotted budget (i.e., MAXEX). If a new local search is possible, then this phase prepares the execution of the new search and passes control back to the second phase. We now provide details of each phase.

The procedure searches for the best values of all parameters simultaneously. That is, each step of the procedure consists of a concurrent search for the best values for all parameters. This contrasts with the approach, typically used in manual fine-tuning, of searching for the best values by modifying the value of one parameter at a time.

Phase 1 (Initial Experiment)

- Perform 2^k experiments corresponding to k parameters with 2 levels defined by the first and third quartile of their range. That is, the low level for parameter i is $\text{ROUND}(l_i + (u_i - l_i)/4)$ and the high level of parameter i is $\text{ROUND}(u_i - (u_i - l_i)/4)$.
- Considering the best and worst results out of the 2^k experiments, determine the best initial level $p_i^*(0)$ and the worst initial level $p_i^-(0)$ for each parameter i . Also store the corresponding responses for each level as well as the significance value. The significance of each parameter is the critical value at which the corresponding F statistic is significant. The significance value is readily available from the analysis of variance related to the 2^k full factorial design.
- Make $\text{NumSol} = 2$ and $\text{NumCall} = 2^k$.

Phase 2 (Local Search Initialization)

- Make $\text{iter} = 0$.
- If $\text{NumSol} = 2$, then, for each parameter i , $p_i^1(\text{iter}) = p_i^*(0) - \varepsilon$, $p_i^2(\text{iter}) = p_i^*(0)$ and $p_i^3(\text{iter}) = p_i^*(0) + \varepsilon$, where $\varepsilon = \frac{\min\{(p_i^*(0) - l_i), (u_i - p_i^*(0))\}}{2}$. If $k > 4$ then the parameter with the largest significance value (as determined by the analysis of variance calculations in phase 1) is set to its best current value.
- If $\text{NumSol} = 3$, then, for each parameter i , $p_i^1(\text{iter}) = p_i^-(0) - \varepsilon$, $p_i^2(\text{iter}) = p_i^-(0)$ and $p_i^3(\text{iter}) = p_i^-(0) + \varepsilon$, where $\varepsilon = \frac{\min\{(p_i^-(0) - l_i), (u_i - p_i^-(0))\}}{2}$. If $k > 4$ then the parameter with the largest significance value (as determined by the analysis of variance calculations in phase 1) is set to its best current value.
- If $\text{NumSol} > 3$ then select two previously generated solutions to be the “seeds” for the new local search. When NumSol is an even number, the selection rule pairs the two best solutions that have not been previously chosen together as seeds. (A list of all previous pairings is kept for this purpose.) When NumSol is an odd number, the best solution is paired with its most diverse corresponding solution, as long as the two have not been paired before. Diversity is measured according to the individual parameter values. Let the two seed solutions be $s1$ and $s2$. Therefore, P_i^{s1} denotes the value of parameter i in the seed solution $s1$ and P_i^{s2} is the value of parameter i in the seed solution $s2$.

- For each parameter i , make $p_i^1(iter) = \min\{P_i^{s1}, P_i^{s2}\}$,
 $p_i^3(iter) = \max\{P_i^{s1}, P_i^{s2}\}$ and $p_i^2(iter) = \frac{p_i^1(iter) + p_i^3(iter)}{2}$. If
 $k > 4$ then the parameter i with the smallest difference between
 $p_i^1(iter)$ and $p_i^3(iter)$ is considered fixed, and therefore all three
levels receive the same value (i.e., $p_i^1(iter) = p_i^2(iter) = p_i^3(iter) =$
 $(p_i^1(iter) + p_i^3(iter))/2$).

Phase 3 (Local Search)

- Make $iter = iter + 1$.
- Perform experiments according to a Taguchi $L_9(3^4)$ design, where the parameter levels are given by $p_i^1(iter-1)$, $p_i^2(iter-1)$ and $p_i^3(iter-1)$, for each parameter i . Let $p_i^*(iter)$ be the best value for parameter i after the analysis. Make then $p_i^1(iter) = p_i^*(iter) - \varepsilon$, $p_i^2(iter) = p_i^*(iter)$ and $p_i^3(iter) = p_i^*(iter) + \varepsilon$, where $\varepsilon = |p_i^*(iter) - p_i^2(iter-1)|$. If $p_i^*(iter)$ is equal to $p_i^2(iter-1)$ or the new value has caused a change in the search direction, then $\varepsilon = \min\{|p_i^*(iter) - p_i^1(iter-1)|, |p_i^3(iter-1) - p_i^*(iter)|\}/2$, with $p_i^1(iter)$ and $p_i^3(iter)$ respectively bounded by l_i and u_i . (This bounding allows the procedure to eventually reach either l_i or u_i if the search continues to be promising in one particular direction.)
- Make $NumCall = NumCall + 9$.
- Determine whether the current solution is a local optimum. If $p_i^1(iter) = p_i^2(iter) = p_i^3(iter)$ or $p_i^1(iter) = p_i^1(iter-1)$ and $p_i^2(iter) = p_i^2(iter-1)$ and $p_i^3(iter) = p_i^3(iter-1)$ for each parameter i , then the solution is a local optimum.
- If the solution is a local optimum, go to Phase 4, otherwise continue the local search (i.e., go to the beginning of Phase 3).

Phase 4 (Update Solution List)

- If $p_i^*(iter)$ is not in the list of solutions, add the solution to the list. Otherwise, add the complement of the solution to the list (in order to diversify the search). The complement for parameter p_i is found with respect to the original limits l_i and u_i as the symmetrical value $p_i = l_i + (u_i - p_i)$.
- Make $NumSol = NumSol + 1$.
- If $NumCall \geq MAXEX$, then stop the search and report the best parameter values found. Otherwise, go to Phase 2.

The operation of the procedure has an appealing graphical representation. A horizontal line represents each experimental factor (i.e., a parameter being fine-tuned in our context). Two dotted vertical lines are used to represent the upper and lower limits associated with the parameter. Tick marks are used to represent the level values at a given iteration. A cross represents the best value found so far. Finally, an ellipse is used to show the search area during a given iteration of the local search procedure. Figure 2 shows the state of the procedure after Phase 1 for an algorithm with two parameters. The first parameter ranges from 20 to 30 and the second one from 0 to 1.6.

— FIGURE 2 —

Phase 1 performs a 2-level experiment using the first and third quartile as the low and high levels for each parameter. After the experiment is performed, Figure 2 indicates that $p_1^*(0) = 23$ and $p_2^*(0) = 1.2$. These values are used to initiate Phase 2, where three levels are identified to initialize Phase 3.

Figure 3 shows the state of the search after Phase 2 is executed immediately after Phase 1. Since $NumSol = 2$, the three initial levels are found by calculating ε as half of the distance between the best level and the closest limit. This space reduction intensifies the search in a region that is deemed promising by virtue of the best levels identified after the experiment in Phase 1. Note that subsequent executions of Phase 2 result in values for the three levels that are always within the initial values. In other words, additional levels for parameter 1 will fall within 21.5 and 24.5, in the same way that additional levels for parameter two will fall within 1.0 and 1.4.

— FIGURE 3 —

When $NumSol > 3$ then the seed solution is constructed using the last two local optimal solutions that have not been yet paired for this purpose. A graphical representation of this situation is depicted in Figure 4.

— FIGURE 4 —

The determination of the new solution in iteration $iter$ of Phase 3 depends on the best values obtained in the previous iteration (i.e., $iter-1$). Figure 5 (a) shows the process when the best value in $iter-1$ occurs at the center of the range (i.e., $p_i^*(iter) = p_i^2(iter-1)$). Figure 5(b) shows the case for which $p_i^*(iter) = p_i^1(iter-1)$, that is, when the best value is the lower level in the previous iteration.

— FIGURE 5 —

The evolution of the local search can be represented as a set of embedded ellipses, as shown in Figure 6. The local search stops when the current ellipse is “small enough,” where the notion of size is implicit in the stopping rules.

— FIGURE 6 —

Figure 7 shows an example of the search trajectory followed by CALIBRA when tuning the RMAX parameter in Boctor (1991), which is one of our case studies in the following section. This figure depicts the signal-to-noise ratio (S/N) corresponding to each local solution. The

S/N ratios are log functions of desired output and are used as the response variable for an analysis of variance in Taguchi methods (Roy, 1990). The aim is to find the combination of the input factors that make the S/N ratio as large as possible, because this means that the signal is as large as possible when compared to the noise. In general there are three basic quality characteristics: smaller-the-better, larger-the-better and nominal-the-best. We use the smaller-the-better because we are trying to minimize the ratio between the objective function values of the heuristic solutions (f_h) and the optimal (or best known) objective function values (f^*) for the problems in the training set. Specifically, our S/N ratio has the following form:

$$\text{S/N ratio} = -10 \text{Log}_{10} [\text{mean of sum of squares of } (f_h / f^*)]$$

The largest value for the S/N ratio defined above is zero and is achieved when $f_h = f^*$ for all the problems in the training set. Figure 7 shows that after 8 local optima the final value of 139 for RMAX is obtained.

— FIGURE 7 —

In order to test the performance of our procedure, we now present several applications of CALIBRA to procedures that were developed and published prior to this study and were fine-tuned using a variety of different methods.

4. CASE STUDIES

In this section, we present the results of applying CALIBRA in six different problem settings:

- Rectilinear Steiner problem (Adenso-Díaz and Laguna, 2001)
- A TS (tabu search) approach for the part-machine grouping problem (Lozano et al., 1999)
- A simulated annealing approach for the part-machine grouping problem (Boctor, 1991)
- Single machine scheduling (Laguna, Barnes and Glover 1991)
- Scheduling tardiness in proportionate flowshops (Adenso-Díaz, 1996)
- Bandwidth packing (Laguna and Glover, 1993)

Before presenting the results, we describe the experimental framework used.

4.1. Experimental Framework

The experimental framework in this subsection sets a common set of rules to carry out the computational tests employing the six problem settings listed above. We assume that anyone interested in fine-tuning an algorithm with CALIBRA has the following two general concerns:

1. To fine-tune the algorithm in a reasonable amount of computer time
2. To verify that the parameter values found with a fraction of the available test problems perform well in the entire set of problems

These two concerns and the speed of the computer used for testing guide the selection of the number of instances to be used in the training set as well as the number of CALIBRA iterations in the computational experiments reported below. For example, if one is interested in completing the fine-tuning process for a given ALGO in 4 hours on a computer that is capable of executing ALGO in an average of 6 seconds per problem instance, then the possible options range from MAXEX = 1 and a training set with 2400 problem instances to MAXEX = 2400 and

1 problem instance in the training set. Neither one of these extremes would be an effective way of achieving the desired results of obtaining robust values for the parameters of ALGO. On one hand, CALIBRA requires a balance between the number of problem instances used for fine-tuning and the value of MAXEX. And on the other hand, the number of instances used for the fine-tuning process should not exceed half of the total number of problem instances available if one wishes to have a hold-out set of problems available for validating the robustness of the parameter values found.

For our computational experiments we set $\text{MAXEX} = 350$. This value takes into account the considerations discussed above in the context of our experiments, however, it should not be taken as a recommended value for running CALIBRA in every situation. As mentioned, above the value of MAXEX relates to the question of available budget for performing experiments. In our case, we also set a maximum computational time of 8 hours on a Pentium 4 computer running at 2.35 GHz. When we consider these two choices and also take into account the average time to execute each ALGO, the number of problem instances in each case is determined as shown in Table 1. Since the execution time of each ALGO varies, the percentage of instances used as training set relative to the total available also varies from case to case.

— TABLE 1 —

Table 1 shows that we choose the minimum between half of the available problem instances and the number of instances associated with executing MAXEX experiments in 8 hours. This results in a unified criterion for performing our tests based on setting $\text{MAXEX} = 350$ and adjusting the number of problem instances in the training set to yield a total computational time of 8 hours or less. The performance criterion for all cases is solution quality. We compare the results from the CALIBRA parameters and the solutions found with the parameter values originally proposed by the authors of each study. We also compare against a simplistic procedure that chooses the midpoint of the range for each parameter. For one case study, we incorporate the use of solution speed for comparison purposes, to show the flexibility of the CALIBRA regarding the selection of alternative objective functions for fine-tuning purposes.

A description of the statistical tests associated with our experiments and a summary of the results appear in Section 4.8. Therefore, we devote Sections 4.2-4.7 to provide a brief description each problem setting along with the parameters that are used during the fine-tuning process. A comparison of the parameter values found with CALIBRA and those originally used by the developers of each procedure is available as an online supplement (<http://or.pubs.informs.org/Pages/collect.html>)

4.2. The Rectilinear Steiner Problem

The rectilinear Steiner problem consists of connecting all the so-called terminal points on a plane in order to minimize the total cost of the edges used, where the cost of the edge connecting two points is equal to the rectilinear distance between the two points. It is well-known that the total cost is minimized by a minimum spanning tree on an augmented set of points, where the additional points are called the set of Steiner points. Unfortunately, it is not possible to determine a priori what points should belong to the set of Steiner points. However, if a grid is drawn with horizontal and vertical lines crossing all the terminal points then it can be verified that the set of Steiner points must be contained in the set of points consisting of all possible intersections on the grid.

Given that the problem is NP-complete, most of the research has focused on the development of heuristic procedures (Beasley, 1992; de Souza and Ribeiro, 1993). Adenso-Díaz and Laguna (2001) have proposed a heuristic that is based on the exploration of a special neighborhood structure using a pseudo-greedy approach. They also use probabilistic rules, based on move attributes derived from the neighborhood structure, to induce search diversification.

Their procedure uses 5 search parameters that were tuned using a Taguchi-based design of experiments. Computational tests were performed using Beasley's set of problem instances (Beasley, 1992). This set consists of 180 instances with number of terminal points ranging from 10 to 500. The 12 problem instances used for training, as specified in Table 1, were selected randomly from the entire set. Specifically, the training set consisted of 3 instances for each problem size with 20, 30, 40 and 50 terminal points.

4.3. A TS approach for the Part-Machine Grouping Problem

An important element in the design of cellular manufacturing systems consists of grouping machines in cells so each machine group can be dedicated to processing a given family of parts. The goal is to minimize the movement of parts between one cell to another, and therefore minimizing the associated costs. The approach of grouping machines is widely used in industry (Wemmerlöv and Hyer, 1989) and the research focuses on the associated clustering problem. (See for example Singh, 1993.)

Lozano, et al. (1999) developed a procedure for the problem of designing effective manufacturing cells. The procedure, which employs 5 parameters, explores possible machine groupings, and based on each configuration, it creates families of parts by solving a linear network flow problem. Lozano, et al. (1999) used 28 problem instances for testing. As indicated in Table 1, fourteen of these instances were selected as the training set to run CALIBRA.

4.4. Simulated Annealing Approach to the Part-Machine Grouping Problem

Boctor (1991) proposed a simulated annealing procedure for the part-machine grouping problem. Boctor's formulation considers a maximum limit M_{\max} for the number of machines per cell and a number of cells given by C_{\max} . The number of cells is an independent factor that makes it possible to attempt the solution of the problem with several C_{\max} values. The values of 5 parameters define an instance of the simulated annealing procedure. Using a set of 10 base cases each consisting of 16 machines and 30 part types and making $M_{\max} = 6, 7, \dots, 12$, Boctor produced 70 problem instances. The optimal solution to these instances is known if $C_{\max} = 3$ when $M_{\max} < 10$ and $C_{\max} = 2$ when $M_{\max} \geq 10$.

Since simulated annealing is highly stochastic in nature, we based our fine-tuning effort on running Boctor's procedure 10 times for each instance in the training set. CALIBRA used the average value obtained for the 10 runs to guide the search.

4.5. Single Machine Scheduling

Laguna, Barnes and Glover (1991) developed a tabu search procedure for the approximate solution of a single machine scheduling problem. The problem consists of minimizing the sum of the setup costs and linear delay penalties when a set of jobs, arriving at time zero, are to be scheduled for sequential processing on a continuously available machine. The method uses the common approach of making a succession of pairwise job exchanges, or swaps, to move

from one trial solution to another. In addition to swap moves, the procedure employs insert moves to expand the definition of the local neighborhood of each trial solution. These moves consist of transferring a single job from one position to another in the sequence.

The procedure employs 5 search parameters. The test set consists of 25 problem instances, with 5 instances for each size of 35, 40, 60, 80 and 100 jobs. CALIBRA was run using 13 randomly selected instances, as indicated in Table 1.

4.6. Minimizing Tardiness in Proportionate Flowshops

The proportionate flowshop refers to a special case of a flowshop problem where the relative processing time in each machine is proportionally the same for all jobs. That is, if a job has a relatively large processing time on a machine compared to the processing time on other machines, then all other jobs also have a relatively large processing time on that machine compared to all other machines. Several heuristic procedures have been developed for the problem of minimizing tardiness in a proportionate flowshop. Adenso-Díaz (1996) developed one of these procedures with a combination of a simulated annealing pre-processor and a tabu search.

The search stops after $nbmax$ iterations without observing an improvement in the objective value of the best solution found. The procedure considers two types of neighborhoods, one based on a swap move of two jobs, and another based on an insertion move of one job. The swap moves are used for most of the search, while the insert moves are only triggered after the search has stalled. The procedure is considered to have stalled when $p*nbmax$ iterations elapse without improvement of the best solution, where $p \in [0,1]$. A property shown in Adenso-Díaz (1992) is used to reduce the size of the neighborhoods by limiting the distance (measured in number of positions) between jobs in a swap or between the moving job and the target position in an insert. The distance is limited between iterations ret and $estab$. The tabu search component of the procedure consisted of a short-term memory with a single static list. The size of the list was set to 7 in all experiments. The set of test problems in this work consists of 720 instances with number of jobs ranging from 10 to 50 and number of machines ranging from 5 to 20.

In order to test CALIBRA on this problem, we selected 40 instances (four for each instance size with 10, 15, 20, 25, ..., 50 jobs, plus four additional instances of sizes 10, 25, 35 and 50). This was done with the purpose of ensuring proper representation of problem instances from all sizes. All selections within each group size were made at random.

4.7. Bandwidth Packing

The bandwidth-packing problem is a combinatorially difficult problem arising in the area of telecommunications. The problem consists of assigning calls to paths in a capacitated graph, such that capacities are not violated and the total profit is maximized. Laguna and Glover (1993) developed a tabu search method for this problem. The method makes use of an implementation of the k -shortest path algorithm, which allows the identification of a controlled set of feasible paths for each call. A tabu search is then performed to find the best path assignment for each call.

The method incorporates a number of features that proved useful for obtaining optimal and near optimal solutions to difficult instances of this problem. These features resulted in 4 search parameters that required fine-tuning. Laguna and Glover (1993) used two types of

problems for testing: (1) instances defined on networks without link costs and (2) instances defined on networks with link costs. We employed a set of 25 instances without link costs for our experiments: 10 originally used by Laguna and Glover, and 15 more from the literature (Anderson, et al. 1993). We chose 13 instances at random to run CALIBRA.

4.8. Summary of Results and Statistical Analysis

Using the parameter values found by CALIBRA, we ran the 6 algorithms on all available problems instances. We also ran the algorithms using a set of parameter values consisting of the midpoint value in the acceptable ranges defined for each parameter (these ranges are also available in the online supplement to this article). It is important to point out that the midpoint is not an arbitrary value since the ranges were configured with the knowledge of the parameter values originally suggested in each study. Nevertheless, we use the midpoints as a base case to compare the results of the search conducted with CALIBRA.

Table 2 shows the improvement of CALIBRA parameter values over the results found with the original parameter values (ORIG) and the midpoint values (MID). Although always with a positive improvement, before testing for statistical significance, this table seems to indicate that only in one case CALIBRA makes a difference regarding the quality of the solutions obtained in the test problems. We refer to the 32.34% improvement of CALIBRA over MID in the Boctor (1991) case.

— TABLE 2 —

We use the results from the same experiment to take a different approach with respect to measuring performance of the three sets of parameter values under consideration (i.e., CALIBRA, ORIG, and MID). Since we have the benefit of knowing the optimal solutions to most of the problem instances and a set of “best known” solutions for all other instances for which optimality has not been confirmed, we use this information to test for statistically significant differences in terms of the number of times each set of parameter values is capable of finding the optimal or best-known solutions.

Table 3 presents the results of this analysis. The table first reports the number of optimal (or best-known) solutions found by each set of parameter values, out of the total number of instances shown in the second column. The last two columns in the table report the 95% confidence intervals for p_1-p_2 for the cases CALIBRA-ORIG and CALIBRA-MID. The test is such that if the interval contains the value of zero, then it cannot be concluded that there is a significant difference between the compared sets of parameter values. The table shows that the statistical tests detect significant differences (marked with an asterisk) between CALIBRA and ORIG only in the Adenso-Díaz (1996) case. Significant differences between CALIBRA and MID are detected in three different cases, as indicated in Table 3.

— TABLE 3 —

An additional way of assessing performance based on solution quality is by counting the number of instances each set of parameter values finds solutions of better, equal or worse quality than the solutions found by another set of parameter values. We use this criterion to compare one more time CALIBRA and ORIG as well as CALIBRA and MID. Table 4 shows the results of such comparison. The BETTER columns show the number of instances in which the CALIBRA parameter values found better solutions than ORIG or MID. The WORSE columns show the number of instances in which the CALIBRA parameter values found worse solutions

than ORIG or MID. The EQUAL columns show the number of instances in which the CALIBRA parameter values found the same solutions than ORIG or MID.

— TABLE 4 —

The 2-tailed p-values in Table 4 correspond to McNemar test for 2×2 contingency tables. We test the following null hypothesis, which indicates that both approaches have the same probability of defeating the other:

$$H_0: \text{Prob}[\text{BETTER}/(\text{BETTER}+\text{WORSE})] = \text{Prob}[\text{WORSE}/(\text{BETTER}+\text{WORSE})] = 0.5$$

The test at a significance level of 5% results in 5 cases for which it detects a significant difference between the number of better solutions found by the CALIBRA parameter values and the values given by ORIG and MID. The outcome of these experiments shows that, within the specified ranges for each parameter value, CALIBRA is capable of finding parameter values that result in performance that is at least as good as the one associated with the values used by the authors of the tested procedures. Furthermore, in 5 cases the CALIBRA parameter values are able to statistically improve upon the results found with the other parameter values. Throughout this experimentation, we have determined that most of the procedures that we selected for testing are quite robust in the acceptable range of values for their parameters. This conclusion is supported by an additional set of statistical experiments where we did not find significant difference in the quality of results between ORIG and MID. However, we address the issue of robustness in more detail in section 4.10.

4.9 Considering Solution Time within a Measure of Performance

In some situations, as in the case of the procedure by Adenso-Díaz (1996) for minimizing tardiness in a proportionate flowshop, the parameter values affect not only the quality of the solutions but also the time required to reach solutions of a given quality. To study the usefulness of CALIBRA under such circumstances, we performed an experiment where the measure of performance accounts for both solution quality and speed. The experiment consists of running CALIBRA using $\frac{f_h}{f_0} + \text{time} * 10^{-4}$ as a measure of performance to guide the search. In this equation, f_h is the objective function value of the solution found with the procedure developed by Adenso-Díaz (1996) using the parameter values under consideration, f_0 is a lower bound on the optimal solution and *time* is the time (in seconds) required to find the heuristic solution with objective function value f_h .

After running CALIBRA on the training set of problems and finding a new set of parameter values ($ret=14$, $estab=0.5$, $nbmax=31$, $p=0.81$, $maxlist=14$), the new parameter values were applied to the entire set of 720 problem instances (80 for each size of 10, 15, ..., 50 jobs). The new parameter values did not match all the solutions found with the original values, however, the solutions were found employing less computational time. The computational effort (measured in CPU seconds) was reduced with a modest sacrifice in solution quality. In particular, solution quality deteriorated on an average of 0.005% while solution time was reduced by 69.87%. Figure 8 shows, for each problem instance, the ratio of the CPU time required with the parameter values found in section 4.6 vs. the time required with the new parameter values. We have omitted the set with $n = 10$ jobs because the procedure processes these problems in less than 0.11 seconds. This is why there are only 640 instead of 720 data points in figure 8.

— FIGURE 8 —

In situations where not only the quality of the solution but also the time to reach solutions of a certain quality are important, it might be interesting to fine-tune a given ALGO using a measure of performance similar to the one we have illustrated in this section. However, this should be done with care because a reduction in solution time may result in a deterioration of solution quality that goes beyond an acceptable level. In our experiment, the tradeoff seems to favor using the new parameter values; however, the decision depends on the importance that the analyst places on finding solutions of the highest quality possible.

4.10 Analyzing the Sensitivity and Robustness of Parameter Values

One of the key questions regarding the fine-tuning of parameters is the sensitivity of ALGO to changes in the parameter values. One possible way of addressing this issue is to try all possible combinations of the parameter values and analyze the resulting data for a large set of problems. As a proxy to this computationally overwhelming task, we have recorded the output of each call to ALGO in the experiments of sections 4.2 and 4.4. For each case, we collected a data set consisting of approximately 350 records (one for each call to ALGO). Each record consists of the quality measure associated with each instance in the training set. For the purpose of this experiment, the measure of quality is an average relative deviation of the solutions found with the parameter values under consideration and the best known solutions to the problems in the training set. The average is calculated over the problems in the training set (32 for Boctor (1991) and 12 for Adenso-Díaz and Laguna (2001)). The histograms in Figures 9 and 10 correspond to the data values associated with each CALIBRA run.

— FIGURES 9 and 10 —

Figures 9 and 10 reveal the difference between the sensitivity of the parameters in Boctor's (1991) simulated annealing method and the one associated with Adenso-Díaz and Laguna's (2001) procedure. The histogram in Figure 9 shows that almost 30% (102/351) of the parameter values tried during the CALIBRA search resulted in an average relative deviation of more than 100%. In contrast, all the parameter values tried on the Adenso-Díaz and Laguna's procedure resulted in average relative deviations smaller than 0.73%. Although there is some sensitivity of the parameter values within this range, the performance difference is very small when compared to the differences detected in Boctor's training set. The results of this experiment show that the use of CALIBRA may be more valuable in the case of Boctor's procedure, since the procedure by Adenso-Díaz and Laguna seems to operate a very high level regardless of the specific values that are used within the specified range.

Robustness of the parameter values is another important issue related to the fine-tuning of algorithms. To measure robustness, we performed one more experiment that consists of utilizing CALIBRA to search for the best parameter values for each instance in the training sets for the procedures of Boctor (1991) and Adenso-Díaz and Laguna (2001). Table 5 summarizes the output of this experiment.

— TABLE 5 —

This table shows, for each case and parameter, simple statistics related to the best parameter values found for the problems in the corresponding training set. The best common parameter values also are included in this table. The best common parameters are those found by CALIBRA when using the entire training set during the search. In addition, the table includes

the average percent improvement achieved by the best individual parameter values when compared with the best common parameter values found in the experiments of sections 4.2 and 4.4. The percent improvement is an estimate of what the analyst is given up for using a single set of parameter values for a class of problems as opposed to the individual best parameter values for each problem. The results in Table 5 indicate that the best common parameter values found for the Adenso-Díaz and Laguna (2001) case are more robust than those found for the Boctor (1991) case. This conclusion is supported by the percent improvement values and the small variability exhibited by the parameter values in the Adenso-Díaz and Laguna case.

5. CONCLUSIONS

Due to the advent of metaheuristic procedures, researchers and developers of optimization software are increasingly becoming more aware of the importance of fine-tuning algorithms (see e.g., Coy, et al. 2001). In this paper, we have presented a support tool for fine-tuning algorithms that is based on experimental design and local search. The goal of this research has been to facilitate the task of finding parameter values for algorithms whose performance depends on using combinations of such values that may not be immediately obvious. We first undertook to describe the procedure and illustrate its underlying ideas graphically. We then used the procedure to illustrate the fine-tuning of six unrelated algorithms from the literature. The results of our experiments were subjected to a statistical analysis in order to detect significant difference in performance. These tests were able to identify some cases in which the parameter values found with CALIBRA performed better than those originally suggested by the authors. In all other cases, the differences were not significant, so the parameter values found with the execution of CALIBRA are not expected to perform neither better nor worse than those originally suggested by the developers of each method.

The current version of CALIBRA is limited in two ways: the number of parameters that is capable of handling and the lack of interaction effect analysis. Procedures that have more than 5 parameters cannot be fine-tuned with the current version of CALIBRA, unless some of the parameters are fixed by the user (as explained at the beginning of section 3). The local search is guided by the analysis of main effects in the experimental design phase. This means that the search is not able to exploit interaction effects and report back to the analyst the significance of interactions among parameters. Hence, CALIBRA should be more effective in situations when the interactions among parameters are negligible.

Supported by our experiences using CALIBRA and by the experiments discussed here, our conclusion is that CALIBRA can be a useful support tool for the process of fine-tuning algorithms. One possible application may relate to narrowing down choices for parameter values, in a situation where the analyst is facing a wide range of possible values. Another application may consist of searching in a somewhat narrow range around parameter values that the analyst has already tested. We anticipate that the benefits of using CALIBRA will be more evident in situations where the algorithm being fine-tuned has parameters whose values have significant impact in performance when changed within the acceptable range.

As a final comment, we point out that CALIBRA can also be used to search for the best possible solution to a given problem instance. This situation arises when one is interested in finding the best solution to a problem for which the only available procedure is heuristic in nature. In this particular situation, CALIBRA may be used to run the heuristic procedure using a range of parameter values with the sole purpose of finding a set of parameter values

that provide the best possible answer to the problem of interest. We encourage the reader to visit http://coruxa.epsig.uniovi.es/~adenso/file_d.html for this and other applications of CALIBRA (including tuning of neural networks).

REFERENCES

- Adenso-Díaz, B. (1992) "Restricted neighborhood in the tabu search for the flowshop problem," *European Journal of Operational Research*, vol. 62, pp. 27-37
- Adenso-Díaz, B. (1996) "An SA/TS mixture algorithm for the scheduling tardiness problem," *European Journal of Operational Research*, vol. 88, pp. 516-524
- Adenso-Díaz, B. and M. Laguna (2001) "A Pseudo-Greedy Heuristic for the Rectilinear Steiner Problem," *International Journal of Operation & Quantitative Management*, vol. 7, pp. 105-118
- Amini, M. M. and R. S. Barr (1993) "Network Reoptimization Algorithms: A Statistically Designed Comparison," *ORSA Journal on Computing*, vol. 5, no. 4, pp. 395-409.
- Amini, M. M. and M. Racer (1994) "A Rigorous Computational Comparison of Alternative Solution Methods for the Generalized Assignment Problem," *Management Science*, vol. 40, no. 7, pp. 868-890.
- Anderson A., K. Fraughnaugh, M. Parker, J. Ryan (1993) "Path Assignment for Call Routing: An Application of Tabu Search," *Annals of Operation Research*, Vol. 41, pp. 301-312.
- Barr, R. S., B. L. Golden, J. P. Kelly, M. G. C. Resende and W. R. Stewart (1995) "Designing and Reporting Computational Experiments with Heuristic Methods," *Journal of Heuristics*, vol. 1, no. 1, pp. 9-32.
- Beasley, J. E. (1992) "A Heuristic for Euclidean and Rectilinear Steiner problems," *European Journal of Operational Research*, vol. 58, pp. 284-292.
- Boctor, F. F. (1991) "A linear formulation of the machine-part cell formation problem," *International Journal of Production Research*, vol. 29, No. 2, pp. 342-356.
- Coy, S. P., B. L. Golden, G. C. Runger and E. A. Wasil (2001) "Using Experimental Design to Find Effective Parameter Settings for Heuristics," *Journal of Heuristics*, vol. 7, no. 1, pp. 77-97.
- Crowder, H. P., R. S. Dembo, and J. M. Mulvey (1978) "Reporting Computational Experiments in Mathematical Programming," *Mathematical Programming*, vol. 15, pp. 316-329.
- DeJong, K. A. and W. M. Spears (1990) "An Analysis of the Interacting Roles of Population Size and Crossover in Genetic Algorithms," *Proceedings of the First Workshop on Parallel Problem Solving from Nature*, Springer Verlag, pp. 38-47.
- de Souza, C. C. and C. C. Ribeiro (1993) "Heuristics for the Minimum Rectilinear Steiner Tree Problem: New Algorithm and a Computational Study," *Discrete Applied Mathematics*, vol. 45, pp. 205-220.

- Fink, A. and S. Voss (2002) "HOTFRAME: A Heuristic Optimization Framework," in *Optimization Software Class Libraries*, S. Voss and D. Woodruff (eds.), Kluwer Academic Publishers, Boston, pp. 81-154.
- Greenberg, H. J. (1990) "Computational Testing: Why, How and How Much," *ORSA Journal on Computing*, vol. 2, no. 1, pp. 94-97.
- Grefenstette, J. J. (1986) "Optimization of Control Parameters for genetic Algorithms," *IEEE Transactions on Systems Man and Cybernetics*, vol. 16, no. 1, pp. 122-128.
- Hedayat, A. S., N. J. A. Sloane and John Stufken (1999) *Orthogonal Arrays: Theory and Applications*, Springer-Verlag, New York.
- Hoaglin, D. C. and D. F. Andrews (1975) "The Reporting of Computation-Based Results in Statistics," *The American Statistician*, vol. 29, no. 3, pp. 122-126.
- Laguna, M., J. W. Barnes and F. Glover (1991) "Tabu Search Methods for a Single Machine Scheduling Problem," *Journal of Intelligent Manufacturing*, vol. 2, pp. 63-74.
- Laguna, M. and F. Glover (1993) "Bandwidth Packing: A Tabu Search Approach," *Management Science*, vol. 39, no. 4, pp. 492-500.
- Lozano, S., B. Adenso-Díaz, I. Eguía and L. Onieva (1999) "A one-step tabu search algorithm for manufacturing cell design," *Journal of Operational Research Society*, vol. 50, no. 5, pp. 509-516.
- Michalewicz, Z. (1994) *Genetic Algorithms + Data Structures = Evolution Programs*, Second-Extended Edition, Springer-Verlag.
- Montgomery, D. C. (1997) *Design and Analysis of Experiments*, 4th edition, Wiley, New York.
- Neter, J., M. H. Kutner, C. J. Nachtsheim and W. Wasserman (1996) *Applied Linear Statistical Models*, Irwin, Homewood, Ill.
- Peace, G. S. (1993) *Taguchi Methods: A Hands-On Approach*, Addison-Wesley Publishing Company, Reading MA.
- Roy, R.K. (1990) *A Primer on the Taguchi Method*, Van Nostrand Reinhold, New York.
- Singh, N. (1993) "Design of Cellular Manufacturing Systems: An Invited Review," *European Journal of Operational Research*, vol. 69, pp. 284-291.
- Song, A. A., A. Mathur and K. R. Pattipati (1995) "Design of Process Parameters Using Robust Design Techniques and Multiple Criteria Optimization," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 25, no. 11, pp. 1437-1446.
- Taguchi, G. (1987) *System of Experimental Design: Engineering Methods to Optimize Quality and Minimize Costs*, Vols. 1 & 2, UNIPUB/Kraus International Publications, White Plains NY.

Taguchi, G. and Y. Yokoyama (1994) *Taguchi Methods: Design of Experiments*, American Supplier Institute, Dearborn MI, in conjunction with the Japanese Standards Association, Tokyo, Japan.

van Breedam, A. (1995) "Improvement Heuristics for the Vehicle Routing Problem Based on Simulated Annealing," *European Journal of Operational Research*, vol. 86, pp. 480-490.

van Laarhoven, P. J. M. and E. H. Aarts (1987) *Simulated Annealing: Theory and Applications*, Kluwer Academic Publishers, London.

Wemmerlöv, U. and N. L. Hyer (1989) "Cellular Manufacturing in the U.S. Industry: A Survey of Users," *International Journal of Production Research*, vol. 27, pp. 1511-1530.

Xu, J., S. Chiu and F. Glover (1998) "Fine-tuning a Tabu Search Algorithm with Statistical Tests," *International Transactions in Operational Research*, vol. 5, no. 3, pp. 233-244.

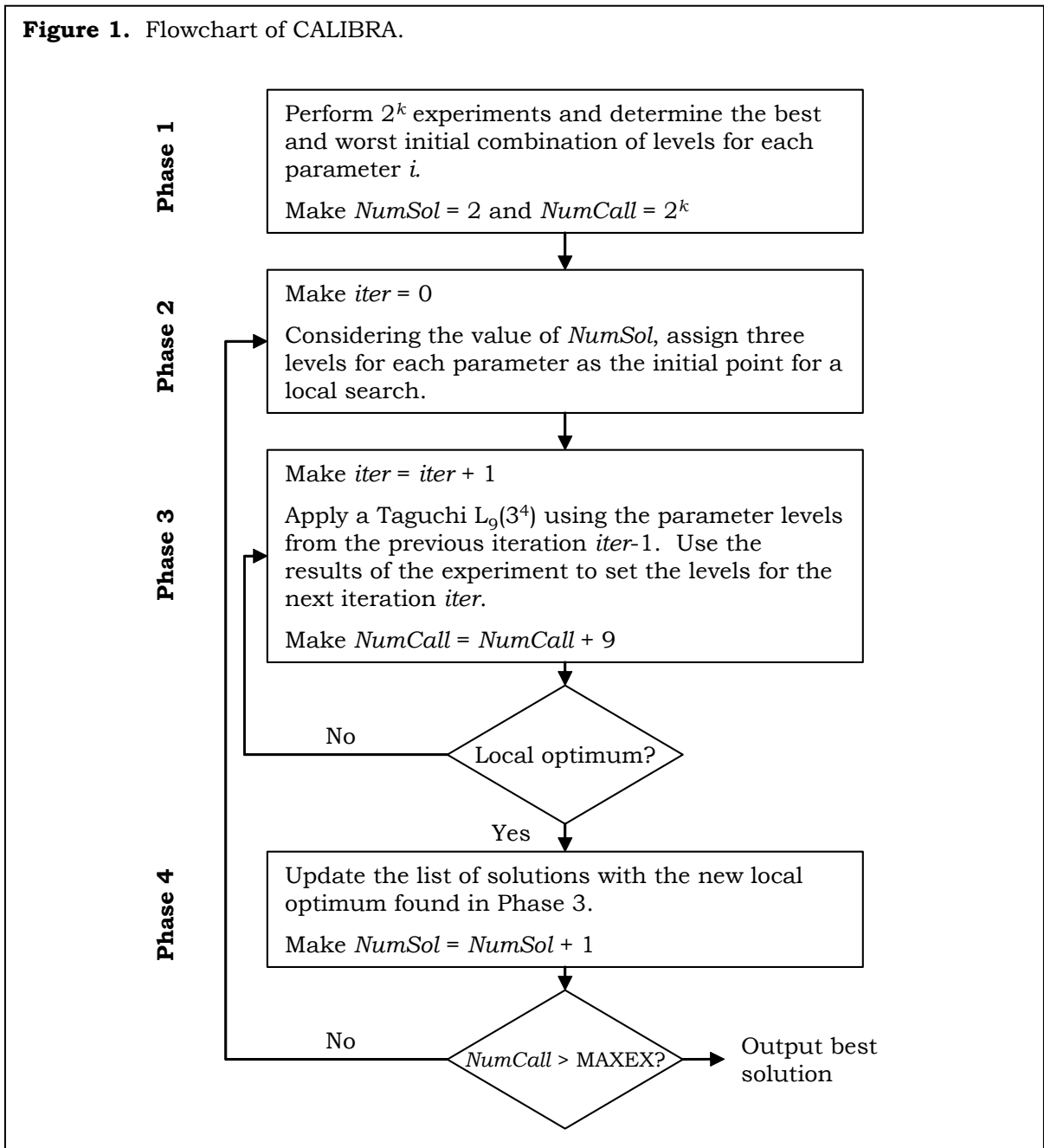


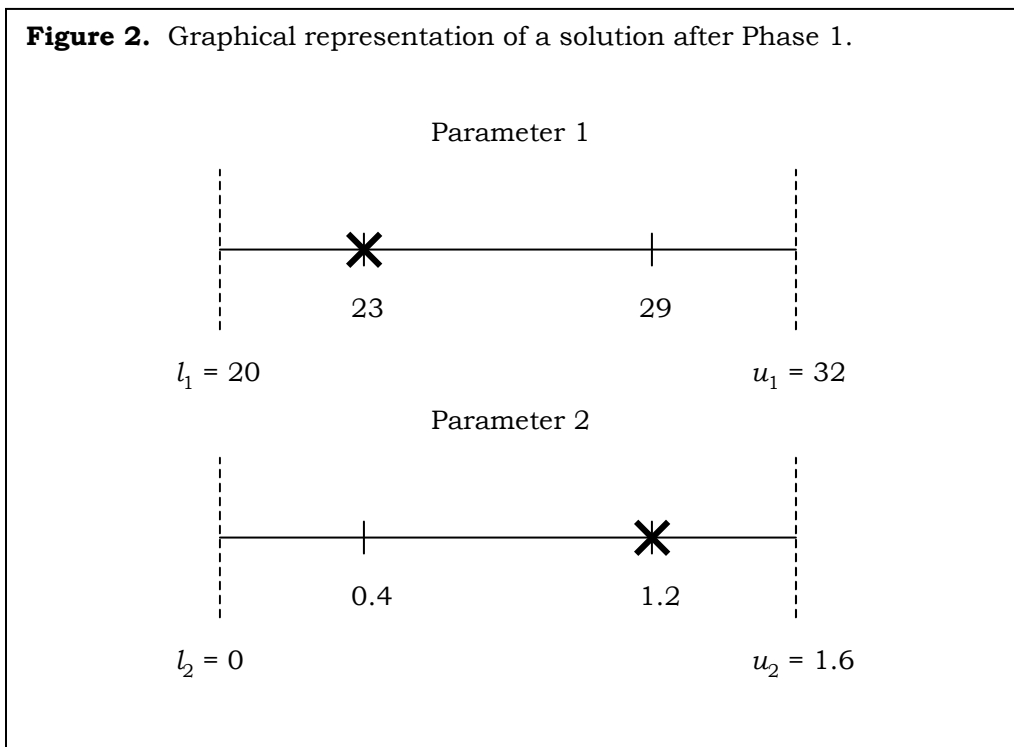
Figure 2. Graphical representation of a solution after Phase 1.

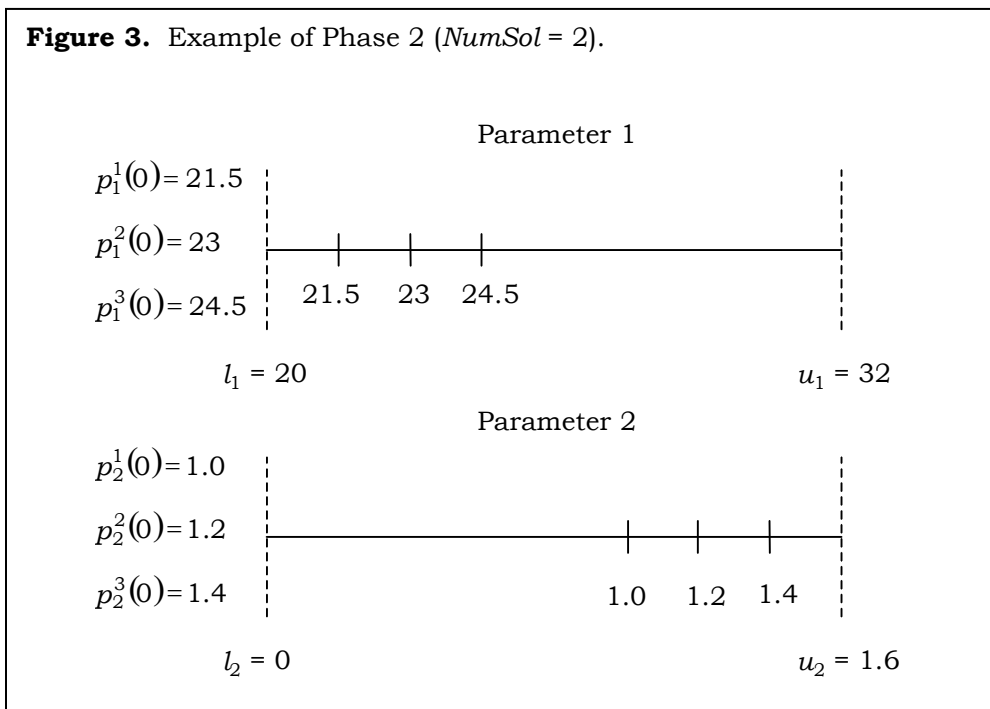
Figure 3. Example of Phase 2 ($NumSol = 2$).

Figure 4. Example of Phase 2 ($NumSol > 3$).

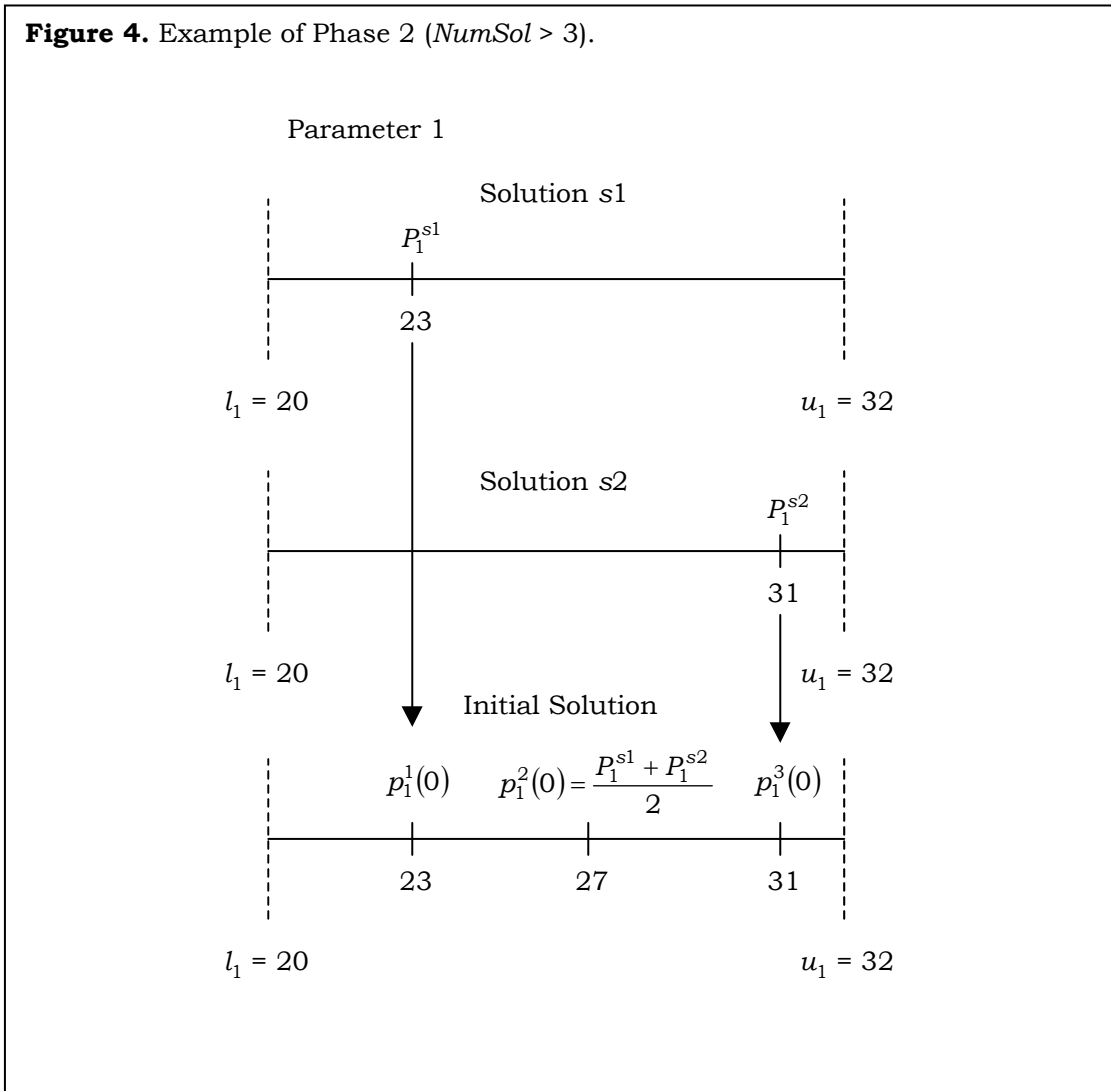


Figure 5. Graphical representation of determining $p_i^n(iter)$ during Phase 3.

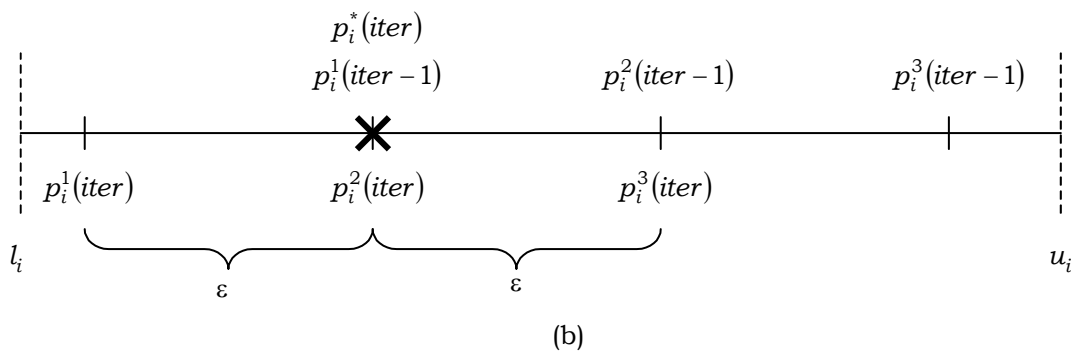
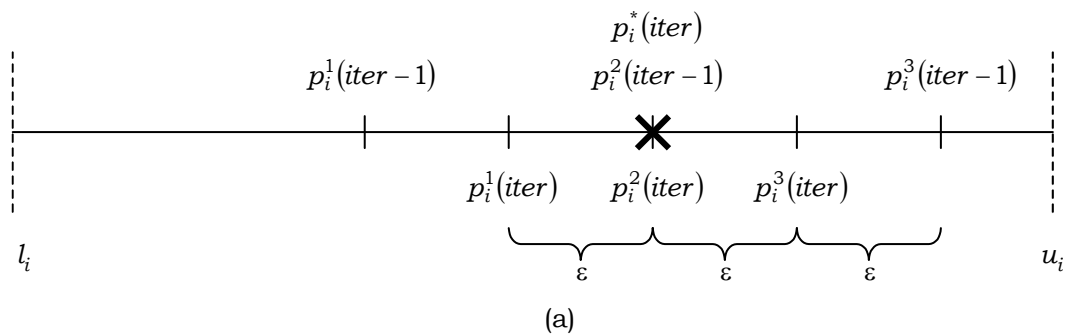


Figure 6. Evolution of Phase 3 until reaching a local optimum P_i^s , for all the five parameters

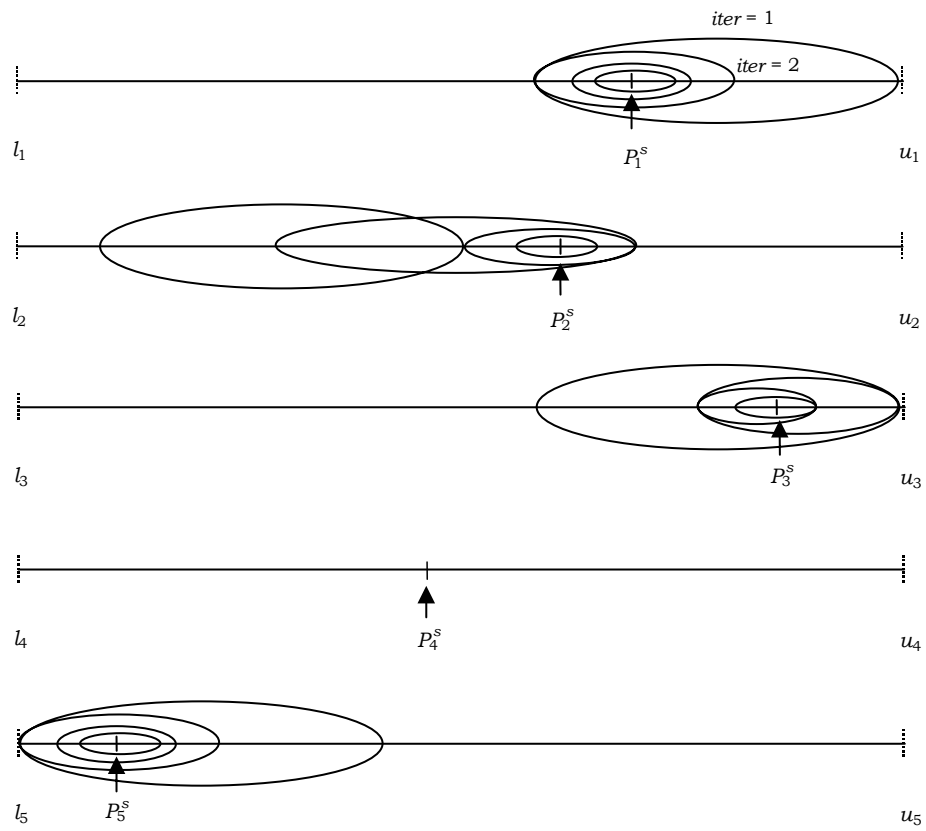


Figure 7. Evolution of the search for parameter #3 (RMAX) in the Boctor (1991) problem. Stars represent local optimal values of this parameter. Final output given by CALIBRA corresponds to local optimum #8.

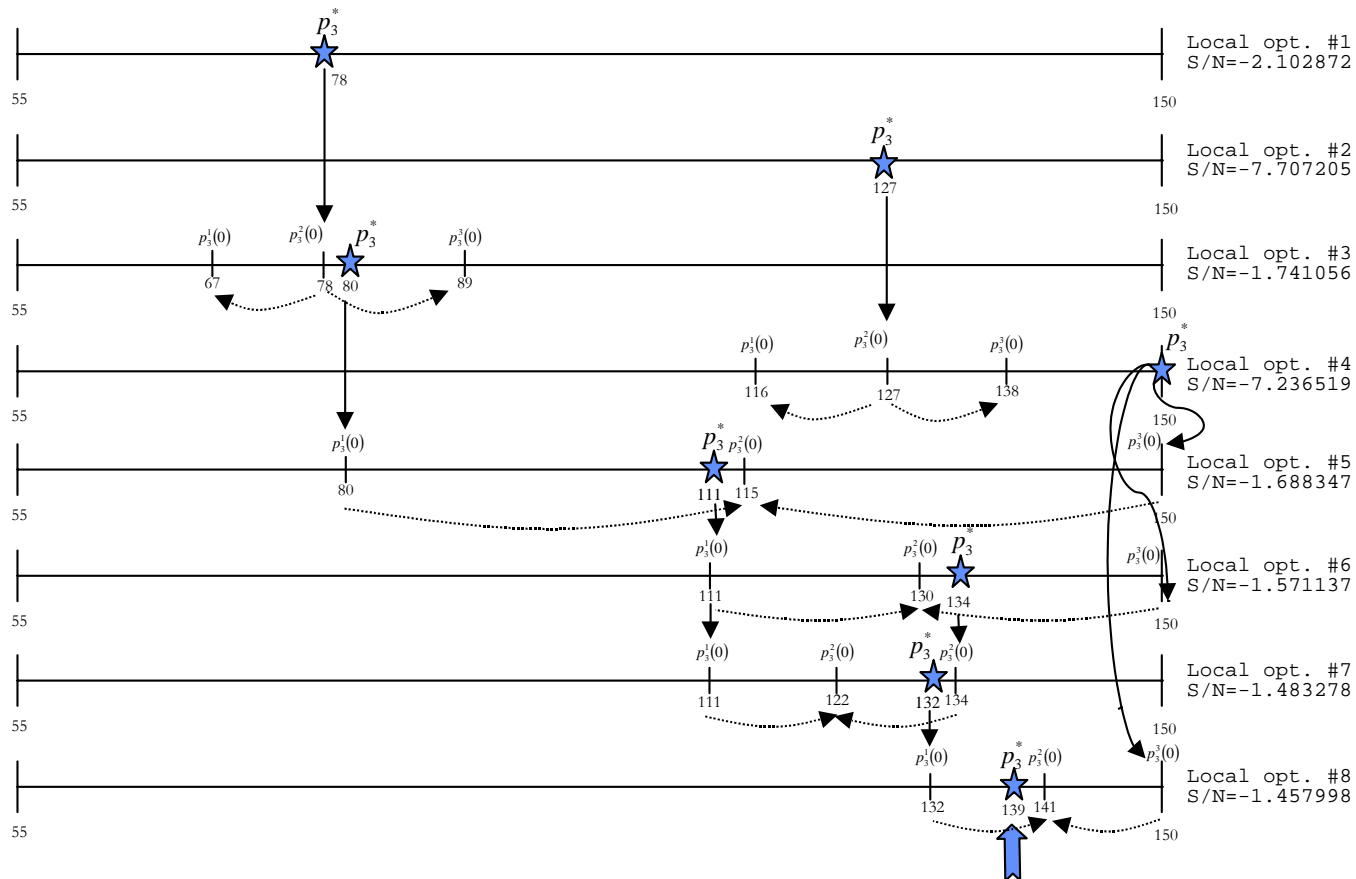


Figure 8. Ratio of the computational times using the parameter values found with CALIBRA when considering the objective function only, and those obtained with CALIBRA when considering both the objective function and the computational time for the metaheuristic of Adenso-Díaz (1996).

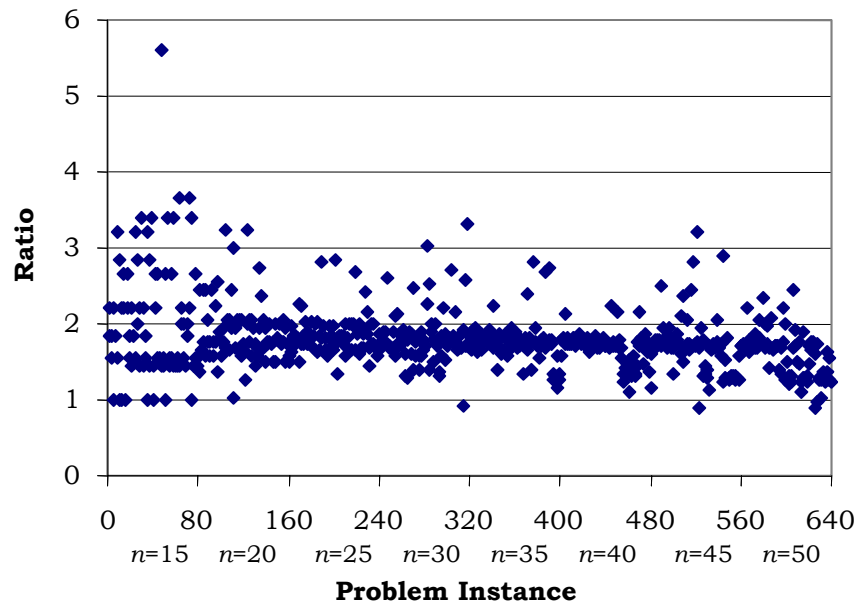


Figure 9. Distribution of average relative deviations of the objective function values found when applying CALIBRA to Boctor's (1991) training set.

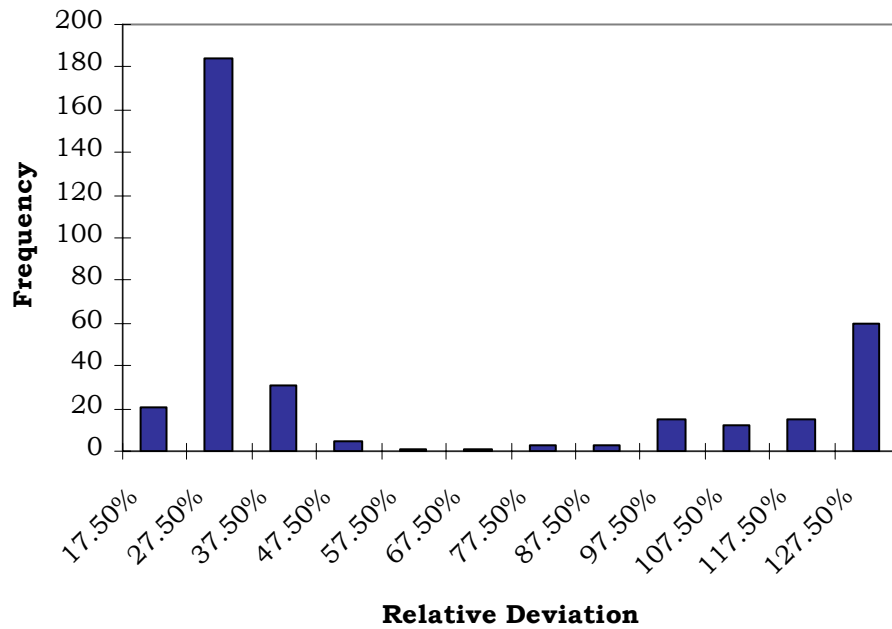


Figure 10. Distribution of average relative deviations of the objective function values found when applying CALIBRA to Adenso-Díaz and Laguna's (2001) training set

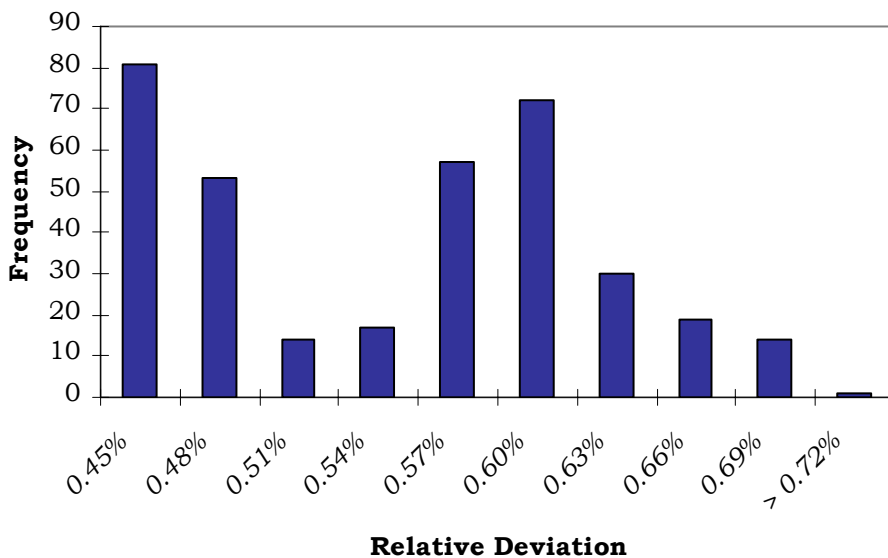


Table 1. Number of problem instances in each training set

Case	Available Instances	Instances for MAXEX = 350 in 8 CPU hours	Instances Used by CALIBRA	Actual Computational Time (hrs.)
Adenso-Díaz and Laguna (2001)	180	12	12	7.87
Lozano <i>et al.</i> (1999)	28	29	14	3.89
Boctor (1991)	70	32	32	8.00
Laguna, Barnes and Glover (1991)	25	343	13	0.42
Adenso-Díaz (1996)	720	40	40	8.10
Laguna and Glover (1993)	25	217	13	0.48

Table 2. Average improvement of CALIBRA parameter values

Case	Average improvement of CALIBRA compared with ORIG	Average improvement of CALIBRA compared with MID
Adenso-Díaz and Laguna (2001)	+0.013%	+0.020%
Lozano <i>et al.</i> (1999)	+0.081%	+0.172%
Boctor (1991)	+2.714%	+32.34%
Laguna, Barnes and Glover(1991)	+0.003%	+0.025%
Adenso-Díaz (1996)	+0.030%	+0.000%
Laguna and Glover (1993)	+0.538%	+0.319%

Table 3. Comparison using number of optimal or best known solutions found

Case	Available Instances	No. of optima or best-known found by			Confidence interval for p1-p2 (CALIBRA-ORIG)	Confidence interval for p1-p2 (CALIBRA-MED)
		CALIBRA	ORIG	MED		
Adenso-Díaz and Laguna (2001)	180	44	51	48	[-0.130,0.052]	[-0.112,0.068]
Lozano <i>et al.</i> (1999)	28	28	27	25	[-0.033,0.104]	[-0.007,0.222]
Boctor (1991)	70	64	58	33	[-0.024,0.200]	[0.309,0.577] *
Laguna, Barnes and Glover (1991)	25	19	15	6	[-0.095,0.415]	[0.283,0.757] *
Adenso-Díaz (1996)	720	692	620	673	[0.071,0.129] *	[0.003,0.049] *
Laguna and Glover (1993)	25	11	12	11	[-0.316,0.236]	[-0.275,0.275]

* Statistically significant difference detected ($\alpha=0.05$)

Table 4. Comparison based on the number of times solutions of better, equal or worse quality are found, using McNemar test.

Case	Available Instances	CALIBRA compared with ORIG			p-value	CALIBRA compared with MID			p-value
		BETTER	WORSE	EQUAL		BETTER	WORSE	EQUAL	
Adenso-Díaz and Laguna (2001)	180	55	51	74	0.771	49	61	70	0.294
Lozano <i>et al.</i> (1999)	28	1	0	27	1.000	3	0	25	0.250
Boctor (1991)	70	6	0	64	0.031 (*)	37	0	33	0.000 (*)
Laguna, Barnes and Glover (1991)	25	10	6	9	0.453	18	0	7	0.000 (*)
Adenso-Díaz (1996)	720	93	16	611	0.000 (*)	37	20	663	0.034 (*)
Laguna and Glover (1993)	25	7	5	13	0.773	7	6	12	1.000

(*) Statistically significant result ($\alpha=0.05$)

Table 5. Results of running CALIBRA to find the best parameter values for each problem in a training set

Case	Statistic	Parameters					Average Improvement
		1	2	3	4	5	
Boctor (1991)	Best common values	11	0.87	139	0.46	424	8.00%
	Average	31.38	0.76	82.94	0.46	405.84	
	Std. Dev.	16.07	0.09	13.77	0.06	66.87	
	Minimum	10.00	0.61	56.00	0.30	200.00	
	Maximum	78.00	0.94	120.00	0.60	500.00	
Adenso-Díaz and Laguna (2001)	Best common values	0.33	0.53	0.04	-2	10	0.06%
	Average	0.30	0.55	0.05	-2.00	13.33	
	Std. Dev.	0.08	0.08	0.02	0.00	4.92	
	Minimum	0.13	0.53	0.04	-2.00	10.00	
	Maximum	0.33	0.82	0.08	-2.00	20.00	