Hindawi Publishing Corporation
International Journal of Engineering Mathematics
Volume 2015, Article ID 167031, 9 pages
http://dx.doi.org/10.1155/2015/167031

Research Article

Improving the Performance of Metaheuristics: An Approach Combining Response Surface Methodology and Racing Algorithms

Eduardo Batista de Moraes Barbosa, Edson Luiz França Senne, and Messias Borges Silva

School of Engineering at Guaratinguetá (FEG), Universidade Estadual Paulista (UNESP), Avenida Doutor Ariberto Pereira da Cunha 333, 12516-410 Guaratinguetá, SP, Brazil

Correspondence should be addressed to Eduardo Batista de Moraes Barbosa; eduardo.bmbarbosa@gmail.com

Received 30 May 2015; Accepted 30 August 2015

Academic Editor: Song Cen

The setup of heuristics and metaheuristics, that is, the fine-tuning of their parameters, exercises a great influence in both the solution process, and in the quality of results of optimization problems. The search for the best fit of these algorithms is an important task and a major research challenge in the field of metaheuristics. The fine-tuning process requires a robust statistical approach, in order to aid in the process understanding and also in the effective settings, as well as an efficient algorithm which can summarize the search process. This paper aims to present an approach combining design of experiments (DOE) techniques and racing algorithms to improve the performance of different algorithms to solve classical optimization problems. The results comparison considering the default metaheuristics and ones using the settings suggested by the fine-tuning procedure will be presented. Broadly, the statistical results suggest that the fine-tuning process improves the quality of solutions for different instances of the studied problems. Therefore, by means of this study it can be concluded that the use of DOE techniques combined with racing algorithms may be a promising and powerful tool to assist in the investigation, and in the fine-tuning of different algorithms. However, additional studies must be conducted to verify the effectiveness of the proposed methodology.

1. Introduction

The fine-tuning of heuristics and metaheuristics is, usually, a tedious and laborious work for almost all researchers. However, it exerts strong influence in the solution process and in the quality of results of optimization problems. Some researchers classify this process as an optimization problem with many variables (e.g., the parameters to be set) subject to several constraints (e.g., ranges of the parameters), where the choice of inappropriate values may result in a poor performance of algorithms and/or low-quality solutions.

Researches about metaheuristics are constantly evolving and cover theoretical developments, new algorithms, and the enhancement techniques to assist the researchers. Since the last decade, there is a growing interest in methods to assist the tuning of these algorithms and reduce the work related to this activity. Therefore, the fine-tuning of metaheuristics is an important field of research in the development context of these algorithms and in the evaluation of problems from areas such as Operations Research and Engineering.

Since the last decade, many researchers (e.g., [1–6] and many others) have been studying the use of different methodologies in order to summarize the process. Broadly, there is a consensus that the fine-tuning of algorithms requires a robust statistical approach, supported by efficient algorithms methods, in order to aid in the process of understanding and also in the effective settings. In that context, the design of experiments (DOE) methodology, a framework of statistical techniques, which enables simultaneous studies of multiple parameters combined between them, as well as the concept of racing algorithms, a prominent method, which automates the fine-tuning of algorithms by streamlining the assessment of alternatives (candidate configurations) and by releasing those that appeared less promising during the evaluation process, must be highlighted.

This paper aims to present an approach combining DOE techniques and racing algorithms to improve the performance of different metaheuristics from distinct nature, such as genetic algorithm (GA) and simulated annealing (SA), where the main difference between them is the way that the searching patterns are implemented. Our approach will be illustrated by means of a case study, where a set of parameters of both algorithms will be simultaneously studied by means of the response surface methodology (RSM), in order to define the space of search (e.g., candidate configurations) of each one, followed by a racing algorithm to define the best fit of the algorithms. The quality of the proposed settings for GA and SA will be evaluated applying the selected metaheuristics in classical optimization problems, such as the travelling salesman problem (TSP) and the scheduling problem to minimizing the total weighted tardiness in a single machine (TWTP), and through the results comparison from the default metaheuristics and ones using the settings suggested by the study of fine-tuning.

The rest of the paper is structured as follows: Section 2 presents an overview about the studied problems (TSP and TWTP), as well as the algorithms that will be used in the case study. The problem of fine-tuning algorithms is presented in Section 3. This section also presents our approach combining RSM and racing algorithm to fine-tune different algorithms. The proposed approach is applied in a case study (Section 4) with different parameters of GA and SA. Section 5 presents the results of case study and its analysis. Our final considerations are in Section 6.

2. Considered Problems and Algorithms

Many optimization problems, especially those related to the real world (resource allocation, facility location, vehicle routing, etc.), cannot be solved exactly considering realistic time limits. Essentially, these problems consist in finding an absolute extreme (maximum or minimum), called optimum, from an objective function with many local extremes (Figure 1).

Generally, such problems are inherently complex and therefore require a lot of computing effort. Some of those problems are classics in the Operations Research field, that is, travelling salesman and scheduling, since it involves a significant number of publications in the specialized literature [7–11].

The travelling salesman problem (TSP) is a classical optimization problem, where the idea is to find the shortest route between a set of given cities, starting and ending at the same city, such that each city should be visited exactly once. A TSP consists of a set C of cities ($c = 1, 2, \ldots, n$) and the corresponding distance d_{ij} of each pair of cities $i, j \in C$, such as $i \neq j$. The problem is classified as symmetric if $d_{ij} = d_{ji}$, $\forall i, j$, or asymmetric if $d_{ij} \neq d_{ji}$, $\forall i, j$ [8].

Scheduling is another classical optimization problem involving tasks that must be arranged in m machines ($m \geq 1$) subject to restrictions, in order to optimize an objective function. In contemporary literature one of the most widely studied goals is that related to dates, especially significant for the industry, due to the need of meeting deadlines.

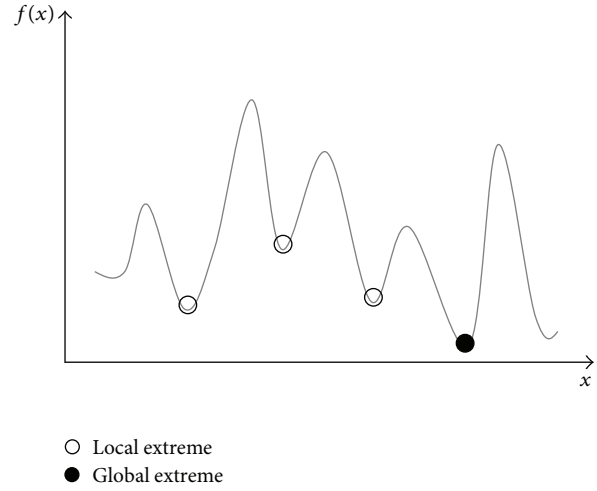

Figure 1: Objective function for a minimization problem.

Among the problems with date restrictions, there is the total weighted tardiness in a single machine [12–14]. These problems consider a set J of tasks ($j = 1, 2, \ldots, n$) and only one machine to process at most one task at a time. When the processing of a task is started, it cannot be stopped. Each task $j \in J$ spends a processing time p_j (time units), positive and continuous, and has a weight w_j, a start time r_j, and a due date d_j. Broadly, the tardiness is the difference between the task due date and its effective completion. The tardiness (T_j) is computed as $\max(0, C_j - d_j)$, where C_j is the time completion of task j. The purpose of this problem is to organize the tasks to find an optimal sequence such that the weighted total delay is minimized.

Both problems are known to be NP-hard [15]; thus, in order to get optimal solutions, a great computing effort is required and demands the use of efficient algorithms.

Metaheuristics are one of the best known approaches to solving problems for which there is no specific and/or efficient algorithm. Many metaheuristics are inspired by metaphors from different knowledge areas, such as biology (genetic algorithms and neural networks) and physics (particle swarm optimization and simulated annealing). Usually, these algorithms differ from each other in terms of searching pattern but offer accurate and balanced methods for diversification (search space exploration) and intensification (exploitation of a promising region), share features, such as the use of stochastic components (involving randomness of variables), and have a variety of parameters that must be set according to the problem under study.

The genetic algorithm (GA) is a population-based method invented by Holland [16], inspired by principles of survival from Darwin's evolution theory. GA simulates an evolution process in which the fitness of individuals (parents) is crucial to generate new individuals (children). On the other hand, the simulated annealing (SA) is a probabilistic method proposed in Kirkpatrick et al. [17] and Černý [18] in order to find the global minimum of an objective function with numerous local minimums. Widely applied to solve optimization problems, SA simulates a physical process from

which a solid is cooled slowly, so that the final product becomes a homogeneous mass to achieve a minimum energy configuration [19].

The main difference between these algorithms is the way that the searching methods are implemented. GA operates on a population of solutions, where the new generations (offspring) are generated from the fittest individuals of previous generations (parents). This feature (survival principle) guarantees an increase in the quality of solutions as new generations are created. On the other hand, the SA performs constant movement between one solution (s) and another (s') according to some predefined neighborhood structure. SA uses a probabilistic test to accept new solutions, and sometimes this feature allows low-quality solutions to be considered in the search algorithm process.

Both GA and SA algorithms have a wide range of parameters (e.g., rates of crossover and mutation and population size, for GA, and initial temperature and its rate of decrement and number of interactions, for SA) that must be tuned before starting the solution of a problem. Since the metaheuristics are extremely dependent on the values assigned to those parameters, they must be carefully studied during the process of fine-tuning, since they can define the success of the algorithm.

3. Improving the Performance of Metaheuristics

A is an algorithm with a set of parameters (V) applied on different problems (P). Thus, the problem of fine-tuning an algorithm can be summarized as a search space:

$$A_{Pi} = \left(\alpha_i \longleftarrow 1 \cdots n_\alpha \times \beta_i \longleftarrow 1 \cdots n_\beta \times \cdots \times \xi_i \longleftarrow 1 \cdots n_\xi\right), \quad i = 1, \ldots, m, \tag{1}$$

where $\alpha, \beta, \ldots, \xi$ are parameters of algorithm A for a given problem P and $1 \cdots n_\alpha, 1 \cdots n_\beta, \ldots, 1 \cdots n_\xi$ are the finite range of values assumed for each parameter. The number of parameters as well as their ranges can vary extensively according to A and P studied, such that $n_\alpha \times n_\beta \times \cdots \times n_\xi$ should be possibly a number of combinations tests of A on P.

Our approach to fine-tune algorithms can be expressed as a procedure that begins with an arbitrary selection of instances from a class of optimization problems and followed by the definition of ranges for each parameter from the algorithm, to apply 2^k full factorial design in order to study the response (effect) of multiple parameters (factors) [20–22]. By means of the full factorial designs it is possible to establish a relationship of cause and effect between factors and response, whose usual representation is an empirical regression model (linear or quadratic) for the process under analysis, such as

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_{12} X_1 X_2 + \varepsilon, \tag{2}$$

where Y is the response, β_i are the coefficients, X_1 and X_2 are factors, and ε is the experimental error.

Factorial designs are useful for identifying the factors influencing the response. However, when the interest is to define the settings of the factors that can optimize the process and produce values (of factors) closer to the optimum, (2) can be insufficient to represent the relationship between factors and response. Thus, the next stage of our approach employs the response surface methodology (RSM) as a fine-tuning tool, to achieve greater proximity of regions with promising settings. The RSM is a framework of statistical and mathematical techniques, which employs factorial designs, regression analysis, and optimization methods in situations where several input parameters influence the performance of a process [23, 24]. Roughly, the RSM works by exploring the neighborhood regions around a promising region.

The result of RSM is a second-order model [22], given by

$$Y = \beta_0 + \beta_1 X_1 + \beta_{11} {X_1}^2 + \beta_2 X_2 + \beta_{22} {X_2}^2 + \beta_{12} X_1 X_2 + \varepsilon, \tag{3}$$

to represent the relationship between response and factors in the new region, and a three-dimensional contour with the shape of the surface (Figure 2).

The final stage of our approach consists in applying a racing algorithm to define the setup of the algorithms. Racing was introduced by Maron and Moore [25]. Differently from a brute-force technique, where all alternatives must be tested by an equal number of times, the racing idea is to decrease the number of alternatives (candidate configurations) as soon as statistical evidence exists to discard them. Thus, at the beginning, a very large set of alternatives should exist, which decreases by the elimination of the worst ones. The racing concept was studied in Birattari et al. [26, 27] by means of the method F-Race, a racing algorithm, where the candidate configurations are eliminated by means of the Friedman statistics. In [28] the authors propose an iterative application of F-Race, so-called I/F-Race. The racing and sharpening are combined [29] in order to increase the efficiency of search. An extension of the I/F-Race procedure by means of a package implemented in R is presented in [30]. Different sampling techniques for F-Race are suggested in [31, 32].

4. Case Study

To illustrate our approach of fine-tuning algorithms, we selected a set of parameters that intuitively seems to influence the performance of the metaheuristics GA and SA, regardless of the studied problem. For GA, we chose the following parameters: mutation probability (pMut), crossover probability (pCros), size of the population (sPop), and number of generations to be computed (nGen). For SA, the considered parameters are initial probability of accepting a solution (pIni), number of temperature stages (iExt), number of iterations during one temperature stage (iInt), and decrease in temperature (dTem).

To generalize our results and compare them among themselves, we use the relative deviation from the optimum, given by

$$\text{Dev} = \frac{f(s) - f(s^*)}{f(s^*)}, \tag{4}$$

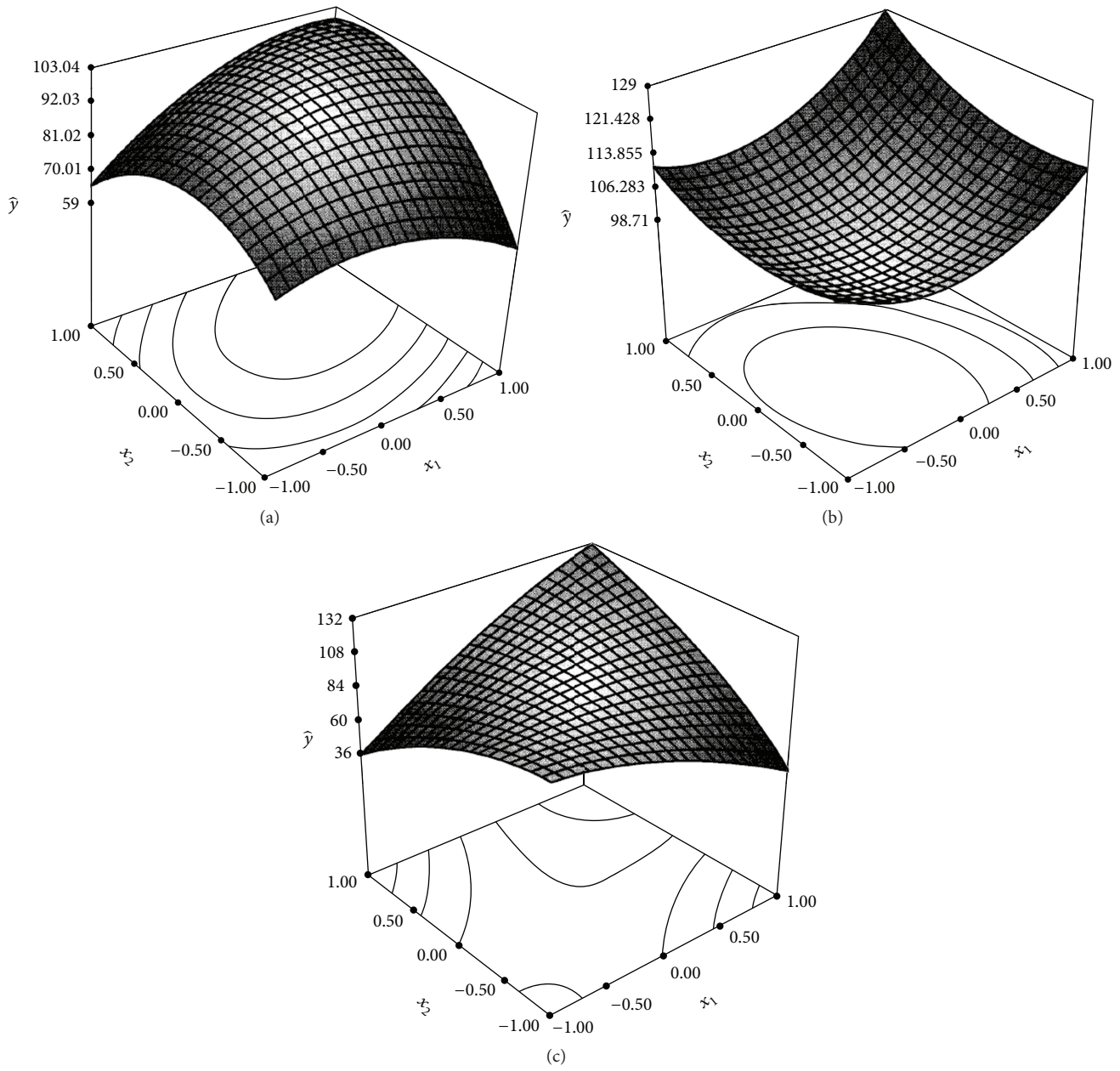

Figure 2: Response surface shapes. (a) Maximum; (b) minimum; (c) saddle point. Source: Montgomery, 2001.

where $f(s)$ is our computed solution and $f(s^*)$ is the best known solution of the problem. Thus, the lower the value of Dev for the metaheuristics, the better the performance of the algorithms.

The parameters and their corresponding levels (low and high) required by a 2^k full factorial design in the first stage of our approach are in Table 1.

Table 1: Parameters settings for GA and SA.

GA param.	pCros	pMut	sPop	nGen
Low	0.01	0.01	10	10
High	0.99	0.99	200	200
SA param.	pIni	iExt	iInt	dTem
Low	0.50	30	1000	0.01
High	0.95	100	2000	0.99

4.1. TSP. The fine-tuning of the metaheuristics GA and SA on TSP uses four arbitrary instances from the TSPLIB (URL: http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/).

The $2^{k=4}$ full factorial design was used here in order to identify the factors that influence the response. Thus, in the first stage of this approach we can conclude for both GA and SA algorithms the following:

(i) All four factors studied are significant for the process, regardless of the instance selected.

Table 2: Parameters settings for GA.

GA param.	Inst. 1	Inst. 2	Inst. 3	Inst. 4
pCros	0.54	0.57	0.63	0.57
pMut	0.59	0.70	0.72	0.86
sPop	110	116	129	113
nGen	176	198	230	229

Table 3: Parameters settings for SA.

SA param.	Inst. 1	Inst. 2	Inst. 3	Inst. 4
pIni	0.74	0.75	0.74	0.76
iExt	64	65	64	65
iInt	1496	1491	1493	1489
dTem	0.55	0.54	0.51	0.53

(ii) There are differences between interactions of the factors according to the instance studied.

The next stage consists in applying the RSM to explore the neighborhood regions around a promising region and get values for each parameter according to the studied instance. The procedure employed consists in simultaneous study of all four parameters of each algorithm until ANOVA shows statistical significance and suggests an empirical model to explain the relationships between them. The direct search algorithm based on the simplex of Nelder and Mead [33] was used in order to optimize the empirical model. The values of the parameters should result in the best responses.

After RSM, the next stage uses a racing algorithm inspired by *F*-Race [26, 27] to define the setup of the algorithms. The main contribution of our approach is combining the power of RSM (a framework of mathematical and statistical techniques employed in the modeling and optimization of problems, where the response is affected by many factors) and the efficiency of racing algorithms. In this context, RSM is a sampling technique that accurately finds the best fit of each parameter. That is, from the results of RSM (Tables 2 and 3) we define a range of values between minimum and maximum of each parameter, such that it forms a space of search of candidate configurations. Thus, the ranges of GA are as follows:

(i) pCros [0.54 and 0.63];

(ii) pMut [0.59 and 0.86];

(iii) sPop [110 and 130];

(iv) nGen [176 and 230].

The same methodology produces the following ranges for SA:

(i) pIni [0.74 and 0.76];

(ii) iExt [64 and 65];

(iii) iInt [1490 and 1500];

(iv) dTem [0.51 and 0.55].

Our idea of using a racing algorithm is to select the as good as possible configuration out of a lot of options. For this study, the settings used for GA are pCros $\in \{0.54, 0.585, 0.63\}$, pMut $\in \{0.59, 0.725, 0.86\}$, sPop $\in \{110, 130\}$, and nGen $\in \{176, 230\}$, and for SA we consider pIni $\in \{0.74, 0.75, 0.76\}$, iExt $\in \{64, 65\}$, iInt $\in \{1490, 1500\}$, and dTem $\in \{0.51, 0.53, 0.55\}$. Each possible combination leads to one different algorithm setting, such that our space of search was composed of 36 different parameter settings for both algorithms. After applying racing algorithm in the space of search we reached the best setting for each algorithm to solve the TSP (Table 4). This table also presents the default values for each metaheuristic in Scilab.

Table 4: Default and suggested parameters settings.

GA param.	pCros	pMut	sPop	nGen
Default	0.70	0.10	100	10
Suggested	0.54	0.79	110	203
SA param.	pIni	iExt	iInt	dTem
Default	0.80	100	1000	0.90
Suggested	0.74	64	1490	0.51

Table 5: Parameters settings for GA.

GA param.	Inst. 1	Inst. 2	Inst. 3	Inst. 4
pCros	0.54	0.51	0.55	0.53
pMut	0.56	0.58	0.54	0.57
sPop	110	121	140	116
nGen	123	133	137	120

Table 6: Parameters settings for SA.

SA param.	Inst. 1	Inst. 2	Inst. 3	Inst. 4
pIni	0.74	0.76	0.74	0.71
iExt	62	61	64	64
iInt	1439	1429	1478	1401
dTem	0.58	0.56	0.58	0.53

4.2. TWTP. This approach begins as previously presented from the arbitrary selection of four instances of the "wt40," a TWTP benchmark with 40 tasks from the OR Library (URL: http://people.brunel.ac.uk/~mastjjb/jeb/info.html).

Here, all four factors studied are also significant for the process, but it is possible to identify differences between their interactions. The results of applying RSM in the instances are presented in Tables 5 and 6.

Once again, we applied a racing algorithm considering the space of search of candidate configurations built from ranges of minimum and maximum of each parameter (Tables 5 and 6). For this problem, the ranges of GA are as follows:

(i) pCros [0.51 and 0.55];

(ii) pMut [0.54 and 0.58];

(iii) sPop [110 and 140];

(iv) nGen [120 and 140].

Table 7: Default and suggested parameters settings.

GA param.	pCros	pMut	sPop	nGen
Default	0.70	0.10	100	10
Suggested	0.52	0.55	110	140
SA param.	pIni	iExt	iInt	dTem
Default	0.80	30	1000	0.90
Suggested	0.71	61	1416	0.56

Table 8: Statistics of the GA after 10 runs on 10 instances of TSP.

Inst.	AD_{GA}	$NOpt_{GA}$	$AD_{GA'}$	$NOpt_{GA'}$
Berlin52	1.84	0	0.32	0
Eil51	1.78	0	0.36	0
Eil76	2.61	0	0.84	0
KroA100	5.09	0	1.91	0
Pr76	2.94	0	0.90	0
St70	3.06	0	0.83	0
Eil101	3.27	0	1.33	0
Lin105	5.28	0	1.70	0
Ch130	5.20	0	2.46	0
Tsp225	7.84	0	4.29	0
μ	**3.89**	—	**1.49**	—
σ	**1.82**	—	**1.14**	—

The same methodology suggests the following ranges for SA:

(i) pIni [0.71 and 0.76];

(ii) iExt [61 and 64];

(iii) iInt [1400 and 1480];

(iv) dTem [0.53 and 0.58].

Here, the settings used for GA are pCros $\in \{0.51, 0.53, 0.55\}$, pMut $\in \{0.54, 0.56, 0.58\}$, sPop $\in \{110, 140\}$, and nGen $\in \{120, 140\}$, while for SA we have pIni $\in \{0.71, 0.735, 0.76\}$, iExt $\in \{61, 64\}$, iInt $\in \{1400, 1480\}$, and dTem $\in \{0.53, 0.555, 0.58\}$, such that our space of search was composed of 36 different parameter settings for both algorithms. Thus, running the racing we reached the following setup for each metaheuristic (Table 7).

5. Analysis of Results

Our results were collected using the scientific software Scilab (http://www.scilab.org) in Intel Core i5TM 1.8 GHz, 6 GB memory, 1 TB hard disc on Windows 8 64 bits.

All results presented in this section were computed by means of (4) and in order to do the comparisons, they were collected before the fine-tuning process (called GA and SA) using the default settings of each metaheuristic in Scilab and then (called GA′ and SA′) with the algorithms set according to Table 4 for TSP and Table 8 for TWTP.

It should be noted that the procedure to improve the performance of metaheuristics (Section 3) can be adapted to many algorithms coming from distinct natures (e.g., tabu search, ant colony optimization, particle swarm optimization, etc.); thus as for the purpose of this paper, we consider sufficiently the use of distinct metaheuristics GA and SA.

Table 9: Statistics of the SA after 10 runs on 10 instances of TSP.

Inst.	AD_{SA}	$NOpt_{SA}$	$AD_{SA'}$	$NOpt_{SA'}$
Berlin52	0.34	0	0.07	0
Eil51	0.35	0	0.06	0
Eil76	0.64	0	0.09	0
KroA100	1.10	0	0.09	0
Pr76	0.39	0	0.05	0
St70	0.60	0	0.06	0
Eil101	1.03	0	0.10	0
Lin105	1.50	0	0.12	0
Ch130	1.22	0	0.11	0
Tsp225	1.64	0	0.38	0
μ	**0.88**	—	**0.11**	—
σ	**0.46**	—	**0.09**	—

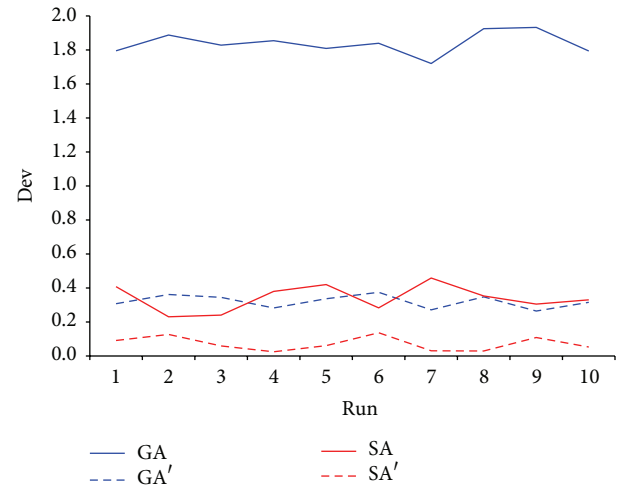

Figure 3: Time series of a single instance (Inst. Berlin52) of TSP.

5.1. TSP. The first set of results (Tables 8 and 9) corresponds to 10 runs of the metaheuristics GA and SA in ten instances of the benchmark of symmetric TSP from TSPLIB. In these tables, column AD is the arithmetic mean of (4) in 10 runs, such as $AD = (1/N)\sum_{i=1}^{N} \text{Dev}(i)$, where $N = 10$; NOpt is the number of times that algorithms reach the optimum for each instance; and μ and σ are arithmetic mean and standard deviation, respectively, of all instances.

The statistics of GA′ and SA′ (Tables 8 and 9) reveal an increase of the performances of both algorithms after the fine-tuning (about 65% and 85% for GA′ and SA′, resp.). It should be highlighted that both metaheuristics are more promising for almost all instances. Those results (Tables 8 and 9) also suggest that the fine-tuning makes the process more stable for SA, since $\mu_{SA'} = 0.11 \pm 0.09$, whereas $\mu_{GA'} = 1.49 \pm 1.14$.

When we analyze the time series of both algorithms (Figure 3) on a single instance (Inst. Berlin52) of TSP, the SA has the best performance if compared with GA, but there is an increase of both performances with the fine-tuning. From

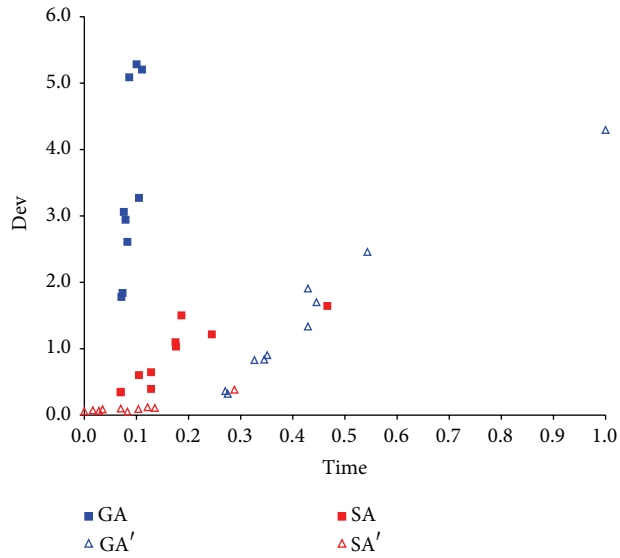

Figure 4: Performance of the metaheuristics over time.

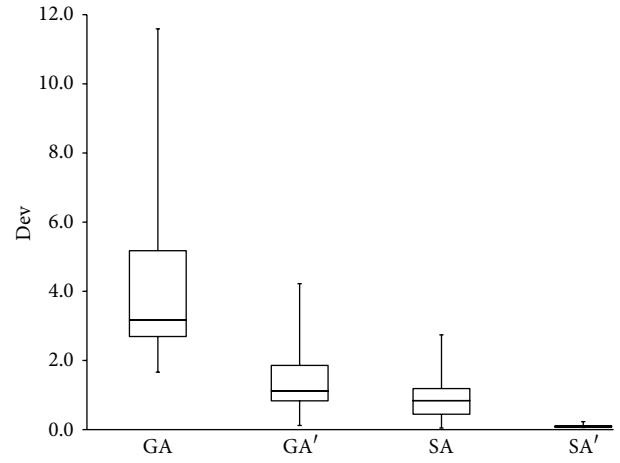

Figure 5: Variability of the studied metaheuristics.

this graph it should be highlighted that the performance of GA′ is very similar to the algorithm SA (before fine-tuning process). However, comparing the metaheuristics with themselves, the fine-tuning process (dashed lines on graph) grants an improvement in terms of quality of solutions for both algorithms.

The substantial increase in the performance of both algorithms (GA and SA) can also be noted when we analyze the normalized execution times (Figure 4), where SA and SA′ show better results. That is, with the fine-tuning process, SA′ presents good performance in lesser time. Nevertheless, the decrease in time is relatively less than the increase in the quality of solutions. On the other hand, for GA′ there is also an improvement in the quality of solutions, but its time is slightly larger.

Through the results (Tables 8 and 9) and supported by the graphical analysis we can highlight that the fine-tuning process produces better results (Figure 5).

Table 10: Statistics of the GA after 10 runs on wt40 instances.

Inst.	AD_{GA}	$NOpt_{GA}$	$AD_{GA'}$	$NOpt_{GA'}$
1	1.63	0	0.08	1
2	0.59	0	0.16	0
3	2.42	0	0.20	0
4	0.53	0	0.15	0
5	0.87	0	0.07	5
6	0.70	0	0.04	3
7	0.74	0	0.06	1
8	0.54	0	0.04	1
9	0.32	0	0.02	0
10	0.48	0	0.03	0
μ	**0.88**	—	**0.08**	—
σ	**0.61**	—	**0.06**	—

Table 11: Statistics of the SA after 10 runs on wt40 instances.

Inst.	AD_{SA}	$NOpt_{SA}$	$AD_{SA'}$	$NOpt_{SA'}$
1	1.13	0	0.04	1
2	0.60	0	0.08	1
3	1.93	0	0.08	0
4	0.55	0	0.04	4
5	0.51	0	0.04	7
6	0.60	0	0.08	1
7	0.72	0	0.05	2
8	0.53	0	0.05	3
9	0.33	0	0.03	0
10	0.56	0	0.05	0
μ	**0.75**	—	**0.05**	—
σ	**0.44**	—	**0.02**	—

5.2. TWTP. In Tables 10 and 11 are the sets of results of 10 runs of the GA and SA in the first ten instances from the OR Library benchmark "wt40." In those tables columns AD, NOpt, μ, and σ have similar meaning as previously described (Section 5.1).

The statistics of GA′ (Table 10) suggests an increment in the quality of results for all instances. Its statistics stands out for instance 5, where about 50% of runs can reach the optimum. SA′ also shows the best results for instance 5; however the optimum can be reached for 70% of runs (Table 11). In general, it is also noted that SA′ can reach the optimum more frequently. The statistics (Tables 10 and 11) suggest that SA′ has more stable results than GA′, since $\mu_{SA'} = 0.05 \pm 0.02$, whereas $\mu_{GA'} = 0.08 \pm 0.06$.

Through the analysis of the time series of both algorithms (Figure 6) on a single instance (Inst. 1) of TWTP, it can be noted that after the fine-tuning process (dashed lines on graph), there is an improvement of about 90% in the performances of both metaheuristics.

Just as previously noted (Section 5.1), SA also presents better results in terms of the execution time (Figure 7). Through the fine-tuning, SA′ can reach the best results in lesser time, whereas GA′ also reaches the best, but in a larger execution time.

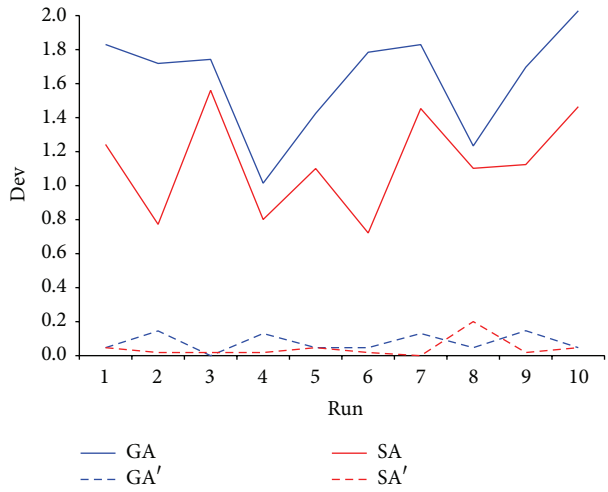

Figure 6: Time series of a single instance (Inst. 1) of TWTP.

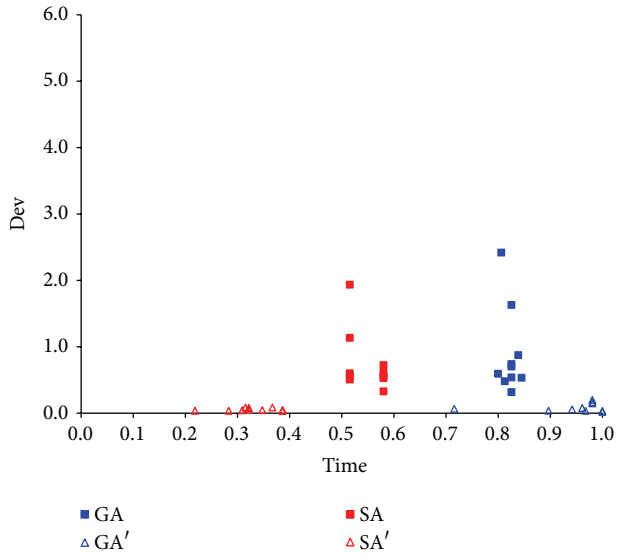

Figure 7: Performance of the metaheuristics over time.

The statistics (Tables 10 and 11) also suggest that the fine-tuning process improves the performance of the algorithms and produces better results, closer to the optimum (Figure 8).

6. Final Considerations

This paper presented a study on fine-tuning of different metaheuristics through a statistical approach combining RSM and racing algorithm. From a case study was investigated the influence of different parameters of the GA and SA, applied in different instances of classical optimization problems, such as TSP and TWTP. The quality of settings for GA and SA was evaluated and compared considering default algorithms settings and ones using the settings suggested by the study of fine-tuning.

The use of RSM allows defining the search space of candidate configurations, explored by means of a racing algorithm

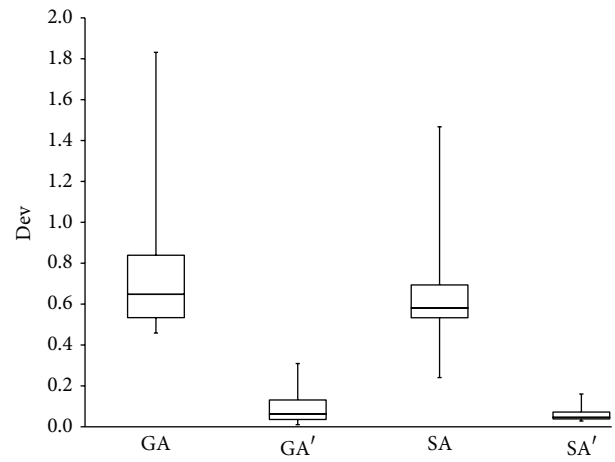

Figure 8: Variability of the studied metaheuristics.

to reach the best fit of each parameter of the metaheuristics to solve different instances of studied problems.

From a case study we collected different results of the TSP and TWTP before and after the fine-tuning of the algorithms. In general, it can be seen that, regardless of the nature of the metaheuristic, the fine-tuning process improves the quality of solutions and allows for both GA and SA achieving better results for different instances of the problems. In the comparisons of our results with themselves, SA has the best results for the TSP (quality of solution), whereas for the TWTP, they both have similar performance, but SA is slightly better than GA. In terms of the execution time, in general, the SA produces good performance in lesser time for both of the studied problems, whereas the GA is the slowest in finding better results.

It is important to note that the metaheuristics GA and SA, as well as the problems TSP and TWTP, were used in this work only to demonstrate our approach combing RSM and racing algorithm. The aim was to verify the effectiveness of the proposed methodology applied in these cases. Our results suggest that the proposed approach may be a promising and powerful tool to assist in the fine-tuning of different algorithms. However, additional studies must be conducted to verify the effectiveness of the proposed methodology, mainly when applied on already well-configured algorithms.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

References

[1] S. P. Coy, B. L. Golden, G. C. Runger, and E. A. Wasil, "Using experimental design to find effective parameter settings for heuristics," *Journal of Heuristics*, vol. 7, no. 1, pp. 77–97, 2001.

[2] T. Bartz-Beielstein, K. E. Parsopoulos, and M. N. Vrahatis, "Design and analysis of optimization algorithms using computational statistics," *Applied Numerical Analysis & Computational Mathematics*, vol. 1, no. 3, pp. 413–433, 2004.

[3] B. Adenso-Díaz and M. Laguna, "Fine-tuning of algorithms using fractional experimental designs and local search," *Operations Research*, vol. 54, no. 1, pp. 99–114, 2006.

[4] K. Y. Chan, M. E. Aydin, and T. C. Fogarty, "Main effect fine-tuning of the mutation operator and the neighbourhood function for uncapacitated facility location problems," *Soft Computing*, vol. 10, no. 11, pp. 1075–1090, 2006.

[5] F. Dobslaw, "A parameter tuning framework for metaheuristics based on design of experiments and artificial neural networks," in *Proceedings of the International Conference on Computer Mathematics and Natural Computing*, pp. 1–4, 2010.

[6] A. Arin, G. Rabadi, and R. Unal, "Comparative studies on design of experiments for tuning parameters in a genetic algorithm for a scheduling problem," *International Journal of Experimental Design and Process Optimisation*, vol. 2, no. 2, pp. 103–124, 2011.

[7] D. Davendra, *Traveling Salesman Problem, Theory and Applications*, InTech, 2010.

[8] R. Matai, S. P. Singh, and M. L. Mittal, "Traveling salesman problem: an overview of applications, formulations, and solution approaches," in *Traveling Salesman Problem, Theory and Applications*, D. Davendra, Ed., InTech, 2011.

[9] D. Karger, C. Stein, and J. Wein, *Scheduling Algorithms*, Chapman & Hall/CRC, 1997.

[10] G. Schmidt, "Scheduling with limited machine availability," *European Journal of Operational Research*, vol. 121, no. 1, pp. 1–15, 2000.

[11] Y. Ma, C. Chu, and C. Zuo, "A survey of scheduling with deterministic machine availability constraints," *Computers and Industrial Engineering*, vol. 58, no. 2, pp. 199–211, 2010.

[12] E. Vallada and R. Ruiz, "Cooperative metaheuristics for the permutation flowshop scheduling problem," *European Journal of Operational Research*, vol. 193, no. 2, pp. 365–376, 2009.

[13] M. M. Mazdeh, A. Hamidinia, and A. Karamouzian, "A mathematical model for weighted tardy jobs scheduling problem with a batched delivery system," *International Journal of Industrial Engineering Computations*, vol. 2, no. 3, pp. 491–498, 2011.

[14] H. Feili, H. Haddad, and P. Ghanbari, "Two hybrid algorithms for single-machine total weighted tardiness scheduling problem with sequence-dependent setup," *American Journal of Scientific Research*, no. 64, pp. 22–29, 2012.

[15] J. van Leeuwen, *Handbook of Theoretical Computer Science*, The MIT Press, 1994.

[16] J. H. Holland, *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Boston, Mass, USA, 1975.

[17] S. Kirkpatrick, C. D. Gelatt Jr., and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.

[18] V. Černý, "Thermodynamical approach to the traveling salesman problem: an efficient simulation algorithm," *Journal of Optimization Theory and Applications*, vol. 45, no. 1, pp. 41–51, 1985.

[19] D. Bertsimas and J. Tsitsiklis, "Simulated annealing," *Statistical Science*, vol. 8, no. 1, pp. 10–15, 1993.

[20] G. E. P. Box, W. G. Hunter, and J. S. Hunter, *Statistics for Experimenters*, John Wiley & Sons, 1st edition, 1978.

[21] D. C. Montgomery, *Design and Analysis of Experiments*, John Wiley & Sons, 5th edition, 2001.

[22] NIST/Sematech, e-Handbook of Statistical Methods, on-line resourse, 2015, http://www.itl.nist.gov/div898/handbook/.

[23] G. E. P. Box and K. B. Wilson, "On the experimental attainment of optimum conditions," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 13, no. 1, pp. 1–45, 1951.

[24] A. I. Khuri and S. Mukhopadhyay, "Response surface methodology," *WIREs Computational Statistics*, vol. 2, pp. 128–149, 2010.

[25] O. Maron and A. W. Moore, *Hoeffding Races: Accelerating Model Selection Search for Classification and Function Approximation*, Robotics Institute, 1993.

[26] M. Birattari, T. Stützle, L. Paquete, and K. e Varrentrapp, "A racing algorithm for configuring metaheuristics," in *Proceedings of the Genetic and Evolutionary Computation Conference*, July 2002.

[27] M. Birattari, Z. Yuan, P. Balaprakash, and T. Stützle, "F-race and iterated F-race: an overview," IRIDIA—Technical Report Series TR/IRIDIA/2009-018, 2009.

[28] S. K. Smit and A. E. Eiben, "Comparing parameter tuning methods for evolutionary algorithms," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '09)*, pp. 399–406, May 2009.

[29] P. Balaprakash, M. Birattari, and T. Stützle, "Improvement strategies for the F-race algorithm: sampling design and iterative refinement," in *Hybrid Metaheuristics: 4th International Workshop, HM 2007, Dortmund, Germany, October 8-9, 2007. Proceedings*, T. Bartz-Beielstein, M. J. B. Aguilera, C. Blum et al., Eds., vol. 4771 of *Lecture Notes in Computer Science*, pp. 108–122, Springer, Berlin, Germany, 2007.

[30] M. López-Ibáñez, J. Dubois-Lacoste, T. Stützle, and M. Birattari, "The irace package: iterated racing for automatic algorithm configuration," Tech. Rep. TR/IRIDIA/2011-004, IRIDIA, 2011, Technical Report Series.

[31] Z. Yuan, M. Birattari, and T. Stützle, "MADS/F-race: mesh adaptive direct search meets F-race," IRIDIA—Technical Report Series TR/IRIDIA/2010-001, 2010.

[32] Z. Yuan, M. A. M. Oca, T. Stützle, and M. Birattari, "Continuous optimization algorithms for tuning real and integer parameters of swarm intelligence algorithms," IRIDIA—Technical Report Series TR/IRIDIA/2011-017, 2011.

[33] J. A. Nelder and R. Mead, "A simplex method for function minimization," *The Computer Journal*, vol. 7, no. 4, pp. 308–313, 1965.