Hindawi
Journal of Optimization
Volume 2017, Article ID 8042436, 7 pages
https://doi.org/10.1155/2017/8042436

Research Article

Improving the Fine-Tuning of Metaheuristics: An Approach Combining Design of Experiments and Racing Algorithms

Eduardo Batista de Moraes Barbosa[1] **and Edson Luiz França Senne**[2]

[1]*Brazilian Institute for Space Research (INPE), Cachoeira Paulista, SP, Brazil*
[2]*Universidade Estadual Paulista (UNESP), Guaratinguetá, SP, Brazil*

Correspondence should be addressed to Eduardo Batista de Moraes Barbosa; eduardo.barbosa@inpe.br

Received 10 February 2017; Accepted 10 April 2017; Published 7 June 2017

Academic Editor: Ferrante Neri

Copyright © 2017 Eduardo Batista de Moraes Barbosa and Edson Luiz França Senne. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Usually, metaheuristic algorithms are adapted to a large set of problems by applying few modifications on parameters for each specific case. However, this flexibility demands a huge effort to correctly tune such parameters. Therefore, the tuning of metaheuristics arises as one of the most important challenges in the context of research of these algorithms. Thus, this paper aims to present a methodology combining Statistical and Artificial Intelligence methods in the fine-tuning of metaheuristics. The key idea is a heuristic method, called Heuristic Oriented Racing Algorithm (HORA), which explores a search space of parameters looking for candidate configurations close to a promising alternative. To confirm the validity of this approach, we present a case study for fine-tuning two distinct metaheuristics: Simulated Annealing (SA) and Genetic Algorithm (GA), in order to solve the classical traveling salesman problem. The results are compared considering the same metaheuristics tuned through a racing method. Broadly, the proposed approach proved to be effective in terms of the overall time of the tuning process. Our results reveal that metaheuristics tuned by means of HORA achieve, with much less computational effort, similar results compared to the case when they are tuned by the other fine-tuning approach.

1. Introduction

Usually, metaheuristic algorithms are adapted to a large set of problems with few modifications on parameters for each specific case. However, this adaptation in some situations demands a huge effort to correctly tune its parameters. Sometimes, due to time restrictions for the tuning process, the algorithms eventually may lead to solutions of poor quality.

The flexibility of metaheuristics allows these algorithms to find acceptable solutions to a wide range of problems, but at the same time it is difficult to obtain good (or, sometimes, optimal) solutions, due to the difficulty of fine-tuning of parameters. Therefore, the tuning of metaheuristics arises as one of the most important research challenges in the context of the design and application of these algorithms.

These challenges are usually of interest to different research communities. In the contemporary literature, researches on statistical techniques stand out and are supported by efficient methods, in order to aid the process of understanding and also to reach effective settings (e.g., [1–7] and many others).

This paper aims to contribute to the literature by presenting a methodology combining Statistical and Artificial Intelligence methods in the fine-tuning of metaheuristics, such as Design of Experiments (DOE) [8] and the concept of Racing [9, 10]. Broadly, our approach employs DOE as a tool to define a parameter search space. Then, we apply the Racing concept to explore the previously defined search space looking for alternatives close to promising candidate configurations, in order to consistently find the good ones. The search process is focused on dynamically created alternatives in an iterative process, which evaluates and discards some of them based on statistical evidences.

The proposed approach brings together the characteristics of different strategies from the literature, such as

CALIBRA [11], I/F-Race [10, 12], and ParamILS [13], in a single heuristic method with the ability to define the search space and efficiently concentrate on searches for the candidate configurations within this search space. This approach will be illustrated by means of a simple case study, where a set of parameters of two distinct metaheuristics (Simulated Annealing, SA, and Genetic Algorithm, GA) will be tuned to solve a classical optimization problem, such as the traveling salesman problem (TSP). The quality of the proposed settings will be compared with a racing tuning approach.

The rest of the paper is structured as follows: Section 2 presents the problem of tuning metaheuristics and our approach combining Statistical and Artificial Intelligence methods to address this problem. In Section 3 there is an overview about the considered problem, as well as the metaheuristics used in a case study. The proposed approach is applied in a case study (Section 4) to fine-tune the metaheuristics SA and GA. Section 4 also presents the case study results and its analyses. Our final considerations are in Section 5.

2. An Approach to the Problem of Tuning Metaheuristics

In the contemporary literature it is possible to find formal definitions related to the problem of fine-tuning the parameters of algorithms [10, 14, 15].

Let M be a metaheuristic with a set of parameters (e.g., $\alpha, \beta, \ldots, \xi$) that must be tuned to solve a class P of problems. The parameters of M can assume a finite set of values and its cardinality can also vary extensively according to M and P. If Θ is a set of candidate configurations, such that θ is any setting of M, then the problem of tuning metaheuristics can be formalized as a state-space:

$$S = (\Theta, P). \tag{1}$$

Broadly, this problem consists of determining the best setting $\theta \in \Theta$ present in S to solve problems P. However, its determination is not always simple and, in the worst hypothesis, it may require a full search in the state-space S.

2.1. The Dynamic of State-Space. In this study we propose an automatic approach to avoid a full search in the state-space (1) and still find a good setting of M to solve P. The idea is to consider the existence of an *agent*, whose *actions* modify the *state* of (1) [16]. In our approach this agent is a heuristic method, which combines robust statistical methods in an iterative process to create candidate configurations and find the good ones, based on statistical evaluations of a wide range of problems. An action (e.g., the creation of alternatives) is valid if it respects some constraints (e.g., the state-space bounds). Thus, given an initial state (e.g., the best known candidate configuration) we apply the heuristic method to explore (1) through the creation of candidate configurations at the neighborhood of a promising alternative.

The actions yielded by the heuristic method modify the state of (1) by creating candidate configurations on demand at the neighborhood of some best known alternative, as a sequence of sets of candidate configurations:

$$\Theta_0 \Longrightarrow \Theta_1 \Longrightarrow \Theta_2 \Longrightarrow \cdots. \tag{2}$$

Such actions can be thought as a directed graph in the state-space, where the nodes are the states and the arcs are the actions (Figure 1).

From step k to $k+1$ the set of candidate configurations is built possibly discarding some alternatives considered statistically inferior. Given that some candidate configurations persist in this set, they are evaluated on more instances. Therefore, finding a solution is equivalent to finding a path in (1); that from an initial state reaches the final state, that is, a good setting for M.

To illustrate this process (Figure 1), let us consider any state-space, where at each iteration k, $m = 3$ candidate configurations are created. At the end of iteration, all alternatives in the set Θ of candidate configurations are evaluated and those with inferior quality are discarded. Therefore, the set Θ is dynamic; that is, its size can increase or decrease. The process continues pursuing the alternatives in the search space until a stopping criterion (e.g., number of alternatives in Θ, runtime limit) is met.

2.2. Automatic Tuning of Metaheuristics. The tuning process begins with an arbitrary selection of n instances ($n > 1$) from a class of optimization problems and follows by defining ranges for the parameters of metaheuristic. The previously selected instances are treated as a training set, on which experimental studies are performed with the Response Surface Methodology (RSM) to define the best parameters settings for each instance. Therefore, at the end of the experimental phase there will be n different settings for each parameter, each one being related to an instance.

The settings identified in the training set ensure diversity for the parameters, and they are used to define the bounds of each parameter, that is, a search space of parameters limited by the maximum and minimum values of each parameter in the training set. Then, the goal is to pursue alternatives dynamically created at the neighborhood of some best known candidate configuration, with respect to the previously defined bounds of the search space. For each of the alternatives, the target algorithm is run in an expanded set of instances, larger than the previous one.

This process (Figure 2) is called Heuristic Oriented Racing Algorithm (HORA), due to the means of exploring the alternatives in the search space, that is, using a heuristic method, and by its evaluation process through a racing method.

3. Considered Problem and Metaheuristics: Overview

Optimization problems are common in many areas (e.g., science, engineering, management, and business) and different domains. Essentially, they involve finding an extreme value (maximum or minimum) called optimal, from a function with numerous local extremes, called objective function.

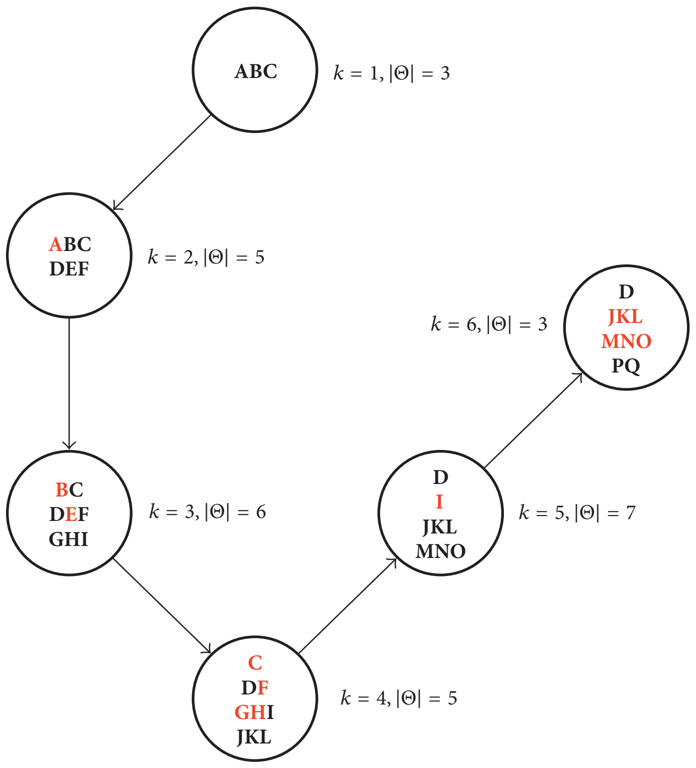


FIGURE 1: Illustrative process of creating (black) and excluding (red) alternatives in the state-space.

Some of those problems are classical in Operations Research area, such as the traveling salesman problem, and involve a significant number of publications in the specialized literature [17–20].

The traveling salesman problem (TSP) is a classical optimization problem, whose idea is to find the shortest route between a set of given cities, starting and ending at the same city, such that each city is 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 that $i \neq j$. The problem is classified as symmetric if $d_{ij} = d_{ji}$, $\forall i, j$, or asymmetric if $d_{ij} \neq d_{ji}$, $\forall i, j$ [20].

This problem is known to be NP-hard [21]; thus, in order to get the optimal solution, a significant computational 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 efficient algorithm. Many metaheuristics are inspired by metaphors from different knowledge areas, like physics (particle swarm optimization and Simulated Annealing) and biology (Genetic Algorithms and neural networks). 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) and 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.

Simulated Annealing (SA) is a probabilistic method proposed by Kirkpatrick et al. [22] and Černý [23] 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 [24]. On the other hand, Genetic Algorithm (GA) is a population-based method invented by Holland [25] inspired on principles of survival from Darwin's evolution theory. GA simulates an evolution process in which the fitness of individuals (parents) is crucial to generating new individuals (children).

The main difference between these algorithms is the searching methods that are implemented. SA performs constant movements between one solution (s) and another (s') according to some predefined neighborhood structure, and it uses a probabilistic test to accept new solutions, which sometimes allows low-quality solutions to be considered in the search process. On the other hand, GA operates on a population of solutions, whose 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.

Both SA and GA algorithms have a wide range of parameters (e.g., initial temperature and its rate of decrease, number of interactions, for SA, and rates of crossover and mutation, population size, for GA) that must be tuned before starting

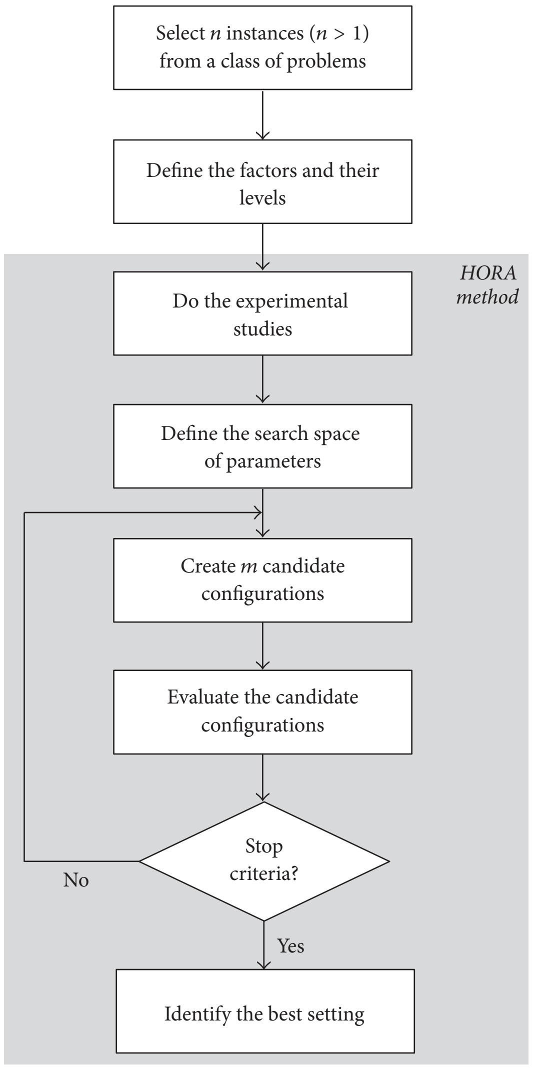

Select n instances ($n > 1$) from a class of problems
Define the factors and their levels
HORA method
Do the experimental studies
Define the search space of parameters
Create m candidate configurations
Evaluate the candidate configurations
Stop criteria?
No
Yes
Identify the best setting

Figure 2: Schema of the proposed approach.

to solve 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.

4. A Case Study

For our study, we selected a set of parameters of each metaheuristic, which are the most frequently used in the literature and seem to influence the performance of both SA and GA, regardless of the studied problem. The considered parameters for SA are value of the initial temperature (T_0), number of iterations on one temperature stage ($\mathrm{SA}_{\max}$), and temperature cooling rate (α), while the chosen parameters for GA are mutation rate (p_m), crossover rate (p_c), population size (μ), and number of generations (g). The parameters (Table 1) were chosen within the region of operability of parameters (their real limits), in order to promote diversity

Table 1: Parameters settings for GA and SA.

SA	Low	High
T_0	1.00e4	1.50e6
$\mathrm{SA}_{\max}$	500	1500
α	0.900	0.980
GA	Low	High
p_m	0.001	0.025
p_c	0.400	0.900
μ	10	100
g	100	1000

in the search space, as well as differences between each particular parameter setting.

The tuning process begins from a training set with $n = 4$ instances arbitrarily selected from the TSP benchmark from the TSPLIB [26]. The studies with DOE were conducted by means of a circumscribed Central Composite Design (CCD) [8], in which some points overcome the previously set limits to ensure an appropriate estimation for the parameters.

Thus, after the experimental studies we have four different results for each parameter, each one being related to an instance. From those results, we defined the search space of parameters, whose limits are the maximum and minimum values of the parameters in the training set. Accordingly, the SA search space is

(i) T_0: $[1.35e5, 1.45e5]$;

(ii) $\mathrm{SA}_{\max}$: $[1466, 1859]$;

(iii) α: $[0.946, 0.947]$.

In the case of GA, we have the following search space:

(i) p_m: $[0.012, 0.027]$

(ii) p_c: $[0.292, 0.774]$

(iii) μ: $[55, 122]$

(iv) g: $[494, 1474]$

Observing the results is noteworthy, where some parameter values are outside of the limits initially defined (Table 1). However, as pointed out before, this occurs due to the chosen CCD, whose axial points establish new limits for a region of interest.

From there, the exploration of the state-space is conducted by means of the HORA heuristic creating the alternatives at the neighborhood of some best known candidate configuration. For each of the alternatives we run the target metaheuristics (e.g., SA and GA) during 15 seconds on an expanded set of instances (e.g., for this study, the expanded set matches 40 instances from the benchmark TSP). This process was repeated 10 times and the results of fine-tuning of the metaheuristics by means of HORA are presented in terms of mean and standard deviation ($\mu \pm \sigma$) in Table 2. This table also presents the total time (in seconds) of the tuning process.

For comparisons, we considered the previously defined search space of parameters and one other fine-tuning approach, such as a racing algorithm based on F-Race

Table 2: Fine-tuning of metaheuristics (HORA).

SA	Settings
T_0	$1.40e5 \pm 2445$
SA_{max}	1584 ± 106
α	0.947 ± 0.000
—	—
t	969
GA	Settings
p_m	0.020 ± 0.005
p_c	0.665 ± 0.123
μ	81 ± 12
g	760 ± 202
t	896

Table 3: Fine-tuning of metaheuristics (*Racing*).

SA	Settings
T_0	$1.39e5 \pm 2411$
SA_{max}	1672 ± 112
α	0.947 ± 0.000
—	—
t	8338
GA	Settings
p_m	0.020 ± 0.007
p_c	0.549 ± 0.164
μ	84 ± 14
g	690 ± 217
t	16740

method [10, 12, 14], from hereon called *Racing*. The settings used for SA are the following: $T_0 = \{1.35e5, 1.37e5, 1.38e5, 1.40e5, 1.42e5, 1.43e5, 1.45e5\}$, $SA_{max} = \{1466, 1564, 1662, 1761, 1859\}$, and $\alpha = \{0.946, 0.947\}$; and for GA we have $p_m = \{0.012, 0.020, 0.027\}$, $p_c = \{0.292, 0.453, 0.613, 0.774\}$, $\mu = \{55, 77, 100, 122\}$, and $g = \{494, 821, 1147, 1474\}$.

These settings were defined in a discrete interval limited by the previously defined search space, with a number of levels that seem to be enough for the algorithms leading to a good result. Thus, each possible combination of parameters leads to a different algorithm setting, such that the search space is composed of 70 and 192 different parameter settings for SA and GA algorithms, respectively. The idea is to use the *Racing* to select a possible good configuration from a number of options, after running the target algorithms for 15 seconds on the same set of instances previously used by HORA. This process was also repeated 10 times and the fine-tuning results of the studied metaheuristics (Table 3) are presented in terms of mean and standard deviation ($\mu \pm \sigma$). The Table also presents the total time (t, in seconds) of this tuning process.

From these results (Table 3) it is possible to note the similarities between HORA and *Racing* in terms of parameter settings, since both approaches employ the same evaluation method for the candidate configurations. But it is possible to highlight that the tuning process using HORA spends about 11.6% and 5.3% of the time required in the fine-tuning process using *Racing*, for SA and GA, respectively.

Table 4: Statistics of the fine-tuning approaches for the metaheuristic SA.

Inst.	gap_H	t_H	gap_R	t_R
Berlin52	0.00	98	0.00	60
St70	1.48	190	1.63	178
Rat99	2.73	260	5.53	243
Rd100	1.78	263	3.67	245
Lin105	0.83	266	1.96	256
Pr124	1.66	285	1.84	271
Bier127	3.12	293	3.40	292
Ch130	2.39	297	4.66	299
KroA200	7.91	300	8.49	299
Ts225	3.23	300	5.32	299
μ	*2.51*	*255*	*3.65*	*244*
σ	*2.04*	*61*	*2.33*	*71*

4.1. Analysis of Results. Although it is not the main purpose of this paper, it is interesting to verify whether the metaheuristics tuned by HORA and *Racing* achieve good results for the TSP. Note that the quality of the results yielded by an algorithm configured by *Racing* can depend heavily on the number of candidate configurations initially established, since this approach performs an exhaustive analysis of the search space of previously defined candidate configurations. Thus, the smaller the candidate configurations space, the greater the probability that the tuning process will finish with a poor parameters setting, compromising the quality of the results (and, even, the execution time) of an algorithm which has been tuned by this process. For HORA approach, this limitation does not exist, since it starts from any candidate configuration and dynamically builds the space of parameters settings.

In general, the metaheuristics employ some degree of randomness to diversify the searches and avoid confinement in the search space. Thus, a single run of these algorithms can result in different solutions from the next run. So, to test the quality of the settings, the experimental results were collected after 5 runs of the metaheuristics SA and GA on the TSP. To compare the results, we use

$$\text{gap} = \frac{f(s) - f(s^*)}{f(s^*)} \times 100, \quad (3)$$

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

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

The results were collected considering the settings of the metaheuristics SA and GA presented in Tables 2 and 3, for HORA and *Racing*, respectively. Tables 4 and 5 present the results for 5 runs of SA and GA, in 10 instances of the TSP

Table 5: Statistics of the fine-tuning approaches for the metaheuristic GA.

Inst.	gap_H	t_H	gap_R	t_R
Berlin52	44.60	109	34.83	52
St70	72.15	103	74.67	69
Rat99	106.69	123	106.69	82
Rd100	116.94	114	114.80	74
Lin105	143.98	95	150.88	61
Pr124	194.38	110	205.06	60
Bier127	99.38	100	88.84	63
Ch130	142.36	125	130.70	103
KroA200	227.62	125	216.68	103
Ts225	245.84	112	282.68	74
μ	*139.39*	*109*	*140.58*	*52*
σ	*62.33*	*11*	*70.97*	*17*

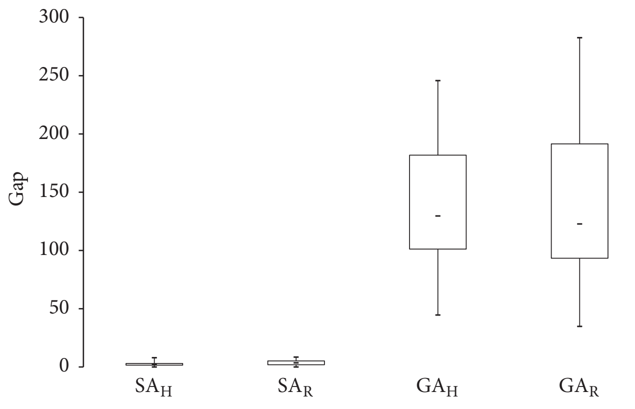


Figure 3: Performance of the studied metaheuristics tuned by different approaches.

benchmark, with the following stopping criteria: number of iterations without changes in the objective function (200 iterations) and maximum running time (300 seconds). In these tables, the results of the approaches are underlined in capital letters H (for HORA) and R (for *Racing*). The column *gap* is the best found value of (3), t is average of the runtime (t), and μ and σ are arithmetic mean and standard deviation of all selected instances.

From the results for the metaheuristic SA (Table 4) we note that the fine-tuning process by means of HORA achieves solutions with better quality. Note that the *gap* of SA tuned by HORA is, on average, about 30% better than the *gap* of the corresponding version tuned by *Racing*, although the running time for SA_H is, on average, about 4.5% greater than for SA_R. This deficit of performance may be related to the parameter T_0 (value of the initial temperature), which is slightly higher according to HORA and can require more time to find a good quality solution.

For the results achieved by GA (Table 5), we also note slightly better quality solutions when the metaheuristic is tuned by HORA, although this version demands, on average, the double of the running time than its corresponding version tuned by *Racing*.

The graphical analysis (Figure 3) reveals differences between the data distributions from the studied metaheuristics, such that the results of SA figure are closer to the optimal solution. According to the results (Tables 4 and 5) and supported by graphical analysis it is possible to confirm that the fine-tuning process by means of HORA can achieve better quality solutions.

5. Final Considerations

This paper presented a method addressed to the problem of tuning metaheuristics. The problem was formalized as a state-space, whose exploration is conducted effectively by a heuristic method combining Statistical (DOE) and Artificial Intelligence (racing algorithms).

The proposed method, called HORA, applies robust statistics on a limited number of instances from a class of problems, in order to define a search space of parameters. Thus, from the alternatives dynamically created at the neighborhood of some best known candidate configuration, it employs a racing method to consistently find the good settings.

From a case study, HORA was applied in the fine-tuning of two distinct metaheuristics. Its results were compared with the same metaheuristics tuned by means of a racing algorithm. The HORA method proved to be effective in terms of overall time of the tuning process, since it demands just a fraction of the time required by the other studied approach. Through the experimental studies it is noted that the metaheuristics SA and GA tuned by HORA can achieve similar results (eventually better) from those obtained by the *Racing* approach, but it is highlighted that the tuning process is much faster with HORA. This better performance can be explained by how the alternatives in the state-space are explored, which consists in creating alternatives at the neighborhood of some best known candidate configuration, and evaluate them by a racing method.

The aim of this study was to present the proposed approach and verify its effectiveness when applied on different metaheuristics. Our results suggest that HORA may be a promising and powerful tool to assist in the fine-tuning of different algorithms. Some additional studies can be conducted to verify its effectiveness considering other metaheuristics and problems, as well as explore other heuristic alternatives to pursue good configurations in the state-space of parameters settings.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

References

[1] F. Dobslaw, "A parameter tuning framework for metaheuristics based on design of experiments and artificial neural networks," in *Proceedings of the 6th International Conference on Natural Computation*, vol. 6, pp. 1–6, Cairo, Egypt, 2010.

[2] S. Lessmann, M. Caserta, and I. M. Arango, "Tuning metaheuristics: a data mining based approach for particle swarm optimization," *Expert Systems with Applications*, vol. 38, no. 10, pp. 12826–12838, 2011.

[3] C. Neumüller, S. Wagner, G. Kronberger, and M. Affenzeller, "Parameter meta-optimization of metaheuristic optimization algorithms," in *Proceedings of 13th International Conference on Computer Aided Systems Theory (EUROCAST 2011)*, vol. 13, pp. 367–374.

[4] J. Ries, P. Beullens, and D. Salt, "Instance-specific multi-objective parameter tuning based on fuzzy logic," *European Journal of Operational Research*, vol. 218, no. 2, pp. 305–315, 2012.

[5] H. Akbaripour and E. Masehian, "Efficient and robust parameter tuning for heuristic algorithms," *International Journal of Industrial Engineering and Production Research, Tehran*, vol. 24, no. 2, pp. 143–150, 2013.

[6] M. Amoozegar and E. Rashedi, "Parameter tuning of GSA using DOE," in *Proceedings of 4th International Conference on Computer and Knowledge Engineering ICCKE 2014*, pp. 431–436, October 2014.

[7] L. Calvet, A. A. Juan, C. Serrat, and J. Ries, "A statistical learning based approach for parameter fine-tuning of metaheuristics," *SORT (Statistics and Operations Research Transactions)*, vol. 40, no. 1, pp. 201–224, 2016.

[8] D. C. Montgomery, *Design and Analysis of Experiments*, John Wiley & Sons, NJ, USA, 8th edition, 2012.

[9] O. Maron and A. W. Moore, "Hoeffding races: accelerating model selection search for classification and function approximation," in *Advances in Neural Information Processing Systems*, pp. 59–66, Morgan Kaufmann, San Mateo, Calif, USA, 1994.

[10] M. Birattari, T. Stützle, L. Paquete, and K. Varrentrapp, "A racing algorithm for configuring metaheuristics," in *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 11–18, New York, NY, USA, 2002.

[11] 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.

[12] P. Balaprakash, M. Birattari, T. Stützle, and M. Dorigo, "Improvement strategies for the F-race algorithm: sampling design and iterative refinement," in *Proceedings of the 4th International Workshop on Hybrid Metaheuristics*, pp. 108–122.

[13] F. Hutter, H. H. Hoos, K. Leyton-Brown, and T. Stützle, "ParamILS: an automatic algorithm configuration framework," *Journal of Artificial Intelligence Research*, vol. 36, pp. 267–306, 2009.

[14] M. Birattari, *Tuning Metaheuristics: A Machine Learning Perspective*, Springer, New York, NY, USA, 2nd edition, 2009.

[15] S. Smit and A. Eiben, "Comparing parameter tuning methods for evolutionary algorithms," in *Proceedings of the IEEE Congress on Evolutionary Computing*, pp. 399–406, IEEE, Trondheim, Norway, May 2009.

[16] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, Pearson, London, UK, 3rd edition, 2009.

[17] D. L. Applegate, R. E. Bixby, V. Chvátal, and W. J. Cook, "The traveling salesman problem: a computational study," in *Princeton Series in Applied Mathematics*, Princeton University Press, Princeton, NJ, USA, 2nd edition, 2007.

[18] G. Laporte, "What you should know about the vehicle routing problem," *Naval Research Logistics*, vol. 54, no. 8, pp. 811–819, 2007.

[19] U. Derigs, *Optimization and Operations Research*, vol. 2, EOLSS Publishers Co. Ltd., Paris, France, 2009.

[20] 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., Chapter 1, pp. 1–24, InTech, 2010.

[21] J. K. Lenstra, A. H. G. Rinnooykan, and P. Brucker, "Complexity of machine scheduling problems," in *Annals of discrete mathematics*, P. L. Hammer, E. L. Johnson, B. H. Korte, and G. L. Nemhauser, Eds., pp. 343–362, Elsevier, Amsterdam, 1977.

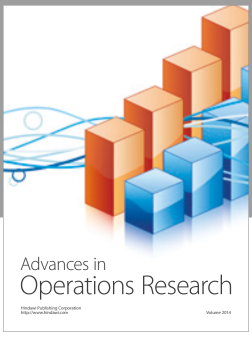
[22] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *American Association for the Advancement of Science. Science*, vol. 220, no. 4598, pp. 671–680, 1983.

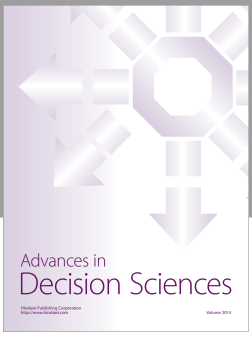
[23] V. Černý, "Thermodynamical approach to the traveling salesman problem: an efficient simulation algorithm," *Journal of Optimization Theory and Applications*, vol. 45, no. 1, pp. 41–51, 1985.

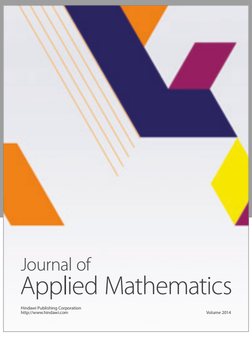
[24] D. Bertsimas and J. Tsitsiklis, "Simulated annealing," *Statistical Science*, vol. 8, no. 1, pp. 10–15, 1993.

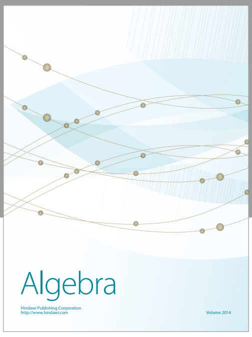
[25] J. H. Holland, *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Boston, Mass, USA, 1975.

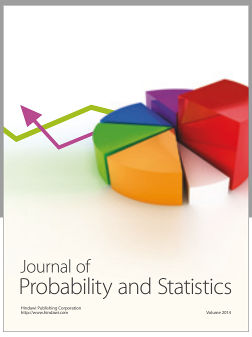
[26] G. Reinelt, "TSPLIB: a traveling salesman problem library," *ORSA Journal on Computing*, vol. 3, no. 4, pp. 376–384, 1991.


Advances in Operations Research


Advances in Decision Sciences


Journal of Applied Mathematics

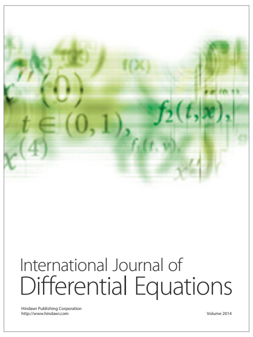

Algebra


Journal of Probability and Statistics

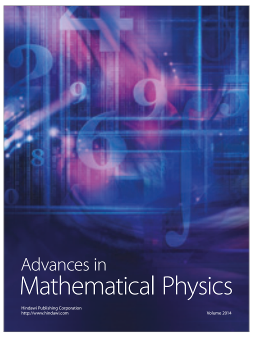
The Scientific World Journal

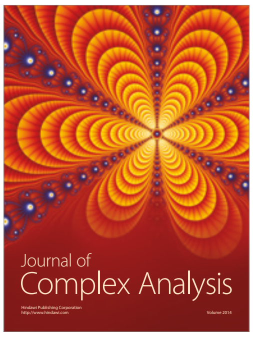
Hindawi

Submit your manuscripts at
https://www.hindawi.com

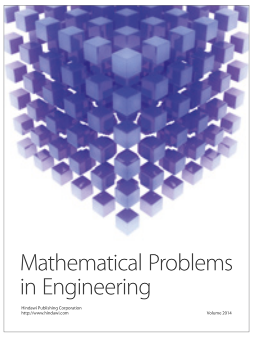

International Journal of Differential Equations

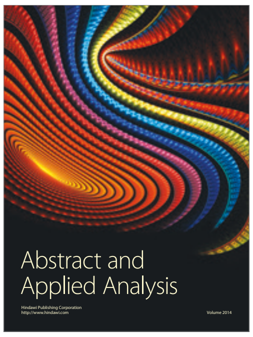
International Journal of Combinatorics

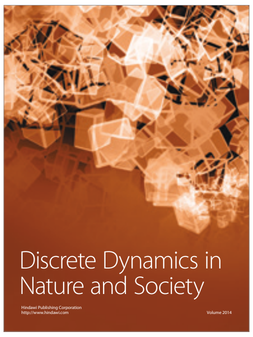

Advances in Mathematical Physics

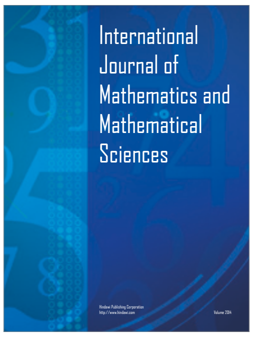

Journal of Complex Analysis

Journal of Mathematics


Mathematical Problems in Engineering

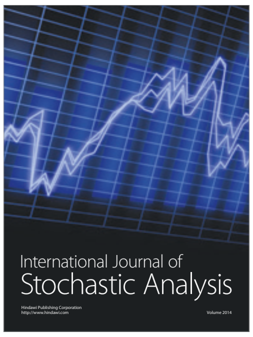

Abstract and Applied Analysis


Discrete Dynamics in Nature and Society


International Journal of Mathematics and Mathematical Sciences

Journal of Discrete Mathematics

Journal of Function Spaces


International Journal of Stochastic Analysis

Journal of Optimization