Computing and Informatics, Vol. 38, 2019, 265–290, doi: 10.31577/cai_2019_2_265

SPECIFIC PARAMETER-FREE GLOBAL OPTIMIZATION TO SPEED UP SETTING AND AVOID FACTORS INTERACTIONS

Rafael Rodríguez-Reche, Rocío P. Prado
Sebastián García-Galán, José Enrique Muñoz-Expósito
Nicolás Ruiz-Reyes

Telecommunication Engineering Department
University of Jaén
Science and Technology Campus of Linares
Jaén, Spain
e-mail: {rrreche, rperez, sgalan, jemunoz, nicolas}@ujaen.es

Abstract. Meta-heuristics utilizing numerous parameters are more complicated than meta-heuristics with a couple of parameters for various reasons. In essence, the effort expected to tune the strategy-particular parameters is far more prominent as the quantity of parameters increases and furthermore, complex algorithms are liable for the presence of further parameter interactions. Jaya meta-heuristic does not involve any strategy-specific parameters and is a one-stage technique. It has demonstrated its effectiveness compared to major types of meta-heuristics and it introduces various points of interest, such as its easy deployment and set-up in industrial applications and its low complexity to be studied. In this work, a new meta-heuristic, Enhanced Jaya (EJaya) is proposed to overcome the inconsistency of Jaya in diverse situations, introducing coherent attraction and repulsion movements and restrained intensity for flight. Comparative results of EJaya in a set of benchmark problems including statistical tests show that it is feasible to increase the accuracy, scalability and exploitation capability of Jaya while keeping its specific parameter-free feature. EJaya is especially suitable for a priori undefined characteristics optimization functions or applications where the set-up time of the optimization process is critical and parameters tuning and interactions must be avoided.

Keywords: Soft computing, optimization, meta-heuristics, parameters complexity, parameters interactions

Mathematics Subject Classification 2010: 90-08

1 INTRODUCTION

Meta-heuristics, in their genuine definition, are solution techniques that organize an association between local solution methods and more complex methodologies to make a procedure able to escape from local optima and playing out a vigorous inquiry of a solution space [8]. Over time, these techniques have incorporated new methodologies to avoid getting trapped on local optima in complex search spaces, particularly those strategies that use at least one neighbourhood structure as a method for characterizing permissible movements to change from a solution to the next, or to implement or destroy solutions in diverse applications.

Some instruments and strategies that have risen up out of research in meta-heuristic techniques have ended up being surprisingly viable, to such an extent that meta-heuristics have moved into the spotlight lately as a favoured line of assault for facing numerous sorts of complex issues, especially those of a combinatorial nature, for example, optimization in robotics [16], cloud and grid computing [6, 7], energy consumption [3], bioinformatics [18], manufacturing planning and scheduling [27], image processing [17], filter modelling [1], etc. While meta-heuristics cannot confirm the optimal character of their solutions, exact methods (which hypothetically can give such an accreditation, if permitted to run long enough) have generally demonstrated to be unable to discover solutions whose quality is near of that provided by the main meta-heuristics, especially for real-world applications, which frequently present a more complex nature. Additionally, a portion of the more effective uses of exact techniques has come to fruition with consolidating meta-heuristic methodologies inside them. These results have propelled further research and utilization of novel and enhanced meta-heuristic procedures.

It is known that there does not exist a single strategy to achieve the more efficient solution for all optimization problems, also explained as the no "free lunch" theorems for optimization [24], and since the characteristics of the problems at hand are unknown a priori in many real applications, the selection of the optimization strategy could be arbitrary. Nevertheless, even considering that the selected strategy is the most suitable for the problem in hand, generally a tuning process must be considered to adjust the value of its control parameters, what delays or even makes not feasible the consideration of optimization processes to increase the system's efficiency. All meta-heuristics are probabilistic strategies that make use of basic control parameters such as population size, number of dimensions, etc. Beyond the regular control parameters, most strategies use particular control parameters. In this way, Genetic Algorithms (GA) consider mutation, crossover and selection rates [8, 9, 28], Particle Swarm Optimization (PSO) requires the specification of inertia weight, social and cognitive controlling factors [10, 13], Artificial Bee Colony (ABC) [11, 12] must define the amount of onlooker, employed and scout bees and also, a bound factor, Harmony Search (HS) makes use of memory and pitch adjusting rates, and the amount of improvisations. Likewise, different algorithms, for example, Differential Evolution (DE), Heat Transfer Search (HTS), Biogeography-Based Optimization (BBO), Adaptive Segregational Constraint Handling Evolutionary Algorithm (AS-

CHEA), etc., require the tuning of strategy-particular parameters [8, 19, 20, 23]. The tuning of the strategy-particular parameters is an exceptionally significant aspect which highly influences the successful execution of most meta-heuristics. An inefficient tuning of strategy-particular parameters either increases the computational cost or offers local optimal solutions. Henceforth, if two meta-heuristics generate comparable results but, however, one is fundamentally less complex than the other, then the simplest is a better strategy [8]. Less complex algorithms have various points of interest, including being easy to deploy and set up in an industrial setting and being less complex to be studied.

Jaya [20] is an extremely simple algorithm, whose main virtue is given by the fact that it is not necessary to configure any control parameter to make it work beyond the own associated to the problem to be solved (e.g., delimitation of the search space, number of variables, fitness function, etc.), which makes it especially suitable for problems where characteristics are unknown a priori. Jaya efficiency has been tested in a diverse test-bed of benchmark functions [20] and the outcomes have been compared to the accomplishments of major types of algorithms, such as GA [9], PSO [13], DE [23], ABC [11] and recent simple meta-heuristics such as Teaching-Learning-Based Optimization (TLBO) [2, 21]. Results show the satisfactory performance of Jaya in a wide range of optimization problems and statistical tests additionally accredit the success of this technique. As stated before, there may not exist a better algorithm for all different types of applications, but Jaya emerges as a competitive strategy to be considered in the field of optimization. What can be stated with certainty is that Jaya is a strategy easy to implement, it requires no strategy-particular parameters and it gives the optimal solutions with slightly less time complexity and exactly the same computational complexity than major types of meta-heuristics such as the ones cited above. Thus, the optimization research community is urged to make changes to Jaya in a way that the strategy can turn out to be a great deal more accurate with a more efficient performance [20].

In this work, Jaya is redefined and the new proposal is called EJaya. EJaya overcomes Jaya inconsistency in diverse conditions through the introduction of coherent attraction and repulsion movements and restrained intensity for flight. The proposal is tested considering a wide range of benchmark problems from the Congress on Evolutionary Computation (CEC) [15]. The field of meta-heuristics, included within what is known as Evolutionary Computation, has its own space within CEC annually where the latest developments are discussed on this matter and a number of objective functions or benchmark functions to be optimized by meta-heuristics algorithms are presented as well as modifications to traditional functions in optimization problems, each with a number of features that have different impacts on the performance of meta-heuristics. Thus, benchmark functions from the CEC for the single objective real-parameter numerical optimization 2014 [15] are considered to test the enhanced strategy performance. Results including statistical tests show that it is feasible to increase the accuracy, scalability and exploitation capability of Jaya while keeping its control parameter-free feature, what makes it especially suitable for a priori undefined characteristics optimization functions or applications

where the set-up time of the optimization is critical and tuning and interactions of parameters must be avoided.

This paper is organized as follows. In Section 2, an analysis of the complexity of meta-heuristics and related works are presented. Next, the fundamentals and accomplishments of Jaya are presented in Section 3 and the proposed meta-heuristic EJaya is explained in Section 4. In Section 5, the experimental results are presented and discussed and finally, in Section 6, the main conclusions of the work are drawn.

2 BACKGROUND

Various measures of complexity exist for meta-heuristics [8]. Some measurements incorporate the quantity of phases of pseudo-code expected to depict the strategy or the quantity of lines of program expected to execute the meta-heuristic. In any case, this kind of measurements are not especially helpful, as they differ in the light of the programming language, the style and the level of the description of the pseudo-code. A more relevant measure for the complexity nature of a meta-heuristic is the quantity of parameters utilized as a part of the strategy. Parameters can be defined as the adaptable factors of a meta-heuristic that can be tuned to adjust its execution. They can be specified statically (e.g., selection rate of 0.4) or depending on the specific case (e.g., selection rate of 0.01 n, where n is the number of variables to be optimized for each individual in the population). In both of these cases, the steady estimation of the parameter or its relation to other factor in the problem in hand must be specified a priori by the strategy designer.

Most sorts of algorithms consider various specific parameters to be determined before their execution.

Table 1 presents fundamental parameters required for main types of strategies. Although these are just examples to show some typical specific parameters in various sorts of algorithms, most meta-heuristics need a diverse amount of parameters. For example, Tabu Search (TS) technique can only have one parameter, the Tabu rundown length. Nevertheless, in [26] TS in the vehicle routing issue utilizes 32 parameters. Similarly, algorithms can require less than the "base" amount of parameters by joining parameters with similar esteem. For example, the GA strategy for the spanning tree problem [25] utilizes only one specific parameter, which lets both control the population size and fix the end criteria.

Meta-heuristics utilizing numerous parameters present further complexity than the methodologies with a couple of parameters for various reasons. To begin with, the cost expected to set up and comprehend a wide set of parameters is far more prominent as the quantity of parameters increases. A brute force strategy aiming to tune m parameter values for each of the n parameters in the problem in hand, must test m^n combinations for each problem instance. Let us consider that three values can be assigned to each parameter of a strategy which requires the use of two parameters and seven parameters. In the first case, this would mean 9 evaluations and in the second case, $3^7 = 2\,187$. If this number could be considered viable, the eval-

Meta-Heuristics	**Specific Control Parameters**
Genetic Algorithms	Mutation probability
	Crossover probability
	Selection operator
Particle Swarm Optimization	Inertia weight
	Social parameter
	Cognitive parameter
Differential Evolution	Mutation rate
	Recombination rate
Artificial Bee Colony	Onlooker bees
	Employed bees
	Scout bees
	Limit
Harmony Search	Distance bandwidth
	Memory size
	Pitch adjustment rate
	Rate of choosing from memory

Table 1. Specific control parameters for major types of meta-heuristics

uations required for a 32 parameter meta-heuristic, $3^{32} = 1\,853\,020\,188\,851\,841$, are not feasible in most real-world applications. Moreover, the number of possibilities for strategies with a higher number of parameters increases exponentially, making the set-up of algorithms much harder. Despite the fact that there are approaches to search for good combinations of parameters, the number of options still increases with the quantity of parameters, which means that the set up is much more troublesome. Greater quantities of parameters additionally make the understanding of the optimization process much more difficult.

Furthermore, the complexity in setting up it is not the only drawback of complex meta-heuristics. A major problem is given by the greater possibilities of a vast parameter set to present complicated parameter interactions. Complex interactions of parameters can derive in, for example, finding numerous local solutions. In general, it is proved that the optimization of parameters independently or in small sets is not effective and the problem becomes harder as the number of parameters increases. There exist a number of works related to parameter interactions. For example, in [4] the researchers could appreciate non-trifling interactions in GA considering just three parameters. It was observed that the adequacy of a given parameter combination is frequently subject to the problem and functions to be optimized and so, it was difficult to classify and automatize the analysis of interactions. Thus, it is frequently extremely hard to keep away from parameter interactions and the level of these interactions increases drastically with the quantity of parameters once again. This has also aimed research in optimization. The recently presented TLBO [2, 20, 21] and Jaya [20] meta-heuristics do not use any strategy-particular parameters. However, it must be noted that TLBO requires two differentiated stages (i.e., educator and

learner stages), whereas Jaya considers only one stage and thus, it is more straightforward to use. Furthermore, as shown in the next section, results for Jaya in a wide range of benchmark functions provide better results than TLBO.

Hence, considering the relevance of the parameters in the performance of meta-heuristics, it is important to propose and analyse new ways of reducing the complexity of meta-heuristics while offering good solutions and this work represents a new effort in this sense.

3 FUNDAMENTALS AND ACCOMPLISHMENTS OF JAYA

Jaya could be defined as a swarm type strategy in which the attraction to the best local is eliminated and replaced by a repulsion to the worst particle in the population [20]. Moreover, the inertial weight of the particles is removed: its value is fixed to the unity and it is not modified at any time during the execution of the algorithm. In addition, particles have no memory: they do not keep a record of the best solution found, neither globally nor locally (i.e., the best position of the swarm and the best position of the particle, respectively), because the particles do not move from their position if they do not find a better solution in the next iteration. Thus, interaction with the best local position of the particle is not considered, but instead a new relationship is added: a movement of escape from the worst position within the swarm in the current iteration. On the other hand, the movement of approach to the best particle in the population is preserved. The intensity of both movements, attraction and repulsion, depends solely on the distance between the particle that makes the movement and those that affect it, that is, the best and worst positions in the swarm, respectively. Hence, there are no adjustment parameters to tune the exploration of the search space (such as c_1 and c_2 in most swarm-based strategies [14]).

Next, the algorithm Jaya is formulated formally. Consider $f(x)$ the objective function to be optimized, m the number of design factors or variables (i.e., $j = 1, 2, \ldots, m$) and n the population size (i.e., $k = 1, 2, \ldots, n$). Also, consider the best individual of the population to be the candidate solution obtaining the current optimal result of $f(x)$ (i.e., $f(x)_{best}$) and, analogously, the worst candidate to be the individual obtaining the current optimal result of $f(x)$ (i.e., $f(x)_{worst}$) in the population. If $X^t_{j,k}$ represents the value of the j^{th} factor or variable for the k^{th} solution during the t^{th} iteration, then this value is modified by following Equation (1):

$$X^{t+1}_{j,k} = X^t_{j,k} + r_{1,j,t}\left(X^t_{j,best} - |X^t_{j,k}|\right) - r_{2,j,t}\left(X^t_{j,worst} - |X^t_{j,k}|\right) \tag{1}$$

where $X^t_{j,best}$ is the value of the j^{th} factor of the best individual, $X^t_{j,worst}$, is the value of the j^{th} variable of the worst candidate and $X^{t+1}_{j,k}$ is the updated value of $X^t_{j,k}$, $r_{1,j,t}, r_{2,j,t}$ being random numbers in the range [0, 1]. The term $r_{1,j,t}\left(X^t_{j,best} - |X^t_{j,k}|\right)$ shows the leaning of the candidate to move towards the best solution or attraction factor $AF^t_{j,k}$ and the term $r_{2,j,t}(X^t_{j,worst} - |X^t_{j,k}|)$ indicates the leaning of the solution

to go away from the worst solution or rejection factor $RF^t_{j,k}$. $X^{t+1}_{j,k}$ performance is tested. All particles whose modification represents an improvement in the final results are kept up and they are considered in the following steps of the strategy. Jaya algorithm is detailed in Algorithm 1. As observed, the algorithm continuously tries to get nearer to the best solution and tries to maintain a strategic distance from the worst solution. Jaya aims to wind up successful in achieving the best solution and thus, it is named Jaya (i.e., triumph). These characteristics provide Jaya a strong convergent behaviour mainly due to its large exploration capacities and the elimination of the possibility for particles to move towards positions offering worse results than the current solution.

Algorithm 1 Jaya pseudo-code

```
-- Data
N: Number of individuals
D: Number of dimensions
-- Algorithm
Population initialization
while !end condition do
    Find (X^t_{j,best}) and worst (X^t_{j,worst}) individual in the population
    for k ¡ N do
        for j ¡ D do
            X^{t+1}_{j,k} = X^t_{j,k} + r_{1,j,t} (X^t_{j,best} - |X^t_{j,k}|) - r_{2,j,t} (X^t_{j,worst} - |X^t_{j,k}|)
        end for
        if Better solution found over particle's actual solution then
            Update particle's solution
        else
            Preserve previous particle's solution
        end if
    end for
end while
```

In [20], Jaya is evaluated in a test-bed of well-known benchmark functions in the optimization literature with diverse features like unimodality and multimodality, separability and non-separability, regularity and non-regularity, etc., where the quantity of design factors and their extents are diverse for every case. To assess the execution of the proposed algorithm based on Jaya in this work, the outcomes obtained by Jaya are contrasted to the outcomes of diverse optimization meta-heuristics, such as GA, PSO, DE, ABC and TLBO. This selection of strategies for comparison considers both the more competitive and well-known strategies (e.g., GA, PSO and DE) and the more recent and simple optimization meta-heuristics (e.g., ABC and TLBO) of those extensively analysed in [20]. Obtained mean results are reproduced in Table 2 for the diverse strategies.

From Table 2 it is observed that the results of Jaya are either equal or more accurate than the rest of the strategies in the test-bed. This can be further proved considering statistical tests. The p-value of Friedman Ranks test of these results is

f	GA	PSO	DE	ABC	TLBO	Jaya
Sphere	1.11e+03	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00
SumSquares	1.48e+02	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00
Beale	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00
Easom	−1.00e+00	−1.00e+00	−1.00e+00	−1.00e+00	−1.00e+00	−1.00e+00
Matyas	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00
Colville	1.49e−02	0.00e+00	4.09e−02	9.29e−02	0.00e+00	0.00e+00
Trid 6	−4.99e+01	−5.00e+01	−5.00e+01	−5.00e+01	−5.00e+01	−5.00e+01
Trid 10	1.93e−01	0.00e+00	0.00e+00	0.00e+00	0.00e+00	−2.10e+02
Zakharov	1.33e−02	0.00e+00	0.00e+00	2.47e−04	0.00e+00	0.00e+00
Schwefel 1.2	7.40e+03	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00
Rosenbrock	1.96e+05	1.50e+01	1.82e+01	8.87e−02	1.62e−05	0.00e+00
Dixon-Price	1.22e+03	6.67e−01	6.67e−01	0.00e+00	6.67e−01	0.00e+00
Foxholes	9.98e−01	9.98e−01	9.98e−01	9.98e−01	9.98e−01	9.98e−01
Branin	3.97e−01	3.97e−01	3.97e−01	3.97e−01	3.97e−01	3.97e−01
Bohachevsky 1	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00
Booth	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00
Michalewicz 2	−1.80e+00	−1.57e+00	−1.80e+00	−1.80e+00	−1.80e+00	−1.80e+00
Michalewicz 5	−4.64e+00	−2.49e+00	−4.68e+00	−4.68e+00	−4.67e+00	−4.68e+00
Bohachevsky 2	6.829e−02	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00
Bohachevsky 3	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00
GoldStein-Price	5.87e+00	3.00e+00	3.00e+00	3.00e+00	3.00e+00	3.00e+00
Perm	3.02e−01	3.60e−02	2.40e−02	4.11e−02	6.76e−04	0.00e+00
Hartman 3	−3.86e+00	−3.63e+00	−3.86e+00	−3.86e+00	−3.86e+00	−3.86e+00
Ackley	1.46e+01	1.64e−01	0.00e+00	0.00e+00	0.00e+00	0.00e+00
Penalized 2	1.25e+02	7.67e−03	2.19e−03	0.00e+00	2.34e−08	0.00e+00
Langerman 2	−1.08e+00	−6.79e−01	−1.08e+00	−1.08e+00	−1.08e+00	−1.08e+00
Angerman 5	2.87e−01	2.13e−01	0.00e+00	2.08e−04	1.55e−05	−1.24e+00
Langerman 10	−0.63e−01	−2.566e−03	−1.05e+00	−4.46e−01	−6.49e−01	−6.20e−01
FletcherPowell 5	4.30e−03	1.45e+03	5.98e+00	1.73e−01	2.20e+00	1.59e−04
FletcherPowell 10	2.95e+01	1.36e+03	7.81e+02	8.23e+00	3.59e+01	5.43e−04

Table 2. Comparative results of Jaya with major types of meta-heuristics (mean)

1.475e−06, indicating that there are significant differences between the algorithms (considering a level of significance of p-value = 0.05). Moreover, in Table 3 the Friedman Ranks Post-Hoc test for Jaya against GA, PSO, DE, ABC and TLBO is presented for peer analysis based on data in Table 2. As shown, Jaya algorithm presents a statistically significant improvement over GA and PSO and no differences are statistically significant for DE, ABC and TLBO. However, considering that Jaya is a much simpler algorithm than DE, ABC and TLBO, it can be inferred that Jaya provides equally good results, but with the difference of its noticeable greater simplicity. Therefore, it can be said that Jaya is a better algorithm, what justifies research to reach improvements like the ones presented in this work.

Also, it is important to highlight that the competitive results of Jaya in comparison to major types of optimization algorithms are achieved in fair conditions in terms of computational effort. As known, the complexity of meta-heuristics can be measured based on two different criteria. Time and computational complexity. On the one hand, time complexity is based on the overall time taken by the different steps of the algorithm involving the generation of initial population, updating solution, etc. In short, Jaya is a swarm-type optimization algorithm directly derived from PSO in which the attraction to the best local particle is eliminated and replaced by a repulsion to the worst particle in the population, and in which the consideration of the inertial weight in the update of particles and the step for its modification is removed. Furthermore, particles have no record of the best solution found and no adjustment parameters (denoted as c_1 and c_2 in PSO) in most swarm-based strategies are considered. Hence, as a whole, the total time complexity is slightly reduced from the canonical PSO in which it is based. On the other hand, the computational complexity refers to the number of function evaluations required to achieve the final result. Also, in [20], it is shown that to test the performance of Jaya compared with the results obtained by the other optimization algorithms such as GA, PSO, DE, ABC and TLBO it is done considering the exactly same amount of function evaluations for the different meta-heuristics and the process is repeated 30 times for each algorithm and benchmark function. Thus, the consistency in the comparison in time and complexity effort is kept in the comparison of the Jaya algorithm with other meta-heuristics.

Algorithm of Study	**Comparison Algorithm**	**p-Value**
	GA	2.137193e−06
	PSO	2.527065e−03
Jaya	DE	5.230299e−01
	ABC	9.147621e−01
	TLBO	8.438326e−01

Table 3. Friedman Ranks Post-Hoc test for Jaya and major types of meta-heuristics (p-values)

4 PROPOSED META-HEURISTIC: EJAYA

From the study of Jaya, two incongruities between its philosophy and its implementation can be found. Firstly, it is an incoherence associated with the attraction and repulsion factors that govern the fundamental equation of Jaya, Equation (1), and secondly, an incoherence related to the intensity of the flight in relation to the distance to the worst particle. Considering these problems and the benefits associated with the simplicity of Jaya, two new improvements are proposed in this work. And the combination of them has resulted in the proposed strategy named EJaya.

4.1 Coherent Attraction and Repulsion Factors

In certain situations, it is possible that the design of Jaya goes against its own philosophy: particles are attracted to the worst solution of the population and repulsed of the best individual of the population. As explained in the previous section, Jaya is designed to attract particles, $X^t_{j,k}$, to the best particle found, $X^t_{j,best}$, and move them away from the worst particle found, $X^t_{j,worst}$, as indicated in Equation (1). However, derived from the analysis of its mathematical formulation, conditions may appear where the opposite behaviour, i.e., a departure from the best particle or an approach to the worst particle, may take place. As can be seen, both the attraction and repulsion factors in Equation (1) involve the absolute value of the position of the moving particle and this caused diverse incongruent situations. In case $X^t_{j,k}$¡ 0 and $|X^t_{j,k}| > |X^t_{j,best}|$, the proposed attraction factor moves the particle away from the overall best one. The same applies to the second part of the equation, in which a movement of rapprochement to the worst particle is suggested if $X^t_{j,k}$¡ 0 and $|X^t_{j,k}| > |X^t_{j,worst}|$.

In Table 4, we have presented four examples (i.e., cases A-D) of different situations where Jaya is wrong in its formulation to achieve its goal. Specifically, a particle X^t_k focusing in dimension $j = 5$ is considered in all cases for simplicity ($X^t_{5,k}$). In case A, the studied conditions are $X^t_{j,k}$¡ 0, $|X^t_{j,k}| > |X^t_{j,best}|$ and $X^t_{j,best} < 0$ when $j = 5$. In this case it can be deduced that being $X^t_{5,k} = -9$ and $X^t_{5,best} = -6$, the attraction factor or Jaya term inducing an approach to the best particle, $AF^t_{5,k} = +r_{1,j,t}\left(X^t_{5,best} - |X^t_{5,k}|\right)$, should give a positive value to let an approach of $X^t_{5,k}$ to $X^t_{5,best}$.

However, the attraction factor given by Jaya provides a negative attraction factor, $AF^t_{5,k} = -15 \cdot r_{1,j,t}$, which would result in a departure from the best particle when it should be an approach to it. On the other hand, case B corresponds to the situation where $X^t_{j,k}$¡ 0, $|X^t_{j,k}| > |X^t_{j,worst}|$ and $X^t_{j,worst} < 0$ when $j = 5$. Considering $X^t_{5,k} = -9$ and $X^t_{5,worst} = -4$ the factor inducing the particle to move away from the worst particle or repulsion factor, $RF^t_{5,k} = -r_{2,j,t}\left(X^t_{5,worst} - |X^t_{5,k}|\right)$, should provide a positive value to increase the distance. However, as can be seen, a positive value for the repulsion factor is obtained, $RF^t_{5,k} = 13 \cdot r_{2,j,t} > 0$, leading to an approach to the worst particle $X^t_{j,worst}$. Case C corresponds to the situation where $X^t_{j,k}$¡ 0, $|X^t_{j,k}| > |X^t_{j,best}|$ and $X^t_{j,best} > 0$ when $j = 5$. In this case $X^t_{5,k} = -9$ and $X^t_{5,best} = 5$, therefore, in this case the attraction factor AF should be positive in order to attract particle $X^t_{5,k}$ to $X^t_{5,best}$ but instead $AF^t_{5,k} = -4 \cdot r_{1,j,t} < 0$. Finally, an example of the situation presented when $X^t_{j,k}$¡ 0, $|X^t_{j,k}| > |X^t_{j,worst}|$ and $X^t_{j,worst} > 0$ for $j = 5$, is shown in case D. In this case, the RF should be negative to achieve a repulsion of $X^t_{5,k}$ from $X^t_{5,worst}$. However, once again, the original algorithm in this situation would get the opposite effect, i.e., an approach to the worst particle, since $RF^t_{5,k} = 6 \cdot r_{2,j,t} > 0$.

Hence, from these examples it can be understood that although Jaya is designed to approach particles to the best current solution and escape from the worst existing solution in the population, the equation of motion that governs Jaya does not always

$$X_k^t = [2\ 7\ 4\ 8\ -9\ 1\ 3\ 4\ 6] \Rightarrow X_{5,k}^t = -9$$

Case A: $X_{j,k}^t < 0, |X_{j,k}^t| > |X_{j,best}^t|, X_{j,best}^t < 0$

$$X_{best}^t = [8\ 1\ 3\ 4\ -6\ 7\ 2\ 3\ 1] \Rightarrow X_{5,best}^t = -6$$

$$X_{5,best}^t - |X_{5,k}^t| = -6 - 9 = -15$$

$$AF_{5,k}^t = +r_{1,j,t}\left(X_{5,best}^t - |X_{5,k}^t|\right) = -15 \cdot r_{1,j,t};\ r_{1,j,t} \in [0,1]$$

$$AF_{5,k}^t = -15 \cdot r_{1,j,t} < 0$$

$$\Rightarrow AF_{5,k}^t \textit{ should be } > 0 \textit{ to attract } X_{5,k}^{t+1} \textit{ to } X_{5,best}^t$$

Case B: $X_{j,k}^t < 0,\ |X_{j,k}^t| > |X_{j,worst}^t|,\ X_{j,worst}^t < 0$

$$X_{worst}^t = [2\ 5\ 7\ 2\ -4\ 9\ 8\ 5\ 9] \Rightarrow X_{5,worst}^t = -4$$

$$X_{5,worst}^t - |X_{5,k}^t| = -4 - 9 = -13$$

$$RF_{5,k}^t = -r_{2,j,t}\left(X_{5,worst}^t - |X_{5,k}^t|\right) = 13 \cdot r_{2,j,t};\ r_{2,j,t} \in [0,1]$$

$$RF_{5,k}^t = 13 \cdot r_{2,j,t} > 0$$

$$\Rightarrow RF_{5,k}^t \textit{ should be } < 0 \textit{ to move away } X_{5,k}^{t+1} \textit{ from } X_{5,worst}^t$$

Case C: $X_{j,k}^t < 0,\ |X_{j,k}^t| > |X_{j,best}^t|,\ X_{j,best}^t > 0$

$$X_{best}^t = [8\ 1\ 3\ 4\ 5\ 7\ 2\ 3\ 1] \Rightarrow X_{5,best}^t = 5$$

$$X_{5,best}^t - |X_{5,k}^t| = 5 - 9 = -4$$

$$AF_{5,k}^t = +r_{1,j,t}\left(X_{5,best}^t - |X_{5,k}^t|\right) = -4 \cdot r_{1,j,t};\ r_{1,j,t} \in [0,1]$$

$$AF_{5,k}^t = -4 \cdot r_{1,j,t} < 0$$

$$\Rightarrow AF_{5,k}^t \textit{ should be } > 0 \textit{ to attract } X_{5,k}^{t+1} \textit{ to } X_{5,best}^t$$

Case D: $X_{j,k}^t < 0,\ |X_{j,k}^t| > |,\ X_{j,worst}^t > 0$

$$X_{worst}^t = [2\ 5\ 7\ 2\ 3\ 9\ 8\ 5\ 9] \Rightarrow X_{5,worst}^t = 3$$

$$X_{5,worst}^t - |X_{5,k}^t| = 3 - 9 = -6$$

$$RF_{5,k}^t = -r_{2,j,t}\left(X_{5,worst}^t - |X_{5,k}^t|\right) = 6 \cdot r_{2,j,t};\ r_{2,j,t} \in [0,1]$$

$$RF_{5,k}^t = 6 \cdot r_{2,j,t} > 0$$

$$\Rightarrow RF_{5,k}^t \textit{ should be } < 0 \textit{ to move away } X_{5,k}^{t+1} \textit{ from } X_{5,worst}^t$$

Table 4. Incoherent cases in Jaya (A-D)

meet the premise of the algorithm. According to the sign and relative position of each particle to the best and worst particles in the population, situations occur in which the particles go far away from the leader and closer to the worst individual, as studied in the previous examples. Alternatively, in this paper we have proposed to eliminate the absolute values of the equation of motion of Jaya, formulating it in the following terms:

$$\begin{gathered} |X_{j,k}^t| \to X_{j,k}^t, \\ X_{j,k}^{t+1} = X_{j,k}^t + r_{1,j,t}\left(X_{j,best}^t - X_{j,k}^t\right) - r_{2,j,t}(X_{j,worst}^t - X_{j,k}^t). \end{gathered} \tag{2}$$

This change ensures that regardless of $X_{j,k}^t$ is greater or lower than 0, $X_{j,best}^t$ or $X_{j,worst}^t$ are greater or lower than 0 and $|X_{j,k}^t|$ is greater or lower than $|X_{j,best}^t|$ or $|X_{j,worst}^t|$, a coherent attraction $AF_{j,k}^t$ or coherent repulsion $RF_{j,k}^t$ is obtained. In Table 5, the new values obtained considering the adopted solution in the cases A-D (Table 4) are presented. As can be seen, in all the cases the proposed amendment manages to offer a coherent response to the value of the particles. Thus, this first variant, Convergent Jaya (CJaya), allows to enhance the convergence of the original algorithm slightly sacrificing its ability to explore.

4.2 Restrained Intensity for Flight

If Equation (2) is analysed, it could be observed that the flight movement, governed by $|X_{j,worst}^t - X_{j,k}^t|$, wins intensity when the distance is longer to the worst particle, which gives rise to contradictory situations: when a particle is close to the overall worst, it moves away from it very slowly, while if it is far from it, it will move away very quickly. In addition, the first proposed variant of Jaya (CJaya), can still lead to situations in which each individual can see its approach to the best particle, whose intensity is governed by $|X_{j,best}^t - X_{j,k}^t|$, diminished by the presence of the worst particle. This problem can be analysed considering the example presented in Table 6, case E. In case E, it can be observed that the escape movement of particle $X_{5,k}^t = -900$ from the worst particle $X_{5,worst}^t = 40\,000$ is more intense than the attraction movement of the particle $X_{5,k}^t = -900$ to the best $X_{5,best}^t = -6$, although it is much farther away from the worst particle. That is, when $|X_{j,best}^t - X_{j,k}^t| < |X_{j,worst}^t - X_{j,k}^t|$ (in this particular case 894 ¡ 40900) the flight movement overshadows the approach towards the best particle. In case E, it is shown that after the proposed movements of approach and flight, the resulting particle, $X_{5,k}^{t+1} = -20\,012$, is farther from the best particle $X_{5,best}^t = -6$ than before carrying out such operations. As can be inferred, it is incoherent that moving a particle away from a bad distant individual can harm its approach to a good nearby solution.

Hence, this work also proposes to perform a moderate flight movement that takes into account a balance between attraction and repulsion between particles. The second variant of the proposal (EJaya) raises the same equation of motion that the first variant (CJaya), Equation (2), but it performs an additional checking of the movement of the particles. Specifically, to prevent the approach to the best

$$X_k^t = [2\ 7\ 4\ 8\ -9\ 1\ 3\ 4\ 6] \Rightarrow X_{5,k}^t = -9$$

Solved case A: $X_{j,k}^t < 0,\ |X_{j,k}^t| > |X_{j,best}^t|,\ X_{j,best}^t < 0$

$$X_{best}^t = [8\ 1\ 3\ 4\ -6\ 7\ 2\ 3\ 1] \Rightarrow X_{5,best}^t = -6$$

$$X_{5,best}^t - X_{5,k}^t = -6 + 9 = 3$$

$$AF_{5,k}^t = +r_{1,j,t}\left(X_{5,best}^t - X_{5,k}^t\right) = 3 \cdot r_{1,j,t};\ r_{1,j,t} \in [0,1]$$

$$AF_{5,k}^t = 3 \cdot r_{1,j,t} > 0$$

$$\Rightarrow AF_{5,k}^t \textit{ attracts } X_{5,k}^{t+1} \textit{ to } X_{5,best}^t$$

Solved case B: $X_{j,k}^t < 0,\ |X_{j,k}^t| > |X_{j,worst}^t|,\ X_{j,worst}^t < 0$

$$X_{worst}^t = [2\ 5\ 7\ 2\ -4\ 9\ 8\ 5\ 9] \Rightarrow X_{5,worst}^t = -4$$

$$X_{5,worst}^t - X_{5,k}^t = -4 + 9 = 5$$

$$RF_{5,k}^t = -r_{2,j,t}\left(X_{5,worst}^t - X_{5,k}^t\right) = -5 \cdot r_{2,j,t};\ r_{2,j,t} \in [0,1]$$

$$RF_{5,k}^t = -5 \cdot r_{2,j,t} < 0$$

$$\Rightarrow RF_{5,k}^t \textit{ moves away } X_{5,k}^{t+1} \textit{ from } X_{5,worst}^t$$

Solved case C: $X_{j,k}^t < 0,\ |X_{j,k}^t| > |X_{j,best}^t|,\ X_{j,best}^t > 0$

$$X_{best}^t = [8\ 1\ 3\ 4\ 5\ 7\ 2\ 3\ 1] \Rightarrow X_{5,best}^t = 5$$

$$X_{5,best}^t - X_{5,k}^t = 5 + 9 = 14$$

$$AF_{5,k}^t = +r_{1,j,t}\left(X_{5,best}^t - X_{5,k}^t\right) = 14 \cdot r_{1,j,t};\ r_{1,j,t} \in [0,1]$$

$$AF_{5,k}^t = 14 \cdot r_{1,j,t} > 0$$

$$\Rightarrow AF_{5,k}^t \textit{ attracts } X_{5,k}^{t+1} \textit{ to } X_{5,best}^t$$

Solved case D: $X_{j,k}^t < 0,\ |X_{j,k}^t| > |,\ X_{j,worst}^t > 0$

$$X_{worst}^t = [2\ 5\ 7\ 2\ 3\ 9\ 8\ 5\ 9] \Rightarrow X_{5,worst}^t = 3$$

$$X_{5,worst}^t - X_{5,k}^t = 3 + 9 = 12$$

$$RF_{5,k}^t = -r_{2,j,t}\left(X_{5,worst}^t - X_{5,k}^t\right) = -12 \cdot r_{2,j,t};\ r_{2,j,t} \in [0,1]$$

$$RF_{5,k}^t = -12 \cdot r_{2,j,t} > 0$$

$$\Rightarrow RF_{5,k}^t \textit{ moves away } X_{5,k}^{t+1} \textit{ from } X_{5,worst}^t$$

Table 5. Incoherent cases in Jaya solved with the first improvement of EJaya (CJaya)

$$X_k^t = [2\ 7\ 4\ 8\ -900\ 1\ 3\ 4\ 6] \Rightarrow X_{5,k}^t = -900$$
$$X_{best}^t = [8\ 1\ 3\ 4\ -6\ 7\ 2\ 3\ 1] \Rightarrow X_{5,best}^t = -6$$
$$X_{worst}^t = [2\ 5\ 7\ 2\ 40\,000\ 9\ 8\ 5\ 9] \Rightarrow X_{5,worst}^t = 40\,000$$
$$r_{1,j,t} = r_{2,j,t} = 0.5$$

Case E: $|X_{j,best}^t - X_{j,k}^t| < |X_{j,worst}^t - X_{j,k}^t|$

$$X_{5,best}^t - X_{5,k}^t = -6 + 900 = 894$$
$$X_{5,worst}^t - X_{5,k}^t = 40\,000 + 900 = 40\,900$$
$$X_{5,k}^{t+1} = X_{5,k}^t + r_{1,j,t}(X_{j,best}^t - X_{5,k}^t) - r_{2,j,t}(X_{j,worst}^t - X_{5,k}^t)$$
$$= -9 + r_{1,j,t} \cdot 894 - r_{2,j,t} \cdot 40\,900 = -20\,012$$
$$\Rightarrow X_{5,k}^{t+1} \textit{ moves away from } X_{5,best}^t$$

Table 6. Incoherent cases in Jaya (E)

particle being eroded by the distance to the worst particle, a check of the movements of attraction and repulsion for each dimension is included, in a way that if the movement of repulsion is greater than the movement of attraction, the movement of repulsion is halved in consecutive iterations (bisection technique or binary-search method) until its magnitude is lower than the movement of attraction, avoiding oscillations of particles in problems with numerous dimensions. Analytically:

$$\textit{while} \quad |X_{j,best}^t - X_{j,k}^t| < |X_{j,worst}^t - X_{j,k}^t|$$
$$|X_{j,worst}^t - X_{j,k}^t| = \frac{|X_{j,worst}^t - X_{j,k}^t|}{2}$$
$$\textit{end}$$
$$X_{j,k}^{t+1} = X_{j,k}^t + r_{1,j,t}\left(X_{j,best}^t - X_{j,k}^t\right) - r_{2,j,t}\left(X_{j,worst}^t - X_{j,k}^t\right)$$

In Table 7, the results of applying EJaya for case E are presented.

It can be observed that when the movement of flight is moderated in relation to the movement towards the best particle, a departure from the worst particle, $X_{5,worst}^t = 40\,000$, can be achieved simultaneously to an approximation to the best one, $X_{5,best}^t = -6$, being the resulting solution $X_{5,k}^{t+1} \simeq 118.4687$ instead of $X_{5,k}^{t+1} = -20\,012$, which was obtained by only applying the first improvement of Jaya, CJaya. The pseudocode of EJaya is shown in Algorithm 2.

$$X_k^t = [2\ 7\ 4\ 8\ -900\ 1\ 3\ 4\ 6] \Rightarrow X_{5,k}^t = -900$$
$$X_{best}^t = [8\ 1\ 3\ 4\ -6\ 7\ 2\ 3\ 1] \Rightarrow X_{5,best}^t = -6$$
$$X_{worst}^t = [2\ 5\ 7\ 2\ 40\,000\ 9\ 8\ 5\ 9] \Rightarrow X_{5,worst}^t = 40\,000$$
$$r_{1,j,t} = r_{2,j,t} = 0.5$$

Solved case E: $|X_{j,best}^t - X_{j,k}^t| < |X_{j,worst}^t - X_{j,k}^t|$

$$X_{5,best}^t - X_{5,k}^t = -6 + 900 = 894$$
$$X_{5,worst}^t - X_{5,k}^t = 40\,000 + 900 = 40\,900$$

$$\textit{while } |X_{5,best}^t - X_{5,k}^t| < |X_{5,worst}^t - X_{5,k}^t| \Rightarrow$$
$$|X_{j,worst}^t - X_{j,k}^t| = \tfrac{|X_{j,worst}^t - X_{j,k}^t|}{2}$$
$$\textit{end}$$
$$|X_{j,worst}^t - X_{j,k}^t| = \tfrac{40\,900}{2^6} = 639.0625$$

$$X_{5,k}^{t+1} = X_{5,k}^t + r_{1,j,t}(X_{j,best}^t - X_{5,k}^t) - r_{2,j,t}(X_{j,worst}^t - X_{5,k}^t)$$
$$= -9 + 0.5 \cdot 894 - 0.5 \cdot 639.0625 \simeq 118.4687$$
$$\Rightarrow X_{5,k}^{t+1} \textit{ is attracted to } X_{5,best}^t$$

Table 7. Incoherent case in Jaya solved with EJaya

5 EXPERIMENTAL EVALUATION AND DISCUSSION

To make an extensive comparison of the proposed improvements to Jaya, experiments of global optimization with functions of various kinds, including low, medium and high number of dimensions D (10, 50, 100), unimodal and multimodal functions, with and without random components and with a search space with restricted and unrestricted areas have been conducted in a way that the reader is provided with a wide range of data to distinguish the strengths and weaknesses of the proposal. Some functions are retrieved from the classic literature on global optimization (unimodal functions) and most of them have been presented in CEC 2014 (multimodal functions) [15]. The characteristics that make up each function are described briefly in Table 8. For each meta-heuristic to be examined, the objective functions are tested considering 40 runs. In each run, 50 random initial solutions are generated and meta-heuristics can evaluate possible solutions up to 2 000 per number of dimension of the objective function times. The set of 40 runs is called an experiment and each experiment is configured to evaluate the objective functions considering 10, 50 and 100 dimensions or variables. For each dimension, solutions in the range

Algorithm 2 EJaya pseudocode

```
—— Data
N: Number of individuals
D: Number of dimensions
—— Algorithm
Population initialization
while !end condition do
    Find (X^t_{j,best}) and worst (X^t_{j,worst}) individual in the population
    for k ¡ N do
        for j ¡ D do
            while |X^t_{j,best} − X^t_{j,k}| < |X^t_{j,worst} − X^t_{j,k}| do
                |X^t_{j,worst} − X^t_{j,k}| = (|X^t_{j,worst} − X^t_{j,k}|)/2
            end while
            X^{t+1}_{j,k} = X^t_{j,k} + r_{1,j,t} (X^t_{j,best} − X^t_{j,k}) − r_{2,j,t} (X^t_{j,worst} − X^t_{j,k})
        end for
        if Better solution found over particle's actual solution then
            Update particle's solution
        else
            Preserve previous particle's solution
        end if
    end for
end while
```

$[-100, 100]$ are accepted unless the objective function specifies a different range, as indicated in Table 8.

This section presents the results obtained by each of the algorithms differentiating the evaluation of unimodal and multimodal functions and finally, conducting a global analysis of all of them. Results for 10, 50 and 100 dimensions (Tables 9, 10 and 11, respectively) are listed by objective functions (rows) and each function contains four sub-rows indicating the average of the found solutions, the best solution reached, the average runtime and the classification of those heuristics depending on the quality of the average reached solution. Finally, a statistical analysis of the results is presented.

Unimodal Functions Analysis

In Tables 9, 10 and 11 it can be observed that the most appropriate heuristic for solving benchmark unimodal functions (f_{01}, f_{02}, f_{03}) is EJaya, which obtains for all dimensions D (10, 50 and 100) results at least an order of magnitude better than its competitor in second place, CJaya. Unimodal functions are simple functions in which algorithms with high qualities for exploitation render exceptionally. Thus, these results are an indication of the high exploitation capability of EJaya, what makes this heuristic particularly suitable for problems with a reduced or limited search space.

Benchmark Function	**Formulation**	**Minima**
High-Conditioned Elliptic	$f_1(x) = \sum_{i=1}^{D} (10^6)^{\frac{i-1}{D-1}} x_i^2$	0
Bent-Cigar	$f_2(x) = x_1^2 + 10^6 \sum_{i=2}^{D} x_i^2$	0
Discus	$f_3(x) = 10^6 x_1^2 + \sum_{i=2}^{D} x_i^2$	0
Rosenbrock	$f_4(x) = \sum_{i=1}^{D-1} \left(100\left(x_i^2 - x_{i+1}\right) + (x_i - 1)^2\right)$	0
Ackley's path	$f_5(x) = -20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}\right) - \exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i)\right) + 20 + e$	0
Weierstrass	$f_6(x) = \sum_{i=1}^{D}\left(\sum_{k=0}^{k_{max}}\left[a^k \cos\left(2\pi b^k (x_i + 0.05)\right)\right]\right) - D\sum_{k=0}^{k_{max}}\left[a^k \cos\left(\pi b^k\right)\right]$	4
Griewank's Function 8	$f_7(x) = \sum_{i=1}^{D} \frac{x_i^2}{4000} - \prod_{i=1}^{D} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	0
Rastrigin	$f_8(x) = \sum_{i=1}^{D}\left(x_i^2 - 10\cos(2\pi x_i) + 10\right)$	0
Modified Swchefel	$f_9(x) = 418.9829D - \sum_{i=1}^{D} g(z_i),\ z_i = x_i + 4.209687462275036\text{e+}002$	–
	$g(z_i) = \begin{cases} z_i \sin(\sqrt{\lvert z_i\rvert}), & \lvert z_i\rvert \le 500, \\ (500 - \mathrm{mod}(z_i, 500))\sin\left(\sqrt{\lvert 500 - \mathrm{mod}(z_i, 500)\rvert}\right) - \frac{(z_i - 500)^2}{10000D}, & z_i > 500, \\ (\mathrm{mod}(\lvert z_i\rvert, 500) - 500)\sin\left(\sqrt{\lvert \mathrm{mod}(\lvert z_i\rvert, 500) - 500\rvert}\right) - \frac{(z_i + 500)^2}{10000D}, & z_i < -500 \end{cases}$	–
Katsuura	$f_{10}(x) = \frac{10}{D^2}\prod_{i=1}^{D}\left(1 + i\sum_{j=1}^{32}\frac{\lvert 2^j x_i - round(2^j x_i)\rvert}{2^j}\right)^{\frac{10}{D^{1.2}}} - \frac{10}{D^2}$	0
HappyCat	$f_{11}(x) = \left\lvert\sum_{i=1}^{D} x_i^2 - D\right\rvert^{\frac{1}{4}} + \frac{0.5\sum_{i=1}^{D} x_i^2 + \sum_{i=1}^{D} x_i}{D} + 0.5$	–
HGBat	$f_{12}(x) = \left\lvert\left(\sum_{i=1}^{D} x_i^2\right)^2 - \left(\sum_{i=1}^{D} x_i\right)^2\right\rvert^{1/2} + \frac{0.5\sum_{i=1}^{D} x_i^2 + \sum_{i=1}^{D} x_i}{D} + 0.5$	–
Expanded Griewank's plus Rosenbrock's	$f_{13}(x) = f_7(f_4(x_1, x_2)) + f_7(f_4(x_2, x_3)) + \ldots + f_7(f_4(x_{D-1}, x_D)) + f_7(f_4(x_D, x_1))$	
Expanded Schaffer Function 6	$f_{14}(x) = g(x_1, x_2) + g(x_2, x_3) + \ldots + g(x_{D-1}, x_D), g(x_D, x_1)$	–
	$g(x, y) = 0.5 + \frac{\left(\sin^2\left(\sqrt{x^2 + y^2}\right) - 0.5\right)}{\left(1 + 0.001\left(x^2 + y^2\right)\right)^2}$	
Langermann	$f_{15}(x) = -\sum_{i=1}^{5} \frac{c_i \cos\left\{\pi\left[(x_1 - a_i)^2 + (x_2 - b_u)^2\right]\right\}}{e^{\frac{(x_1 - a_i)^2 + (x_2 - b_u)^2}{\pi}}}$	−5.1621259
Eggholder	$f_{16}(x) = -x_1 \sin\left(\sqrt{\lvert x1 - x2 - 47\rvert}\right) - (x_2 + 47)\sin\left(\sqrt{\lvert \frac{1}{2}x_1 + x_2 + 47\rvert}\right)$	−959.64066
Holder's table	$f_{17}(x) = -\left\lvert e^{\left\lvert 1 - \frac{\sqrt{x_1^2 + x_2^2}}{\pi}\right\rvert}\sin(x_1)\cos(x_2)\right\rvert$	–
DropWave	$f_{18}(x) = -\frac{1 + \cos\left(12\sqrt{\sum_i^D x_i^2}\right)}{2 + 0.5\sum_{i=1}^{D} x_i^2}$	0
Bohachevsky	$f_{19}(x) = \sum_i^{D-1}\left[x_i^2 + 2x_{i+1}^2 - 0.3\cos(3\pi x_i) - 0.4\cos(4\pi x_{i+1}) + 0.7\right]$	0
Whitley	$f_{20}(x) = \sum_{i=1}^{D}\sum_{j=1}^{D}\left[\frac{\left(100(x_i^2 - x_j)^2 + (1 - x_j)^2\right)^2}{4\,000} - \cos\left(100\left(x_i^2 - x_j\right)^2 + (1 - x_j)^2\right) + 1\right]$	0

Table 8. Benchmark functions testbed

f	Stats	Jaya	CJaya	EJaya	f	Stats	Jaya	CJaya	EJaya
f01	Mean	2.163e−02	8.794e−06	4.893**e**−11	**f11**	Mean	3.709e−01	1.633e−01	1.456**e**−01
	Best	5.739e−03	1.352e−06	8.380**e**−12		Best	3.011e−01	8.737e−02	5.084**e**−02
	Runtime (s)	2.101e+00	2.081e+00	2.127**e**+00		Runtime (s)	9.867e+00	9.849e+00	1.036**e**+01
	Rank	3	2	**1**		Rank	3	2	**1**
f02	Mean	7.577e+00	2.714e−03	1.559**e**−08	**f12**	Mean	2.411e−01	1.331e−01	1.279**e**−01
	Best	1.907e+00	5.027e−04	1.769**e**−09		Best	1.490e−01	6.492e−02	7.453**e**−02
	Runtime (s)	2.071e+00	2.049e+00	2.151**e**+00		Runtime (s)	9.645e+00	9.525e+00	1.021**e**+01
	Rank	3	2	**1**		Rank	3	2	**1**
f03	Mean	1.436e−04	3.124e−08	1.633**e**−13	**f13**	Mean	1.487e−01	9.338e−02	5.099**e**−02
	Best	2.060e−05	5.067e−09	2.139**e**−14		Best	1.273e−02	4.638e−02	2.637**e**−03
	Runtime (s)	2.081e+00	2.064e+00	2.130**e**+00		Runtime (s)	9.847e+00	9.663e+00	9.967**e**+00
	Rank	3	2	**1**		Rank	3	2	**1**
f04	Mean	8.751**e**+11	9.059e+11	9.203e+11	**f14**	Mean	0.000**e**+00	5.551e−18	0.000e+00
	Best	1.237**e**+11	1.558e+11	2.087e+11		Best	0.000**e**+00	0.000e+00	0.000e+00
	Runtime (s)	2.081**e**+00	2.054e+00	2.128e+00		Runtime (s)	3.057**e**+00	3.085e+00	3.163e+00
	Rank	**1**	2	3		Rank	**1**	3	2
f05	Mean	2.000**e**+01	2.000e+01	2.000e+01	**f15**	Mean	−5.162**e**+00	−5.162**e**+00	−5.162**e**+00
	Best	2.000**e**+01	2.000e+01	2.000e+01		Best	−5.162**e**+00	−5.162**e**+00	−5.162**e**+00
	Runtime (s)	2.084**e**+00	2.092e+00	2.115e+00		Runtime (s)	1.036**e**+01	1.030**e**+01	1.059**e**+01
	Rank	**1**	2	3		Rank	**1**	**2**	**3**
f06	Mean	1.800e+02	1.800e+02	1.800**e**+02	**f16**	Mean	−9.596e+02	−9.596e+02	−9.596**e**+02
	Best	1.800e+02	1.800e+02	1.800**e**+02		Best	−9.596e+02	−9.596e+02	−9.596**e**+02
	Runtime (s)	2.090e+00	2.049e+00	2.123**e**+00		Runtime (s)	2.971e+00	3.094e+00	3.147**e**+00
	Rank	2	3	**1**		Rank	3	2	**1**
f07	Mean	5.802e−01	5.232e−01	4.015**e**−01	**f17**	Mean	−9.140**e**+18	−9.140**e**+18	−9.140**e**+18
	Best	3.096e−01	2.868e−01	2.256**e**−01		Best	−9.140**e**+18	−9.140**e**+18	−9.140**e**+18
	Runtime (s)	2.047e+00	2.059e+00	2.115**e**+00		Runtime (s)	3.068**e**+00	3.040**e**+00	3.066**e**+00
	Rank	3	2	**1**		Rank	**1**	**2**	**3**
f08	Mean	4.250e+01	3.932e+01	2.619**e**+01	**f18**	Mean	−9.362e−01	−9.362e−01	−9.362**e**−01
	Best	2.954e+01	1.975e+01	9.750**e**+00		Best	−9.362e−01	−9.362e−01	−9.362**e**−01
	Runtime (s)	2.084e+00	2.070e+00	2.129**e**+00		Runtime (s)	1.039e+01	1.015e+01	1.058**e**+01
	Rank	3	2	**1**		Rank	2	3	**1**
f09	Mean	1.325e−04	1.273e−04	1.273**e**−04	**f19**	Mean	2.776**e**−17	5.551e−17	3.886e−17
	Best	1.280e−04	1.273e−04	1.273**e**−04		Best	0.000**e**+00	0.000e+00	0.000e+00
	Runtime (s)	2.057e+00	2.068e+00	2.117**e**+00		Runtime (s)	1.038**e**+01	1.016e+01	1.044e+01
	Rank	3	2	**1**		Rank	**1**	3	2
f10	Mean	0.000**e**+00	0.000**e**+00	0.000**e**+00	**f20**	Mean	4.039e+01	4.852e+01	3.604**e**+01
	Best	0.000**e**+00	0.000**e**+00	0.000**e**+00		Best	0.000e+00	0.000e+00	9.894**e**+00
	Runtime (s)	2.095**e**+00	2.110**e**+00	2.149**e**+00		Runtime (s)	1.040e+01	1.031e+01	1.059**e**+01
	Rank	**1**	**2**	**3**		Rank	2	3	**1**

Table 9. Benchmark functions f01-f20 results with Jaya and improvements of Jaya, CJaya and EJaya, $D = 10$

Multimodal Functions Analysis

Multimodal functions have many local minima, and they are more difficult to optimize than unimodal functions. Hence, the end results of this type of functions are more relevant since they mirror the capacity of the strategy to get away from local optima and finding a result close to the global optimum [22]. In this case, although there is no clear supremacy of an algorithm, CJaya and EJaya emerge as the most effective meta-heuristics in optimizing the benchmark functions. This can be observed from Tables 9, 10 and 11 for functions $f_{04} - f_{20}$. CJaya and EJaya are also more scalable than Jaya, since increasing the number of dimensions in the experiments significantly improves their efficiency, and they scale positions in the rank.

f	Stats	Jaya	CJaya	EJaya	f	Stats	Jaya	CJaya	EJaya
f01	Mean	3.005e+02	5.849e+00	1.007**e**−05	**f11**	Mean	1.307e+00	6.960e−01	6.543**e**−01
	Best	2.404e+00	2.258e−03	7.029**e**−07		Best	1.083e+00	4.676e−01	4.874**e**−01
	Runtime (s)	4.572e+01	4.531e+01	4.708**e**+01		Runtime (s)	4.517e+01	4.584e+01	4.736**e**+01
	Rank	3	2	**1**		Rank	3	2	**1**
f02	Mean	3.828e−03	1.674e−05	3.141**e**−09	**f12**	Mean	1.382e+00	6.722**e**−01	6.740e−01
	Best	1.453e−03	3.388e−06	9.776**e**−10		Best	1.125e+00	2.959**e**−01	3.345e−01
	Runtime (s)	4.546e+01	4.536e+01	4.761**e**+01		Runtime (s)	4.567e+01	4.549**e**+01	4.699e+01
	Rank	3	2	**1**		Rank	3	**1**	2
f03	Mean	3.828e−03	1.674e−05	3.141**e**−09	**f13**	Mean	2.396e+04	1.277e+00	3.971**e**−01
	Best	1.453e−03	3.388e−06	9.776**e**−10		Best	1.869e+01	5.963e−01	7.343**e**−03
	Runtime (s)	4.546e+01	4.536e+01	4.761**e**+01		Runtime (s)	4.551e+01	4.522e+01	4.740**e**+01
	Rank	3	2	**1**		Rank	3	2	**1**
f04	Mean	4.979**e**+12	4.988e+12	4.987e+12	**f14**	Mean	2.776e−18	0.000**e**+00	5.551e−18
	Best	4.747**e**+12	4.749e+12	4.708e+12		Best	0.000e+00	0.000**e**+00	0.000e+00
	Runtime (s)	4.568**e**+01	4.514e+01	4.686e+01		Runtime (s)	3.013e+00	3.016**e**+00	3.057e+00
	Rank	**1**	3	2		Rank	2	**1**	3
f05	Mean	2.000e+01	2.000**e**+01	2.000e+01	**f15**	Mean	−5.162**e**+00	−5.162**e**+00	−5.162**e**+00
	Best	2.000e+01	2.000**e**+01	2.000e+01		Best	−5.162**e**+00	−5.162**e**+00	−5.162**e**+00
	Runtime (s)	4.612e+01	4.568**e**+01	4.732e+01		Runtime (s)	4.538**e**+01	4.533**e**+01	4.698**e**+01
	Rank	3	**1**	2		Rank	**1**	**2**	**3**
f06	Mean	4.900e+03	4.900**e**+03	4.900e+03	**f16**	Mean	−9.589e+02	−9.580e+02	−9.596**e**+02
	Best	4.900e+03	4.900**e**+03	4.900e+03		Best	−9.596e+02	−9.596e+02	−9.596**e**+02
	Runtime (s)	4.540e+01	4.494**e**+01	4.693e+01		Runtime (s)	2.936e+00	2.976e+00	3.026**e**+00
	Rank	3	**1**	2		Rank	2	3	**1**
f07	Mean	2.628e−02	1.441e−02	4.800**e**−03	**f17**	Mean	−9.140**e**+18	−9.140**e**+18	−9.140**e**+18
	Best	2.090e−05	6.990e−08	3.203**e**−11		Best	−9.140**e**+18	−9.140**e**+18	−9.140**e**+18
	Runtime (s)	4.655e+01	4.605e+01	4.724**e**+01		Runtime (s)	2.972**e**+00	2.926**e**+00	3.075**e**+00
	Rank	3	2	**1**		Rank	**1**	**2**	**3**
f08	Mean	4.780e+02	4.330e+02	4.191**e**+02	**f18**	Mean	−5.885e−03	−2.969e−02	−1.862**e**−01
	Best	3.802e+02	3.059e+02	3.465**e**+02		Best	−1.651e−02	−7.774e−02	−2.888**e**−01
	Runtime (s)	4.689e+01	4.648e+01	4.794**e**+01		Runtime (s)	4.544e+01	4.550e+01	4.711**e**+01
	Rank	3	2	**1**		Rank	3	2	**1**
f09	Mean	2.768e+01	9.228e+00	6.364**e**−04	**f19**	Mean	7.752e+00	5.088e+00	2.512**e**+00
	Best	7.062e−04	6.367e−04	6.364**e**−04		Best	2.554e+00	4.720e−01	3.801**e**−07
	Runtime (s)	4.636e+01	4.554e+01	4.737**e**+01		Runtime (s)	4.607e+01	4.565e+01	4.716**e**+01
	Rank	3	2	**1**		Rank	3	2	**1**
f10	Mean	4.231e−05	0.000**e**+00	0.000**e**+00	**f20**	Mean	1.998e+05	2.250e+03	2.184**e**+03
	Best	0.000e+00	0.000**e**+00	0.000**e**+00		Best	1.715e+03	1.256e+03	1.940**e**+03
	Runtime (s)	4.550e+01	4.529**e**+01	4.705**e**+01		Runtime (s)	4.536e+01	4.550e+01	4.684**e**+01
	Rank	3	**1**	**2**		Rank	3	2	**1**

Table 10. Benchmark functions f01-f20 results with Jaya and improvements of Jaya, CJaya and EJaya, $D = 50$

From this analysis, it is noteworthy that CJaya and EJaya obtain better or equal (when all the algorithms reach the global minimum) results than Jaya more than half of the times: for $D = 10$ this condition occurs in 14 functions, and for $D = 50$ and $D = 100$ this occurs in 17 functions out of 20. Moreover, EJaya reaches the top position in the rank in most cases, sometimes with a lead of several orders of magnitude over Jaya, and most of the time providing an improvement between 20 and 60 percentage points over. Although statistical tests results are to be presented in the next section, it can be concluded in view of these experiments that both CJaya as EJaya, and especially the latter, offer an improvement in the performance of Jaya, with very small increases in runtime (around 5 % in the case of EJaya).

f	Stats	Jaya	CJaya	EJaya	f	Stats	Jaya	CJaya	EJaya
f01	Mean	9.525e+03	1.689e+00	1.147**e**−04	**f11**	Mean	1.581e+00	7.363**e**−01	7.407e−01
	Best	1.978e+01	3.283e−02	4.053**e**−06		Best	1.377e+00	5.524**e**−01	6.278e−01
	Runtime (s)	1.792e+02	1.775e+02	1.866**e**+02		Runtime (s)	1.796e+02	1.789**e**+02	1.872e+02
	Rank	3	2	**1**		Rank	3	**1**	2
f02	Mean	1.621e+04	7.249e+00	7.974**e**−03	**f12**	Mean	1.574e+00	7.183e−01	7.068**e**−01
	Best	2.506e+03	1.578e+00	1.622**e**−03		Best	1.405e+00	3.325e−01	3.416**e**−01
	Runtime (s)	1.788e+02	1.784e+02	1.872**e**+02		Runtime (s)	1.795e+02	1.834e+02	1.938**e**+02
	Rank	3	2	**1**		Rank	3	2	**1**
f03	Mean	3.689e−02	2.463e−05	1.684**e**−08	**f13**	Mean	1.437e+07	1.281e+01	1.682**e**+00
	Best	6.813e−03	4.110e−06	5.686**e**−09		Best	3.025e+05	4.059e+00	1.359**e**+00
	Runtime (s)	1.793e+02	1.776e+02	1.870**e**+02		Runtime (s)	1.802e+02	1.776e+02	1.848**e**+02
	Rank	3	2	**1**		Rank	3	2	**1**
f04	Mean	1.000e+13	9.936**e**+12	1.003e+13	**f14**	Mean	2.776e−18	0.000**e**+00	2.776e−18
	Best	9.701e+12	9.527**e**+12	9.524e+12		Best	0.000e+00	0.000**e**+00	0.000e+00
	Runtime (s)	1.811e+02	1.803**e**+02	1.845e+02		Runtime (s)	5.908e+00	5.886**e**+00	6.021e+00
	Rank	2	**1**	3		Rank	2	**1**	3
f05	Mean	2.001e+01	2.000**e**+01	2.000e+01	**f15**	Mean	−5.162**e**+00	−5.162**e**+00	−5.162**e**+00
	Best	2.000e+01	2.000**e**+01	2.000e+01		Best	−5.162**e**+00	−5.162**e**+00	−5.162**e**+00
	Runtime (s)	1.803e+02	1.798**e**+02	1.852e+02		Runtime (s)	1.802**e**+02	1.798**e**+02	1.847**e**+02
	Rank	3	**1**	2		Rank	**1**	**2**	**3**
f06	Mean	1.980e+04	1.980e+04	1.980**e**+04	**f16**	Mean	−9.558e+02	−9.580e+02	−9.596**e**+02
	Best	1.980e+04	1.980e+04	1.980**e**+04		Best	−9.596e+02	−9.596e+02	−9.596**e**+02
	Runtime (s)	1.800e+02	1.780e+02	1.846**e**+02		Runtime (s)	6.016e+00	5.971e+00	6.005**e**+00
	Rank	3	2	**1**		Rank	3	2	**1**
f07	Mean	7.049e−03	3.078e−03	2.772**e**−03	**f17**	Mean	−9.140**e**+18	−9.140**e**+18	−9.140**e**+18
	Best	6.537e−05	2.726e−08	3.404**e**−11		Best	−9.140**e**+18	−9.140**e**+18	−9.140**e**+18
	Runtime (s)	1.802e+02	1.784e+02	1.852**e**+02		Runtime (s)	5.861**e**+00	5.918**e**+00	6.016**e**+00
	Rank	3	2	**1**		Rank	**1**	**2**	**3**
f08	Mean	1.100e+03	7.472**e**+02	9.940e+02	**f18**	Mean	−5.684e−04	−2.680e−03	−1.271**e**−02
	Best	6.421e+02	4.371**e**+02	4.835e+02		Best	−9.462e−04	−5.352e−03	−3.199**e**−02
	Runtime (s)	1.789e+02	1.785**e**+02	1.858e+02		Runtime (s)	1.780e+02	1.770e+02	1.841**e**+02
	Rank	3	**1**	2		Rank	3	2	**1**
f09	Mean	3.285**e**−03	1.846e+01	3.691e+01	**f19**	Mean	2.986e+01	2.553e+01	1.492**e**+01
	Best	1.809**e**−03	1.273e−03	1.273e−03		Best	1.966e+01	1.680e+01	8.398**e**+00
	Runtime (s)	1.790**e**+02	1.776e+02	1.840e+02		Runtime (s)	1.782e+02	1.781e+02	1.857**e**+02
	Rank	**1**	2	3		Rank	3	2	**1**
f10	Mean	2.367e−04	1.637**e**−07	2.447e−06	**f20**	Mean	3.600e+08	9.062e+03	7.475**e**+03
	Best	0.000e+00	0.000**e**+00	0.000e+00		Best	8.277e+06	4.638e+03	9.412**e**+02
	Runtime (s)	1.806e+02	1.783**e**+02	1.852e+02		Runtime (s)	1.784e+02	1.777e+02	1.843**e**+02
	Rank	3	**1**	2		Rank	3	2	**1**

Table 11. Benchmark functions f01-f20 results with Jaya and improvements of Jaya, CJaya and EJaya, $D = 100$

Statistical Tests

The best known procedure for multiple comparison to check differences between more than two related samples is the Friedman Ranks test [5]. Given a certain value of statistical significance limit (generally, p-value = 0.05) it is determined whether the null hypothesis H0 (H0: algorithms have a similar behaviour) can be rejected. On the other hand, it must be born in mind that the main drawback is that Friedman Ranks test can only detect significant differences with respect to any multiple comparison, but it is unable to establish appropriate comparisons between some considered algorithms. When the goal is not only to know whether there are differences between the methods, but to compare which relations have more statistical significance, a series of ad-hoc hypotheses using a post-hoc test must be done after the Friedman Ranks test. In this work, the Friedman Ranks Post-Hoc test

is used to get the unadjusted p-values that allow the comparison of the statistical significance among all the implemented algorithms. The Friedman Ranks Post-Hoc test obtains a set of p-values that determine the degree of rejection of each null hypothesis for each pair of algorithms. It must be noted that for this analysis it is not possible to make a comparison between pairs by type, such as in the Wilcoxon test because they involve a set of results where the Family-Wise Error Rate (FWER) is not controlled.

The Friedman Ranks and Friedman Ranks Post-Hoc tests are conducted in this work for the set of algorithms in the case of 10, 50 and 100 dimensions for average value, better results as well as runtime.

Stats	**D = 10**	**D = 50**	**D = 100**
Mean	**3.379e−04**	**1.008e−04**	**4.800e−04**
Best	**2.410e−02**	**5.615e−04**	**2.390e−02**
Runtime	**6.772e−07**	**7.126e−08**	**1.219e−08**

Table 12. Friedman Ranks test p-values for mean, best and runtime results

First, in Table 12, Friedman Ranks test results are presented. It can be observed that for all the cases the Friedman Ranks test indicates that the null hypothesis should be rejected, that is, there are statistically significant differences (considering a statistical significance limit of p-value = 0.05) between the strategies for the different analysed results (i.e., mean, best and runtime) and dimensions (i.e., 10, 50 and 100), so it makes sense to conduct a Friedman Ranks Post-Hoc test in each case. Table 13 shows p-values for 10, 50 and 100 dimensions, for all the parameters to be analysed and for all the possible comparisons. First, regarding mean results, it is observed that EJaya improves Jaya with statistical significance in all dimensions while CJaya improves Jaya with statistically significance for medium and high dimensions, 50 and 100, respectively. In addition, it can be seen that EJaya and CJaya show similar results in a statistical significance sense. As for the best results analysis, the statistical significance corroborates the results related to the mean, being EJaya once again the strategy presenting the best behaviour. Finally, as expected, the introduction of modifications to achieve coherent attraction and repulsion factors and restrained intensity for flight slows down Jaya variants as more control checks are introduced. In this way, it can be observed that statistically significant differences are found in runtime between EJaya and both Jaya and CJaya, except for high dimensions case (i.e., 100), no statistical significance is found between CJaya and Jaya.

6 CONCLUSIONS

In recent years, several meta-heuristics have been proposed and research in this sense is still in the spotlight due to the wide range of applications requiring more efficient optimization procedures in industrial and scientific areas. A major issue in the implementation and comparison of meta-heuristics is given by their simplicity, which

Stats	Comparison	D = 10	D = 50	D = 100
Mean	EJaya-CJaya	2.305e−01	1.709e−01	8.521e−01
	Jaya-CJaya	6.895e−02	**5.116e−02**	**3.522e−03**
	Jaya-EJaya	**4.097e−04**	**9.807e−05**	**5.1319e−04**
Best	EJaya-CJaya	5.882e−01	5.891e−01	4.602e−01
	Jaya-CJaya	2.302e−01	1.656**e**−02	3.097**e**−03
	Jaya-EJaya	**2.391e−02**	**5.763e−04**	**3.070e−05**
Runtime	EJaya-CJaya	**5.571e−07**	**7.333e−08**	**1.051e−08**
	Jaya-CJaya	3.289e−01	2.537e−01	**1.965e−02**
	Jaya-EJaya	**4.184e−04**	**2.468e−04**	**4.474e−03**

Table 13. Friedman Ranks Post-Hoc test p-values for mean, best and runtime results

is directly related to the number of parameters and their associated interactions. Jaya algorithm is a recent and simple meta-heuristic offering better or comparable results to a large set of major meta-heuristics nowadays such as GA, PSO, DE, ABC and TLBO. In this work, Jaya algorithm is improved to offer a more efficient specific parameter-free meta-heuristic of one phase, EJaya. EJaya includes coherent attraction and repulsion movements and restrained intensity for flights that overcome limitations of Jaya and the contributions of these improvements to general success are analysed gradually (CJaya and EJaya) to valid each of the suggested incremental solutions. The proposal has been tested on a set of various standard benchmark functions from CEC. The results obtained by EJaya offer more efficient results than Jaya in terms of mean and best accomplishments, as proved by statistical tests showing statistical significance, a higher exploitation capability and scalability. Finally, it must be highlighted that EJaya does not require tuning of algorithm specific parameters what may be beneficial to applications where the setup of optimization may be critical for the whole performance of the systems. Hence, this work presents one more step towards the development of simpler and fast optimization strategies.

In future works, EJaya will be applied to a problem of practical importance nowadays that can greatly benefit from the simplicity and speed of EJaya in its set-up. Specifically, EJaya will be considered for the learning of Fuzzy Rule-Based Schedulers in Cloud Computing. Cloud Computing is a very dynamic environment where a fast knowledge acquisition strategy can achieve significant improvement in terms of time and power saving in the allocation of workload among the large number of involved computational resources. Further, results will be compared to other meta-heuristics.

Acknowledgments

This work was financially supported by the Research Projects TEC2015-67387-C4-2 and 2011-TIC-7278.

REFERENCES

[1] Bindiya, T. S.—Elias, E.: Meta-Heuristic Evolutionary Algorithms for the Design of Optimal Multiplier-Less Recombination Filter Banks. Information Sciences, Vol. 339, 2016, pp. 31–52, doi: 10.1016/j.ins.2015.12.018.

[2] Chinta, S.—Kommadath, R.—Kotecha, P.: A Note on Multi-Objective Improved Teaching-Learning Based Optimization Algorithm (MO-ITLBO). Information Sciences, Vol. 373, 2016, pp. 337–350, doi: 10.1016/j.ins.2016.08.061.

[3] Chou, J.-S.—Ngo, N.-T.: Time Series Analytics Using Sliding Window Metaheuristic Optimization-Based Machine Learning System for Identifying Building Energy Consumption Patterns. Applied Energy, Vol. 177, 2016, pp. 751–770, doi: 10.1016/j.apenergy.2016.05.074.

[4] Deb, K.—Agrawal, S.: Understanding Interactions Among Genetic Algorithm Parameters. Foundations of Genetic Algorithms V, Morgan Kaufmann, 1999, pp. 265–286.

[5] Derrac, J.—García, S.—Molina, D.—Herrera, F.: A Practical Tutorial on the Use of Nonparametric Statistical Tests as a Methodology for Comparing Evolutionary and Swarm Intelligence Algorithms. Swarm and Evolutionary Computation, Vol. 1, 2011, No. 1, pp. 3–18, doi: 10.1016/j.swevo.2011.02.002.

[6] García-Galán, S.—Prado, R. P.—Expósito, J. E. M.: Swarm Fuzzy Systems: Knowledge Acquisition in Fuzzy Systems and Its Applications in Grid Computing. IEEE Transactions on Knowledge and Data Engineering, Vol. 26, 2014, No. 7, pp. 1791–1804, doi: 10.1109/TKDE.2013.118.

[7] García-Galán, S.—Prado, R. P.—Expósito, J. E. M.: Rules Discovery in Fuzzy Classifier Systems with PSO for Scheduling in Grid Computational Infrastructures. Applied Soft Computing, Vol. 29, 2015, pp. 424–435, doi: 10.1016/j.asoc.2014.11.064.

[8] Gendreau, M.—Potvin, J.-Y.: Handbook of Meta-Heuristics. Springer US, 2010.

[9] Goldberg, D. E.: Genetic Algorithms in Search, Optimization and Machine Learning. 1st Edition. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1989.

[10] Jordehi, A. R.: Enhanced Leader PSO (ELPSO): A New PSO Variant for Solving Global Optimisation Problems. Applied Soft Computing, Vol. 26, 2015, pp. 401–417, doi: 10.1016/j.asoc.2014.10.026.

[11] Karaboga, D.—Basturk, B.: A Powerful and Efficient Algorithm for Numerical Function Optimization: Artificial Bee Colony (ABC) Algorithm. Journal of Global Optimization, Vol. 39, 2007, No. 3, pp. 459–471, doi: 10.1007/s10898-007-9149-x.

[12] Karaboga, D.—Basturk, B.: On the Performance of Artificial Bee Colony (ABC) Algorithm. Applied Soft Computing, Vol. 8, 2008, No. 1, pp. 687–697, doi: 10.1016/j.asoc.2007.05.007.

[13] Kennedy, J.—Eberhart, R.: Particle Swarm Optimization. Proceedings of the IEEE International Conference on Neural Networks (ICNN'95), Vol. 4, 1995, pp. 1942-1948, doi: 10.1109/ICNN.1995.488968.

[14] Kennedy, J.—Eberhart, R. C.—Shi, Y.: Swarm Intelligence. Springer, 2001.

[15] Liang, J. J.—Qu, B. Y.—Suganthan, P.: Problem Definitions and Evaluation Criteria for the CEC 2014 Special Session and Competition on Single Objective Real-Parameter Numerical Optimization. Technical Report, Zhengzhou University, China and Nanyang Technological University, Singapore, 2013, IIT Kanpur, India, 2014.

[16] Liu, S. Q.—Kozan, E.: A Hybrid Metaheuristic Algorithm to Optimise a Real-World Robotic Cell. Computers and Operations Research, Vol. 84, 2017, pp. 188–194, doi: 10.1016/j.cor.2016.09.011.

[17] Mesejo, P.—Ibáñez, Ó.—Cordón, Ó.—Cagnoni, S.: A Survey on Image Segmentation Using Metaheuristic-Based Deformable Models: State of the Art and Critical Analysis. Applied Soft Computing, Vol. 44, 2016, pp. 1–29, doi: 10.1016/j.asoc.2016.03.004.

[18] Minetti, G.—Leguizamón, G.—Alba, E.: An Improved Trajectory-Based Hybrid Metaheuristic Applied to the Noisy DNA Fragment Assembly Problem. Information Sciences, Vol. 277, 2014, pp. 273–283, doi: 10.1016/j.ins.2014.02.020.

[19] Patel, V. K.—Savsani, V. J.: Heat Transfer Search (HTS): A Novel Optimization Algorithm. Information Sciences, Vol. 324, 2015, pp. 217–246, doi: 10.1016/j.ins.2015.06.044.

[20] Rao, R. V.: Jaya: A Simple and New Optimization Algorithm for Solving Constrained and Unconstrained Optimization Problems. International Journal of Industrial Engineering Computations, Vol. 7, 2016, No. 1, pp. 19–34, doi: 10.5267/j.ijiec.2015.8.004.

[21] Rao, R. V.: Teaching-Learning-Based Optimization Algorithm. Springer International Publishing, Cham, 2016, pp. 9–39.

[22] Rashedi, E.—Nezamabadi-Pour, H.—Saryazdi, S.: GSA: A Gravitational Search Algorithm. Information Sciences, Vol. 179, 2009, No. 13, pp. 2232–2248, doi: 10.1016/j.ins.2009.03.004.

[23] Storn, R.—Price, K.: Differential Evolution – A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. Journal of Global Optimization, Vol. 11, 1997, No. 4, pp. 341–359, doi: 10.1023/A:1008202821328.

[24] Wolpert, D. H.—Macready, W. G.: No Free Lunch Theorems for Optimization. IEEE Transactions on Evolutionary Computation, Vol. 1, 1997, No. 1, pp. 67–82.

[25] Xiong, Y.—Golden, B.—Wasil, E.: A One-Parameter Genetic Algorithm for the Minimum Labeling Spanning Tree Problem. IEEE Transactions on Evolutionary Computation, Vol. 9, 2005, No. 1, pp. 55–60, doi: 10.1109/4235.585893.

[26] Xu, J.—Kelly, J. P.: A Network Flow-Based Tabu Search Heuristic for the Vehicle Routing Problem. Transportation Science, Vol. 30, 1996, No. 4, pp. 379–393, doi: 10.1287/trsc.30.4.379.

[27] Zhang, L.—Wong, T. N.: Solving Integrated Process Planning and Scheduling Problem with Constructive Metaheuristics. Information Sciences, Vol. 340-341, 2016, pp. 1–16, doi: 10.1016/j.ins.2016.01.001.

[28] Zhu, Q.—Lin, Q.—Du, Z.—Liang, Z.—Wang, W.—Zhu, Z.—Chen, J.—Huang, P.—Ming, Z.: A Novel Adaptive Hybrid Crossover Operator for Multiobjective Evolutionary Algorithm. Information Sciences, Vol. 345, 2016, pp. 177–198, doi: 10.1016/j.ins.2016.01.046.

Rafael Rodríguez-Reche received his B.Eng. degree in computer science engineering from University of Jaén, Jaén, Spain, in 2014 and his M.Sc. degree in telecommunication engineering in 2016. At present, he collaborates as a researcher at the Telecommunication Engineering Department of the University of Jaén. His current research interests include blockchain, artificial intelligence, cloud computing, scheduling, machine learning and optimization.

Rocío P. Prado received her M.Sc. degree in telecommunication engineering from Seville University, Seville, Spain, in 2008 and her Ph.D. degree in telecommunication engineering with European Mention from University of Jaén, Jaén, Spain, in 2011. At present, she is Associate Professor with the Telecommunication Engineering Department of the Jaén University. Her current research interests include artificial intelligence, machine learning, grid/cloud computing and scheduling. She is an active member of the research group "Signal Processing for Telecommunication Systems" (TIC-188 of the PAI) and she is in the editorial board of 19 JCR-indexed international journals such as IEEE Transactions on Fuzzy Systems, Applied Soft Computing and IEEE Transactions on Data and Knowledge Engineering.

Sebastián García-Galán received his M.Sc. and Ph.D. degrees in telecommunication engineering from the Málaga University and the Technical University of Madrid (UPM), in 1995 and 2004, respectively. Since 1999, he is Associate Professor at the Telecommunication Engineering Department of the Jaén University. He is a member of the research group "Signal Processing for Telecommunication Systems" (TIC-188 of the PAI) and the European Society for Fuzzy Logic And Technology (EUSFLAT). His areas of research interest are artificial intelligence, grid/cloud computing, speech and audio analysis. Also, he is a reviewer of several journals indexed in JCR. He is involved in research projects of the Spanish Ministry of Science and Education and of private companies.

José Enrique Muñoz-Expósito received his M.Sc. degree in telecommunication engineering from Málaga University, Spain and Ph.D. in telecommunication engineering from Jaén University, Spain. Since 2003, he has been Associate Professor at the Telecommunication Engineering Department of the Jaén University. His research interests include speech and audio analysis, artificial intelligence, grid and cloud computing. He is currently Senior Researcher with the chair for "Signal Processing and Telecommunication Systems" (TIC-188 of the PAI). He is involved in research projects and works also for the editorial reviewer board of the Annual Telematics Engineering Conferences (JITEL).

Nicolás Ruiz-Reyes received his M.Sc. and Ph.D. degrees in telecommunication engineering from the Technology University of Madrid and the University of Alcalá, in 1993 and 2001, respectively. Since 2010 he has been Full Professor at the Telecommunication Engineering Department of the University of Jaén, he is also the head of the research group TIC-188-PAI. He is a member of the IEEE Signal Processing Society and Audio Engineering Society. His areas of research interest are signal processing and its applications to communications. He is involved in research projects of the Spanish Ministry of Science, European Commission and private companies.