Opposition-based moth swarm algorithm

Article in Expert Systems with Applications  · June 2021
DOI: 10.1016/j.eswa.2021.115481

Some of the authors of this publication are also working on these related projects:

- Special issue on Bio-inspired algorithms and Bio-systems, Impact Factor: 1.230 View project
- Advances in Soft Computing and Machine Learning in Image Processing View project
An opposition-based moth swarm algorithm for global optimization

Diego Oliva\(^a,b\), Sara Esquivel-Torres\(^b\), Salvador Hinojosa\(^b\), Marco Pérez-Cisneros\(^b\), Valentin Osuna-Enciso\(^b\), Noé Ortega-Sánchez\(^b\), Gaurav Dhiman\(^c\), Ali Asghar Heidari\(^d,e,1\)

\(^a\) División de electrónica y computación, Universidad de Guadalajara, CUCIEL, Av. Revolución 1500, Guadalajara, Jal, Mexico
\(^b\) School of Computer Science and Robotics, Tomsk Polytechnic University, Tomsk, Russia
\(^c\) School of Surveying and Geospatial Engineering, College of Engineering, University of Tehran, Tehran, Iran
\(^d\) Department of Computer Science, School of Computing, National University of Singapore, Singapore, Singapore

**Article Info**

Article history:
Received 30 December 2019
Received in revised form 24 November 2020
Accepted 23 June 2021
Available online xxx

**Keywords**
Moth swarm algorithm
Opposition-based learning
Optimization techniques
Metaheuristics

**Abstract**

Nowadays, resource-optimizing techniques are required in many engineering areas to obtain the most appropriate solutions for complex problems. For this reason, there is a trend among researchers to improve existing swarm-based algorithms through different evolutionary techniques and to create new population-based methods that can accurately explore the feature space. The recently proposed Moth swarm algorithm (MSA) inspired by the orientation of moths towards moonlight is an associative learning mechanism with immediate memory that uses Lévy mutation to cross-population diversity and spiral movement. The MSA is a population-based method used for tackling complex optimization problems. It presents an adequate capacity for exploration and exploitation trends; however, due to its nature of operators, this type of method is prone to get stuck in sub-optimal locations, which affects the speed of convergence and the computational effort to reach better solutions. To mitigate these shortcomings, this paper proposes an improved MSA that combines opposition-based learning (OBL) as a mechanism to enhance the exploration drifts of the basic version and increase the speed of convergence to obtain more accurate solutions. The proposed approach is called OBMSA. It has been tested for solving three classic engineering design problems (welded beam, tension/compression spring, and pressure vessel designs) with constraints, 19 benchmark functions comprising 7 unimodal, 6 multimodal, and 6 composite functions. Experimental results and comparisons provide evidence that the performance and accuracy of the proposed method are superior to the original MSA. We hope the community utilizes the proposed MSA-based approach for solving other complex problems.

© 2021

1. Introduction

Metaheuristic Algorithms (MA) are computational techniques inspired by natural or social phenomena (Talbi, 2009) and they are utilized to solve complex multimodal real-world problems, particularly from the optimization point of view (Jiang, Wang, Wu, & Geng, 2017; Zavala, Nebro, Luna, & Coello Coello, 2014). The MA as optimization solvers are part of the intelligent expert systems. They try to simulate and extract intelligent behavior from different aspects of nature to reach a set of computational rules for optimization purposes.

Classical optimization methods only guarantee to find the best solution if a group of conditions is satisfied, while MA has few requirements to resolve an optimization problem; for instance, it does not need any gradient operation to find the optimal. Instead, it uses a population of search agents, or candidate solutions, distributed inside a bounded multidimensional search space; those agents interchange information and move toward the best solutions found so-far by using simple operators and at least one selection step (Spears, De Jong, Bäck, Fogel, & De Garis, 1993; Xu et al., 2019; Yang, 2013).

In the literature exist several MA, being one of the first the Genetic Algorithms, which mimic the natural selection that occurs after a group of individuals evolve and adapt to environmental changes (Holland, 1992). This method, together with Evolutionary Strategies (Schwefel, 1984), Evolutionary Programming (Burgin, 1973), Genetic Programming (Koza, 1991), and more recently Differential Evolution (Storn & Price, 1995), conform the main block in MA, called, Evolutionary Computation (Spears et al., 1993). This metaheuristics group utilizes evolution-inspired operators, such as mutation, recombination, and selection of the fittest. The second block of MA is Swarm Intelligence (SI) (Karaboga & Akay, 2009). More recent and bigger than Evolutionary
Computation, this group has many approaches, like Particle Swarm Optimization (PSO), which possess operators based on the behavior of bird flocks or fish schools (Kennedy & Eberhart, 1995). Other approaches are Artificial Bee Colony (Bolaji, Khader, Al-Betar, & Awadallah, 2013; Karaboga, 2005), Social Spider Optimization (Cuevas, Cienfuegos, Zaldívar, & Perez-Cisneros, 2013), Moth Flame Optimization (MFO) (Mirjalili, 2015) and Moth Swarm Algorithm (MSA) (Mohamed, Mohamed, El-Gaafary, & Hemeida, 2017), which is a similar approach to MFO. The most extended survey on this kind of metaheuristics is in (Papinelli & Lopes, 2011). MA are flexible, stochastic approaches that help to realize the solutions to multimodal and complex optimization problems (Fausto, Reyna-Orta, Cuevas, Andrade, & Perez-Cisneros, 2019). The importance of this kind of methods can be clearly seen in the number of methods that every year have been published and used in real-world applications (Dokeroglu, Sevinc, Kucukyilmaz, & Cosar, 2019; Shen et al., 2016; Wang et al., 2017).

One of the main drawbacks of MA for optimization is a slow and early convergence that stuck the search process in suboptimal values (Chen et al., 2020; Hu et al., 2021). In this sense, several proposals extend the original metaheuristics’ performance by using hybridizations with other techniques or even exact optimization methods (Shan et al., 2021; Tu et al., 2021; Zhang et al., 2021; Zhao et al., 2021). For instance, the authors in (Zhang and Hui, 2017a) propose a framework to combine the Bat Algorithm and the concept of cooperating multiagents to solve both a group of benchmark functions and the economic load dispatch problem, with competitive results when compared against the original algorithm. Another interesting work comes up with two modifications to a PSO (Zhang & Hui, 2016). The approach, called multiagent coordination optimization, considers consensus protocols alongside PSO’s swarm intelligence properties; the algorithms were easily programmed using the parallel parfor in Matlab, and the results over three real-world problems suggest that the algorithm is competitive compared to the original MA in terms of accuracy and convergence speed. Other examples of hybridizations can be found in (Blum, Puchinger, Raidl, & Roli, 2011; Talibi, 2002). In this context, the motivation for proposing a new MA or modifications is the No-Free Lunch (NFL) theorem, which states that any MA can outperform others MA in a given family of problems, but not in all families of problems (Joyce & Herrmann, 2018; Wolpert & Macready, 1997). Therefore, the search for new or improvements to the original MA stills being an open research problem (Zhao et al., 2019).

The MSA is a metaheuristic that mimics moths’ behavior looking for food and phototaxis at night. These insects are very similar to butterflies, but they are more active during the night. The MSA differs from the MFO in the operators that modify the population. The MSA algorithm divides a population into pathfinders, prospectors, and onlookers. For exploration and diversity, the pathfinders employ a combination of crossover and mutation with Lévy flights. The prospectors move with a logarithmic spiral that helps stabilize the search space exploration and exploitation drifts (the right balance between them provides a more efficient optimizer). The positions of onlookers are updated using adaptive Gaussian walks. Moreover, the control parameters update with the iterative process. Such facts provide a flexible algorithm that can escape from suboptimal values. In general, MSA shows good capabilities for solving complex optimization problems, MSA has been successfully used to find the optimal power flow (Mohamed et al., 2017), for analysis of clusters in machine learning (Yang, Luo, Zhang, Wu, & Zhou, 2017). image processing (Lüque-Chang et al., 2021; Zhou, Yang, Ling, & Zhang, 2018), optimal power flow (Shilaja & Arunprasad, 2019), among others. However, MSA’s main disadvantage is the slow convergence (Guvenc, Duman, & Hinislöglu, 2017).

The authors in (Tizhoosh, 2005) introduced Opposition-Based Learning (OBL) to improve the convergence properties of MA. The OBL is also part of computational intelligence theories; it takes a candidate solution and generates its opposite position in the feature space. Using a simple rule, the OBL verifies if the opposite or the candidate solution has the best objective function value. Usually, the approach calculates the opposite positions at the initialization step or when an operator modifies the set of feasible solutions. The OBL has demonstrated efficacy in improving several MA. For example, (Bulbul, Pradhan, Roy, & Pal, 2015) proposed an opposition-based Krill Herd (KH) algorithm for the economic load dispatch problem. To solve numerical optimization problems, (Verma, Aggarwal, & Patodi, 2016) modified a firefly algorithm with the opposition-based approach. Other proposals increase the convergence speed in MA by applying OBL over Electromagnetism-Like Optimization (Cuevas, Oliva, Zaldívar, Perez, & Pajares, 2014) or Differential Evolution (ODE) (Rahnamayan, Tizhoosh, & Salama, 2008a), to mention only some examples. The opposition-based rule has been utilized to improve the estimation of parameters in control engineering using the Shuffled Frog Leaping (SFL) algorithm (Ahammad & Alavi-Rad, 2015). In this context, the opposition-based and dimensional-based Firefly Algorithm (ODFA) (Verma et al., 2016) improved the results’ accuracy of FA. The use of OBL has also been extended to multiobjective optimization (Ma et al., 2014). All of these works show that OBL is an excellent mechanism to achieve better results in optimization problems.

This paper presents an improved version of the MSA called the Opposition-Based Moth Swarm Algorithm (OBMSA). The proposal combines the opposition-based learning and the MA operators to enhance the original MSA’s performances and convergence trends. We tested the approach over a standard set of mathematical benchmark problems. Moreover, we tested OBMSA over benchmark engineering problems to prove that it can solve real-life optimization problems, efficiently. Comparisons with other similar approaches indicate that the developed OBMSA can provide better results in terms of accuracy and efficiency. In addition, the OBMSA, as part of the computational intelligence methods, is a tool that can be extended to several fields of application. Different metrics and statistical validations support the experiments and comparisons. Hence, the contributions of the proposed OBMSA can be summarized as:

- An improved population-based optimization algorithm is proposed that uses the OBL to increase the exploration capabilities of MSA.
- The use of OBL to enhance the efficacy of the basic MSA in two parts of the search procedure.
- The application of the OBMSA for constrained engineering problems.

Moreover, the proposed approach also contributes to the domain of the intelligent expert systems by including a modern optimization tool that permits to have more accurate solutions for a set of benchmark problems.

The paper’s organization is as follows: Section 2 introduces the standard MSA, whereas Section 3 describes the main conceptualization points of OBL. The explanation of the proposed OBMSA is presented in Section 4, while Section 5 presents the experimental setup and the comparisons. Finally, Section 6 presents the conclusions of this paper.

2. Preliminaries

In this section, the basic concepts of the standard MSA and theory related to the OBL are introduced. First of all, a biological profile of moths that helps to understand the operators of MSA are introduced. After that, the concept of opposite numbers and the learning rule of OBL are presented.
2.1. Moth Swarm Algorithm

The nocturnal behavior of moths is the inspiration for the Moth Swarm Algorithm (MSA), introduced in 2017 (Mohamed et al., 2017). In the algorithm, the exploration–exploitation balance considers a partition of candidate solutions forming the population:

- Pathfinders (to explore new regions of the search space).
- Prospectors (to exploit the new areas get by the pathfinders).
- Onlookers (to exploit the best areas found by the prospectors).

As with other metaheuristics, this one starts by initializing a population:

\[ x_i = \text{rand} \cdot (u_i - l_i) + l_i, \quad \forall i \in \{1, 2, ..., n\} \]

where \( u_i \) and \( l_i \) are upper and lower limits of the search space, \( x_i \) is the candidate solution, \( n \) is the population size, \( d \) is the problem’s dimensionality, and \( \text{rand} \) is a random value drawn from a uniform distribution.

2.1.1. Pathfinder phase

To generate the pathfinders’ crossover it is necessary to calculate the dispersal degree and the variation coefficient at iteration:

\[ \sigma^2 = \frac{1}{n} \sum_{i=1}^{n} (x^i - p^i)^2 \]

where \( n \) is the number of pathfinders, and

\[ \mu = \frac{1}{d} \sum_{i=1}^{n} x^i \]

In the MSA the crossover points are those with the lowest dispersal values, according to:

\[ j \in c_p \quad \text{if} \quad \sigma^2 \leq \mu \]

From those, only \( n_c \in c_p \) crossover points are utilized to create new sub-trial pathfinder vectors \( \vec{v}_p = [v_{p1}, v_{p2}, ..., v_{pn_c}] \) from the original pathfinder \( \vec{x}_p = [x_{p1}, x_{p2}, ..., x_{pn_c}] \) as follows:

\[ v_{pi} = \begin{cases} x^i_p & \text{if} \quad j \in c_p \\ \mu & \text{if} \quad j \notin c_p \end{cases} \quad \forall i \in \{1, 2, ..., n_c\} \]

where \( L_{p1} \) and \( L_{p2} \) are independent variables computed from the Lévy \( \alpha \)-stable distribution (Mantegna, 1994). The set of indexes \( r \) must be only selected from the pathfinder solutions, and those positions are updated using the mutated variables extracted from the sub-trail vector according to the following equation:

\[ \vec{p}_p = [p^i_p, p^i_p, ..., p^i_p] \quad \forall i \in \{1, 2, ..., n_c\} \]

Finally, MSA applies a selection strategy between the trial and the original pathfinders defined as:

\[ x^*_{p1} = \begin{cases} x^i_p & \text{if} \quad f(x^i_p) \geq f(x^i_p) \\ \mu & \text{otherwise} \end{cases} \]

The probability of selecting the next pathfinder is defined as:

\[ p_p = \frac{\beta f_p}{\sum_{i=0}^{n} f_p} \]

which uses the luminescence intensity calculated by the next equation:

\[ f_p = \frac{1 + f_p}{1 + \beta f_p} \quad \text{if} \quad f_p > 0 \]

2.1.2. Prospector phase

From the pathfinders, \( n_p \) individuals are selected as prospectors; this number is dynamically modified as in the next equation:

\[ n_p = \text{round} \left( (1 - \frac{r}{T}) \times (n - n_c) \right) \]

With \( T \) being the maximum iteration number. To simulate a prospector moth moving in a spiral way around a pathfinder (as in its natural counterpart (Böyadzhiev, 1999)), MSA uses the following definition:

\[ x^i_p = \begin{cases} x^i_p + \epsilon_1 \cdot \text{best}_p - x^i_p & \text{if} \quad j \in c_p \\ \epsilon_1 \cdot \text{best}_p & \text{otherwise} \end{cases} \quad \forall i \in \{1, 2, ..., n_p\} \]

where \( \epsilon_1 \) is a random number utilized to give the spiral form to the prospector path, whereas \( r = 1 - \frac{r}{T} \).

2.1.3. Onlooker phase

The onlookers are the moths with the lowest luminescent intensities moving towards the shiniest source of light (Duman, 2018); in MSA, the onlookers phase is utilized to intensify the exploitation of promisory spots of the search space. The onlooker group is further divided according to two movement rules: Gaussian walks, and associative learning mechanism with immediate memory. In the first case, the onlooker in the actual iteration is obtained by:

\[ x^*_{p1} = x^i_p + \epsilon_2 \cdot \text{best}_p - x^i_p \quad \forall i \in \{1, 2, ..., n_c\} \]

where \( \epsilon_2 \) is uniformly distributed random numbers, \( \text{best}_p \) is the global best candidate solution, \( n = \text{round}(n/2) \) is the number of onlookers, and \( \epsilon_1 \) is a normal random number calculated as

\[ \epsilon_1 \sim \text{random}(\text{size}(d)) \]

The behavior of the moths considering associative learning as well as short term memory, is updated according to:

\[ x^*_{p1} = x^i_p + 0.001 \cdot G + \left( 1 - \frac{2}{G} \right) \cdot \epsilon_2 \cdot \left( \text{best}_p - x^i_p \right) + \left( \frac{2G}{\alpha} \right) \cdot \epsilon_3 \cdot \left( \text{best}_p - x^i_p \right) \quad \forall i \in \{1, 2, ..., n_c\} \]
with \( n_o = n_u - n_p \) being the number of onlookers that performs associative learning and short term memory, \( 1 - \frac{q}{G} \) is a cognitive factor, \( 2q/G \) is a social factor, \( best_p \) is the best light source from the pathfinder group and \( G \sim N \left( x_1^* - x_{1\text{best}}^{\text{max}} - x_1^* \right) \).

The MSA pseudocode, showing all the steps of the original algorithm, is shown in Algorithm 1.

2.2. Opposition-based learning

Inspired by sudden and radical changes of social revolutions, the concept of OBL was proposed in 2005 as a new approach to increase the convergence of evolutionary techniques when they are applied to complicated problems, in particular to search and optimization (Tizhoosh, 2005). The main idea is that every candidate solution of an optimization problem in the actual iteration can be constructed either by considering the gained experience in the evolution of the algorithm or by mere random guesses, as in the population’s initialization. In the second case, the convergence speed could be probabilistically increased by generating an opposed point for every candidate solution not only at the beginning of the algorithm but also at every iteration (Rahnamayan et al., 2008a). The OBL considers that searching in the direction of the original candidate solution as well as at the opposite direction could be beneficial to find the global optimum in a more efficient manner. Therefore, an important concept is the opposite number, defined in the next subsection.

2.2.1. Opposite number

Let \( x = \{x_1, x_2, \ldots, x_D\} \) be a coordinate in a \( D \)-dimensional space (e.g., a candidate solution for an optimization problem), with \( x \in \mathbb{R}^D \) defined between lower and upper limits as \( x_i \in [l_i, u_i], \quad i = 1, \ldots, D \); then the opposite candidate solution \( \bar{x} = \{\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_D\} \) can be defined as:

\[
\bar{x}_i = l_i + u_i - x_i, \quad i = 1, \ldots, D
\]

(18)

Table 1

Moth Swarm Algorithm pseudocode.

1. Initialize the population
2. Calculate fitness of the swarm; sort population according to its fitness; select \( n_p \) moths as pathfinders; identify prospectors and onlookers.
3. \( \text{while} \quad t < T \)
4. pathfinder phase
5. prospector phase
6. onlooker phase
7. identify new light sources and type of each moth
8. \( \text{endwhile} \)
9. print global best

For an optimization problem, the application of OBL is simple and straightforward; let suppose that \( f(x) \) is the objective function utilized to measure the value of the candidate solution \( x \), which is randomly initialized. As the iterations advance, a candidate solution and its opposed will be calculated (\( x \) and \( \bar{x} \)), and evaluated \( f(x) \) and \( f(\bar{x}) \). Without loss of generality, for a minimization problem if \( f(x) \geq f(\bar{x}) \) is true, then the opposite candidate solution is kept; otherwise, the original is preserved. The previous condition is tested at every iteration of the algorithm, usually until a stop criterion is reached (Rahnamayan, Tizhoosh, & Salama, 2008b). As a simple OBL example, let consider Fig. 1, where a 2D objective function is shown in Fig. 1a, while in Fig. 1b are displayed four randomly initialized candidate solutions and their corresponding opposed ones. According to the level lines, only individuals \( x_1, x_3, \bar{x}_1, \bar{x}_3 \) will be retained after the evaluation, whereas individuals \( x_2, x_4, \bar{x}_2, \bar{x}_4 \) will be discarded for the next iteration (Figs. 2, 3, 4).

3. Proposed MSA-based approach

In this section, the OBMSA is proposed as the combination of the abilities of the MSA algorithm with the OBL technique to increase the exploration of the search space, which generates an increase in the precision of the optimal solution. OBMSA improves the performance of the MSA algorithm, which suffers from several unfavorable aspects such as; Slow convergence and being trapped in local solutions features that prevent increasing the exploration of the search space and the efficient use of time. The proposed algorithm aims to avoid these situations, considering the opposite value while covering the search space by considering the two options for the calculated point. With this modification, it is 50% more likely to find optimal solutions in a shorter time. However, as the NFL theorem states that an optimization algorithm cannot solve all problems effectively, every modification of an algorithm must be evaluated (Wolpert & Macready, 1997).

The improvement of the proposed method is carried out in two parts; The first performs the initialization of the population with OBL; the process begins by choosing the \( N \) particles close to the optimal solution. Consequently, a more suitable population is obtained, and an increase in the rate of acceleration of the algorithm. The second part updates the new generations of the swarm particles called prospector moths, the most suitable are chosen to avoid clogging in the local optimal and explore the search space in such a way that it is efficient to find the desired solution.

In the following subsections, the stages mentioned above are explained, and Fig 5 shows the comparative diagram of the MSA and OBMSA algorithms.

Fig. 1. An OBL example: a) objective function, and b) initial population of four candidate solutions and their opposed.
3.1. Initial population based on the opposition

In the absence of prior knowledge, it is common to create the initial population of an algorithm by generating particles with random numbers. Therefore, we can find starting candidate solutions even when the problem is unknown (Rahnamayan et al., 2008a). The initialization of the population based on opposition consists of 3 steps;

1) Randomly initiate the population, that is, create the first positions of the moths with the following equation:
\[ x_j = \text{rand}(0, 1) \left( x^\text{max}_j - x^\text{min}_j \right) + x^\text{min}_j \]

\( \forall i \in \{1, 2, 3, ..., n\}, j \in \{1, 2, 3, ..., d\} \)

where \( x^\text{max}_j \) and \( x^\text{min}_j \) are the upper and lower bounds of the search space, respectively.

2) Calculate the opposite population; once the initial population is obtained, calculate for each moth its corresponding opposite value before selecting the type of each moth.

3) Select from the set formed by the union of \( p \) and \( \tilde{p} \) the \( N_c \) positions of the most suitable moths as initial population. The following equation presents the minimization rule used to choose the values that exist between the two populations.

\[
x = \begin{cases} 
\bar{x} & \text{if } f(\bar{x}) < f(x) \\
x & \text{otherwise}
\end{cases}
\]

Once the fitness has been calculated, and the initial population has been established, the type of each moth is chosen.

3.2. New generations based on the opposition

An algorithm can change the individuals in its population to get new candidate solutions that are more suitable than the current population. In that step, the algorithm dynamically calculates the opposite population using the OBL technique. Thus, new generations based on the opposite calculate the opposite of each variable based on the minimum and maximum values of that variable in the current population (Rahnamayan et al., 2008a).

The new generations based on the opposite technique applied only to the population of prospector and onlooker moths. The pathfinder moths will always be the best \( N_c \) solutions. The opposite population is determined by Eq. (9) using the techniques of OBL. The opposite population is joined with the prospector population, and the fitness is evaluated; then, the moths are ranked from best to worst, and the first halves are preserved for the next generation. This step is repeated in every generation.
3.3. Convergence of the OBMSA

The main drawback of the MSA is its slow convergence speed. Hence, the OBMSA is proposed to improve this characteristic while also increasing the exploration ability of the algorithm. The convergence can be assured since MSA is a random search algorithm that retains the best solution found at each generation (Solis & Wets, 1981). Under such a scenario, a random search algorithm could not reach the optimal solution if the algorithm gets stagnated. However, on the MSA the pathfinders are designed to prevent such situation from updating their position in interaction with other moths (Jevtic, Jovanovic, Radosavljevic, & Klimenta, 2017), and their convergence characteristics have been analyzed on domain-specific cases (Sayed, Kamel, Yu, & Jurado, 2020; Yang, Luo, Zhang, Wu, & Zhou, 2017). Even more, the inclusion of the OBL mechanism does not harm the convergence of the algorithm, but it increases the population diversity and helps to avoid premature convergence by evaluating the candidate solution and its mirrored counterpart (Kang, Xiong, Zhou, & Meng, 2018). In any case, the OBL and the behavior of the pathfinders can avoid the local stagnation of the OBMSA.

4. Experiments and discussion

The proposed method was compared with the standard version of MSA to formulate an alternative version that solves global optimization problems in a reduced time and with greater precision; this is reflected in multimodal problems and even in compound problems. The results of the experiments indicate that the precision of the found solutions and the speed of convergence were superior in most of the selected functions compared to other optimization methods. The performance of the proposed OBMSA algorithm was tested against the MSA, and other well-known optimization algorithms such as the PSO, DE, MFO, and HS. Finally, the Wilcoxon non-parametric test was performed with the MSA and the other algorithms. In this section, the results are discussed in the following subsections: Section 4.1 Test functions, Section 4.2 Performance measures, Section 4.3 Comparison between MSA and OBMSA, Section 4.4 Comparison with other algorithms, Section 4.5 OBMSA applied to classical engineering problems, 4.6 Results and discussions.
4.1. Benchmark functions

For the experimental study, it is common to compare the performance of an algorithm using mathematical functions established in the literature, which have known global optimal (Digalakis & Margaritis, 2002; Molga & Smutnicki, 2005), a set of 19 benchmark functions have been chosen to evaluate the performance of the proposed algorithm that includes unimodal, multimodal and composite types, in order to observe the results in each different type. Table 1 shows each of the original test functions, which consider the set of functions presented in (Mirjalili, 2015).

In this article, the benchmarks functions are named by the character “F” followed by a number; according to their properties, the test functions are divided into three different sets: unimodal, multimodal, and composite. The unimodal functions go from F1 to F7 and have a clear global optimum; these functions are used to know the accuracy of the optimization algorithm. The multimodal functions take the interval from F8 to F13 and have a large number of local optimum; multimodal
Table 3
Bench mark functions.

<table>
<thead>
<tr>
<th>ID</th>
<th>Eq.</th>
<th>Dim</th>
<th>Lower</th>
<th>Upper</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>F_1</td>
<td>f_1(x) = \sum_{i=1}^{n} x_i^2</td>
<td>30</td>
<td>-100</td>
<td>100</td>
<td>Unimodal</td>
</tr>
<tr>
<td>F_2</td>
<td>f_2(x) = \sum_{i=1}^{n}</td>
<td>x_i</td>
<td>+ \prod_{i=1}^{n}</td>
<td>x_i</td>
<td></td>
</tr>
<tr>
<td>F_3</td>
<td>f_3(x) = \sum_{i=1}^{n} \left( \sum_{j=1}^{i} x_j \right)^2</td>
<td>30</td>
<td>-100</td>
<td>100</td>
<td>Unimodal</td>
</tr>
<tr>
<td>F_4</td>
<td>f_4(x) = \max_{1 \leq i \leq n} {</td>
<td>x_i</td>
<td>}</td>
<td>30</td>
<td>-100</td>
</tr>
<tr>
<td>F_5</td>
<td>\sum_{i=1}^{n} (100(x_{i+1} - x_i)^2 + (x_i - 1)^2)</td>
<td>30</td>
<td>-30</td>
<td>30</td>
<td>Unimodal</td>
</tr>
<tr>
<td>F_6</td>
<td>\sum_{i=1}^{n} (x_i + 0.5)^2</td>
<td>30</td>
<td>-100</td>
<td>100</td>
<td>Unimodal</td>
</tr>
<tr>
<td>F_7</td>
<td>\sum_{i=1}^{n} (x_i^4 + \text{random}(0,1))</td>
<td>30</td>
<td>-1.28</td>
<td>1.28</td>
<td>Unimodal</td>
</tr>
<tr>
<td>F_8</td>
<td>\sum_{i=1}^{n} x_i \sin \left( \sqrt{\frac{x_i}{i}} \right)</td>
<td>30</td>
<td>-500</td>
<td>500</td>
<td>Unimodal</td>
</tr>
<tr>
<td>F_9</td>
<td>\sum_{i=1}^{n} (0.2 \sin(2\pi x_i) + 10) \cdot n</td>
<td>30</td>
<td>-5.12</td>
<td>5.12</td>
<td>Multimodal</td>
</tr>
<tr>
<td>F_{10}</td>
<td>-20 \exp \left( -0.2 \sqrt{\sum_{i=1}^{n} x_i^2} \right) - \exp \left( \frac{\sum_{i=1}^{n} \cos(2\pi x_i)}{5} \right) + 20 + \epsilon</td>
<td>30</td>
<td>-600</td>
<td>600</td>
<td>Multimodal</td>
</tr>
<tr>
<td>F_{11}</td>
<td>\frac{1}{\sqrt{\sum_{i=1}^{n} x_i^2}} - \prod_{i=1}^{n} \cos \left( \frac{x_i}{\sqrt{i}} \right) + 1</td>
<td>30</td>
<td>-50</td>
<td>50</td>
<td>Multimodal</td>
</tr>
<tr>
<td>F_{12}</td>
<td>\frac{1}{\sqrt{\sum_{i=1}^{n} x_i^2}} - \prod_{i=1}^{n} \cos \left( \frac{x_i}{\sqrt{i}} \right) + 1</td>
<td>30</td>
<td>-50</td>
<td>50</td>
<td>Multimodal</td>
</tr>
<tr>
<td>F_{13}</td>
<td>\left( \sin^3(\theta_{xy}) + \sum_{i=1}^{n} (x_i - 1)^2 [1 + \sin^2(3\pi x_i + 1)] + (x_n - 1)^2 [1 + \sin^2(3\pi x_n + 1)] \right) + \sum_{i=1}^{n}</td>
<td>x_i</td>
<td>, 5, 100, 4</td>
<td>30</td>
<td>-50</td>
</tr>
<tr>
<td>F_{14}</td>
<td>f_{15}, f_{16}, \ldots, f_{20} = \text{Sphere Function}</td>
<td>2</td>
<td>-65536</td>
<td>65536</td>
<td>Compound</td>
</tr>
<tr>
<td>F_{15}</td>
<td>f_{15}, f_{16}, \ldots, f_{20} = \text{Griewanks Function}</td>
<td>2</td>
<td>-5</td>
<td>5</td>
<td>Compound</td>
</tr>
<tr>
<td>F_{16}</td>
<td>f_{15}, f_{16}, \ldots, f_{20} = \text{Griewanks Function}</td>
<td>2</td>
<td>-5</td>
<td>5</td>
<td>Compound</td>
</tr>
<tr>
<td>F_{17}</td>
<td>f_{15}, f_{16}, \ldots, f_{20} = \text{Griewanks Function}</td>
<td>2</td>
<td>[-5,0]</td>
<td>[10,15]</td>
<td>Compound</td>
</tr>
<tr>
<td>F_{18}</td>
<td>f_{15}, f_{16}, \ldots, f_{20} = \text{Sphere Function}</td>
<td>2</td>
<td>-5</td>
<td>5</td>
<td>Compound</td>
</tr>
<tr>
<td>F_{19}</td>
<td>f_{15}, f_{16}, \ldots, f_{20} = \text{Sphere Function}</td>
<td>2</td>
<td>-5</td>
<td>5</td>
<td>Compound</td>
</tr>
<tr>
<td>F_{20}</td>
<td>f_{15}, f_{16}, \ldots, f_{20} = \text{Sphere Function}</td>
<td>2</td>
<td>-5</td>
<td>5</td>
<td>Compound</td>
</tr>
<tr>
<td>F_{21}</td>
<td>f_{15}, f_{16}, \ldots, f_{20} = \text{Sphere Function}</td>
<td>2</td>
<td>-5</td>
<td>5</td>
<td>Compound</td>
</tr>
<tr>
<td>F_{22}</td>
<td>f_{15}, f_{16}, \ldots, f_{20} = \text{Sphere Function}</td>
<td>2</td>
<td>-5</td>
<td>5</td>
<td>Compound</td>
</tr>
<tr>
<td>F_{23}</td>
<td>f_{15}, f_{16}, \ldots, f_{20} = \text{Sphere Function}</td>
<td>2</td>
<td>-5</td>
<td>5</td>
<td>Compound</td>
</tr>
<tr>
<td>F_{24}</td>
<td>f_{15}, f_{16}, \ldots, f_{20} = \text{Sphere Function}</td>
<td>2</td>
<td>-5</td>
<td>5</td>
<td>Compound</td>
</tr>
<tr>
<td>F_{25}</td>
<td>f_{15}, f_{16}, \ldots, f_{20} = \text{Sphere Function}</td>
<td>2</td>
<td>-5</td>
<td>5</td>
<td>Compound</td>
</tr>
<tr>
<td>F_{26}</td>
<td>f_{15}, f_{16}, \ldots, f_{20} = \text{Sphere Function}</td>
<td>2</td>
<td>-5</td>
<td>5</td>
<td>Compound</td>
</tr>
<tr>
<td>F_{27}</td>
<td>f_{15}, f_{16}, \ldots, f_{20} = \text{Sphere Function}</td>
<td>2</td>
<td>-5</td>
<td>5</td>
<td>Compound</td>
</tr>
<tr>
<td>F_{28}</td>
<td>f_{15}, f_{16}, \ldots, f_{20} = \text{Sphere Function}</td>
<td>2</td>
<td>-5</td>
<td>5</td>
<td>Compound</td>
</tr>
<tr>
<td>F_{29}</td>
<td>f_{15}, f_{16}, \ldots, f_{20} = \text{Sphere Function}</td>
<td>2</td>
<td>-5</td>
<td>5</td>
<td>Compound</td>
</tr>
<tr>
<td>F_{30}</td>
<td>f_{15}, f_{16}, \ldots, f_{20} = \text{Sphere Function}</td>
<td>2</td>
<td>-5</td>
<td>5</td>
<td>Compound</td>
</tr>
</tbody>
</table>

Benchmark functions are used to evaluate the ability of an algorithm to avoid local stagnation in suboptimal solutions. The composite functions ranging from F14 to F19 are the combination of multimodal and unimodal test functions with rotations and biases increasing their complexity. All of the benchmark functions are extracted from the special session of the IEEE CEC 2005 (Suganthan et al., 2005). These functions are similar to search spaces of real-world problems, and they are widely used to assess the balance between exploration and exploitation of a method (Črepinšek, Liu, & Mernik, 2013).
For the non-parametric statistical test, the experiments are conducted with 35 independent executions with N number of population agents corresponding to 30, and the number of iterations is 1000 for functions 1 to 13 and 500 for functions 14 to 19. The overall performance comparison of the algorithm is calculated using performance metrics, where 35 independent runs are evaluated to calculate the mean and standard deviation. The experiments are implemented with MATLAB 9.0 (R2016a), windows 10 64-bit on an IntelCore i3-5005U processor at 2.00 GHz and 6 GB in RAM. Statistics based on aptitude are explained in Section 4.2 performance measures.

4.2. Performance metrics

The performance of the algorithms is measured using statistical metrics calculated according to fitness of the test functions listed in Table 3. The convergence speed comparison is performed by counting the number of function calls (NFC); in this metric, a small value means higher speed before reaching the maximum number of function calls. For these tests, 35 independent executions were performed; the stop criterion used is a margin of error of $10^{-8}$ and a maximum number of function calls of $10^5$. To compare the convergence speed, the following equations are used:

1) Acceleration rate AR

The acceleration rate (AR) is defined as:

$$ AR = \frac{NFC_{\text{MSA}}}{NFC_{\text{OBMSA}}} $$

(21)

where NFC corresponds to the number of function calls where if AR greater than 1 indicated that OBMSA is faster than MSA. The average of the AR is used to determine the percentage of approximate acceleration of an algorithm. For N test functions the $AR_{\text{avg}}$ is determined by the following equation:

$$ AR_{\text{avg}} = \frac{1}{N} \sum_{i=1}^{N} AR_i $$

(22)

2) Number of function calls.

The number of function calls refers to the times that the algorithms successfully finds the optimum for every test function, and it is measured as the success rate (SR) defined as:

$$ SR_{\text{MSA}} = \frac{\text{NVT}_{\text{MSA}}}{\text{total number of trials}} $$

$$ SR_{\text{OBMSA}} = \frac{\text{NVT}_{\text{OBMSA}}}{\text{total number of trials}} $$

(23)

The ratio of success rate (RSR) is determined as:

$$ RSR = \frac{SR_{\text{MSA}}}{SR_{\text{OBMSA}}} $$

(24)

4.3. Comparison between MSA and OBMSA

The MSA optimization method is compared with the proposed OBMSA in terms of convergence speed and robustness through the solution of 19 benchmark functions which are previously defined in Table 1. The results of the comparison of the MSA and the OBMSA are shown in Table 2; the last row shows the average $AR_{\text{avg}}$ acceleration rate of 19 test functions with a value of 5.54, which means that OBMSA exceeds the MSA in an average of 55% faster, and it should be clarified that in the function F3, the proposed OBMSA reaches the function calls.

4.4. Comparison of OBMSA and other algorithms

1) Average fitness values

The following line presents the equation used to calculate the average, where $v_i$ corresponds to the fitness values of each function and N is the number of executions or runs.

$$ \mu = \frac{1}{N} \sum_{i=1}^{N} v_i $$

(25)

2) Standard deviation STD

$$ STD = \sqrt{\frac{1}{N-1} \sum_{i=1}^{N} (v_i - \mu)^2} $$

(26)

4.4.1. Algorithm parameters

In this experimental Section, five optimization methods were used in dealing with 19 functions to make a fair comparison between MFO (Mirjalili, 2015), MSA (Mohamed et al., 2017), PSO, DE, Harmony Search (HS) (Yang, 2009), and the proposed OBMSA. The parameters were configured with the same criteria for all methods, as well as for testing unimodal, multimodal, and composite functions. The configuration of the parameters is recorded in Table 3.

4.4.2. Unimodal and multimodal functions

The results showing the comparison of the above-mentioned algorithms in the unimodal functions ranging from F1 to F7 are presented in Table 6; Table 7 presents the comparison of the multimodal functions ranging from F8 to F13. Table 4 presents the average and Table 5, the standard deviation. All data were obtained by the algorithms after 35 executions for each function.

In Table 4 it is possible to observe that the OBMSA has found results closer to the optimum in F1, F2, and F5, F7 of the unimodal type, while the MSA won in the F3 function; the DE won in F6. On multimodal functions, the OBMSA obtained better results in F8 to F11 func-

Table 4

<table>
<thead>
<tr>
<th>Dim</th>
<th>NFC_MSA</th>
<th>NFC_OBMSA</th>
<th>AR</th>
<th>RSR</th>
</tr>
</thead>
<tbody>
<tr>
<td>F1</td>
<td>30</td>
<td>63,780</td>
<td>4434</td>
<td>14.38</td>
</tr>
<tr>
<td>F2</td>
<td>30</td>
<td>64,500</td>
<td>5946</td>
<td>10.85</td>
</tr>
<tr>
<td>F3</td>
<td>30</td>
<td>66,660</td>
<td>100,014</td>
<td>0.67</td>
</tr>
<tr>
<td>F4</td>
<td>30</td>
<td>66,480</td>
<td>18,150</td>
<td>3.86</td>
</tr>
<tr>
<td>F5</td>
<td>30</td>
<td>62,640</td>
<td>1410</td>
<td>44.43</td>
</tr>
<tr>
<td>F6</td>
<td>30</td>
<td>100,020</td>
<td>88,296</td>
<td>1.13</td>
</tr>
<tr>
<td>F7</td>
<td>30</td>
<td>76,830</td>
<td>100,014</td>
<td>0.77</td>
</tr>
<tr>
<td>F8</td>
<td>30</td>
<td>100,020</td>
<td>100,014</td>
<td>1.00</td>
</tr>
<tr>
<td>F9</td>
<td>30</td>
<td>60,100</td>
<td>9512</td>
<td>6.32</td>
</tr>
<tr>
<td>F10</td>
<td>30</td>
<td>61,600</td>
<td>17,466</td>
<td>3.53</td>
</tr>
<tr>
<td>F11</td>
<td>30</td>
<td>60,400</td>
<td>9900</td>
<td>6.10</td>
</tr>
<tr>
<td>F12</td>
<td>30</td>
<td>100,020</td>
<td>82,788</td>
<td>1.21</td>
</tr>
<tr>
<td>F13</td>
<td>30</td>
<td>62,780</td>
<td>26,466</td>
<td>2.37</td>
</tr>
<tr>
<td>F14</td>
<td>2</td>
<td>100,020</td>
<td>62,214</td>
<td>1.61</td>
</tr>
<tr>
<td>F15</td>
<td>4</td>
<td>46,140</td>
<td>54,600</td>
<td>0.85</td>
</tr>
<tr>
<td>F16</td>
<td>2</td>
<td>100,020</td>
<td>100,014</td>
<td>1.00</td>
</tr>
<tr>
<td>F17</td>
<td>2</td>
<td>4080</td>
<td>1022</td>
<td>3.95</td>
</tr>
<tr>
<td>F18</td>
<td>2</td>
<td>870</td>
<td>1626</td>
<td>0.54</td>
</tr>
<tr>
<td>F19</td>
<td>3</td>
<td>100,020</td>
<td>100,014</td>
<td>1.00</td>
</tr>
<tr>
<td>AR_avg</td>
<td></td>
<td></td>
<td></td>
<td>5.54</td>
</tr>
</tbody>
</table>
Table 5 shows the standard deviation calculated for all the algorithms. It can be observed that the OBMSA obtains better results than the rest of the methods, reaching the minimum values in functions F1, F2, F3, F4, F7, F9, F10, F11, F16, followed by the MSA with similar results except in the F16. The algorithm that follows in the rank is the with the DE in functions F6, F12, F13, F15, and F19. Then, the algorithm that continues is the HS reaching the minimum values for functions F8 and F14. Finally, MFO with only one function that obtains the minimum the F17 and the PSO with the F18.

4.4.3. Wilcoxon non-parametric test

Wilcoxon is a non-parametric test applied by pairs; its objective is to detect significant differences between two samples belonging to different algorithms, that is, the behavior of two algorithms. Table 6 shows the p values obtained in the test, which indicates that the OBMSA algorithm is statistically significantly superior. The functions in which OBMSA achieves results greater than 0.05 have been highlighted, even though OBMSA does not provide better results in a little less than half of the established functions, the values reached for p shows that the results of this algorithm are very competitive. Notice that in Table 6 N/A means that the algorithms statistically similar.

4.5. OBMSA applied to problems with restrictions

Any problem in which certain parameters must be determined to satisfy constraints can be formulated as an optimization problem (Arora, 2011). There are functions in which the global optimum is located in a small area and difficult to locate within the search space, which in some cases becomes extensive; these problems have a large number of inequality restrictions. In this section, we took three engineering problems used to analyze the performance and efficiency of the OBMSA optimization algorithm. The idea is to test the OBMSA in constrained optimization problems. In this kind of functions, penalization methods are regularly used. In the experiments performed over the engineering problems, the OBMSA employs an easy penalization function called the death penalty, where a very large number is assigned to the particle if it does not comply with any of the restrictions posed in the problem. For the tests, a maximum of 100 iterations was assigned for each execution, and 35 solutions were carried out.

4.5.1. Design problem of a welded beam

The welded beam design belongs to a list of structural problems consisting of an objective function and five non-linear restrictions used to test optimization methods (Deb, 1991), it is a practical design problem to minimize the cost of manufacturing a T-beam and the weld required to attach it to the S-frame, the main objective is to find the feasible set of dimensions of the design variables h, l, t and b to carry a certain load (P) (Lee & Geem, 2005; Coello Coello, 2000b) where h is the weld width, l is the length of the weld, t is the height and b is the width of the beam as shown in Fig. 6. Penalty functions are used to transform this restricted problem into an unrestricted problem (Hwang & He, 2006) (Fig. 7).

The mathematical definition of the objective function is formally expressed as follows:

\[ X = [x_1, x_2, x_3, x_4] = [h, l, t, b] \]  

Minimize: \[ F(X) = 1.10471x_1^2 + 0.04811x_1x_2(14.0 + x_2) \]

Subject to: \[ g_1(X) = r(X) - r_{max} \leq 0 \]
\[ g_2(X) = s(X) - s_{max} \leq 0 \]
\[ g_3(X) = x_1 - x_4 \leq 0 \]
\[ g_4(X) = 0.10471x_1^2 + 0.04811x_1x_4(14.0 + x_2) - 5.0 \leq 0 \]
Table 6  
Average of unimodal, multimodal, and composite functions (The bold values represent the best solutions).

<table>
<thead>
<tr>
<th>FUNCTION</th>
<th>OBMSA</th>
<th>MSA</th>
<th>MPO</th>
<th>PSO</th>
<th>DE</th>
<th>HS</th>
<th>F_{min}</th>
</tr>
</thead>
<tbody>
<tr>
<td>F1</td>
<td>0.000E+00</td>
<td>0.000E+00</td>
<td>2.000E+03</td>
<td>1.548E-09</td>
<td>2.654E-12</td>
<td>1.3786E+00</td>
<td>0</td>
</tr>
<tr>
<td>F2</td>
<td>5.205E-194</td>
<td>1.029E-182</td>
<td>3.200E+01</td>
<td>3.346E-01</td>
<td>3.634E-08</td>
<td>1.606E-01</td>
<td>0</td>
</tr>
<tr>
<td>F3</td>
<td>5.022E-276</td>
<td>1.702E-290</td>
<td>1.872E+04</td>
<td>1.525E+03</td>
<td>2.527E+04</td>
<td>5.737E+03</td>
<td>0</td>
</tr>
<tr>
<td>F4</td>
<td>1.136E-165</td>
<td>4.533E-170</td>
<td>6.911E+01</td>
<td>7.668E+00</td>
<td>1.962E+00</td>
<td>6.561E+04</td>
<td>0</td>
</tr>
<tr>
<td>F5</td>
<td>2.777E+01</td>
<td>2.795E+01</td>
<td>8.245E+03</td>
<td>2.477E+02</td>
<td>4.701E+01</td>
<td>3.553E+02</td>
<td>0</td>
</tr>
<tr>
<td>F6</td>
<td>1.834E-02</td>
<td>2.135E-02</td>
<td>1.714E+03</td>
<td>3.336E-10</td>
<td>2.692E-12</td>
<td>1.253E+00</td>
<td>0</td>
</tr>
<tr>
<td>F7</td>
<td>1.900E-04</td>
<td>2.305E-04</td>
<td>4.511E+00</td>
<td>4.954E-02</td>
<td>2.824E-02</td>
<td>4.762E-01</td>
<td>0</td>
</tr>
<tr>
<td>F8</td>
<td>-1.157E+04</td>
<td>-1.167E+03</td>
<td>-8.299E+03</td>
<td>-8.112E+03</td>
<td>-1.241E+04</td>
<td>-1.257E+04</td>
<td>-415.5</td>
</tr>
<tr>
<td>F9</td>
<td>0.000E+00</td>
<td>0.000E+00</td>
<td>1.683E+02</td>
<td>3.942E+01</td>
<td>5.833E+01</td>
<td>5.927E+01</td>
<td>0</td>
</tr>
<tr>
<td>F10</td>
<td>8.882E-16</td>
<td>8.882E-16</td>
<td>1.376E+01</td>
<td>3.846E-02</td>
<td>4.770E-07</td>
<td>7.381E-01</td>
<td>0</td>
</tr>
<tr>
<td>F11</td>
<td>0.000E+00</td>
<td>0.000E+00</td>
<td>2.067E+01</td>
<td>3.392E-02</td>
<td>1.626E-09</td>
<td>9.043E-01</td>
<td>0</td>
</tr>
<tr>
<td>F12</td>
<td>3.076E-03</td>
<td>1.041E-04</td>
<td>5.953E-01</td>
<td>3.260E-02</td>
<td>3.992E-13</td>
<td>1.292E-02</td>
<td>0</td>
</tr>
<tr>
<td>F13</td>
<td>1.169E-01</td>
<td>9.086E-02</td>
<td>5.244E-01</td>
<td>7.264E-02</td>
<td>1.626E-02</td>
<td>2.716E-01</td>
<td>0</td>
</tr>
<tr>
<td>F14</td>
<td>2.599E+00</td>
<td>4.192E+00</td>
<td>2.515E+00</td>
<td>1.252E+00</td>
<td>1.052E+00</td>
<td>9.980E-01</td>
<td>1</td>
</tr>
<tr>
<td>F15</td>
<td>4.426E-03</td>
<td>3.210E-03</td>
<td>2.036E-03</td>
<td>2.383E-03</td>
<td>1.286E-03</td>
<td>1.318E-03</td>
<td>0.0003</td>
</tr>
<tr>
<td>F16</td>
<td>-1.032E+00</td>
<td>-1.032E+00</td>
<td>-1.032E+00</td>
<td>-1.032E+00</td>
<td>-1.032E+00</td>
<td>-1.032E+00</td>
<td>-1.031E+00</td>
</tr>
<tr>
<td>F17</td>
<td>3.976E-01</td>
<td>3.976E-01</td>
<td>3.976E-01</td>
<td>3.976E-01</td>
<td>3.976E-01</td>
<td>3.976E-01</td>
<td>0.398</td>
</tr>
<tr>
<td>F18</td>
<td>7.629E-01</td>
<td>1.149E+01</td>
<td>3.000E+00</td>
<td>3.000E+00</td>
<td>3.000E+00</td>
<td>3.000E+00</td>
<td>3</td>
</tr>
<tr>
<td>F19</td>
<td>-3.863E+00</td>
<td>-3.863E+00</td>
<td>-3.863E+00</td>
<td>-3.863E+00</td>
<td>-3.863E+00</td>
<td>-3.863E+00</td>
<td>-3.86</td>
</tr>
</tbody>
</table>

$g_{1}(X) = 0.125 - x_{1} \leq 0$
$g_{2}(X) = d(x) - d_{max} \leq 0$
$g_{3}(X) = P - P_{c}(X) \leq 0$

where $r(x) = \sqrt{\left(r'\right)^{2} + 2r'^{T}r'' + (r'')^{2}}$

\[ r' = \frac{P}{\sqrt{2}v_{1}v_{2}}, \quad r'' = \frac{MR}{J}, \quad M = P \left( \frac{L + 2r}{2} \right) \]

\[ R = \sqrt{\frac{x_{1}^{2}}{4} + \left( \frac{x_{1} + x_{2}}{2} \right)^{2}} \]

\[ J = 2 \left\{ \sqrt{2x_{1}x_{2}} \left[ \frac{x_{1}^{2}}{12} + \left( \frac{x_{1} + x_{2}}{2} \right)^{2} \right] \right\} \]

\[ \sigma(X) = \frac{4P}{x_{1}s_{1}^{2}}, \quad \delta(X) = \frac{4P^{3}}{3x_{1}^{2}s_{1}^{4}} \]

\[ P_{e}(X) = \frac{4.013E}{} \text{E}^{\frac{v_{1}^{2}}{2}} (1 - x_{1})^{\frac{E}{2L}} \int_{0}^{E} \frac{dG}{G} \]

$P = 6000ib, \quad L = 14in, \quad E = 30 \times 10^{6}psi, \quad G = 12 \times 10^{6}psi$

$r_{max} = 13,600psi, \quad \sigma_{max} = 30,000psi, \quad \delta_{max} = 0.25in$

where $0.1 \leq x_{1}, x_{2} \leq 2$,
$0.1 \leq x_{1}, x_{2} \leq 10$

The OBMSA algorithm was applied to the problem of design optimization of a welded beam, the optimum results obtained were compared with previous solutions presented by Lee and Geem for HS (Lee & Geem, 2005), Hu, Eberhart, and Shi in PSO (Hu, Eberhart, & Shi, 2000).
### 4.5.2. The problem of tension/compression of the design of a spring

The purpose of this engineering problem is to minimize the weight of spring (Arora, 2011; Coello Coello, 2000b) subject to constant stress restrictions, over tension frequency, minimum deflection, outer diameter limit, and design variables.

The design variables are:
- D Coil diameter
- d wire diameter
- N number of active coils.

The problem is then formally expressed:

\[
\text{Considerring : } X = [x_1, x_2, x_3] = [d, D, N] \tag{28}
\]

Minimize: \( F(X) = (N + 2)Dd^2 \)

Subject to:

\[
g_1(X) = 1 - \frac{d^2N}{71985 d^2} \leq 0
\]

\[
g_2(X) = \frac{1}{125660 (Dd^2 - d^2)} + 1 - \frac{1}{5108d^2} \leq 0
\]

\[
g_3(X) = 1 - \frac{140.45d}{D^2N} \leq 0
\]

\[
g_4(X) = \frac{D + d}{1.5} - 1 \leq 0
\]

where \( 0.05 \leq x_1 \leq 2 \), \( 0.25 \leq x_2 \leq 1.3 \), \( y \leq x_3 \leq 15 \).

Table 8 presents the results of the tests performed, which indicate that the OBMSA functions effectively in solving this problem by delivering a better design compared to the listed algorithms.

### 4.5.3. Pressure vessel design problem

This design problem corresponds to a cylindrical container that is covered on both sides with cylindrical lids as can be seen in Fig. 8; the problem comprises the material and welding costs of the cylindrical container in order to minimize the total cost (Abd ElAziz et al., 2017; Coello Coello, 2002).

The optimization operation tries to determine the optimal value of four design variables: \( T_0 \) thickness of the casing, \( L \) length of the cylindrical section without considering the head, \( R \) is the inner radius, and \( Th \) that corresponds to the thickness of the head. In Fig. 8, these design variables are appreciated, where \( T_0 \), as well as \( T_3 \) must contain integer values multiples of 0.0625 in. that are available thicknesses of the rolled steel plates, \( R \) and \( L \) are continuous variables (Kannan & Kramer, 1994).

The design problem is then formally expressed:

\[
\text{Considerring : } X = [x_1, x_2, x_3, x_4] = [T_0, T_3, R, L] \tag{29}
\]

Minimize: \( 0.6224x_1x_3x_4 + 1.7781x_2x_3^2 + 3.1661x_2^2x_4 + 19.84x_2^3x_3 \)

Subject to:

\[
g_1(X) = -x_1 + 0.0193x_3 \leq 0
\]

\[
g_2(X) = -x_3 + 0.00954x_3 \leq 0
\]

\[
g_3(X) = -\pi x_2^3x_4 - \frac{1}{2} \pi x_1^3 + 1296000 \leq 0
\]

\[
g_4(X) = x_4 - 240 \leq 0
\]

where: \( 0 \leq x_1, x_2 \leq 99, 10 \leq x_3, x_4 \leq 200 \).

The optimal results obtained were compared with previous solutions presented in previous works by Lee and Geem for HS (Lee & Geem, 2005), Hu, Eberhart, and Shi in PSO (Hu et al., 2003), Abd Elaziz, Oliva, and Xiong for OBSCA (Abd ElAziz et al., 2017), Huang applied to the algorithm DE (Huang et al., 2007), Mirjalili in MFO (Mirjalili, 2015) and GWO (Mirjalili, Mirjalili, & Lewis, 2014), Coello in (Coello Coello, 2000b) as shown in Table 7, based on the values obtained OBMSA delivers a better set of design variables compared to the compared algorithms.
Table 9
Comparison of the results of the welded beam design problem.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>h</th>
<th>l</th>
<th>t</th>
<th>b</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>OBMSA</td>
<td>0.23081</td>
<td>3.0690</td>
<td>8.9885</td>
<td>0.2085</td>
<td>1.7220</td>
</tr>
<tr>
<td>HS (Lee &amp; Geem, 2005)</td>
<td>0.2442</td>
<td>6.2231</td>
<td>8.2915</td>
<td>0.24</td>
<td>2.3807</td>
</tr>
<tr>
<td>PSO (Hu et al., 2003)</td>
<td>0.20573</td>
<td>3.47049</td>
<td>9.03662</td>
<td>0.20573</td>
<td>1.7248</td>
</tr>
<tr>
<td>GWO (Mirjalili et al., 2014)</td>
<td>0.2057</td>
<td>3.4784</td>
<td>9.03681</td>
<td>0.2058</td>
<td>1.7262</td>
</tr>
<tr>
<td>OBSCA (Abd ElAziz et al., 2017)</td>
<td>0.2082</td>
<td>3.0661</td>
<td>8.9885</td>
<td>0.2088</td>
<td>1.7224</td>
</tr>
<tr>
<td>DE (Huang, Wang, &amp; He, 2007)</td>
<td>0.2066</td>
<td>3.4914</td>
<td>8.9547</td>
<td>0.2102</td>
<td>1.7480</td>
</tr>
<tr>
<td>MFO (Mirjalili, 2015)</td>
<td>0.2057</td>
<td>3.4703</td>
<td>9.0364</td>
<td>0.2057</td>
<td>1.7245</td>
</tr>
<tr>
<td>ABC (Akay &amp; Karaboga, 2012)</td>
<td>0.2058</td>
<td>3.4705</td>
<td>9.0367</td>
<td>0.2057</td>
<td>1.7249</td>
</tr>
<tr>
<td>Coello (Carlos A. Coello Coello, 2000a)</td>
<td>0.2088</td>
<td>3.4205</td>
<td>8.9975</td>
<td>0.2100</td>
<td>1.7484</td>
</tr>
<tr>
<td>GA (Deb, 1991)</td>
<td>0.2489</td>
<td>6.173</td>
<td>8.1789</td>
<td>0.2533</td>
<td>2.4300</td>
</tr>
</tbody>
</table>

Table 10
Comparison of the results of the problem stop/compression of the spring design.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>d</th>
<th>D</th>
<th>N</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>OBMSA</td>
<td>0.0607</td>
<td>0.2847</td>
<td>2.0852</td>
<td>0.0043</td>
</tr>
<tr>
<td>HS (Lee &amp; Geem, 2005)</td>
<td>0.0515</td>
<td>0.3499</td>
<td>12.0764</td>
<td>0.0127</td>
</tr>
<tr>
<td>PSO (Hu et al., 2003)</td>
<td>0.0518</td>
<td>0.3577</td>
<td>11.2446</td>
<td>0.0127</td>
</tr>
<tr>
<td>GWO (Mirjalili et al., 2014)</td>
<td>0.05169</td>
<td>0.3567</td>
<td>11.2882</td>
<td>0.01266</td>
</tr>
<tr>
<td>OBSCA (Abd ElAziz et al., 2017)</td>
<td>0.0523</td>
<td>0.3173</td>
<td>12.5486</td>
<td>0.01262</td>
</tr>
<tr>
<td>DE (Huang, Wang, &amp; He, 2007)</td>
<td>0.0517</td>
<td>0.3548</td>
<td>11.4108</td>
<td>0.0127</td>
</tr>
<tr>
<td>MFO (Mirjalili, 2015)</td>
<td>0.0520</td>
<td>0.3642</td>
<td>10.8685</td>
<td>0.0127</td>
</tr>
<tr>
<td>ABC (Akay &amp; Karaboga, 2012)</td>
<td>0.05174</td>
<td>0.3581</td>
<td>11.2038</td>
<td>0.01266</td>
</tr>
<tr>
<td>Coello (Coello Coello, 2000a)</td>
<td>0.0515</td>
<td>0.3517</td>
<td>11.6322</td>
<td>0.0127</td>
</tr>
</tbody>
</table>

Based on the results obtained, we can conclude that OBMSA results are better than all other algorithms.

5. Conclusions and future works

This article introduced an improved intelligent system for global optimization called OBMSA. The proposed approach considers the use of the OBL that is an intelligent rule that permits to improve the search process in optimization algorithms. The OBMSA used the OBL in two parts; the first is in the initialization and permits to have a higher range of action in the early iterations. The second is after the prospector moths were updated; this permits us to have a better perspective of the search space for the exploration. The results obtained from the OBMSA have been compared with various optimization methods included the original MSA. The intelligent systems used in the experiments permit a fair comparison and also support the use of this kind of approaches for solving complex optimization problems. The combination of an intelligent rule as the OBL substantially increases the performance of the standard MSA. The experiments showed that the OBMSA is 55% faster in convergence speed than the MSA, and the accuracy of the solution is therefore improved. Based on the obtained values, we can conclude that the OBMSA is better than the MSA and those other intelligent algorithms such as those with which it was compared in the precision of the obtained solution as well as in the convergence. In the tests using engineering problems, it can be concluded, based on the obtained results, that the OBMSA presents considerable results. In both mathematical and engineering problems, the improved results from the use of OBL permit to explore the search space in two directions at the same time. This fact helps the search procedure to analyze different regions and decide which is more probably to find the global optimal. Moreover, the OBL also permits that the algorithm escapes from the suboptimal regions. However, intelligent systems as the OBMSA are not perfect. The main limitation of the proposed method is that the OBL process implies more function evaluations, and it could affect the computational time depending on the problem.

In future work, we expect to include some analysis that permits us to decide when it is necessary to use the OBL; this will help to reduce the function calls. The OBMSA should also be tested in some real applications, for example in image processing for image segmentation. The field of solar cells and photovoltaic modules is also a future branch of research. We also want to explore the implementation of the OBMSA for solving the problem of future selection in data mining. Finally, it will be interesting to modify the OBMSA to solve multiobjective optimization problems. These research directions are strongly related to the use of expert and intelligent systems in real applications.

CRediT authorship contribution statement

Diego Oliva: Supervision, Methodology, Conceptualization, Investigation, Validation, Data curation, Formal analysis, Writing - review & editing. Sara Esquivel-Torres: Software, Resources, Writing - original draft, Writing - review & editing. Salvador Hinojosa: Software, Conceptualization, Data curation, Formal analysis, Resources, Writing - review & editing. Marco Pérez-Cisneros: Formal analysis, Writing - review & editing. Valentin Osuna-Enciso: Formal analysis, Writing - review & editing. Noé Ortega-Sánchez: Conceptualization, Data curation, Resources. Gaurav Dhiman: Data curation, Formal analysis, Resources, Writing - review & editing. Ali Asghar Heidari: Conceptualization, Investigation, Formal analysis, Writing - review & editing.
Declaraction of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References


